User's Manual

Renesas Synergy™ Software Package（SSP）v1.3.2ユーザーズマニュアル（参考資料）

Renesas Synergy™ プラットフォーム

本資料の第 1 章から第 5 章は英語版 Rev.1.13 を日本語に翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントをご参照ください。

資料番号 R01US0315EU0113、リビジョン Rev.1.13、発行日 2017 年 11 月 20 日の翻訳版です。

Renesas Electronics
www.renesas.com

Rev.1.13　2018.04

SSP User's Manual

第 1 章 Renesas Synergy Software Package (SSP)

1.1 Renesas Synergy™ ソフトウェアパッケージの概要

1.1.1 SSP ユーザーズマニュアルの概要

このマニュアルでは、Renesas Synergy Software Package を使用して、Synergy マイクロコントローラシリーズ対応のアプリケーションを作成する方法について説明します。下の図では、SSP ユーザーズマニュアルの API リファレンスコンポーネントが青で示されています。このマニュアルではまた、SSP でのプログラミング手順を理解するため、e^2 studio ISDE の説明やチュートリアルと例など、その他のコンポーネントについても記載されています。

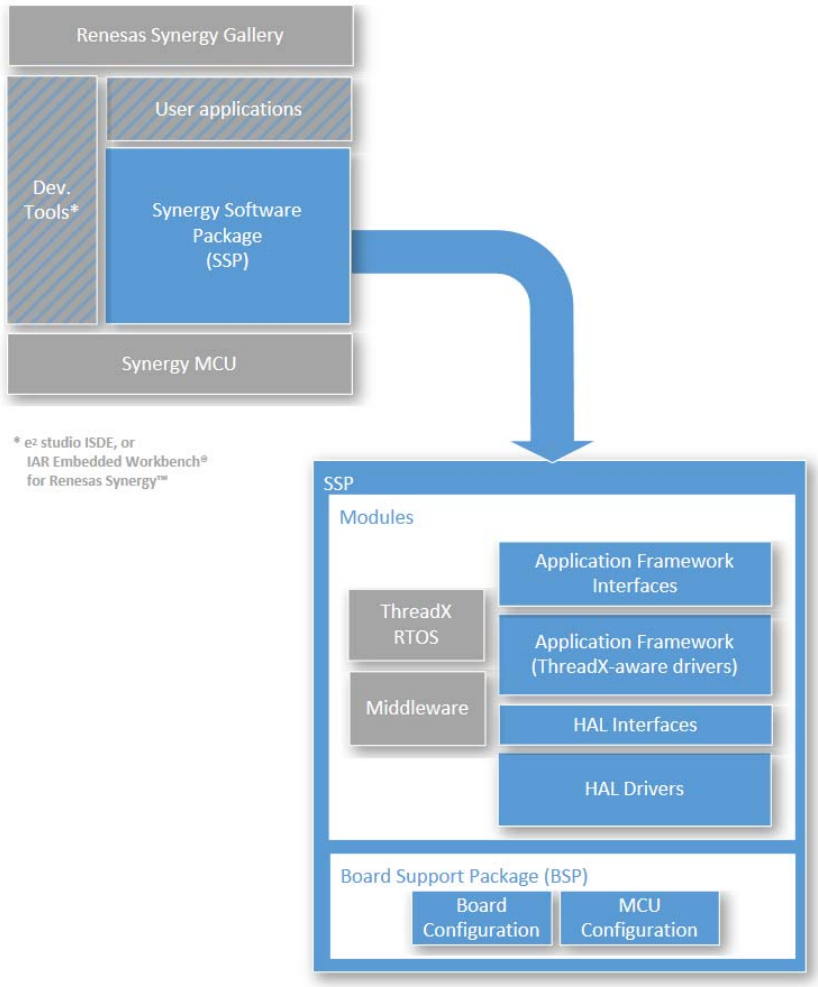


図 1: Synergy Software Package（SSP）のドキュメント

1.1.2 このマニュアルに含まれる内容

SSP アーキテクチャおよび SSP でのボードとチップレベルのサポートについては、以下を参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

SSP を使用したプログラミングおよび e^2 studio ISDE の概要説明については、以下を参照してください。

e^2 studio ISDE ユーザーガイド

入門レベルのチュートリアルとアプリケーション例については、以下を参照してください。

- チュートリアル : Your First Synergy Project - Blinky

- チュートリアル : Using HAL Drivers - Programming the WDT

「モジュールの概要」では SSP モジュールについて説明します。以下を参照してください。

- フレームワークレイヤー
- HAL レイヤー

このドキュメントには、以下の SSP コンポーネントの API リファレンスドキュメントが含まれています。

- API Reference: Framework Interfaces: ThreadX 対応のフレームワークモジュールへのインタフェース
- API Reference: Framework Layer: ThreadX 対応のフレームワークドライバーモジュール
- API Reference: HAL Interfaces: ハードウェア抽象化レイヤー（HAL）モジュールへのインタフェース
- API Reference: HAL Layer: RTOS 独立の HAL ドライバーモジュール
- Board Support Package: ボード固有、かつマイクロコントローラ固有の設定モジュールが含まれたボードサポートパッケージ（BSP）

第 2 章　SSP の概要

2.1 目的

Renesas Synergy™ プラットフォームの一部である Renesas Synergy™ Software Package（SSP）は、使いやすく拡張性に優れた高品質の組み込みシステム設計ソフトウェアを提供するための完全な統合ソフトウェアパッケージです。Synergy ソフトウェアプラットフォームを使用することで、完全に統合された認証済みの組み込みソフトウェアプラットフォームが提供する、完全に最適化され、強化された業界屈指のリアルタイムオペレーティングシステム（RTOS）、ミドルウェア、通信スタック、機能ライブラリ、アプリケーションフレームワーク、ハードウェア抽象化された低レベルデバイスドライバーを利用でき、迅速な商品化が可能になります。

2.2 概要

SSP は、以下の 4 つの主要レイヤーで構成されています。

- API Reference: Framework Interfaces フレームワークライブラリは、ハードウェアを機能のユースケースとして抽象化する共通インタフェースを介して Synergy ハードウェアペリフェラルに接続します。「インタフェースレイヤー」は、関数とパラメータの定義が入ったヘッダーファイルのグループであり、コード用のスペースは含まれていません。
- API Reference: Framework Layer 「フレームワークレイヤー」は、RTOS 統合されたドライバーおよび有益なアプリケーションコードで構成されています。
- API Reference: HAL Interfaces 「HAL レイヤー」インタフェースは、RTOS 独立の HAL レベルのドライバーに接続します。
- API Reference: HAL Layer HAL レイヤーのドライバーとハードウェアレジスタによって、インタフェースが実装されます。

2.3 使いやすさ

SSP は、統一的で直観的な API とそれに関する充実したドキュメントを提供しています。各モジュールには詳細なユーザードキュメントに加え、各関数のコードサイズや実行時間などを記載したソフトウェアデータシートが用意されています。

2.4 スケーラビリティ

ユーザーは、フレームワークレイヤー、インタフェースレイヤー、HAL レイヤーの中から、アプリケーションのニーズに最も適したレイヤーを使用してプラットフォーム機能と統合することができます。各モジュールをさらに拡張するには、パラメータチェッキングなどのビルド時オプションをコンパイルアウトすることで、小さく効率的なコードを作成できます。

2.5 SSP アーキテクチャ

2.5.1 SSP アーキテクチャ

このセクションでは、Renesas Synergy ソフトウェアパッケージ（SSP）アーキテクチャと SSP アプリケーションプログラミングインタフェース（API）の使用方法について説明します。

2.5.1.1 SSP の概要

マイクロコントローラの複雑さが増すにつれ、幅広い知識がないと想定どおりに動作させることが難しくなっています。 SSP は、そのようななかで IoT アプリケーション用の組み込みソフトウェアの開発に革新をもたらすものです。 SSP を使用すると、まったく新しい強力なソフトウェアインタフェースにより、コーディングにかかる時間を短縮しつつ、確実な開発プロセスを実現することができます。このソフトウェアでは、数か月を費やしてベースとなるコードを開発するのではなく、個別のアプリケーションコードを作成することによって、ハードウェアレベルのインタフェースを実装できます。

2.5.1.2 SSP 用語

Term	説明	Reference
Module	モジュールは、ペリフェラルドライバーから単なるソフトウェアまでのすべてに該当します。各モジュールは、ソースコード、ドキュメント、およびユーザーがコードを効率的に使用するために必要な要素を含む 1 つのフォルダで構成されます。モジュールは独立した単位ですが、他のモジュールに依存する場合があります。SSP モジュールの例として、UART ドライバー (UART Interface)、オーディオ再生フレームワーク （タイマ、 DMA、 DAC ドライバー (Audio Framework Interface) に依存する）、およびメッセージフレームワーク (Messaging Framework Interface) が挙げられます。複数のモジュールを組み合わせてユーザーに必要な機能を提供することで、アプリケーションを構築できます。	SSP モジュール
BSP	Board Support Package の略語。 SSP では、他の SSP モジュールが問題なく連動するために最低限必要な基礎を BSP によって提供します。	BSP アーキテクチャ
Callback Function	この用語はイベント発生時に呼び出される関数を意味します。たとえば、バスエラー割り込みハンドラは、 r_bsp. 内に実装されています。ユーザーがバスエラーの発生を知りたいと思うことはよくあります。そのようなときは、 r_bsp. にコールバック関数を提供すると、ユーザーに通知を送ることができます。実際にバスエラーが発生したときは、r_bsp が提供されたコールバック関数にジャンプして、ユーザーがそのエラーを処理できます。割り込みコールバック関数は長くならないように慎重に扱う必要があります。なぜなら、この関数が呼び出されると、現在の割り込みが中断されてMCUがまた別の割り込みに入ることになるため、保留中の割り込みがすべて後回しにされるからです。	-

Term	説明	Reference
Interface	「SSP インタフェース」セクションを参照してください（SSP インタフェース）。SSP 内のすべてのインタフェースは、API Reference: Framework Interfaces と API Reference: HAL Interfaces に記載されています。	SSP インタフェース
Instance	「SSP インスタンス」セクションを参照してください（SSP インスタンス）。	SSP インスタンス
Module Instance	あるモジュールの単一で独立した構成。	-
Application	ユーザーが所有者として維持管理するコード。アプリケーションコードは、Renesas から提供されるサンプルアプリケーションコードをベースに作成することができますが、責任はユーザーにあります。	An example for a simple application is included as tutorial in this manual: チュートリアル : Using HAL Drivers - Programming the WDT
Driver	ドライバーは、MCU 上のレジスタを直接変更する特殊なモジュールです。	-
Stacks	SSP アーキテクチャは、モジュールが連動してスタックを形成するように設計されています。最上位のモジュールから最下位のモジュールまでの依存関係が特定のスタックを形成します。	SSP スタック
Layer/Level	スタックは、複数のモジュールのレイヤーで構成されます。各レイヤーは、1 つ上のレイヤーの要件に応じて 1 つまたは複数のモジュールで構成されます。レイヤーとレベルは区別なく使用されます。	SSP 事前定義レイヤー

2.5.1.3 SSP モジュール

モジュールは SSP の中核的構成要素です。モジュールを使用すれば、さまざまなことを実行できますが、どのモジュールでも、上位に機能を提供し下位に機能を要求するという基本概念は共通しています。

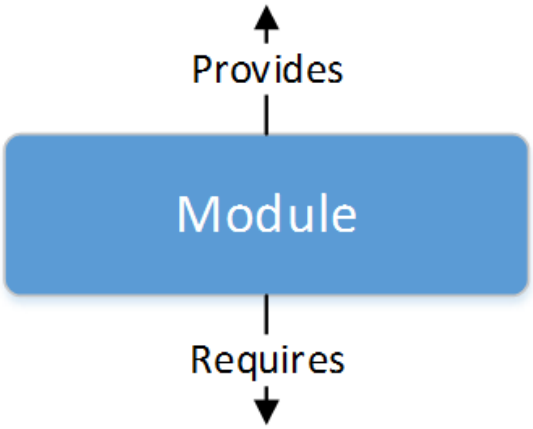


図 2:　モジュール

モジュールによって提供される機能の量に制限はありませんが、通常は意味をなさなくなる限界点が存在します。提供される機能が多すぎると、将来、そのモジュールの再利用が困難になります。提供される機能が不十分だと、モジュールを想定どおりに機能させるための複雑さとオーバーヘッドが不必要に増加します。

最も単純な SSP アプリケーションは、最上位にユーザーアプリケーションを据えた 1 つのモジュールで構成されます。

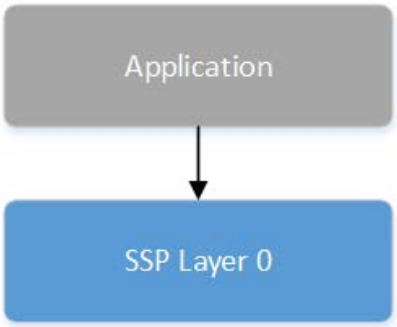


図 3:　アプリケーションを含むモジュール

簡単にするために、現時点ではボードサポートパッケージ（BSP）を無視します。上の図では、BSP は最下位レイヤー（SSP Layer 0）の下に配置されています。

2.5.1.4 SSP スタック

モジュールが上下に重なると、SSP スタックが形成されます。このスタック処理は、あるモジュールが提供しているものと別のモジュールが必要としているものが一致したときに実行されます。たとえば、オーディオ再生フレームワークモジュールには転送インタフェースが必要ですが、それはデータトランスファコントローラ（DTC）ドライバーモジュールで実現できます。オーディオ再生モジュールに DTC コードが組み込まれる代わりに、それらが 2 つのモジュールに分割されています。これにより、基礎となるモジュールの再利用が可能になり、多くのメリットが得られます。

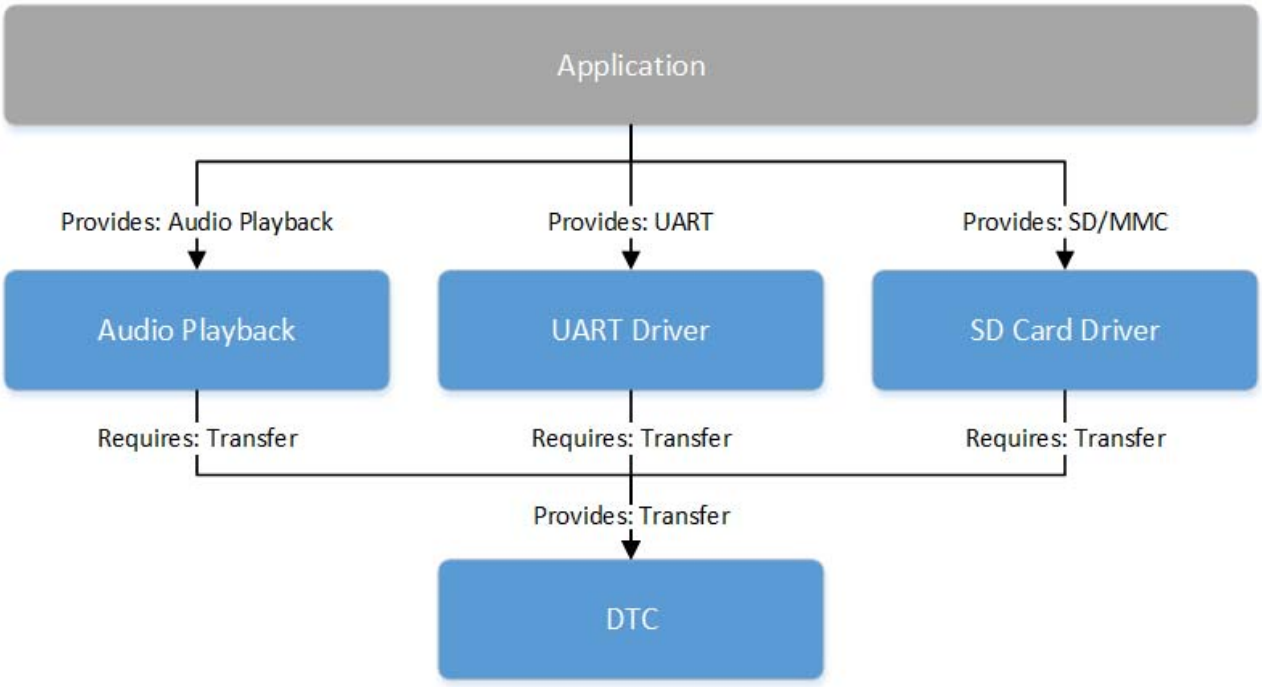


図 4:　スタック – オーディオ再生

SSP モジュールを使ってレイヤーをスタックに追加し続けることにより、Synergy MCU と高レベルでやり取りすることができます。

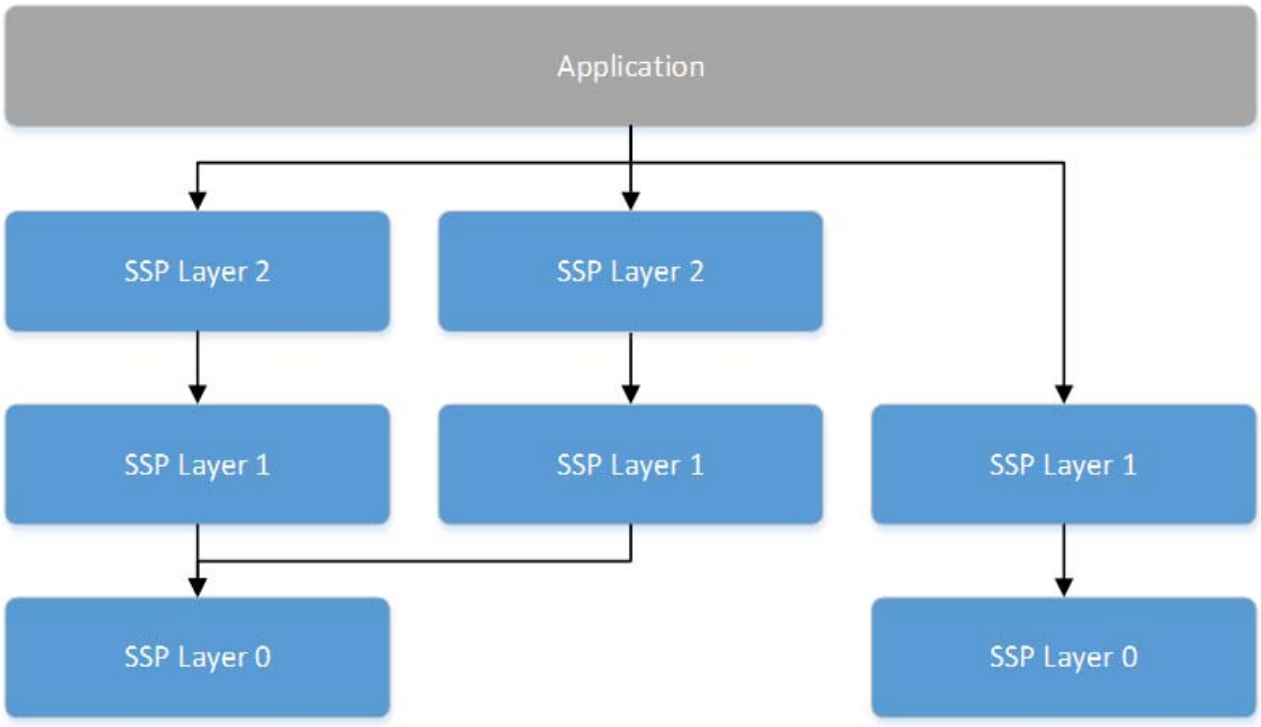


図 5: スタック

モジュールをスタックできると、アーキテクチャ全体の柔軟性が保証されるため、大きなメリットが得られます。モジュールが他のモジュールに直接依存している場合は、アプリケーション機能がさまざまなユーザー設計で機能できないときに問題が発生します。モジュールが再利用可能なことを保証するには、同じ機能を提供する他のモジュールに対してモジュールをスワップアウトできる必要があります。SSP アーキテクチャは、SSP インタフェースの使用を通じてモジュールをスワップインまたはスワップアウトできる柔軟性を提供しています。

2.5.1.5 SSP インタフェース

アーキテクチャレベルでは、モジュールはインタフェースを通じて共通機能を提供します。この共通性によって、同じインタフェースに接続されているモジュールを区別せずに使用できます。インタフェースは、2 つのモジュール間の契約と見なすことができます。モジュールは、契約内で同意された情報を使用して連動することに同意します。

Synergy MCU デバイスでは、別々のペリフェラルが重複していることがあります。たとえば、I2C 通信は IIC ペリフェラルまたは簡易 I2C モードの SCI ペリフェラルの使用を通して実現できます。両方のペリフェラルから提供される機能のレベルに違いがあります。I2C モードの SCI は、フル機能を備えた IIC の機能の一部しかサポートしません。

インタフェースは、ほとんどのユーザーが期待する共通機能のサポートを提供することを目的としています。つまり、IIC などの高度なペリフェラルのなかには、インタフェースで使用できないものがあります。このような機能のほとんどは、インタフェース拡張機能を介して使用することができます。

設計上、インタフェースはヘッダーファイルとして実装されます。すべてのインタフェースヘッダーファイル名は '_api.h'. で終わります。ここからは、インタフェースの構成要素について説明します。

SSP インタフェース列挙

可能な場合、インタフェースは関数パラメータと構造体メンバーに対して型指定列挙を使用します。以下に例を示します。

```
typedef enum e_i2c_addr_mode
{
  I2C_ADDR_MODE_7BIT = 1, ///< Use 7-bit addressing mode
  I2C_ADDR_MODE_10BIT ///<; Use 10-bit addressing mode
} i2c_addr_mode_t;
```

列挙を使用すると、パラメータで使用可能な値を決定する際の不確実性を取り除くことができます。また、列挙オプションは型の名前が使用可能なオプションに基づいてプレフィックスされる厳密な命名規則に従っていることにも注意してください。命名規則と e^2 studio で使用可能なオートコンプリート機能を組み合わせると、高度に読み取りやすいコードを維持しながら、コーディングが短時間で完成するメリットを得ることができます。

SSP インタフェースデータ構造体

少なくとも、すべての SSP インタフェースには、制御構造体、構成構造体、およびインスタンス構造体という 3 つのデータ構造体が含まれています。

制御構造体はモジュールを使用する場合の一意の識別子として使用されます。 SSP モジュールが唯一のペリフェラルドライバーの場合は、この制御構造体がチャネル番号に置き換えられます。その場合は、モジュールに対するすべての関数呼び出しでチャネル番号が使用されるため、コードで動作対象のペリフェラルチャネルを特定できます。 SSP モジュールはデバイスドライバーに制限されないため、制御構造体が使用されます。ユーザーは、制御構造体用のストレージを割り当ててから、その構造体へのポインタを open() 呼び出しに渡します。この時点で、モジュールが必要に応じて構造体を初期化します。その後、ユーザーがそれ以降のすべてのモジュール呼び出しに対して制御構造体へのポインタを送る必要があります。制御構造体の内容はモジュールによって使用されるものであるため、変更されないようにする必要があります。 SSP リリース間でデータ構造が一定であるとは限らないため、制御構造体からのデータリードも避ける必要があります。一般的には、制御構造体をブラックボックスのように取り扱うようにします。

制御構造体の内容はインスタンスに固有です。つまり、インタフェースのインスタンスが 2 つある場合、その制御構造体の型はまったく異なるものになります。1 つのインタフェースに存在する制御構造体の名前は、<interface>_ctrl_t. です。次の例は I2C インタフェースの制御構造体の例です。

```
typedef void  i2c_ctrl_t;
```

インタフェース制御構造体の型はいずれも *void* なので、インタフェース制御構造体を割り当てることはできません。代わりに、これらの型はインスタンス制御構造体のプレースホルダーになります。インスタンス制御構造体はインスタンスヘッダーファイルで定義され、名前は <instance>_instance_ctrl_t. です。次の例は、IIC(r_riic) および SCI(r_sci_i2c) の I2C インスタンス制御構造体を示しています。

```
/** riic Instance control structure to be used with I2C Interface. */
typedef struct st_riic_instance_ctrl
{
  i2c_cfg_t info; ///< Information describing I2C device
  uint32_t open; ///< Flag to determine if the device is open
```

```
  void * p_reg; ///< Base register for this channel

/** More members specific to riic Instance. */

} riic_instance_ctrl_t;

/** sci_i2c Instance control structure to be used with I2C Interface. */

typedef struct st_sci_i2c_instance_ctrl
{
  i2c_cfg_t info; ///< Information describing I2C device
  uint32_t open; ///< Flag to determine if the device is open
  void * p_reg; ///< Base register for this channel

  /** More members specific to sci_i2c Instance. */

} sci_i2c_instance_ctrl_t;
```

インタフェースを使用する際は、インスタンス制御構造体を割り当て、インタフェース制御構造体の代わりに使用する必要があります。上の例を使って説明すると、SCI-I2C インスタンスを使用しているのであれば、型 sci_i2c_instance_ctrl_t の構造体を割り当て、インタフェースで i2c_ctrl_t が参照される場合には常にそちらを使用するようにします。統合ソリューション開発環境は、ユーザーに代わって正確な制御構造体の割り当てを行います。

malloc() 関数と free() 関数を使用した動的メモリ割り当ては SSP モジュールでは使用されません。

設定構造体は、 open() 呼び出し中のモジュールの初期設定に使用されます。この構造体は、チャネル、割り込み優先順位、ビットレート、動作モードなどのメンバーで構成されます。この構造体は、モジュールへの入力にのみ使用されます。この構造体は一意にする必要がなく、必要に応じてユーザーが初期化後に破棄することができます。

```
/** I2C configuration block */
typedef struct st_i2c_cfg
{
  /** Generic configuration */
  ///< Identifier recognizable by implementation
  uint8_t channel;
  ///< Device's maximum clock rate from enum i2c_rate_t
  i2c_rate_t rate;
  uint16_t slave; ///< The address of the slave device
  ///< Indicates how slave fields should be interpreted
  i2c_addr_mode_t addr_mode;
  uint16_t sda_delay; ///< The SDA output delay
  uint8_t rxi_ipl; ///< Receive interrupt priority
  uint8_t txi_ipl; ///< Transmit interrupt priority
  uint8_t tei_ipl; ///< Transmit end interrupt priority
  uint8_t eri_ipl; ///< Error interrupt priority

  /** DTC/DMA support */
  ///< DTC instance for I2C transmit.Set to NULL if unused.
```

```
  transfer_instance_t const * p_transfer_tx;
  ///< DTC instance for I2C receive. Set to NULL if unused.
  transfer_instance_t const * p_transfer_rx;

  /** Parameters to control software behavior */
  ///< Pointer to callback function
  void (* p_callback)(i2c_callback_args_t * p_args);
  ///< Pointer to the user-provided context
  void const * p_context;

  /** Implementation-specific configuration */
  ///< Any configuration data needed by the hardware
  void const * p_extend;

} i2c_cfg_t;
```

上は I2C インタフェースの設定構造体の例です。最後の 3 つの構造体メンバー (p_callback、 p_context、および p_extend) は、ほぼすべてのモジュール設定に共通です。

p_callback および *p_context* メンバーについては、「SSP インタフェース : コールバック関数」のセクションで説明します。

p_extend メンバーは、特定のインスタンスの現在のインタフェースを拡張するために使用されます。インタフェースは最も一般的な機能をサポートするように設計されています。インタフェースのインスタンスがそれ自体を正しく構成するために、特別な情報が必要な場合があります。そうした情報が不要な場合もありますが、ユーザーが特定のアプリケーションに合わせてモジュールを調整するために必要になる場合もあります。その場合、ユーザーは *p_extend* メンバーを通じて、基礎となるインスタンスに追加の構成情報を渡すことができます。このメンバーを介して渡される情報は基礎となるインスタンスによって定義されるため、ユーザーはその構造体に忠実に従う必要があります。無効な情報が基礎となるドライバーに渡された場合は、インスタンスでそのデータを正しく処理できないため、正常な動作が保証されません。詳細については、「インタフェース拡張機能」の項を参照してください。

構成構造体に現在のインタフェースに適用されるメンバーだけが入っていることも重要です。同じスタック内の複数のレイヤーで同じ設定パラメータが定義されていると、オプションをどこで変更するかを知るのが難しくなります。たとえば、UART のボーレートはドライバーレイヤーでのみ定義されます。UART インタフェースを使用するすべてのレイヤーは、ドライバーレイヤーで指定されているボーレートに依存し、独自の構成構造体でボーレートを提供することはありません。

SSP インタフェースコールバック関数

コールバック関数を使用すれば、モジュールはイベント発生時に非同期的にユーザーアプリケーションに通知することができます。イベントの例には、 UART チャネル経由でのバイト受信があります。ユーザーアプリケーションコードで割り込みに対処するには、コールバックが必要です。 SSP モジュールが Synergy MCU ペリフェラルの割り込みを定義して処理します。ユーザーが SSP モジュールと同時に割り込みサービスルーチンを定義しようとしても、そのコードはビルドされません。そのため、 SSP モジュールを使用し、割り込み発生時に呼び出す関数を登録しておくことで、ユーザーアプリケーションが割り込みに対処できるようになります。

コールバック関数はユーザーアプリケーション内で定義する必要があります。この関数は常に *void* を返し、1 つのパラメータに対して 1 つの構造体を使用します。構造体の typedef はモジュールのインタフェースで指

定され、<interface>_callback_args_t. という名前になります。構造体の内容はインタフェースによって異なる可能性がありますが、 *event* と *p_context* の 2 つは共通です。

event メンバーは、アプリケーションが、コールバックが呼び出された理由を特定するために使用されます。前の UART の例では、バイトが受信された、すべてのバイトが送信された、またはフレーミングエラーが発生した場合にコールバックがトリガされた可能性があります。*event* メンバーはインタフェースで指定される列挙です。

p_context メンバーは、ユーザーが指定したデータをコールバック関数に渡すために使用されます。多くの場合、コールバック関数は、複数のチャネルまたはモジュールインスタンス間で共有されます。コールバックが発生すると、それを処理するコードがコールバック対象のモジュールインスタンスを特定するためにコンテキスト情報を必要とします。たとえば、コールバックで SSP API コールを発行する場合は、少なくともコールバックで制御構造体を使用する必要があります。これを簡単にするために、ユーザーは制御構造体へのポインタを *p_context* として指定できます。コールバックが発生するときは、コールバック構造体で渡される制御構造体を使用できます。

コールバック関数は割り込みサービスルーチン内から呼び出されます。そのため、ユーザーのシステムのリアルタイム性能に影響を与えないようにコールバック関数はできるだけ短くする必要があります。フラッシュインタフェースコールバックのスケルトン関数の例を以下に示します。

```
static void flash_callback (flash_callback_args_t * p_args)
{
    /** See what event caused this callback. */
    switch (p_args->event)
    {
        case FLASH_EVENT_ERASE_COMPLETE:
        /**Handle event. */
        break;

        case FLASH_EVENT_WRITE_COMPLETE:
        /**Handle event. */
        break;

        case FLASH_EVENT_BLANK:
        /**Handle event. */
        break;

        case FLASH_EVENT_NOT_BLANK:
        /**Handle event. */
        break;

        case FLASH_EVENT_ERR_DF_ACCESS:
        /** Handle error. */
        break;

        case FLASH_EVENT_ERR_CF_ACCESS:
        /** Handle error. */
        break;

        case FLASH_EVENT_ERR_CMD_LOCKED:
        /** Handle error. */
```

```
        break;
    }
}
```

モジュールがユーザーアプリケーション内で直接使用されていない（たとえば、スタックの最上位レイヤーではない）場合は、そのコールバック関数が上記モジュールによって処理されます。UART インタフェースモジュールが必要なコンソールインタフェースモジュールが存在する場合は、コンソールモジュールが UART のコールバック関数を制御および使用します。この場合、ユーザーはアプリケーションコード内で UART モジュール用のコールバック関数を作成する必要がありません。

SSP インタフェース API 構造体

すべてのインタフェースには、サポートされているすべてのインタフェース関数の関数ポインタを含む API 構造体が含まれています。コメントを削除した、デジタル / アナログ変換 (DAC) 用の構造体の例を以下に示します。

```
typedef struct st_dac_api
{
    ssp_err_t (*open)(dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg);
    ssp_err_t (*close)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*write)(dac_ctrl_t *p_ctrl, dac_size_t *p_value);
    ssp_err_t (*start)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*stop)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*versionGet)(ssp_version_t *p_version);

} dac_api_t;
```

API 構造体は、モジュールを同じインタフェースのインスタンスである他のモジュールに対して簡単にスワップインまたはスワップアウトできるようにするために使用されます。上記 DAC インタフェースを使用したアプリケーションの例を見てみましょう。

Synergy MCU は DAC ペリフェラルを内蔵しています。DAC インタフェースで DAC API 構造体が使用されていない場合は、アプリケーションから直接モジュールを呼び出すことができます。下の例では、アプリケーションは r_dac モジュールで提供されている R_DAC_Write 関数を呼び出しています。

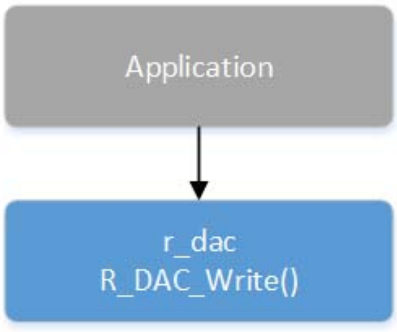


図 6: DAC 書き込みの例

ここで、MCU 上で使用可能な DAC チャネルだけでは足りず、r_dac_external. という名前の新しい外部 DAC モジュールを追加する場合を考えます。外部 DAC は通信に I2C を使用します。アプリケーションは 2 つのモジュールを区別する必要があるため、複雑さが増してアプリケーションに対する依存度が高まります。

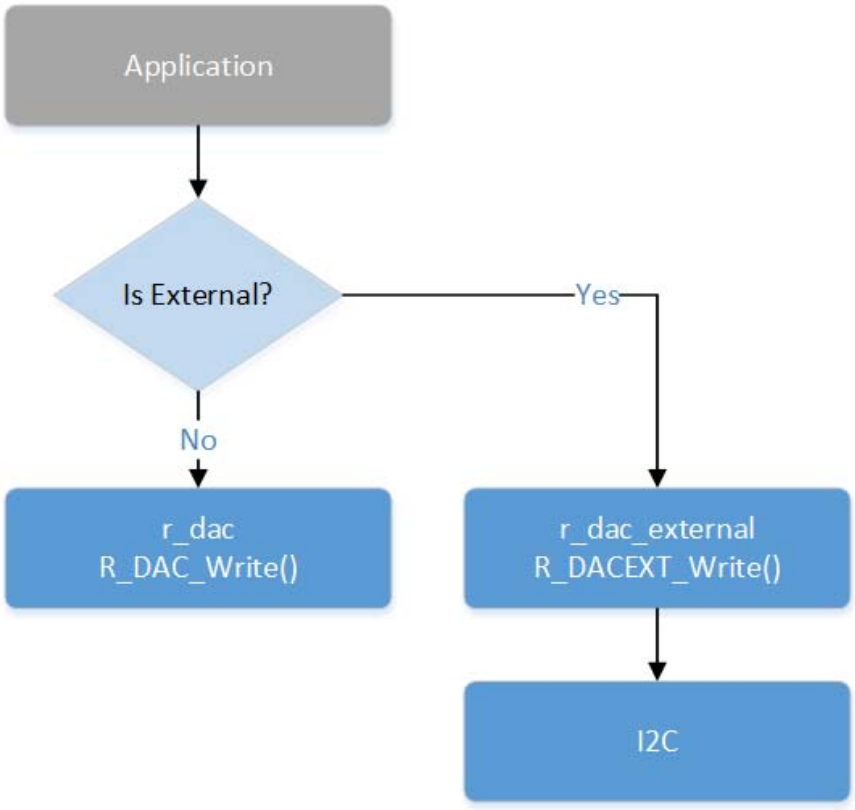


図 7: 2 つの書き込みモジュールを使用した DAC 書き込み

インタフェースと API 構造体を使用すると、抽象化された DAC を使用できます。つまり、外部ロジックが不要で、アプリケーションは特定のハードコードされたモジュールに依存しません。代わりに、アプリケーションは、任意の数のモジュールで実装可能な DAC インタフェース API に依存します。

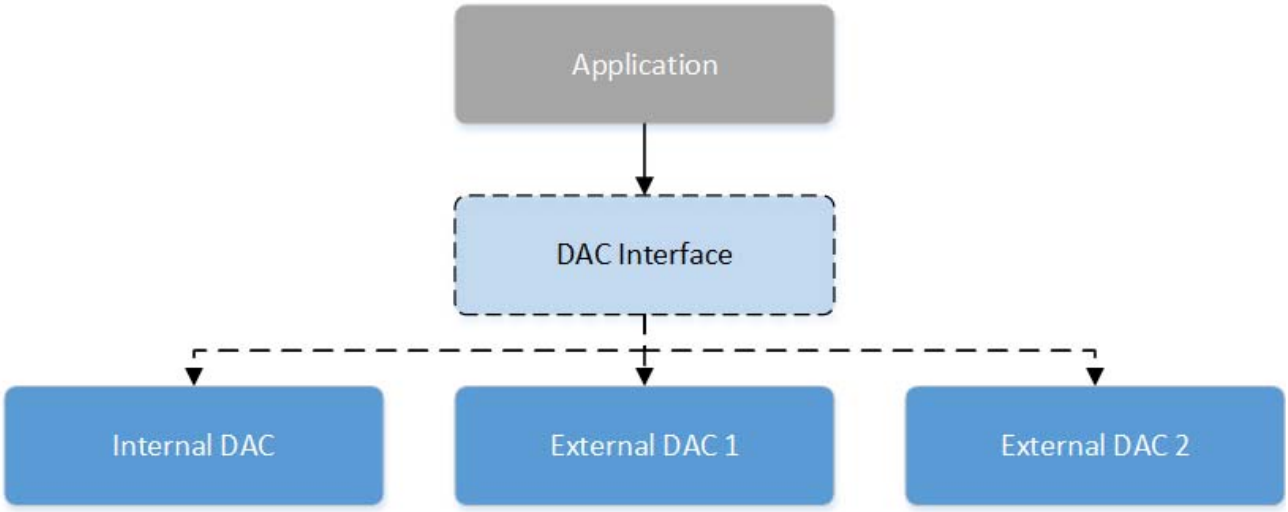


図 8: DAC インタフェース

API 構造体の内部関数は共通の名前に従っています。ほとんどのモジュールが open() 関数と close 関数のペアを使用します。 open() 関数は他の関数の前に呼び出す必要があります。唯一の例外は、ユーザーが指定した情報に依存しない versionGet() 関数です。

共通して使用される他の関数は、read()、write()、get()、および set() です。関数名は名詞の後に動詞が続くように設計されています。名前の例を以下に示します。

- read()、 write()、 writeRead()
- statusGet()
- calendarAlarmSet()、 calendarAlarmGet()
- accessWindowSet()、 accessWindowClear()

SSP インタフェースバージョン情報

すべてのインタフェースが versionGet() 関数を提供します。この関数は ssp_version_t という型の構造体を返します。この構造体は、インタフェース （API）用と現在使用されている基礎となるインスタンス用の 2 つのバージョンで構成されます。

```
/** Common version structure */
typedef union st_ssp_version
{
/** Version id */
uint32_t version_id;
/** Code version parameters */
struct
{
    uint8_t code_version_major; ///< Code major version
    uint8_t code_version_minor; ///< Code minor version
    uint8_t api_version_major; ///< API major version
    uint8_t api_version_minor; ///< API minor version
```

```
};
} ssp_version_t;
```

API バージョンは変更されないことが理想ですが、稀に変更される場合があります。API を変更するには、さかのぼってコードを変更する必要があります。コードバージョン、つまり、現在のインスタンスのバージョンは頻繁に更新される可能性があります。バグの修正、機能強化、および追加機能のすべては、コードバージョンの増加につながる可能性があります。ユーザーコードがインスタンスから提供される拡張機能を使用している場合、コードバージョンの変更では、ユーザーコードの変更のみが必要です。

SSP インスタンス

インタフェースは指定された機能を指示するのに対して、インスタンスは実際にそれらの機能を実装します。インスタンスはそれぞれ、特定のインタフェースに関連付けられます。インスタンスは、インタフェースの列挙、データ構造体、および API プロトタイプを使用します。これにより、インタフェースを使用するアプリケーションは必要に応じてインスタンスをスワップアウトすることができます。

Synergy MCU では、一部のペリフェラルがインタフェースとインスタンスの 1 対 1 マッピングを使用しますが、それ以外のペリフェラルは 1 対多マッピングを使用します。下の例では、IIC と SPI がそれぞれ 1 つのインタフェースにしかマッピングされていないのに対して、 SCI は 3 つのインタフェースを実装しています。

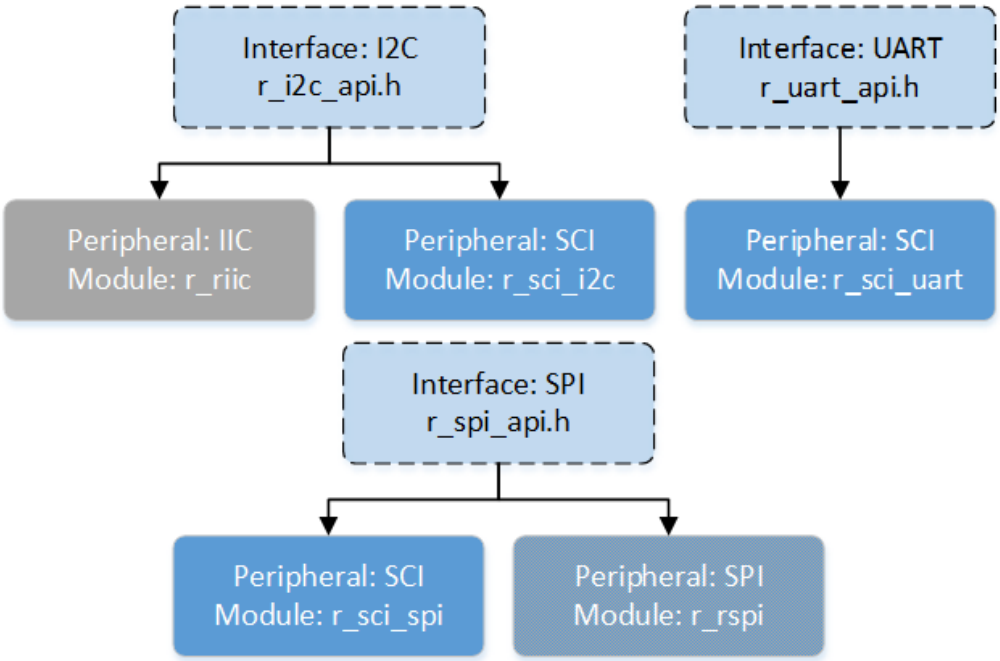


図 9:　　インスタンス

SSP インスタンスの API 構造体

各インスタンスには、インタフェースの API を実装するその関数を含む固定のグローバル構造体が含まれています。この構造体の名前は、 g_<interface>_on_<instance>　　として標準化されています。例として、g_spi_on_rspi、g_transfer_on_dtc、g_adc_on_adc などが挙げられます。この構造体は、インスタンスヘッダーファイル（それぞれ、 r_rspi.h、 r_dtc.h、および r_adc.h) 内の extern 経由で使用できます。

2.5.1.6 ビルド時設定

すべてのモジュールに、ビルド時設定ヘッダーファイルが割り当てられています。ほとんどの構成オプションは実行時に提供されます。稀にしか使用されない、または、モジュールのすべてのインスタンスに適用される一部のオプションはビルド時に移動される場合があります。ビルド時構成オプションを使用するメリットは、未使用の機能を削除することでコードサイズを小さくできる可能性があることです。性能を強化することもできます。

すべての SSP モジュールに、それぞれのパラメータチェックを有効化または無効化するビルド時オプションが 1 つ以上付属しています。 SSP モジュールは、可能な限り、関数引数の妥当性をチェックします。ユーザーは、テストの完了時にこの機能を無効化して、コードスペースを節約し、実行速度を上げることができます。

2.5.1.7 インタフェース拡張機能

インスタンスでは、インタフェースから提供されるよりも多くの情報が必要になることがあります。この状況は次の 2 つの場合に発生する可能性があります。

- あるインスタンスがインタフェースのほとんどのインスタンスに共通しない特別な機能を提供している。
- あるインタフェースをどうしても汎用的にしなければならない。インタフェースが汎用的になるほど、使用可能なインスタンスの数が増えます。この典型例がブロックメディアインタフェースです。

```
/** Interface Configuration */
typedef struct st_sf_block_media_cfg
{
    uint32_t block_size; ///< Block size in bytes
    void const * p_extend; ///< Instance dependent configuration
} sf_block_media_cfg_t;
```

ブロックメディアインタフェースの構成構造体は意図的に小さく作られています。これにより、ほぼ無限のインスタンスが可能になります。例として、SD カード、SPI フラッシュ、SDRAM、USB などが挙げられます。インスタンスごとに別々の構成情報が必要です。これは、 *p_extend* パラメータ経由で情報を提供することによって実現されます。この *p_extend* 内で提供される構成データはインスタンス間で同じではありませんが、その後の API 呼び出しは同じです。これは、 1 か所を変更するだけで済むことを意味します。

必ずしも、インタフェース拡張機能を使用する必要はありません。一部のインスタンスについては、機能がすべてインタフェースで提供されるため、拡張機能がありません。大抵の場合、 *p_extend* メンバーを NULL に設定できます。NULL が指定され、インスタンスが拡張機能を提供している場合、インスタンスはこれを、デフォルトのオプションが使用されるものと見なします。インタフェース拡張機能が提供されているかどうかと、その使用が必須か任意かについては、各インスタンスのドキュメントで確認できます。

2.5.1.8 SSP 事前定義レイヤー

SSP には、 2 つの事前定義レイヤー（ドライバーレイヤーとフレームワークレイヤー）が付属しています。これらのレイヤーは、モジュールが別々のフォルダー内に存在し、プレフィックスが異なるため、簡単に識別できます。ドライバーレイヤーモジュールは *ssp/src/driver* フォルダに配置され、フレームワークレイヤーモジュールは *ssp/src/framework* フォルダに配置されます。ドライバーレイヤー内のモジュールは *r_* プレフィックスで始まりますが、フレームワークレイヤーモジュールは *sf_* プレフィックスで始まります。

両レイヤー間の機能的な主な違いとして、ドライバーレイヤーモジュールには RTOS に対応したペリフェラルドライバーでなければならないという制限がありますが、RTOS オブジェクトの使用や RTOS API コールを行うことはありません。これは、RTOS のある（またはない）アプリケーションでドライバーレイヤーモジュールを使用できることを意味します。

フレームワークレイヤーモジュールはセマフォ、ミューテックス、イベントフラグなどの RTOS オブジェクトを自由に使用できます。フレームワークモジュールは、必要に応じて独自の RTOS オブジェクトを作成することもできます。ハードウェアにアクセスする必要があるフレームワークレイヤーモジュールは、通常、ドライバーレイヤーインタフェースを通じてそれを行います。複数のペリフェラルを同時に使用しなければならず、複数の個別インタフェースを使用することが実際的でない特殊な場合は、例外を認めることができます。

2.5.1.9 SSP ファイルの構造

SSP のファイル構造の概要を以下に示します。

```
ssp
+---inc
|   +---bsp
|   +---driver
|   |   +---api
|   |   \---instances
|   \---framework
|       +---api
|       \---instances
---src
    +---bsp
    +---driver
    |   \---r_module
    \---framework
        \---sf_module
ssp_cfg
+---bsp
+---driver
\---framework
```

ベースとなる *ssp* フォルダの直下では、フォルダが source フォルダと include フォルダに分岐しています。include フォルダーは、include パスの参照とセットアップを容易にするために source から分離されています。*ssp/inc* フォルダと *ssp/src* フォルダの下に、同じフォルダのセット（*bsp*、*driver*、*framework*）があります。

BSP と違って、SSP の 2 つの事前定義レイヤー（ドライバーとフレームワーク）が存在します。BSP と違って、SSP の 2 つの事前定義レイヤー（ドライバーとフレームワーク）が存在します。ドライバーレイヤーモジュールは *ssp/src/driver* フォルダーに位置していますが、フレームワークレイヤーモジュールは *ssp/src/framework* フォルダーに位置しています。include ツリーの下では、ドライバーレイヤーフォルダーとフレームワークレイヤーフォルダーにそれぞれ 2 つずつのフォルダー (*api* と *instances*) が含まれています。*api* フォルダーには、そのレイヤー用のインタフェースヘッダーファイルが含まれています。instances* フォルダーには、そのレイヤー用のインスタンスヘッダーファイルが含まれています。両方のレイヤーが内部的に平坦であるため、プロジェクトに必要な include パスの数は制限されます。

ssp_cfg フォルダーには、モジュールごとの構成ヘッダーファイルが保存されます。そのレイアウトは、*ssp* フォルダーと同じで、BSP、Driver、および Framework レイヤーが別々の平坦なディレクトリで構成されます。これらのヘッダーファイルで提供される情報については、「ビルド時設定」の項を参照してください。

2.5.1.10 SSP 接続レイヤー

SSP モジュールは再利用可能かつスタック可能に設計されています。モジュールは他のモジュールには依存しませんが、他のインタフェースには依存することを覚えておいてください。ニーズに最適なインスタンスを使用したインタフェースを自由に構築できます。

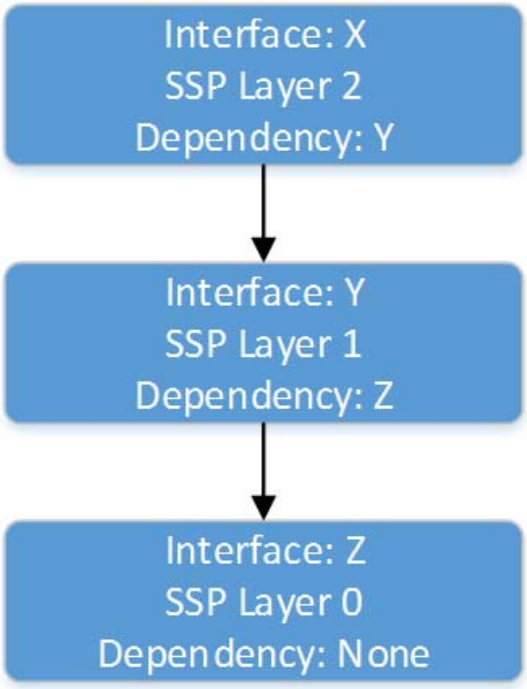


図 10: 接続レイヤー

上の図では、インタフェース Y がインタフェース X から依存されていると同時に、インタフェース Z に依存しています。インタフェース X はインタフェース Y に依存しているのみです。また、インタフェース X はインタフェース Z を認識していません。レイヤーを簡単にスワップアウトできるようにするには、このようにする必要があります。これを下の図に示します。

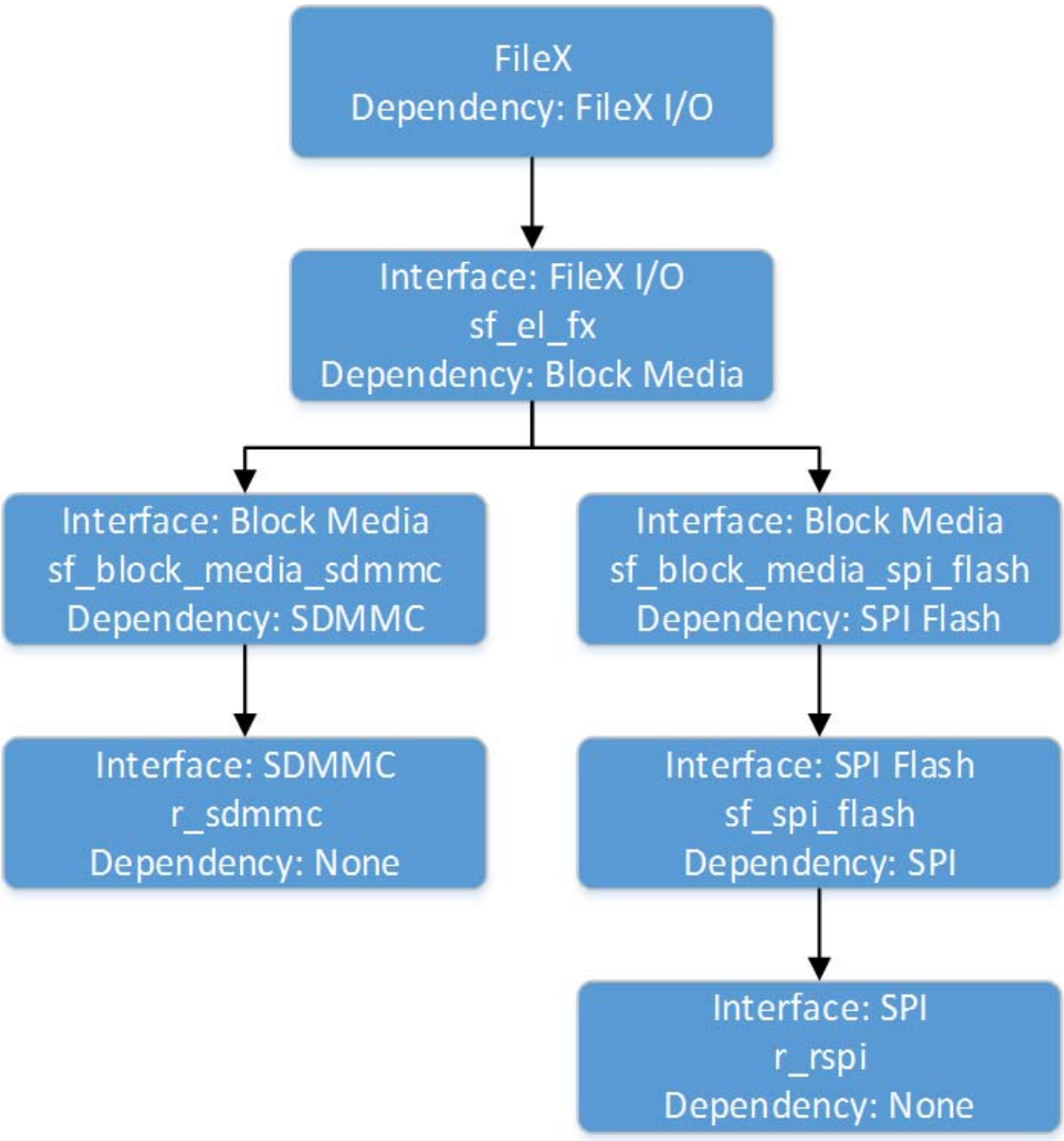


図 11: FileX インタフェースを使用した接続レイヤー

この例では、2 つのストレージメディア（SDMMC と SPI フラッシュ）上の Express Logic, Inc. FileX ファイルシステムを使用しています。SPI フラッシュインタフェースは SPI フラッシュプロトコルを処理しますが、実際の SPI バス通信には SPI インタフェースが必要です。 SDMMC インタフェースはプロトコルとバス通信を処理するため、何にも依存しません。

2.5.1.11 実際の SSP アーキテクチャ

SSP スタック内の各レイヤーには、その依存対象の API 関数を呼び出す役割があります。このことは、ユーザーが自身のやり取りするレイヤーでしか API 関数を呼び出さないと言い換えることもできます。上の FileX の例では、ユーザーはアプリケーションコード内で FileX 関数を呼び出すだけです。内部的に、この後 FileX は FileX I/O を呼び出し、次いで FileX I/O はブロックメディアインタフェースモジュールを呼び出します。ブロックメディアインタフェースは複数のドライバーを呼び出すことができます。上位レイヤーモジュールでは少なくとも、依存しているインタフェースの *open()* 関数を呼び出します。

モジュールを使用してアプリケーションを作成するには、以下のようにします。

1) 呼び出す open() 関数を決定します。依存関係はインタフェースに基づくので、モジュールは呼び出すインスタンスを何らかの方法で識別できなければなりません。
2) 設定パラメータを決定します。モジュールは、伝達する構成情報を認識する必要もあります。モジュールが特定の構成パラメータを設定しなければならない場合もあります。その場合は、モジュールがそれらの構成構造体メンバー自体を設定してから、残りの構造体を伝達します。残りの設定構造体メンバーはモジュール外部で指定する必要があります。
3) モジュールインスタンス固有であり、上位レイヤーモジュールで割り当て可能な制御構造体を提供します。

つまり、モジュールインスタンスとやり取りするには次のポインタが必要です。

- インスタンスの API 構造体へのポインタ
- モジュールインスタンスの構成構造体へのポインタ
- モジュールインスタンスの制御構造体へのポインタ

これは、任意のモジュールを使用するのに十分な情報です。 API 構造体はインスタンスに固有である唯一の構造体であり、*モジュールインスタンス*固有ではありません。これは、 API 構造体が同じインスタンスの複数の使用で変化しないためです。 SPI が SCI チャネル 0 および 2 で使用されている場合は、両方のモジュールインスタンスで同じ API 構造体が使用されますが、構成および制御構造体は異なります。

モジュールインスタンスをより使いやすくするため、これらのすべての部分は各インタフェースで見つかったインスタンス構造体内にカプセル化されます。これらの構造体は、<interface>_instance_t. という標準化された名前を持っています。 WDT インタフェースからの例を下に示します。

```
/** This structure encompasses everything that is needed to use an instance of t
his interface. */
typedef struct st_wdt_instance
{
    wdt_ctrl_t * p_ctrl; ///< Pointer to the control structure forthis instance
    wdt_cfg_t const * p_cfg; ///< Pointer to the configuration structure for thi
s instance
    wdt_api_t const * p_api; ///< Pointer to the API structure for this instance

} wdt_instance_t;
```

インタフェースに対する依存関係を持つ上部レイヤーモジュールは、インスタンス構造体を使用して、そのインタフェースのインスタンスとやり取りするのに必要なすべての情報を保持できます。上の WDT の例に続いて、スレッド監視フレームワークインタフェース構成構造体を下に示します。このスレッド監視インタフェースは WDT インタフェースに依存しています。

```
/** Configuration for Thread Monitor framework */
typedef struct st_sf_thread_monitor_cfg
{
    wdt_instance_t * p_lower_lvl_wdt; ///< Pointer to lower level watchdog insta
nce
    bool profiling_mode_enabled; ///< Enables or disables profiling mode
    UINT priority; ///< Priority of thread monitor thread
    void const * p_extend; ///< Instance dependent configuration

} sf_thread_monitor_cfg_t;
```

スレッド監視モジュールは、WDT インタフェースとやり取りするのに必要なすべての情報を *p_lower_lvl_wdt* 構造体メンバー内に持っています。

場合により、モジュール依存関係はインタフェースではなくインスタンスで定義されます。たとえば、ブロックメディアインタフェースは、SDMCC、SPI フラッシュ、または他の多くのインスタンスで実装できます（も参照）。実装方法は広範囲にわたるため、特定のインタフェースのインスタンス構造体をブロックメディアインタフェースの構成構造体で直接使用することはできません。ブロックメディアインタフェースの構成構造体を再び下に示します。

```
/** Interface Configuration */
typedef struct st_sf_block_media_cfg
{
    uint32_t block_size; ///< Block size in bytes
    void const * p_extend; ///< Instance dependent configuration

} sf_block_media_cfg_t;
```

インスタンス構造体ポインタが指定されていることに注意してください。この理由は、前述のようにブロックメディアインタフェースが汎用的すぎて特定のインタフェースに対する依存関係を強制できないためです。

モジュールがブロックメディアなどの汎用インタフェースのインスタンスで、他のモジュールに依存している場合は、インタフェースの *p_extend* 構成メンバー経由で参照される拡張機能構造体内に下位レイヤーポインタを配置します。これは、インタフェースの拡張を強制せずにモジュールスタッキングを可能にして、多数のオプション構成メンバーを使用できるようにするために必要です。

```
typedef struct st_block_media_on_sdmmc_cfg
{
    sdmmc_instance_t const * const p_lower_lvl_sdmmc; ///< Pointer to SDMMC inst
ance structure

} sf_block_media_on_sdmmc_cfg_t;
```

2.5.1.12 SSP モジュールの使用

ここでは、 SSP モジュールの使用方法に関する一般的な情報を提供します。

インタフェースの選択

最初に必要な機能用のインタフェースを選択します。たとえば、UART 通信には UART インタフェースを使用します。

インタフェースの適切なインスタンスの検索

インタフェースを選択したら、それに対応するインスタンスを選択します。インタフェースの既知のインスタンスのリストは、インタフェースのドキュメンテーションコメント内に記載されています。選択したインスタンスのヘッダーファイルを、そのインスタンスを使用するアプリケーションのソースファイルにインクルードします。

制御構造体と構成構造体の割り当て

e^2 studio 統合ソリューション開発環境には、インタフェースおよびインスタンス構成構造体のパラメータを設定するためのグラフィカルユーザーインタフェースが備わっています。また、GUI で構成されたそれらの構造体は、アプリケーションコードにインクルード可能なアプリケーション固有のヘッダーファイルに自動的に含められます。

ISDE で構成が処理される方法については、『ISDE ユーザーズガイド』の e^2 studio ISDE ユーザーガイドを参照してください（プロジェクトの設定）。

構成および制御構造体の型は、それぞれ <interface>_ctrl_t および <interface>_cfg_t, という標準名に従います。ISDE では、ISDE が作成するアプリケーション固有のヘッダーファイル内に、これら両方の構造体用のストレージが割り当てられます。ISDE の [Properties] ビューを使用し、必要に応じて構成構造体のメンバーの値を設定します。多くのメンバーが、使用可能なオプションで列挙を参照可能な型指定列挙です。

インタフェースにコールバック関数が付属している場合は、最初にソースコードでその関数を宣言して定義する必要があります。戻り値は常に型 *void* で、関数へのパラメータは <interface>_callback_args_t. という名前の型指定構造体です。関数が定義されたら、その名前を構成構造体の *p_callback* メンバーに割り当てます。コールバックでコンテキスト情報が必要な場合は、*p_context* メンバーへのポインタを指定します。統合ソリューション開発環境で選択したモジュールの [Properties] ウィンドウを通じて、コールバック関数名を割り当てることができます。

インタフェース拡張機能が提供されているかどうかを確認するには、インスタンスドキュメントを参照してください。提供されている場合は、*<interface>_*on_*<instance>_cfg_t*. という名前のインスタンスヘッダーファイルに含まれています。インタフェースの構成構造体と同様に、複数のメンバーで構成される場合があります。これらのメンバーは、統合ソリューション開発環境を使用して設定することができます。

インタフェースのインスタンス構造体を使用したやりとり

インスタンス構造体を作成した後は、必要に応じてインスタンスとやり取りできます。下に示すのは、SCI で実装されている UART インタフェースのインスタンス構造体を構築するコードです。e^2 studio を使用する場合は、次のコードが自動的に生成されることに注意してください。

```
/** Include the header file of the Instance. */
/** This will in turn include the r_uart_api.h Interface */
#include “r_sci_uart.h”

/** Allocate Instance specific control structure. This will be used in place of
uart_ctrl_t. */
sci_uart_instance_ctrl_t my_uart_ctrl;

/** Setup extended UART configuration on SCI. */
uart_on_sci_cfg_t my_uart_extended_cfg =
{
```

```
/** Set extended configuration members... */

};

/** Configure standard UART Interface. */
uart_cfg_t my_uart_cfg =

{
    .data_bits = UART_DATA_BITS_8,
    /** Continue configuring other members... */
    .p_extend = &my_uart_extended_cfg
};

/** Setup instance structure */

uart_instance_t my_uart = {
    .p_ctrl = &my_uart_ctrl,
    /** Extended configuration is brought through in p_extend */
    .p_cfg = &my_uart_cfg,
    .p_api = &g_uart_on_sci ///< Defined in r_sci_uart.h
};
```

これでインスタンス構造体は準備完了です。UART インタフェースとやり取りできます。e^2 studio では、インスタンス構造体の名前は、統合ソリューション開発環境の [Properties] ウィンドウでモジュールインスタンスを構成するときに指定した*名前*です。

```
ssp_err_t err;
/** Initialize UART */
err = my_uart.p_api->open(my_uart.p_ctrl, my_uart.p_cfg);

/** Check return for errors. */
if (SSP_SUCCESS != err)
{

    /** Handle error. */

}

/** Use other Interface functions. */
err = my_uart.p_api->write(my_uart.p_ctrl, ...);
err = my_uart.p_api->read(my_uart.p_ctrl, ...);
```

2.5.1.13 コーディングスタイル

C99 の使用

SSP では、 ISO/IEC 9899:1999 (C99) C プログラミング言語規格が使用されます。使用する C99 で導入された特定の機能には、標準の整数型 (stdint.h)、ブール型 (stdbool.h)、指定初期化子、および宣言とコードを組み合わせる機能が含まれます。

API パラメータ内での const の使用

const 修飾子は必要に応じて API パラメータと一緒に使用されます。実例を以下に示します。

```
ssp_err_t (* open)(flash_ctrl_t * const p_ctrl,
                   flash_cfg_t const * const p_cfg);
```

絶対確実というわけではありませんが、これにより SSP コード内で追加のチェックが実行されるため、変更すべきではない引数に対する変更を防ぐうえで役立ちます。

Weak シンボル

Weak シンボルは SSP 内で使用される場合と SSP と一緒に使用される場合があります。このシンボルを使用すると、ユーザーがオプション関数を定義していない場合にもプロジェクトをビルドできます。

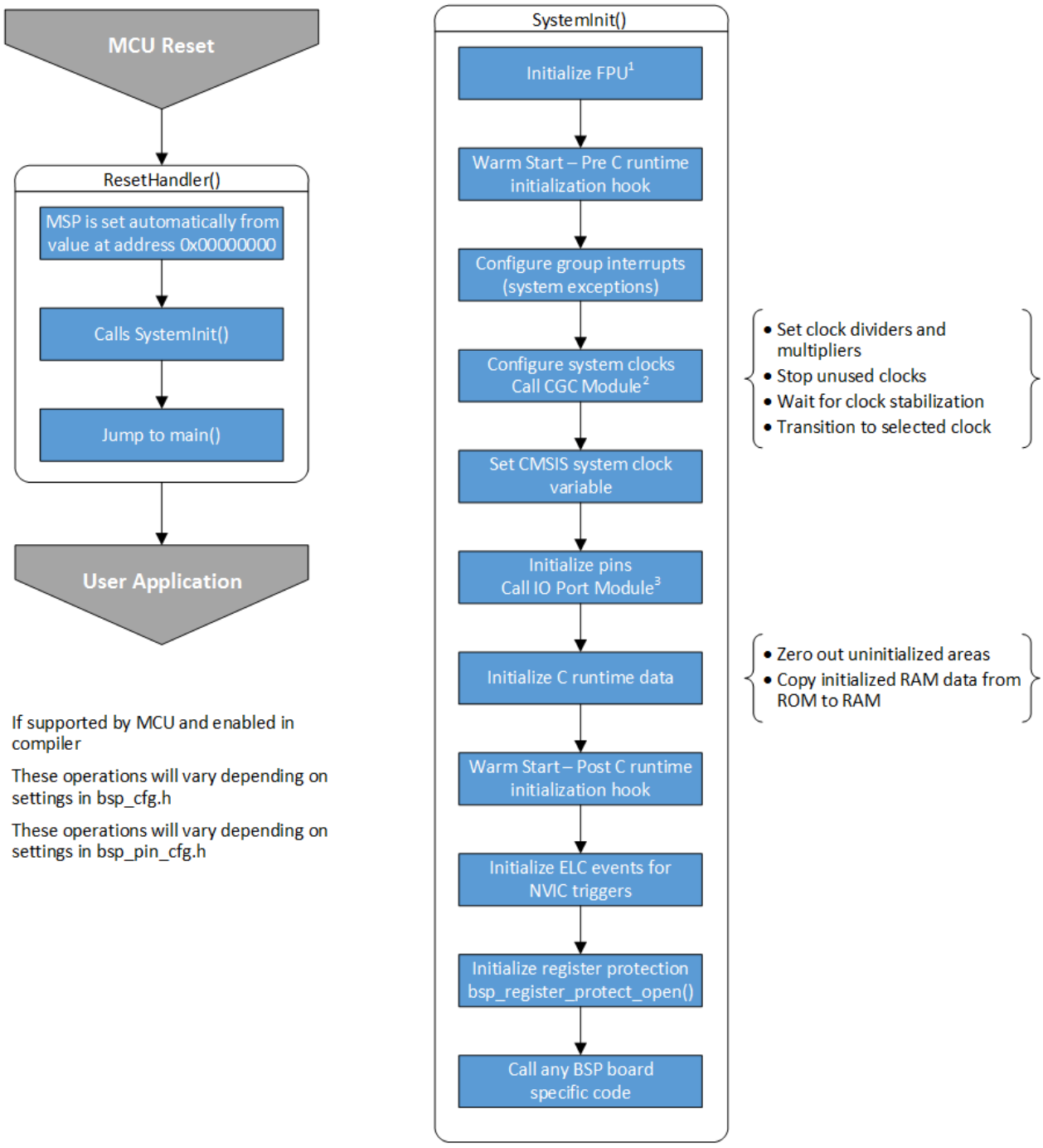
2.6 BSP アーキテクチャ

2.6.1 BSP アーキテクチャ

このセクションでは、BSP（ボードサポートパッケージ）について説明します。 API リファレンスについては、 Board Support Package を参照してください。 BSP はボード固有であり、結果的に MCU 固有です。

2.6.1.1 BSP の機能

BSP には、リセット以降の MCU からユーザーアプリケーション （main() 関数など）に到達できるようにする責任があります。ユーザーアプリケーションに到達する前に、 BSP は、スタック、ヒープ、クロック、割り込み、 C ランタイム環境をセットアップします。また、 BSP は、ポート I/O ピンを設定およびセットアップし、ボード固有の初期化も行います。



1. If supported by MCU and enabled in compiler
2. These operations will vary depending on settings in bsp_cfg.h
3. These operations will vary depending on settings in bsp_pin_cfg.h

図 12: BSP のフロー

2.6.1.2 BSP 関連の用語

Term	意味
Filename_xxxx.c	'xxxx' は MCU の種類を表します。たとえば、S7G2 MCU を参照する場合は Filename_S7G2.c になります。
BSP	Board Support Package の略語。BSP には、一般に特定のボードに関連するソースファイルがあります。
Callback Function	この用語はイベント発生時に呼び出される関数を意味します。ユーザーは、NMI システム例外の発生を知りたい可能性があります。ユーザーに通知するため、グループ割り込み（すべてが NMI に結び付けられている例外のグループ）に対してコールバック関数を設定できます。NMI が発生すると、BSP は指定されたコールバック関数にジャンプして、ユーザーがエラーを処理できます。割り込みコールバック関数は長くならないように慎重に扱う必要があります。これは、この関数が呼び出されたときに、MCU はまだ割り込みの内部にあり、保留中の割り込みが後回しにされるためです。

2.6.1.3 BSP のディレクトリ構造

BSP は、 MCU 情報、ボード固有情報、 CMSIS 情報が格納されるフォルダーに編成されます。

Synergy は CMSIS に準拠しており、 CMSIS-Core に基づいています。そのためには、下記の CMSIS の要件と命名規則に従う必要があります。

- プロセッサペリフェラルに対する標準化された定義
  - NVIC （ネスト型ベクトル割り込みコントローラ）
  - Systick （システムティックタイマ）
  - MPU （メモリ保護ユニット）
- プロセッサ機能にアクセスするための標準化されたアクセス関数
  - NVIC_SetPriority()
  - NVIC_EnableIRQ
- システム例外ハンドラーのための標準化された関数名
  - Reset_Handler()
  - SysTick_Handler()
- システム初期化のための標準化された関数。
  - SystemInit() – system_S7G2.c で S7G2 MCU 用に定義
- クロック速度情報のための標準化されたソフトウェア変数

– SystemCoreClock

2.6.1.4 BSP の設定

BSP は非常にデータ駆動型であり、ほとんどの機能は、設定ファイルの内容に基づいて設定されます。設定ファイルは、ユーザーによって指定された設定を表しており、[Generate Project Content] ボタンをクリックしたときに、ISDE によって生成されます。

2.6.1.5 BSP 設定

次の表では、BSP のそれぞれの変更可能な設定について説明します。これらの設定の多くは MCU 固有であり、サポートされるそれぞれの MCU で使用可能な設定に違いがあります。

表 : BSP 設定オプション

BSP Property	説明
Part number	MCU 部品番号
ram_size_bytes	この MCU パッケージで使用可能な RAM
rom_size_bytes	この MCU パッケージで使用可能な ROM
data_flash_size_bytes	この MCU パッケージで使用可能なデータフラッシュ
package_style	パッケージのスタイル（たとえば BGA）
package_pins	この MCU パッケージのピン数
series	MCU パーツシリーズ
Main stack size (bytes)	メインスタックのサイズ。0 よりも大きい必要があります。
Process stack size (bytes)	プロセススタックのサイズ。このスタックの使用はオプションです。0 の場合、PSP の使用が無効になります。
Heap size (bytes)	ヒープのサイズ（バイト単位）。0 の場合、ヒープが無効になります。

BSP Property	説明
OFS0 register settings: IWDT Start Mode, IWDT Timeout Period, IWDT Dedicated Clock Frequency Divisor, IWDT Window End Position, IWDT Window Start Position, IWDT Reset Interrupt Request Select, IWDT Stop Control, WDT Start Mode Select, WDT Timeout Period, WDT Clock Frequency Division Ratio, WDT Window End Position, WDT Window Start Position, WDT Reset Interrupt Request, WDT Stop Control and the program flash area of the flash memory.	詳細は MCU ユーザーマニュアルを参照してください。
OFS1 register settings: Voltage Detection 0 Circuit Start, Voltage Detection 0 Level, HOCO Oscillation Enable. S3 MPU has MPU configuration settings.	詳細は MCU ユーザーマニュアルを参照してください。
MPU - Enable or disable PC Region 0	アクセスウィンドウ保護のための開始ブロックアドレス
MPU - PC0 Start, MPU - PC0 End, MPU - Enable or disable PC Region 1, MPU - PC1 Start, MPU - PC1 End, MPU - Enable or disable Memory Region 0, MPU - Memory Region 0 Start, MPU - Memory Region 0 End, MPU - Enable or disable Memory Region 1, MPU - Memory Region 1 Start, MPU - Memory Region 1 End, MPU - Enable or disable Memory Region 2, MPU - Memory Region 2 Start, MPU - Memory Region 2 End, MPU - Enable or disable Memory Region 3, MPU - Memory Region 3 Start, MPU - Memory Region 3 End	セキュア MPU ROM レジスタ設定。詳細は MCU ユーザーマニュアルを参照してください。
ID code 1, ID code 2, ID code 3, ID code 4	ブートモードとデバッガーアクセス保護のための ID コードを設定します。
MCU Vcc (mV)	一部のモジュール（LVD など）では、MCU に供給される電圧を知る必要があります。この情報はここから取得されます。
Parameter checking	パラメータチェックのグローバル設定を有効にするか無効にするかを定義します。ローカルモジュールはこの値をデフォルトで取得しますが、ローカルにオーバーライドすることができます。
Assert Failures	アサーション失敗が発生したときに行われる処理を定義します。
Error Log	エラーを ssp_error_log に記録するかどうかを定義します。

2.6.1.6 BSP の設定ファイル

設定ファイルは、BSP により、ROM レジスタ、クロック、割り込み、ELC イベント、および初期ピンを設定するために使用されます。これらの設定ファイルは、ssp_cfg\bsp にあります。

bsp_cfg.h

この構成ファイルは、BSP システム設定の値を含みます。これは、ISDE の BSP プロパティタブで変更可能な設定です。これには、ROM レジスタ設定、スタックおよびヒープサイズ、パラメータチェック、エラーロギングの制御が含まれています。

一部のレジスタは ROM 内にあるため、コンパイル時に設定する必要があります。これには、いくつかのオプション設定メモリ (OFS) レジスタと、特定のメモリ保護レジスタが含まれます。

オプション設定メモリは、リセット後の MCU の状態を決定します。たとえば、IWDT の設定と有効化、電圧検出の有効化、HOCO 発振の有効化を行うことができます。これらのレジスタが設定されている場合、操作は、MCU のリセットベクトルがフェッチされ、例外が開始される前に完了します。

一部の Synergy MCU にはメモリ保護ユニット (MPU) が搭載されています。MPU は、プログラム可能なデバイスであり、各種メモリ領域について、メモリアクセス権限（たとえば、特権アクセス専用またはフルアクセス）と、メモリ属性（バッファリング可能、キャッシング可能など）を定義するために使用できます。MPU は、最大で 8 個のプログラム可能メモリ領域をサポートでき、それぞれが独自のプログラム可能開始アドレス、サイズ、設定を持ちます。

ISDE は、指定された MPU 設定の値を設定することで、これらのメモリ領域を設定します。これらのレジスタは慎重に設定する必要があります。設定が正しくないと、必要なメモリ領域にアクセスできなくなったり、MCU 全体にアクセスできなくなる可能性があります。

2.6.1.7 BSP のピン設定

アプリケーションで使用するピンは、ISDE のピンコンフィギュレーターで設定できます。ピンの設定を参照してください。

2.6.1.8 BSP のクロック設定

すべてのシステムクロックは、BSP の初期化時に、bsp_clock_cfg.h の設定に基づいて設定されます。これらの設定は、ISDE の [Clocks] タブ設定で指定したクロック設定情報が基になります。

- 比較的低速（たとえば 32 kHz) のクロック上で開始される可能性があることから、クロック構成は、起動処理を高速化するため、C ランタイム環境を初期化する前に実行されます。
- BSP は、選択したクロックが安定するために必要な遅延を実装します。

bsp_clock_cfg.h

この設定ファイルは、システムクロック設定の値を表します。これは、ISDE の [Clocks] タブで変更可能な設定です。

クロックの設定を参照してください。

2.6.1.9 システム割り込み

Synergy MCU は、Cortex-M Arm アーキテクチャに基づいているため、ネスト型ベクトル割り込みコントローラ (NVIC) が例外と割り込み設定、優先順位付け、割り込みマスクを処理します。Arm アーキテクチャでは、NVIC が例外を処理します。一部の例外はシステム例外と呼ばれます。システム例外は、ベクトルテーブルの先頭に静的に配置され、ベクトル番号 1 から 15 を占めます。ベクトル 0 は、メインスタックポインタ (MSP) 用に予約されています。残りの 15 個のシステム例外を以下に示します。

- リセット
- NMI
- Cortex-M4 ハード障害ハンドラー
- Cortex-M4 MPU 障害ハンドラー
- Cortex-M4 バス障害ハンドラー
- Cortex-M4 使用率障害ハンドラー
- 予約
- 予約
- 予約
- 予約
- Cortex-M4 SVCall ハンドラー
- Cortex-M4 デバッグモニターハンドラー
- 予約
- Cortex-M4 PendSV ハンドラー
- Cortex-M4 SysTick ハンドラー

NMI とハード障害例外は、リセット以降に有効になっており、優先度は固定です。その他の例外の優先度は設定可能であり、一部は無効にできます。

2.6.1.10 グループ割り込み

グループ割り込みは、マスク不可能な割り込み (NMI) をトリガできる 12 のソースを表すために使用される用語です。NMI が発生した場合、NMI ハンドラーは NMISR（ステータスレジスタ）を調べ、割り込みのソースを判定します。NMI 割り込みはすべての割り込みより優先され、CPU 割り込みとしてのみ使用でき、データトランスファコントローラ (DTC) またはダイレクトメモリアクセスコントローラ (DMAC) のアクティブ化を行うことはできません。

グループ割り込みのソースとしては、以下のものが考えられます。

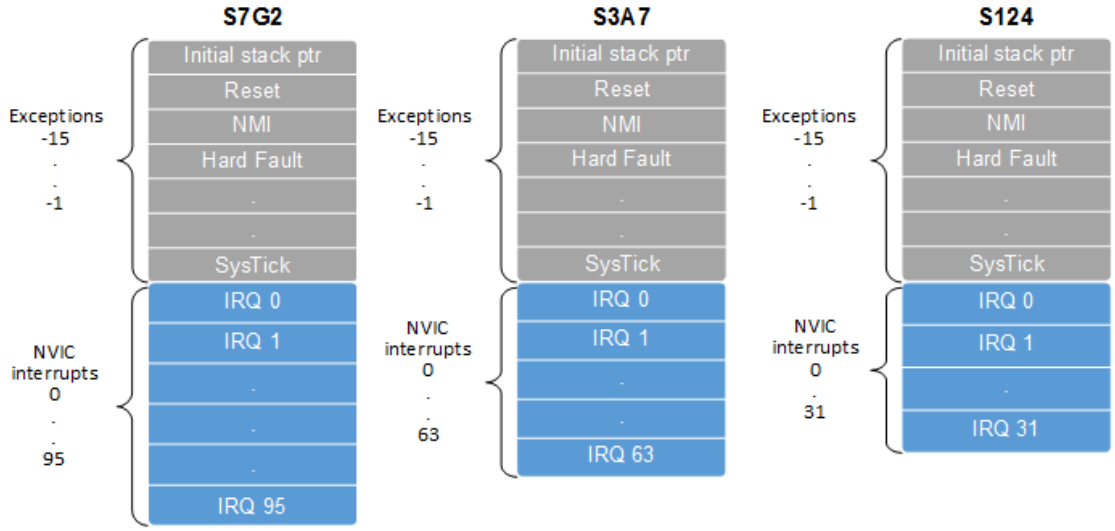
- IWDT アンダーフロー / 更新エラー
- WDT アンダーフロー / 更新エラー
- 電圧監視 1 割り込み
- 電圧監視 2 割り込み
- VBATT 監視割り込み
- 発振停止検出
- NMI ピン
- RAM パリティエラー
- RAM ECC エラー
- MPU バススレーブエラー

- MPU バスマスターエラー
- MPU スタックエラー

ユーザーは、BSP API 関数 R_BSP_GroupIrqWrite を使用してコールバックを登録することで、1 つ以上のグループ割り込みの通知を有効にすることができます。NMI 割り込みが発生すると、NMI ハンドラーは、割り込み原因に対してコールバックが登録されているかどうかを確認し、登録されている場合はそのコールバック関数を呼び出します。

前述のように、ベクトルテーブルの最初の 16 個のスロットはすでにシステム例外に割り当てられています。スロット 16 以降はユーザーが設定可能な割り込みです。外部割り込みか、ペリフェラルで生成される割り込みを設定できます。

NVIC 割り込みテーブルのサイズは、以下のように Synergy MCU の種類によって異なります。



NVIC 割り込みベクトルテーブルの空きスロットの数は少なく見えるかもしれませんが、BSP では、割り込みを生成できるイベントが最大で 512 個定義されます。BSP は、イベントマッピングを使用することで、ユーザーが有効にしたイベントを NVIC 割り込みにマッピングします。S7G2 MCU では、これらのイベントのうち 96 個だけを一度にアクティブにできますが、ユーザーは、アクティブなイベントを生成するイベントを柔軟に選択できます。

下図は S7G2 の割り込みベクトルテーブルを示しています。

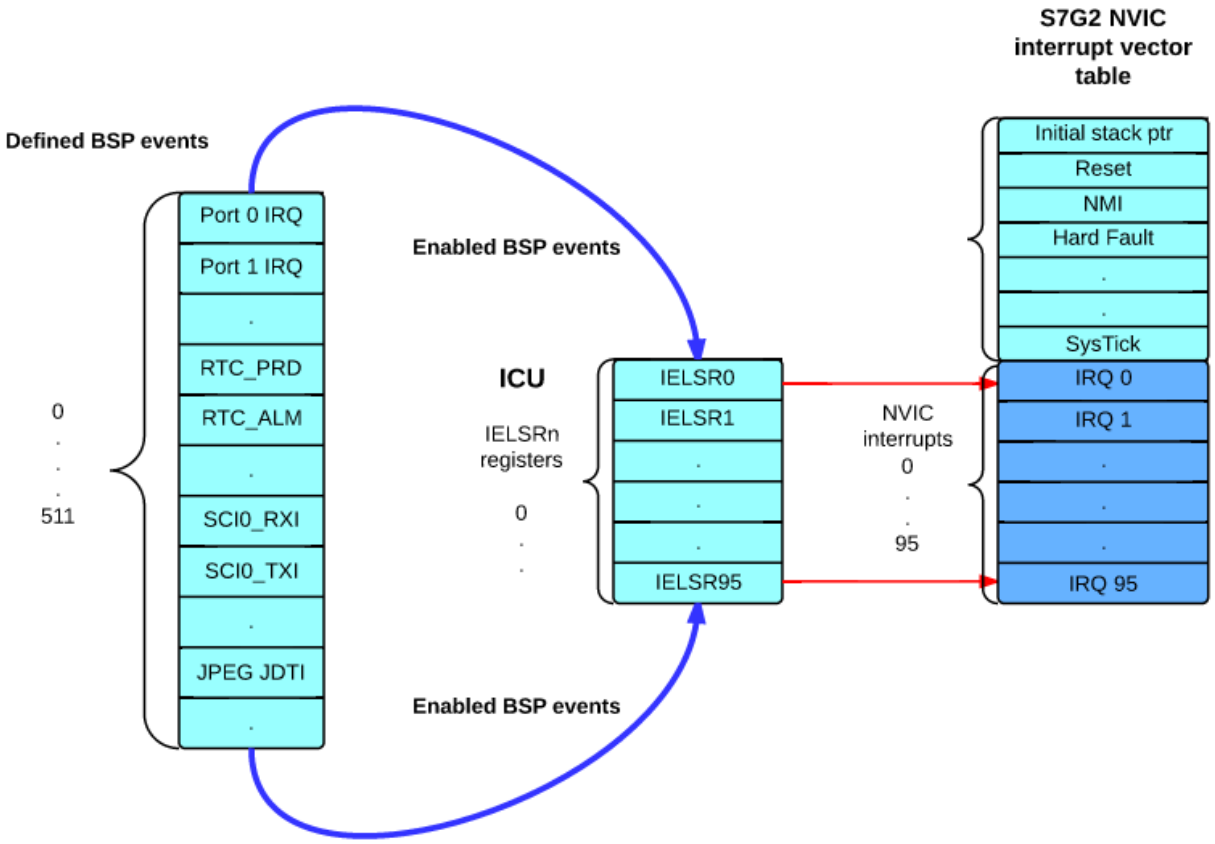


図 13: NVIC 割り込みベクトルテーブル

割り込み要因として関心があるイベントのみをユーザーが選択できるようにすることで、高速でイベント固有の割り込みサービスルーチンを提供できます。

たとえば、他のマイクロコントローラにおいて、標準の NVIC 割り込みベクトルテーブルには、SCI0（シリアル通信インタフェース）用の単一のベクトルエントリが格納される可能性があります。そのための割り込みサービスルーチンは、割り込みの「本当の」ソースのステータスレジスタを確認する必要があります。Synergy の実装では、関心がある SCI0 イベントごとに 1 つのベクトルエントリがあります。標準の NVIC テーブルと Synergy S7G2 NVIC テーブルの違いを下に示します。

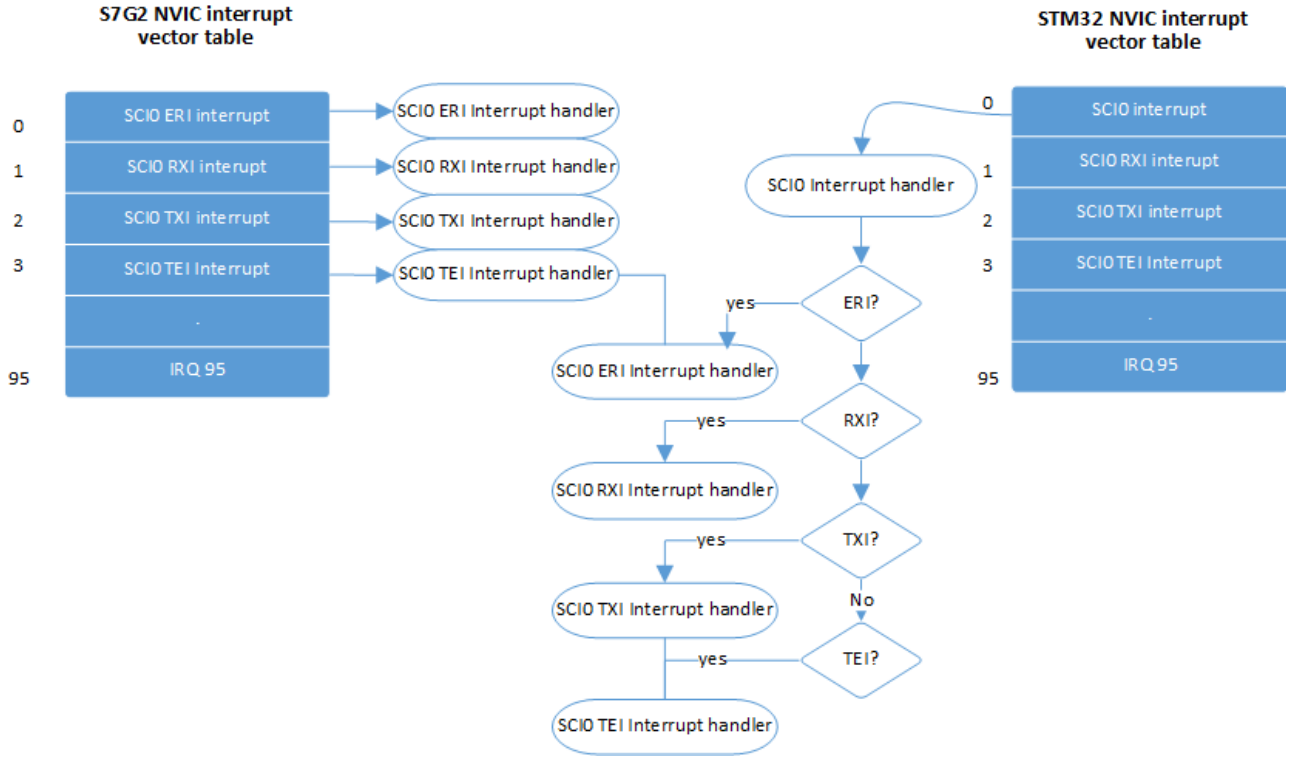


図 14: NVIC 割り込みベクトルテーブルの例

割り込みの構成は統合ソリューション開発環境によって処理されます。モジュールにより使用される割り込みを選択すると、ベクトルテーブルのエントリを割り当てるのに必要なコードが生成され、適切な ICU ELC イベントにリンクされます。

割り込みが発生した場合に、ごく初期に行う必要がある処理の 1 つは、 BSP によって割り当てられた NVIC 割り込みスロットに対応する割り込み番号と共に R_BSP_IrqStatusClear をコールすることです。

R_BSP_IrqStatusClear は、特定の割り込みの割り込みステータスフラグ (IR) をクリアします。特定の割り込みの割り込みステータスフラグ (IR) をクリアします。割り込みがトリガされると、IR ビットがセットされます。

ここに示す例で、優先度が割り当てられたエントリ （たとえば BSP_IRQ_CFG_ICU_IRQ0) では、対応する weak ハンドラーのアドレスが、次に使用可能なベクトルスロットに格納されます。可能性のあるすべて??割り込みソースがこのようにして反復処理されます。定義されている割り込みがベクトルテーブルに設定されます。

bsp_irq_cfg.h

現在では使用しておらず、将来的に削除される予定のレガシーファイルです。割り込みの構成は現在、全面的に統合ソリューション開発環境によって処理されています。

ベクトルテーブルエントリ

HardFault_Handler, などのシステム例外は、弱い参照として定義されます。このため、ユーザー独自のハンドラを定義し、特定の例外についてデフォルトハンドラをオーバーライドすることができます。

ベクトルテーブル内の他のエントリはすべて、ISDE で生成されます。その際、ベクトルとそれに対応するベクトル情報構造体を定義するマクロによって、 ROM テーブルリンカセクションにエントリが生成されます（ベクトルには .vector.*、ベクトル情報には .vector_info.*）。

CMSIS の system_xxxx.c では、ウォームスタートコールバック関数 R_BSP_WarmStart() の weak 定義（と関数本体）もあることに注意してください。この関数は、 weak 宣言と同じファイルに定義されているため、「デフォルトの」実装として呼び出されます。本体をユーザーアプリケーションにコピーし、必要に応じて変更することで、関数をオーバーライドできます。リンカーはこれを「strong」参照として認識して使用します。

ウォームスタートコールバック

BSP がボードをリセット状態から起動している最中に、ユーザーがコールバックを要求できる場所が 2 つあります。これらは、「Pre C」および 「Post C」ウォームスタートコールバックとして定義されます。

前述のように、この関数は R_BSP_WarmStart() としてすでに weak 定義されているため、アプリケーションコードで関数を再定義する （または、 CMSIS の system_xxxx.c から既存の本文をコピーする）だけで、コールバックを作成できます。R_BSP_Warmstart() は、実行中のウォームスタートコールバックの種類を示すイベントパラメータを受け取ります。

```
/** Different warm start entry locations in the BSP. */

typedef enum e_bsp_warm_start_event

{

    BSP_WARM_START_PRE_C = 0, // Called almost immediately after reset.

    /* No C runtime environment, clocks, or IRQs. */

    BSP_WARM_START_POST_C // Called after clocks and C runtime environment have
been set up.

} bsp_warm_start_event_t;
```

この関数は、有効 / 無効にされず、 BSP のスタートアップの一部として、両方のイベントに対して必ず呼び出されます。そのため、ユーザーがオーバーライドしている場合にはコールされない関数本体が必要です。関数の本体は system_xxxx にあります。この関数を使用するには、この関数をユーザーのコードにコピーし、要件を満たすように変更します。

Pre C ウォームスタートコールバック

このコールバックはリセット後ほぼすぐに呼び出されます。この時点では、 C ランタイム環境、クロック、IRQ がセットアップされていません。

「Pre C」ウォームスタートコールバックにユーザーが関心を持つ理由

以下にいくつかの例を挙げます。

- スタートアップ処理の一部としてのセーフティコード （たとえば破壊的なメモリテスト）の実行。
- クラッシュダンプ調査の一環としてのグローバルメモリの検査。
- すでに実行している RTC の再初期化の防止。

Post C ウォームスタートコールバック

このコールバックは、クロックと C ランタイム環境がセットアップされた後で呼び出されます。

「Post C」ウォームスタートコールバックにユーザーが関心を持つ理由

以下にいくつかの例を挙げます。

- クロックがセットアップされている必要のあるテストの実行。
- ADC の診断。
- ROM/ 外部メモリシステムのチェック。

2.6.1.11 カスタム BSP ボードサポート

いずれかの Synergy MCU に基づいて独自のボードを開発している場合はどうしますか。

以前のバージョンでは、外部コマンドラインユーティリティの Custom BSP Creator ツールがあり、これを使用して、e^2 studio 内からカスタム BSP を作成することができました。カスタム BSP の作成は現在、統合ソリューション開発環境内から行うことができます。これについては 「統合ソリューション開発環境使用上の注意」（3 章）で説明します。

2.6.1.12 BSP API 関数

BSP にはパブリック関数があり、BSP を使用するすべてのプロジェクトで使用できます。この関数を使用すると、 BSP でサポートされる MCU とボード全体で共通する機能にアクセスできます。

- R_BSP_SoftwareLockInit: BSP は、アトミックロックを実装するための API 関数を提供しています。これらのロックは、セマフォやミューテックスなどのコードの重要な領域を守るために使用できます。この関数は、単に定義されたロック構造体を BSP_LOCK_UNLOCKED に初期化します。
- R_BSP_SoftwareLock: 渡されたロックの取得を試みます。排他的ロード命令と排他的ストア命令は、排他的読み取り / 変更 / 書き込みを、入力ロックに対して実行するために使用されます。渡されたロックの取得を試みます。
  - 排他的ロード命令と排他的ストア命令は、排他的読み取り / 変更 / 書き込みを、入力ロックに対して実行するために使用されます。
  - この処理は以下のようになります。排他的ロード (LDREXB) を使用してロックの値を読み取ります。
  - ロックを使用できる場合は、ロック値を変更して確保します。
  - 返されたステータスビットをテストし、書き込みが実行されたかどうかを判断します。
- R_BSP_SoftwareUnlock: 既存のソフトウェアロックを解放します。
- R_BSP_HardwareLock: ハードウェアロックはソフトウェアロックに似ています。: 既存のソフトウェアロックを解放します。: ハードウェアロックはソフトウェアロックに似ています。実際に、BSP のソフトウェアロック関数は、ハードウェアロック関数によって呼び出されます。たとえば、フラッシュ API の open() 関数が呼び出された場合、フラッシュハードウェアのロックを取得して、フラッシュ API の close が呼び出されるまでロックを保持します。
- R_BSP_HardwareUnlock: 既存のハードウェアロックを解放します。上のフラッシュの例で、フラッシュ API の close 関数がこの関数を呼び出すことが考えられます。

- R_BSP_GroupIrqWrite: サポートされるいずれかのグループ割り込みに対し、コールバック関数を登録します。サポートされるいずれかのグループ割り込みに対し、コールバック関数を登録します。前述のように、割り込みは 12 個あり、すべて NMI 例外にマッピングされています。NMI が発生した場合、NMI_Handler が NMISR（ステータスレジスタ）を参照し、割り込みのソースを決定します。コールバック引数に NULL が渡された場合は、過去に登録されたコールバックが登録解除されます。
- R_BSP_IrqStatusClear: 特定の割り込みの割り込みステータスフラグ (IR) をクリアします。: 特定の割り込みの割り込みステータスフラグ (IR) をクリアします。割り込みがトリガされると、IR ビットがセットされます。
- R_BSP_SoftwareDelay: ブロッキングソフトウェア遅延を実装します。: ブロッキングソフトウェア遅延を実装します。遅延は、システムクロックレートに基づいて実装されます。
- R_BSP_VersionGet: BSP のバージョンを返します。
- R_BSP_LedsGet: ボード上の LED に関する情報を返します。
- R_BSP_ModuleStop : ストップビットを設定する必要のあるモジュールを指定します。
- R_BSP_ModuleStart : ストップビットをクリアする必要のあるモジュールを指定します。
- R_BSP_CacheOff(): ROM キャッシュをオフにして、以前の状態に戻します。
- R_BSP_CacheSet(): キャッシュを特定の状態（オンまたはオフ）に設定します。
- R_BSP_RegisterProtectEnable: レジスタ保護を有効にします。保護されたレジスタに書き込むことはできません。レジスタ保護は、保護レジスタ (PRCR) と MPC の書き込み保護レジスタ (PWPR) を使用することで可能になります。保護が可能なレジスタは、3 つのグループのいずれかにグループ化されます。
  - BSP_REG_PROTECT_CGC - クロック生成回路に関連するレジスタ。
  - BSP_REG_PROTECT_OM_LPC_BATT - 動作モード、低電力消費、バッテリーバックアップ機能に関連するレジスタ。
  - BSP_REG_PROTECT_LVD – LVD（低電圧検出）に関連するレジスタ。

BSP レジスタ保護機能は、参照カウンタを利用して、特定のレジスタを指定した後で別の関数を呼び出すアプリケーションで、そのレジスタ保護設定が誤って変更されないようにします。

- RegisterProtectDisable() が呼び出されるたびに、それぞれの参照カウンタがインクリメントされます。
- RegisterProtectEnable() が呼び出されるたびに、それぞれの参照カウンタがデクリメントされます。

どちらの関数も、参照カウンタがゼロの場合は、保護状態のみを変更します。

以下の例に示すように、参照カウンタがないと、MODULE1 が保護解除したレジスターを MODULE2 が再度保護し、MODULE1 が書き込めなくなります。

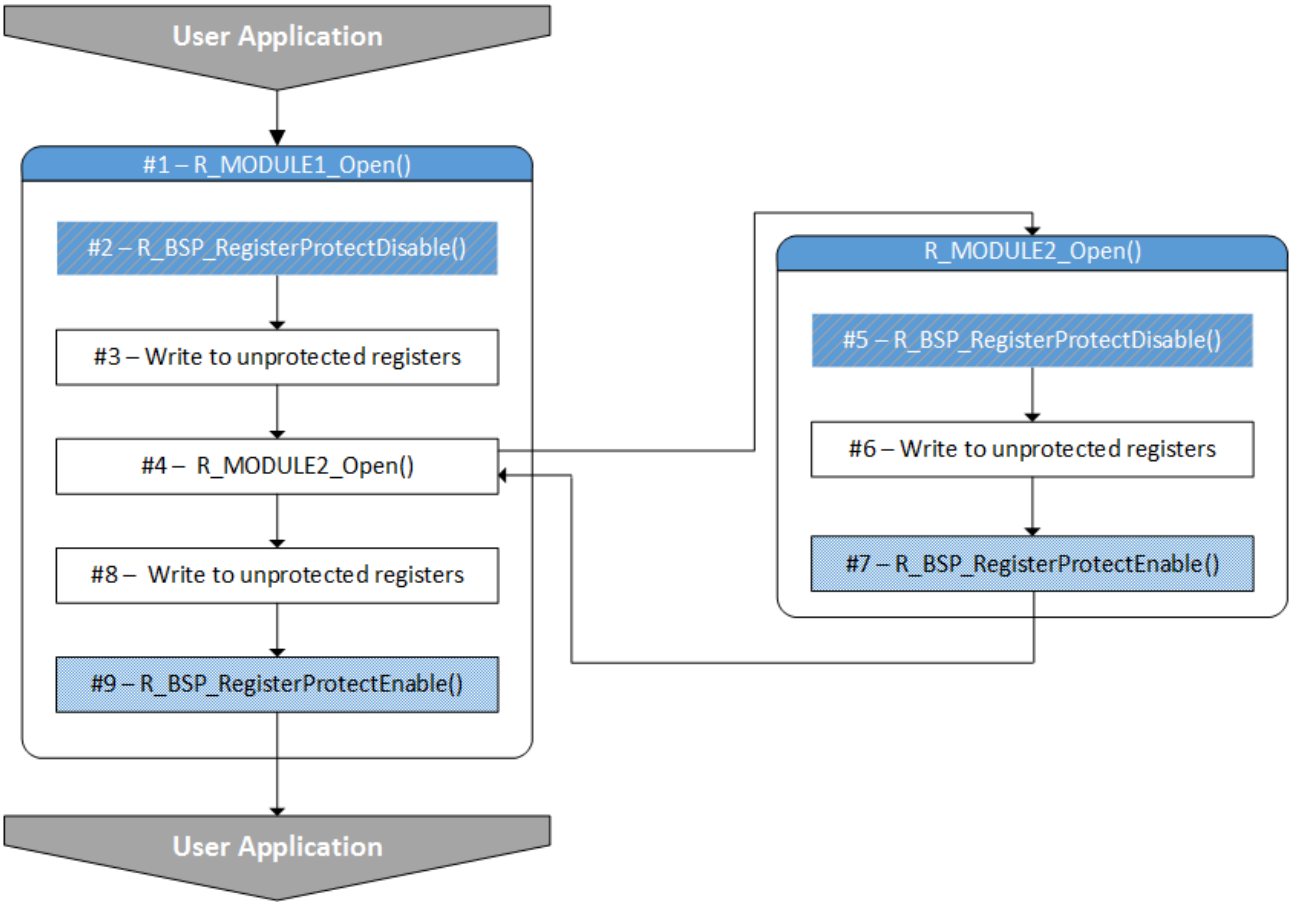


図 15:　レジスタ保護

- R_BSP_RegisterProtectDisable: レジスタ保護を無効にします。保護されていないレジスタに書き込むことができます。レジスタ保護は、保護レジスタ (PRCR) と MPC の書き込み保護レジスタ (PWPR) を使用して無効にします。ここでも上記のレジスタのグループ化が適用されます。

第 3 章　開発の開始

Renesas Synergy Software Package (SSP) を使用して開発を開始するには、 e2 studio をダウンロードして、対象となる Synergy 開発と評価ボードを入手し、この章にあるチュートリアルに従って実行します。e2 studio ISDE ユーザーガイドとチュートリアルには、簡単なアプリケーションから始められる詳細なインストラクションが含まれています。 SSP を始めるには、これらのページを参照してください。

- e2 studio ISDE ユーザーガイド
- チュートリアル : Your First Synergy Project - Blinky
- チュートリアル : Using HAL Drivers - Programming the WDT
- IAR Embedded Workbench for Renesas Synergy
- Synergy Standalone Configurator （SSC）とは
- SSP v1.3.z へのアップグレード

3.1 e2 studio ISDE ユーザーガイド

ここでは、e2 studio ISDE（統合ソリューション開発環境）を使用して、Renesas Synergy Software Package (SSP) を使用したアプリケーションを作成する方法について説明します。 SSP のアーキテクチャは、 e2 studio ISDE をどのように使用して Synergy アプリケーションを開発するかを直接決定します。このマニュアルに含まれる SSP アーキテクチャの詳細については、次のドキュメントを参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

e2 studio で生成および構築された簡単なサンプルプロジェクトについては、次を参照してください :

- チュートリアル : Using HAL Drivers - Programming the WDT
- チュートリアル : Your First Synergy Project - Blinky

このマニュアルのすべてのユーザーガイドでは、e2 studio ISDE を使用してドライバーを構成し、アプリケーションを開発する方法について説明しています。以下を参照してください。

- HAL レイヤー : HAL レイヤーユーザーガイド
- フレームワークレイヤー : フレームワークレイヤーユーザーガイド

e2 studio の幅広いヘルプコンテンツも参照してください。[Help] > [Help Contents] をクリックしてアクセスできます。

3.1.1 e2 studio ISDE の概要

e2 studio ISDE（統合ソリューション開発環境）は、コードの開発、ビルド、デバッグを包含した開発ツールです。 ISDE はオープンソース Eclipse IDE および関連する C/C++ Development Tooling(CDT) に基づいています。特に Synergy MCU では、Synergy プロジェクト用に、Synergy Software Package (SSP) を使用してコードを

設定および自動生成するための、いくつかのグラフィカルユーザーインタフェース (GUI) ウィザードが ISDE に備わっています。ISDE にはスマートマニュアルも取り込まれており、ドライバーとデバイスのドキュメントが、コード中でツールヒントの形で利用できます。

図 16: e2 studio の開始画面

e2 studio 統合ソリューション開発環境と Synergy Project Editor のコンフィギュレータは、特定のアプリケーションに必要な SSP モジュールをできるだけ簡単かつ迅速に選択し、プロジェクトに追加して構成できるように開発されています。 ISDE は、 Synergy MCU アプリケーション用に SSP のすべての要素を設定するグラフィカルユーザーインタフェースを提供しています。 ISDE は、 HAL モジュールとフレームワークモジュールに加えて、 RTOS スレッド、セマフォ、ミューテックス、イベントフラグ、キューを追加および設定できます。これにより、 RTOS のサポートをアプリケーションに非常に簡単に追加できます。プロジェクトが生成されたら、必要に応じて任意のモジュールと設定値を再設定できます。ISDE は構成ビューの選択から完全かつ正確な構成コードを生成するため、アプリケーションコードの記述に専念することができます。

プロジェクトの一部となる SSP の要素は、 e2 studio 統合ソリューション開発環境に含まれている Synergy Project Editor の [Threads] タブに表示されます。 SSP の構成構造とパラメータはすべて XML ファイルにマッピングされます。統合ソリューション開発環境ではこの XML ファイルにより、 [Properties] ビューに構成可能なオプションの視覚的リストを提示できます。モジュールを設定するためのコードを生成するのに加えて、 XML はモジュールの依存関係情報も提供します。

プロジェクトに SSP モジュールを追加すると、 e2 studio 統合ソリューション開発環境はそのモジュールの依存関係を確認し、すべての必要なドライバーとフレームワークモジュールを追加して、適切なソフトウェアスタックを作成します。ユーザーによる選択が必要な依存関係がある場合は、[Stacks] ペインでそのモジュールがハイライトされ、選択可能なオプションが統合ソリューション開発環境により表示されます。

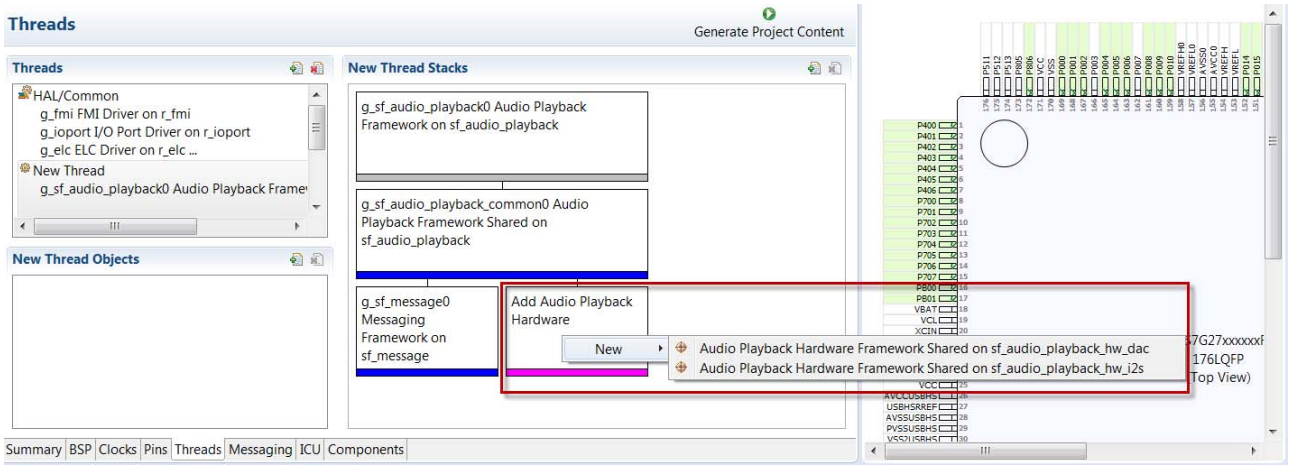


図 17: ISDE の依存関係チェック

エラーは、HAL/共通モジュールまたは新しいスレッドモジュールペインのドライバー名の横にフラグされます。また、[Problems] ビューでエラーを確認できます。

3.1.2 e2 studio ISDE の前提条件

3.1.2.1 Synergy キットの取得

SSP でアプリケーションを開発するには、いずれかの Renesas Synergy キットでスタートします。Renesas Synergy キットは e2 studio ISDE とシームレスに統合するよう設計されています。いくつかの種類のキットが提供されています。

- 開発キット (DK)
- スターターキット (SK)
- プロモーションキット （PK）
- 製品サンプル (PE)
- アプリケーションサンプル （AE）

すべての Synergy キットに関する注文情報、クイックスタートガイド、ユーザーマニュアル、その他のドキュメントが http://renesassynergy.com で入手できます。

3.1.2.2 PC 要件

e2 studio ISDE を使用するには、ご使用のパソコンが次の最小要件を満たしていることを確認してください。

- Windows 7 または Windows 10 と Intel i5 または i7、あるいは AMD A10-7850K または FX
- メモリ : 8 GB DDR3 または DDR4 DRAM （16 GB DDR4/2400 MHz RAM を推奨）
- 最低 2GB の空きハードディスク容量

3.1.2.3 e2 studio および SSP のインストール

e2 studio ISDE および SSP の詳細なインストール手順とインストーラーは、 Renesas Synergy Gallery ウェブサイト https://synergygallery.renesas.com. にあります。e2 studio のリリースノートをレビューして、e2 studio のバージョンが選択された SSP バージョンをサポートしていることを確認します。

関連項目 :SSP リリースの処理

3.1.2.4 ツールチェーンの選択

e2 studio ISDE は、 GNU Arm コンパイラや IAR ツールチェーンなど、いくつかのツールチェーンとツールチェーンバージョンと連携できます。特定のバージョンの GNU Arm コンパイラが e2 studio インストーラに含まれており、その SSP バージョンで実行できることが検証されています。

Arm 用 IAR ツールチェーンと e2 studio を連携させるには、IAR Embedded Workbench for Renesas Synergy（IAR EW for Synergy）をインストールします（Synergy Gallery から入手できます。ただし、IAR によるライセンス供与が必要です）。IAR で Synergy プロジェクトを始める前に、必ず IAR Embedded Workbench for Arm Eclipse プラグインをインストールします （[Help] メニューの IAR Embedded Workbench プラグインマネージャーを使用）。

3.1.2.5 SSP ライセンス設定

デフォルトで、 SSP ダウンロードには SSP の評価ライセンスが含まれます。 Synergy Project の構成中にライセンスファイルを指定するようプロンプトされた場合は、インストール手順のライセンスに関する説明を参照してください。評価ライセンスファイルは以下のディレクトリにあります :<e2_studio_base_dir>/internal/projectgen/arm/Licenses/

評価ライセンスで、完全に機能する SSP バージョンを使用することができます。これは、 SSP のすべてのモジュールをコンパイルして、アプリケーションにリンクできることを意味します。ただし、フレームワークレイヤーのモジュールのソースコードは暗号化されており、変更できません （Express Logic X-Ware コンポーネントを含む） 。この場合でも、選択したモジュールのソースコードを表示させることができます。このタイプの暗号化コードは保護コードと呼ばれます。詳細は、ライセンスファイルを参照してください。

HAL レイヤーのモジュールのソースコードは保護されておらず、 ssp/src/<module>/<module>.c ファイルをクリックして e2 studio ISDE から表示できます。

他のライセンスタイプでは、統合ソリューション開発環境のデバッグフェーズ中にセキュアデバッグビューで保護コードを表示できるか、選択されたソースに対して完全なリードまたはライトアクセスを行うことができます。

下の表では、利用可能なライセンスと機能を示しています。

License Type	利用可能	利用不可
Evaluation License	コンパイル / リンク	保護ソースの保存 / 編集。すべてのモジュールを見ることはできません。詳細については、ライセンスファイルを参照してください。

License Type	利用可能	利用不可
Development/Production License	Secure Viewer での保護ソースのコンパイル / リンクおよび表示	保護ソースの保存 / 編集
Source License	Secure Viewer での保護ソースのコンパイル / リンクおよび表示。選択したソース（ソースライセンスによる）は平文のテキストファイルとして編集および保存できます。	ソースライセンスに含まれない保護ソースファイルの保存 / 編集

3.1.3 Synergy プロジェクトとは

e2 studio では、すべての SSP アプリケーションは Synergy プロジェクトに編成されます。Synergy プロジェクトの設定には以下の部分があります。

1) プロジェクトの作成
2) プロジェクトの設定

e2 studio ISDE には、Synergy プロジェクト専用の多数のプロジェクトウィザードおよび構成ウィンドウがあります。Synergy Project Generator を使って新しい Synergy プロジェクトを作成することも、Synergy Project Editor を使って既存のプロジェクトの構成を編集することもできます。

e2 studio を起動してワークスペースを選択すると、選択したワークスペースで以前に保存したすべてのプロジェクトが読み込まれ、[Project Explorer] ビューに表示されます。各プロジェクトには、configuration.xml という関連する構成ファイルがあり、プロジェクトのルートディレクトリに置かれています。

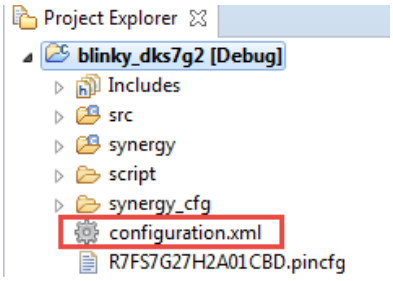


図 18: e2 studio のプロジェクト構成ファイル

configuration.xml ファイルをダブルクリックすると Synergy Project Editor が開き、このプロジェクトに関連する構成設定を表示または変更できます。プロジェクト構成を編集するには、Synergy Configuration パースペクティブが e2 studio ウィンドウの右上で選択されていることを確認してください。

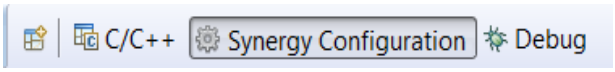


図 19: e2 studio Synergy Configuration パースペクティブ

注: Synergy プロジェクト構成（つまり、 configuration.xml ファイル）を保存すると、毎回、すべてのプロジェクト設定を含む詳細な Synergy プロジェクトレポートファイル (synergy_cfg.txt) が生成されます。ファイルの差分は、テキスト差分表示ツールを使用して簡単に確認することができます。生成されたファイルはプロジェクトのルートディレクトリに配置されます。

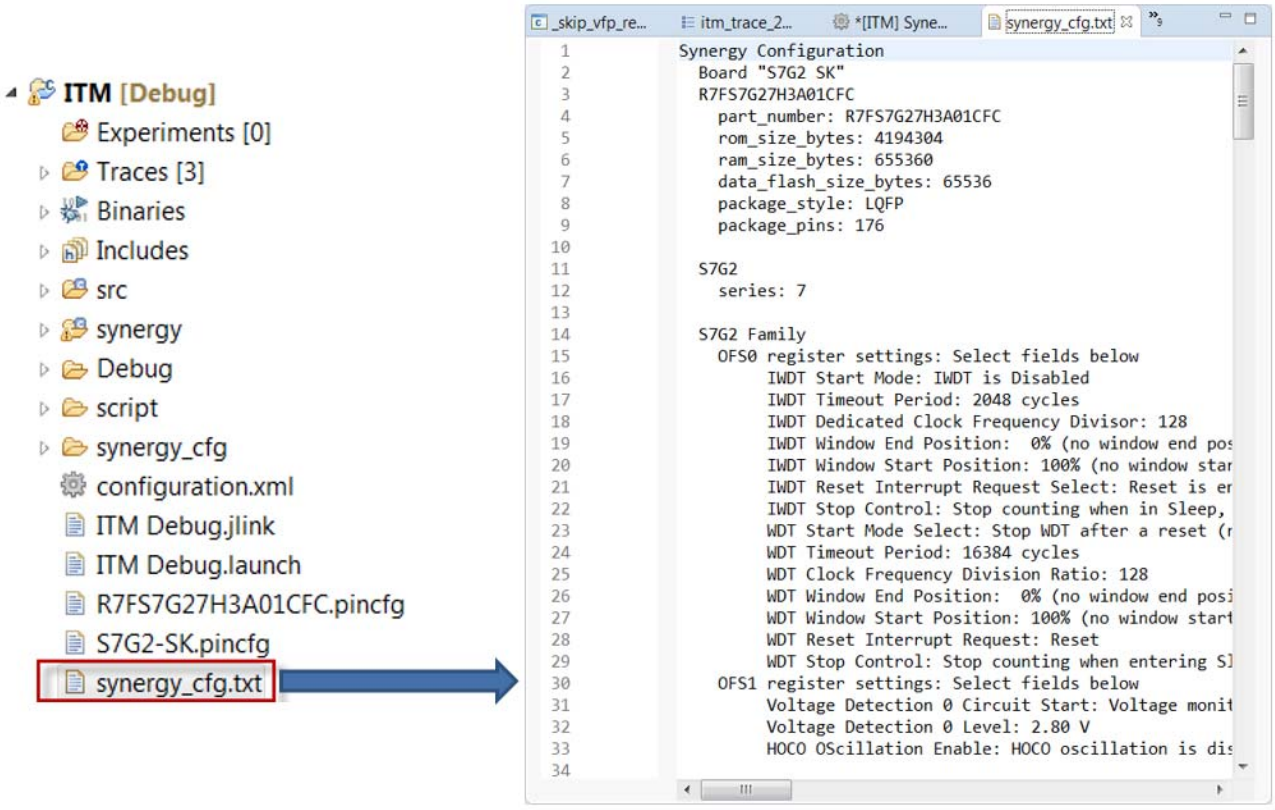


図 20: Synergy プロジェクトレポート

Synergy Project Editor にはいくつかのタブがあります。個別のタブの構成ステップとオプションについては、続くセクションで取り上げます。

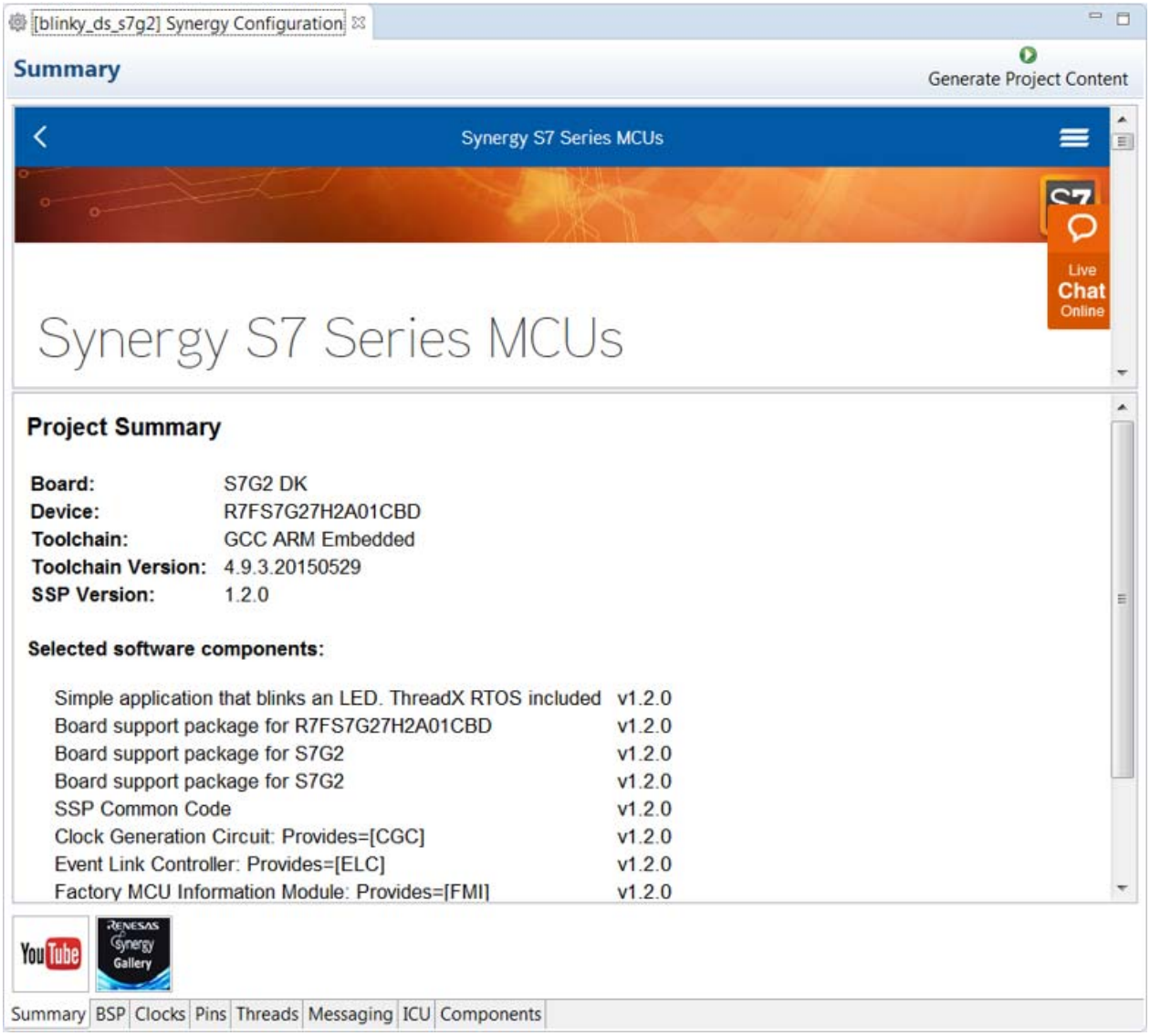


図 21: e2 studio Project Editor

3.1.4 Synergy プロジェクトの作成

このセクションでは、 Synergy C または C++ プロジェクトを作成する詳細な手順について説明します。プロジェクトを作成すると、ハードウェア （クロック、ピン、割り込みなど）とアプリケーションの一部であるすべての SSP モジュールのパラメータを簡単に構成できます。

Synergy Project Generator で新しい Synergy C または C++ プロジェクトを作成するには、プロジェクト名を指定し、アプリケーション用にハードウェアを選択し、ライセンスの確認または追加を行い、ツールチェーンを選択し、さらにプロジェクトテンプレートの選択により事前構成されたクロック、ピン、MCU 関連の設定を選びます。

3.1.4.1 新規 Synergy C または C++ プロジェクトの作成

Synergy アプリケーションでは、次の方法で新しいプロジェクトを Synergy プロジェクトとして生成します。

1) 新規の Synergy **C** プロジェクトを作成する場合には、[File] > [New] > [Synergy C Project] をクリックします。

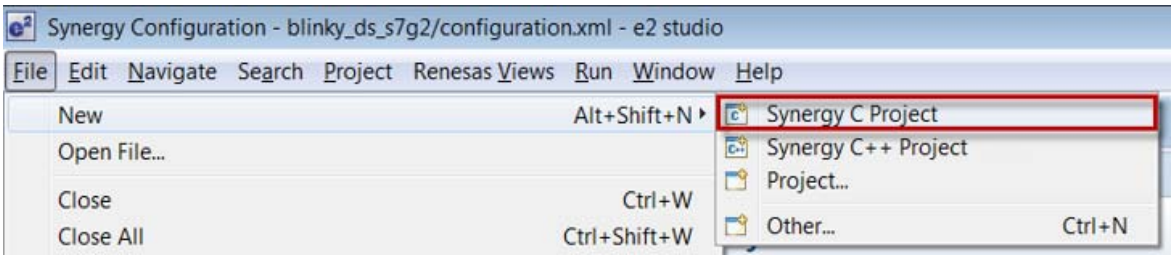


図 22: [New Synergy Project]

新規の Synergy **C++** プロジェクトを作成する場合には、 [File] > [New] > [Synergy C++ Project] をクリックします。 Synergy C++ Project Generator により、新しい C++ ソースファイルを Synergy プロジェクトに追加して、適正にビルドすることができます。

注 : SSP v1.1.z は C++ ベースのアプリケーションをサポートしていないため、C++ ベースのアプリケーションで直接使用することはできません。C++ アプリケーションを作成するには、SSP v1.2.0 を使用します。

2) プロジェクト名と場所を選択します。

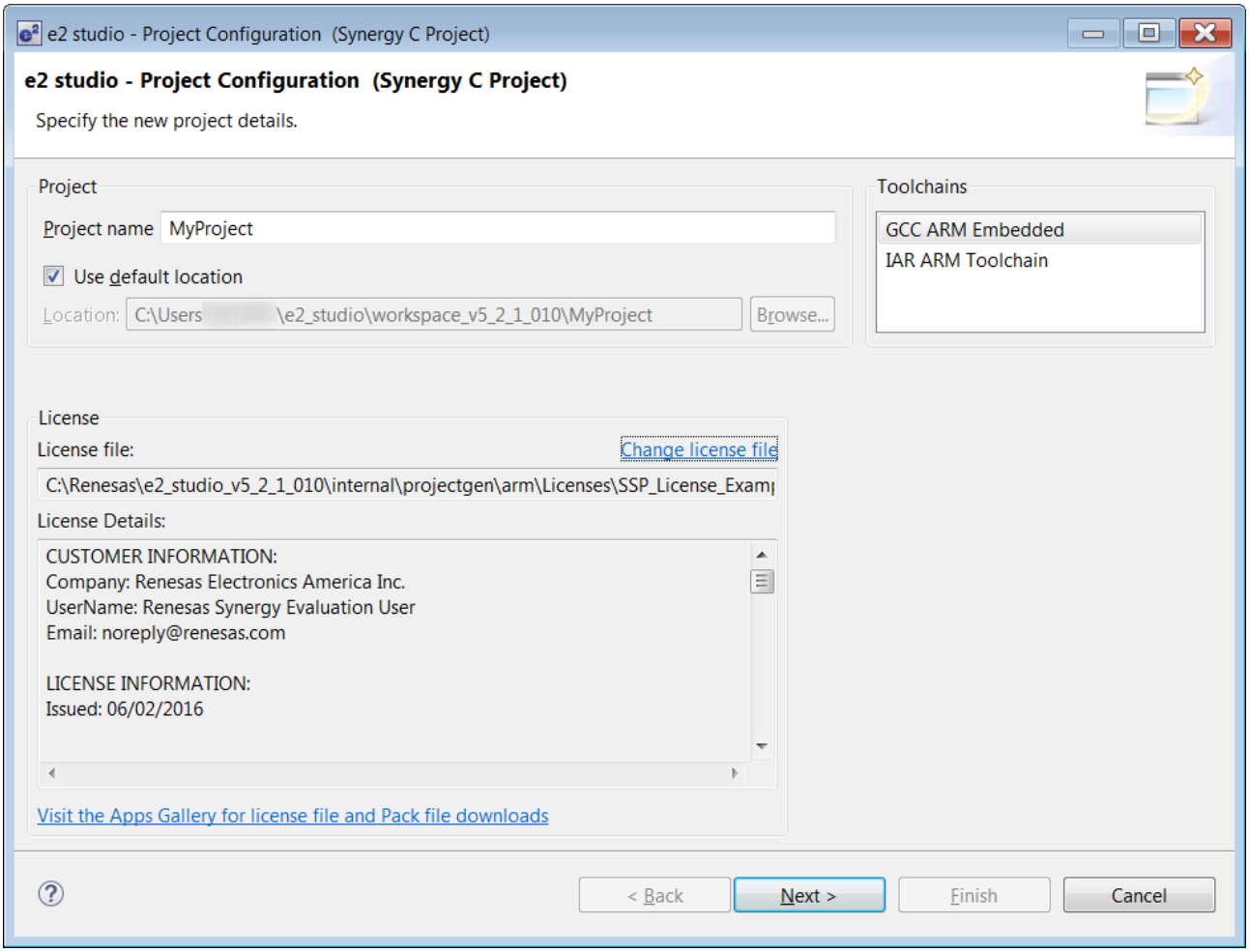


図 23: Synergy Project Generator（画面 1）

3) 最新の SSP のライセンスファイルがあることを確認します。SSP ダウンロードパックには評価ライセンスが含まれていますが、開発・生産ライセンスやソースライセンスなども利用可能です。ライセンスファイルの読み込み方法については、Synergy ウェブサイトにある『Installation Guide』を参照してください。e2 studio および SSP を初めてインストールした場合、ライセンスペインは空です。それ以外の場合、ライセンスペインにはインストールした最新のライセンスが表示されます。複数のライセンスがある場合は、いずれかを選択できます。ライセンスファイルは以下のディレクトリにあります：<e2_studio_base_dir>/internal/projectgen/arm/Licenses/

4) **[Next]** をクリックします。

3.1.4.2 ボードとツールチェーンの選択

次の [Project Configuration] ウィンドウで、ハードウェアとソフトウェアの環境を選択します。

1) プロジェクトで使いたい SSP のバージョンを選択します。
2) アプリケーションのボードを選択します。既存の Synergy キットを選択するか、独自の BSP 定義を持つ任意の Synergy デバイスの 「カスタムユーザーボード」を選択することができます。

   注 : 独自のカスタムボードパックを作成するには、カスタムボードパック （「カスタム BSP」）の作成の章を参照してください。
3) ツールチェーンのバージョンを選択します。これは GCC ツールチェーンでのみ利用できます。
4) デバッガを選択します。 J-Link Arm デバッガーがあらかじめ選択されています。
5) [Next] をクリックします。

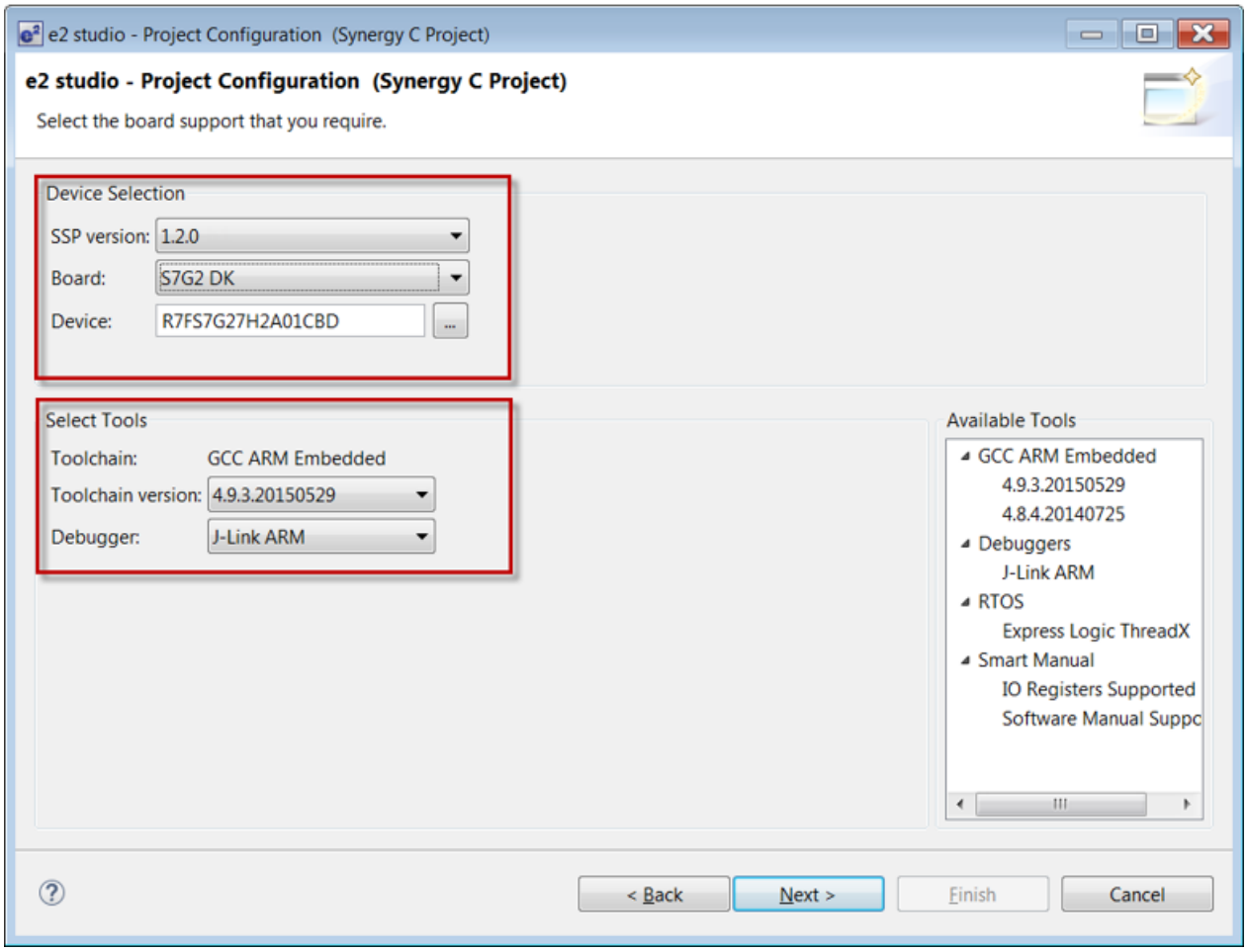


図 24: Synergy Project Generator （画面 2）

3.1.4.3 プロジェクトテンプレートの選択

続くウィンドウで、利用可能なテンプレートのリストからプロジェクトテンプレートを選択して、[Finish] をクリックします。

注: 独自のアプリケーションを開発する場合、「BSP」など、使用するボードの基本テンプレートを選択します。プロジェクトのモジュールを構成する際にいつでも RTOS サポートを追加できます。

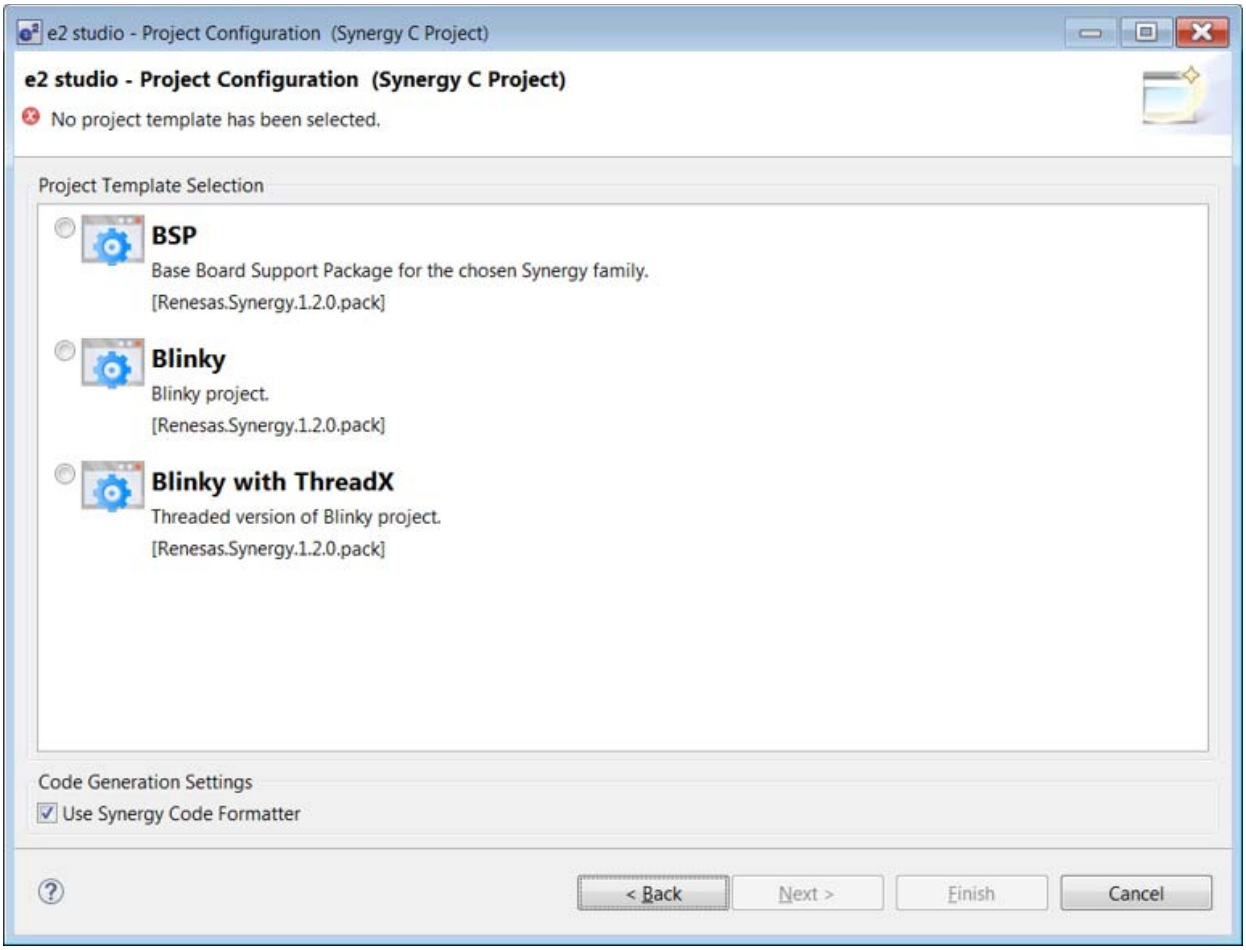


図 25: Synergy Project Generator（画面 3）

デフォルトでは、この画面には現在の SSP パックに含まれるテンプレートが表示されます。

プロジェクトが作成されると、ISDE は Synergy Project Editor で現在のプロジェクト構成の概要を表示します。

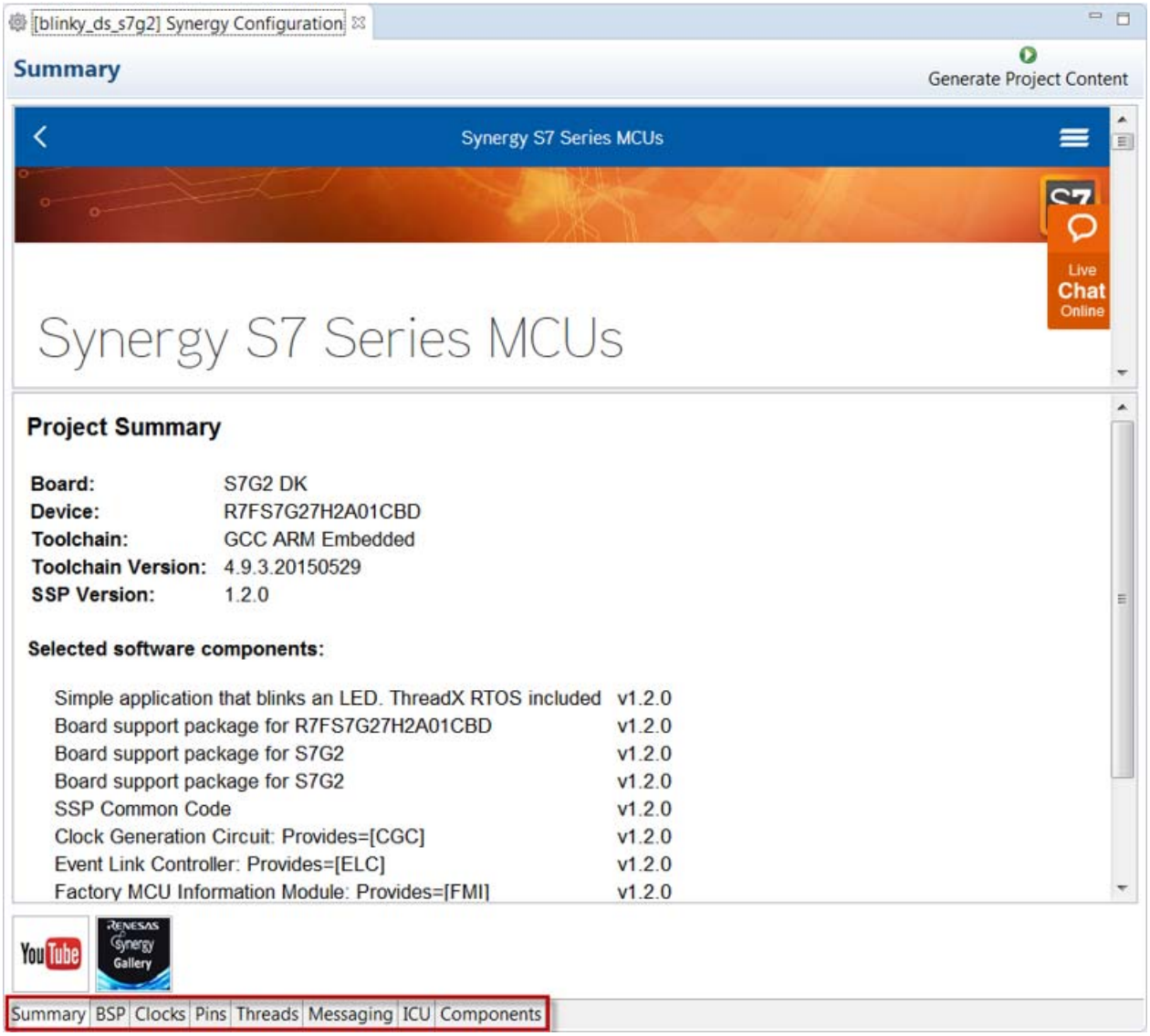


図 26: Synergy Project Editor と利用可能なエディタータブがあります。

Synergy Project Editor ビューの下で、自分のプロジェクトの複数のアスペクトを構成できるタブがあります。

- [BSP] タブでは、初期プロジェクト選択からボード別のパラメータを変更できます。
- [Clocks] タブでは、プロジェクトの MCU クロック設定を構成できます。
- [Pins] タブでは、各ポートピンの電気特性と機能をペリフェラルレベルまたは個々のピンレベルで構成できます。
- [Threads] タブでは、 RTOS およびそれ以外のアプリケーションの SSP モジュールとドライバーの追加と、モジュールとドライバーの構成を実行できます。このタブで選択したモジュールまたはドライバ

ーごとに、[Properties] ビューでは構成パラメータ、割り込み優先順位、ピンの選択へのアクセスを提供します。

- [Messaging] タブでは、 ThreadX ベースプロジェクトのメッセージフレームワークを構成できます。[Messaging] タブは、 e2 studio バージョン 5.0 以降に含まれています。
- [ICU] タブでは、割り込みを有効化し、割り込み優先順位を割り当てられます。

  注 : SSP v1.2.0 以降、ICU タブは使用されなくなり、空になっています。その理由は、SSP v1.2.0 以降、割り込みの優先順位をグローバルに構成するのではなく、割り込みを使用する SSP モジュールで直接設定するようになったからです。Synergy プロジェクトで用いられるモジュールごとに、使用するペリフェラルとチャネルに基づいて割り込みの優先順位が特定され、モジュールのプロパティに自動で転送されます。
- [Components] タブでは、選択したモジュールの概要を提供しています。ここでは、特定の SSP リリースのドライバーおよびアプリケーションサンプルコードを追加することもできます。しかし、基本的には [Threads] タブを使用して、 SSP モジュールをプロジェクトに追加するようにしてください。

3.1.5 プロジェクトの設定

プロジェクトには 2 つのレベルの構成があります。

- [BSP]、[Clocks]、および [Pins] のタブは、リセット後にユーザーコードが実行される前の MCU の初期構成を決定します。プロジェクトの作成時にプロジェクトテンプレートを選択することにより、 ISDE は選択したボードに適したデフォルト値を設定します。それらのデフォルト値は必要に応じて変更できます。
- [Threads] タブでは、プロジェクトに SSP モジュールを追加でき、アプリケーションの必要に応じてモジュールの構成パラメータを設定できます。メッセージングフレームワークは ThreadX ベースのアプリケーションに不可欠な部分であるため、 [Messaging] タブではメッセージングが必要な各スレッドのメッセージングフレームワークを設定できます （e2 studio バージョン 5.0 以降の場合）。

3.1.5.1 ISDE での BSP の設定

[BSP] タブには、現在選択されているボード（選択されている場合）とデバイスが表示されます。[Properties] ビューは、下図の Synergy パースペクティブの左下に表示されています。

*注 : [Prop*erties] ビューが表示されていない場合、トップメニューバーで [Window] > [Show View] > [Properties] *をクリックします。*

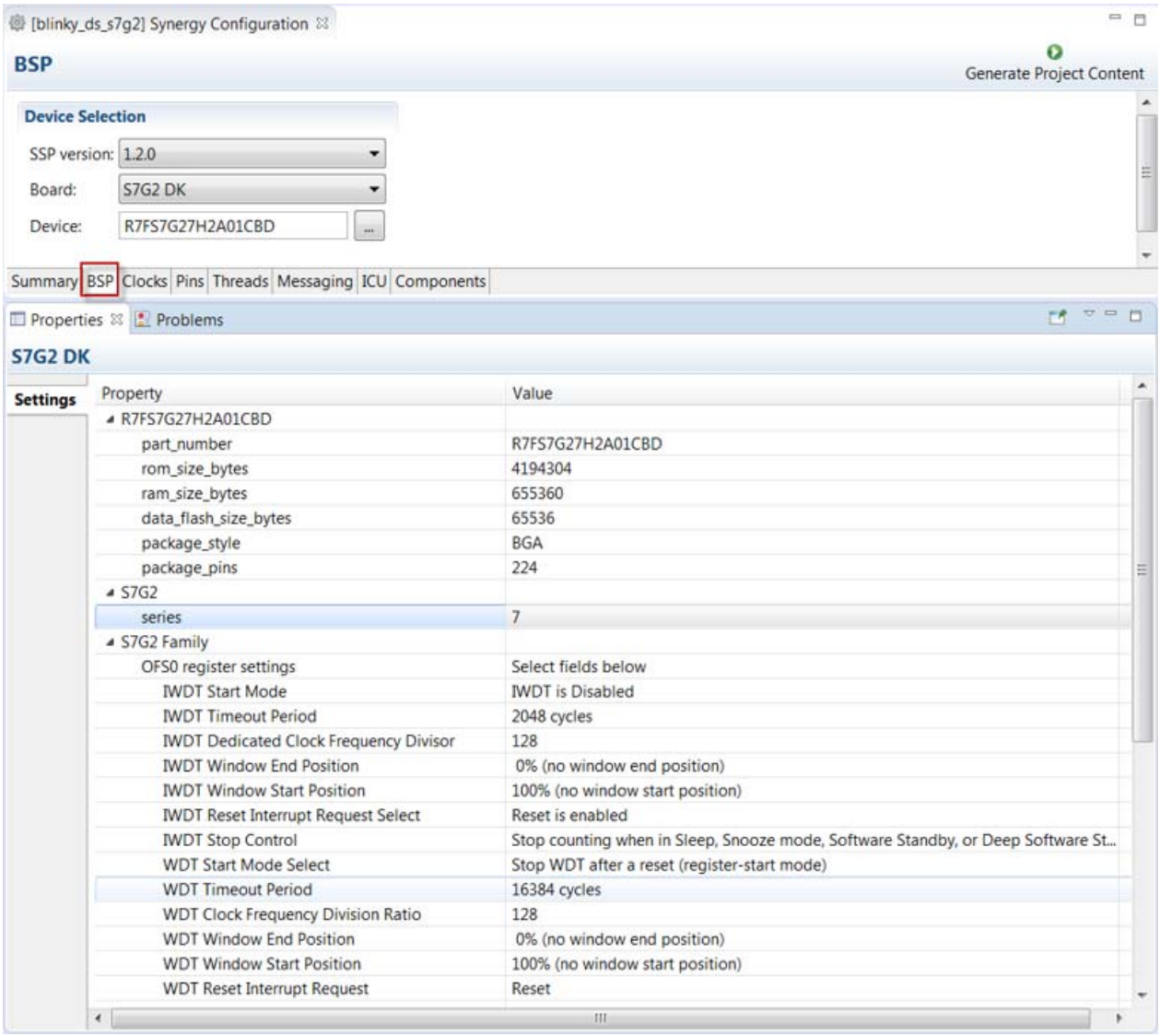


図 27: ISDE の [BSP] タブ

[BSP] タブでは、 [Properties] ビューに BSP で使用できる構成可能なオプションが表示されます。これらのオプションは、必要に応じて変更できます。BSP は、MCU ハードウェア上の SSP レイヤーです。ISDE は、入力フィールドをチェックし、無効な入力を通知します。たとえば、スタックサイズには有効な数値のみを入力できます。

[Generate Project Content] ボタンを押すと、 BSP の構成内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_cfg.h

このファイルが存在しない場合は作成されます。

***注:** このファイルは、*[GENERATE PROJECT CONTENT] *ボタンを押すたびに上書きされるため、編集しないでください。*

3.1.5.2 クロックの設定

[Clocks] タブには、MCU のクロックツリーがグラフィカルに表示され、各種のクロック除算値とソースを変更できます。クロック設定が無効な場合、問題のあるクロック値は赤く強調表示されます。この設定でもコードを生成できますが、正しい動作は保証されません。次の図で、USB クロック UCLK の除算値が変更され、クロック周波数が必要な 48 MHz ではなく 60 MHz になっています。このパラメータは赤く表示されます。

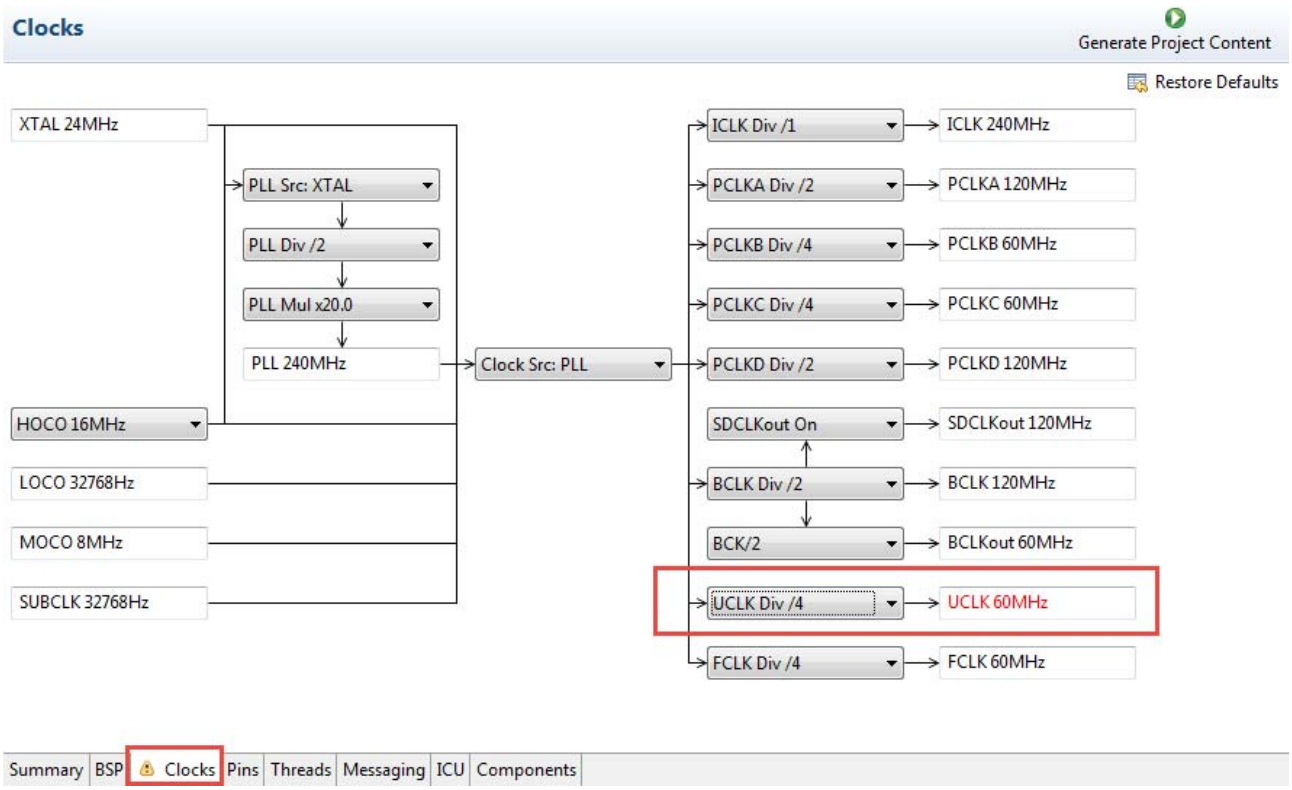


図 28: [ISDE Clocks] タブ

[Generate Project Content] ボタンを押すと、クロック構成の内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_clock_cfg.h

このファイルが存在しない場合は作成されます。

***注:** このファイルは、*[GENERATE PROJECT CONTENT] *ボタンを押すたびに上書きされるため、編集しないでください。*

3.1.5.3 ピンの設定

[Pins] タブでは、MCU のピンを柔軟に設定できます。多数のピンが多数の機能を提供できるため、ペリフェラルごとに設定できます。たとえば、SCI でシリアルチャネルを選択すると、そのモジュールとチャネルの送受信ピンの配置に関する複数のオプションが提供されます。設定されたピンは、[Package] ビューに緑色で表示されます。

注:[Package] ビューウィンドウがISDEで開いていない場合、トップメニューバーから [Window] > [Show View] > [Pin Configurator] > [Package] を選択して開きます。

[Pins] タブでは、ピンまたはペリフェラルごとにエラーをハイライトし、オプションを表示することで、ピンが高度に多重化された大規模なパッケージを簡単に構成できます。DK-S7G2 などの特定のボードでプロジェクトテンプレートを選択した場合、ボードに接続された一部のペリフェラルは事前に選択されています。

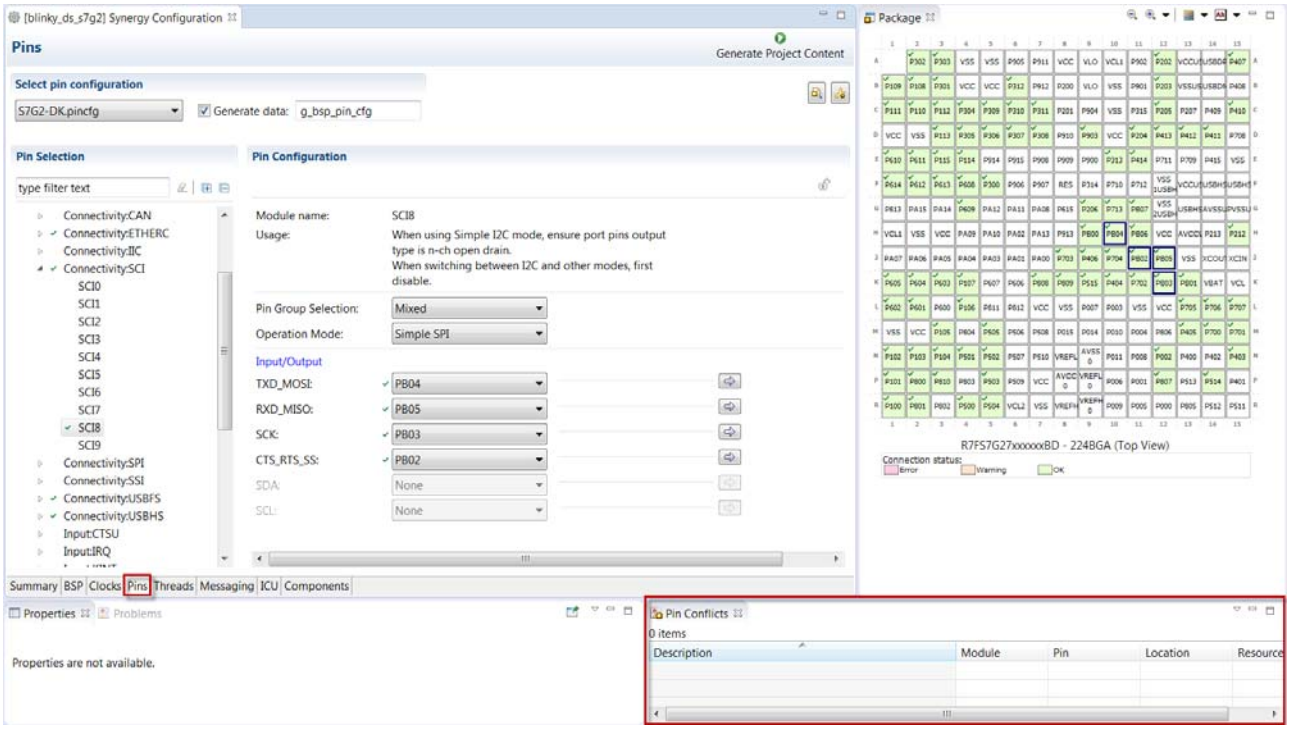


図 29: [ISDE Pins] タブ

ピンコンフィギュレータは、競合チェッカーを内蔵しています。同じピンが別のペリフェラルや I/O 機能に割り当てられた場合、ピンがパッケージビューで赤く表示され、メインの [Pins] タブの [Pin Selection] ペインと [Pin Configuration] ペインの赤い四角形の中に白い十字とともに表示されます。[Pin Conflicts] ビューにはすべての衝突が表示されるため、容易に識別および修正できます。

下の例では、ポートピン P100、P101、P102 は既に外部メモリペリフェラルに割り当てられ、使用されているので、これらのポートをシリアル通信インタフェース 0（SCI0）につなげようとすると、ダングリング接続エラーが発生します。このエラーを修正するには、別の SCI インタフェースを選択するか、タブの左側にある [Pin Selection] ペインで競合するペリフェラル（この場合は外部メモリペリフェラル）を無効にします。

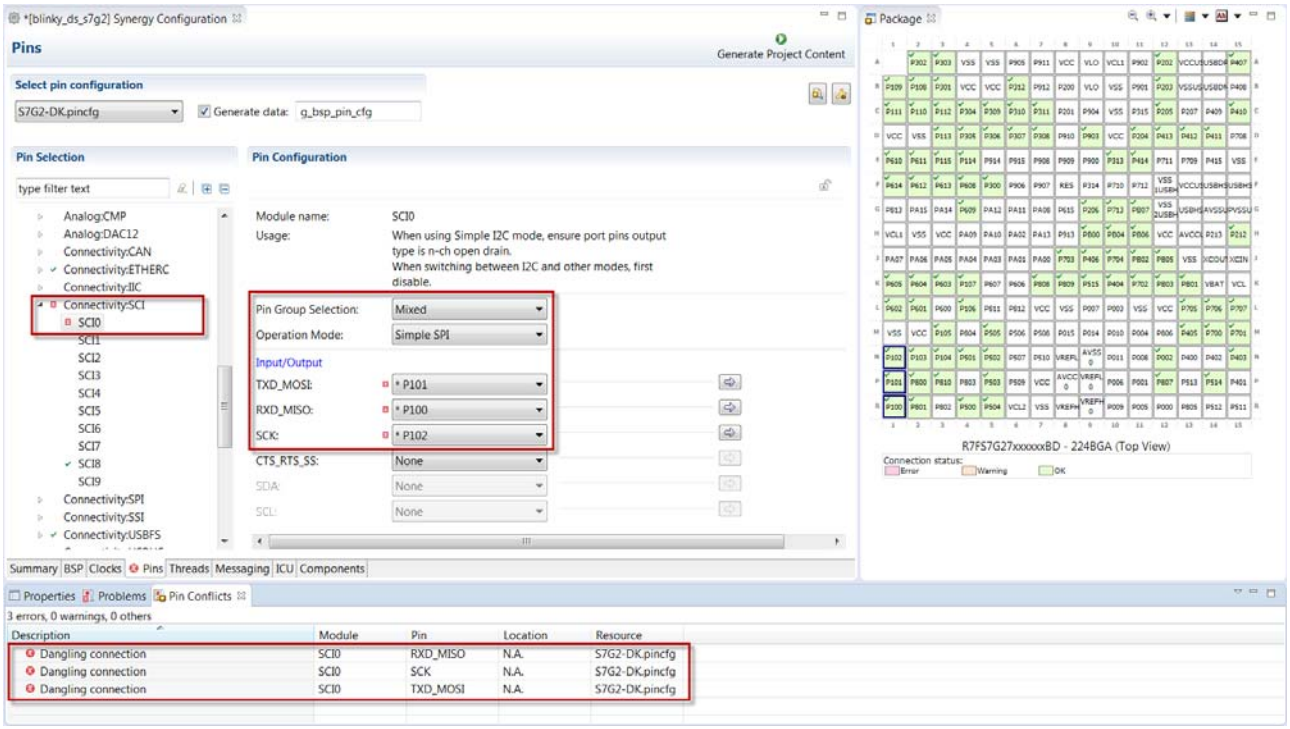


図 30: ISDE ピンコンフィギュレーター

ピンコンフィギュレータの [Package] ビューには、選択された各ピンの電気特性または機能特性が表示されます。

[Package] ビューでは、以下の表示を選択できます。

- 接続ステータス（デフォルト）
- ドライブ能力
- 動作モード
- 出力タイプ
- プルアップ

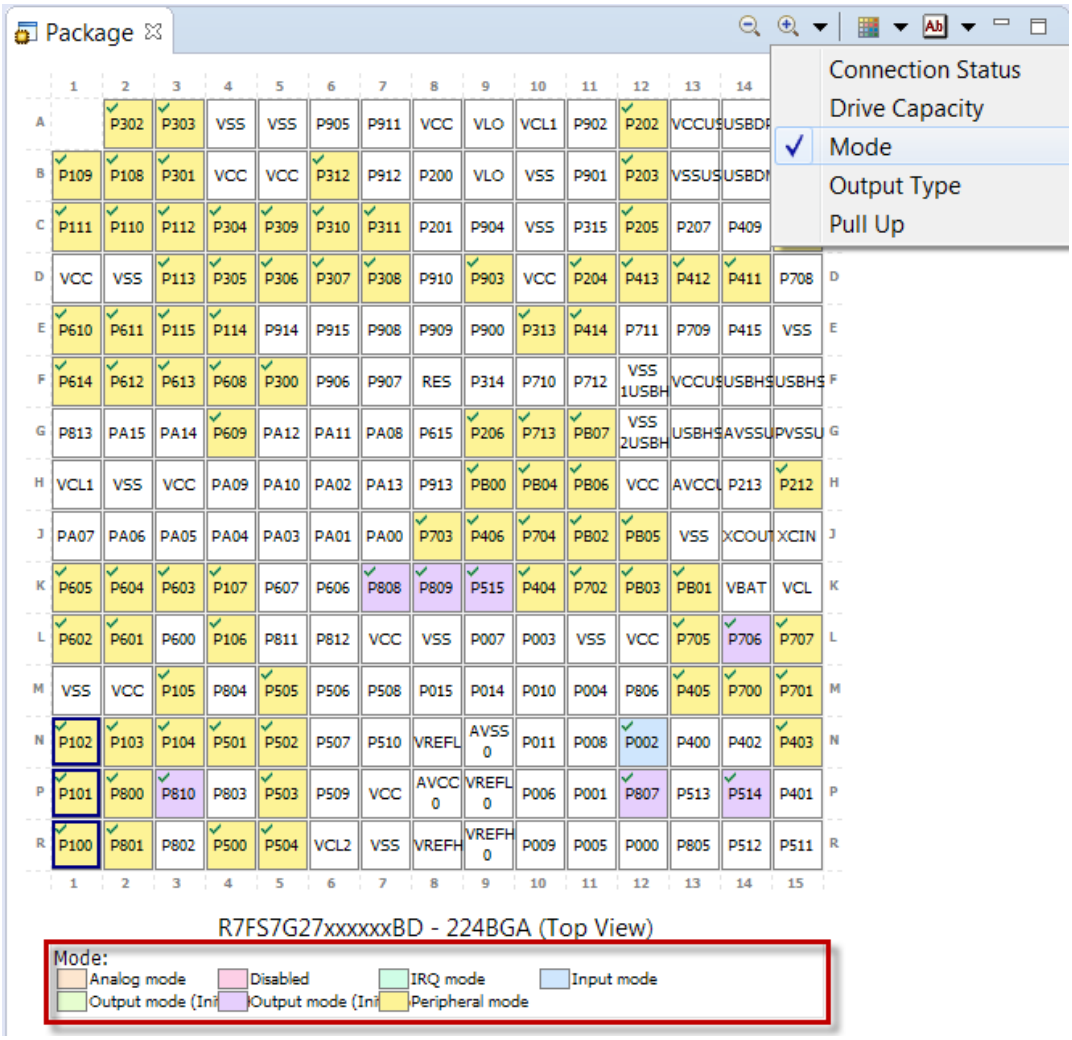


図 31: 統合ソリューション開発環境ピンコンフィギュレータの [Package] ビューに表示された各ピンの動作モード

ピンコンフィギュレータの [Import] ボタンをクリックして、既存の（互換性のある）ピン構成をプロジェクトに「インポート」することができます（e2 studio v5.2.1 以降）。この場合に行われるのは、部分的なインポート（つまり、現在のピン構成設定に対する追加）です。インポート機能が現在のピン構成とインポートされるピン構成との間に競合を検出した場合は、以下のオプションが提示されます。

- インポート操作をキャンセルする
- 競合を無視して、とにかく競合のある設定をインポートする
- 競合のある設定をインポートせずに、インポート操作を続ける

[Generate Project Content] ボタンを押すと、ピン構成の内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_pin_cfg.h および

src/synergy_gen/pin_data.c_

これらのファイルが存在しない場合は作成されます。

注: *これらのファイルは、*[GENERATE PROJECT CONTENT] *ボタンを押すたびに上書きされるため、編集しないでください。*

ピンに関する情報を簡単に共有できるよう、統合ソリューション開発環境はピン構成設定を csv 形式でエクスポートし、csv ファイルを synergy_cfg/ssp_cfg/bsp/<pincfg_file_name>.csv にコピーします。

ピン構成は、プロジェクトのルートレベルの Synergy プロジェクトレポート（ファイル 'synergy_cfg.txt'）にも含まれています。Synergy 構成（つまり、configuration.xml ファイル）を保存するたびに、Synergy プロジェクトレポートが作成または更新されます。

注: *SSP v1.1.z を使用して作成されたピン構成ファイル（*.pincfg*）には、SSP v1.2.0 で提供される新しいピンコンフィギュレータ XML ファイルとの互換性がありません。e² studio v5.2.1（およびそれ以降）には、SSP v1.1.z から SSP v1.2.0 プロジェクトへのアップグレード時に起動されるピン構成アップグレードユーティリティが含まれています。このユーティリティは、SSP v1.1.z ピン構成ファイルをアップグレードして SSP v1.2.0 で作成されるピンコンフィギュレータ XML との互換性を持たせ、e² studio のコンソールに出力します。また、ピン構成アップグレードユーティリティは手動で起動することもできます。その場合は、[Project Explorer] ビューで SSP v1.1.z ピン構成ファイルを右クリックし、[Migrate Pin Configuration] を選択します。*

3.1.6 スレッドとドライバーの追加

すべての ThreadX ベースの Synergy プロジェクトには、少なくとも 1 つの RTOS スレッドと、そのスレッドを実行する SSP モジュールのスタックが 1 つ含まれています。[Threads] タブは、スレッドに適切な SSP モジュールを追加して、スレッドのプロパティと各スレッドに関連するモジュールのプロパティを構成できるグラフィカルユーザーインタフェースです。スレッドを設定すると、その構成に応じて ISDE が自動的にコードを生成します。

任意のドライバー、さらに一般的にはスレッドに追加する任意のモジュールに対し、統合ソリューション開発環境は他のモジュールとのすべての依存関係を自動的に解決し、適切なスタックを作成します。このスタックは、選択されたモジュールとモジュールオプションに応じ、ISDE によって自動的に [Threads] ペインに入力表示されます。依存関係の要件を満たすモジュールが 2 つ以上ある場合、ドロップダウンメニューからモジュールを選択するよう表示されます。

たとえば、スレッドにオーディオ再生フレームワークを追加する際には、再生のフレームワークとして DAC または I2S を選択する必要があります。

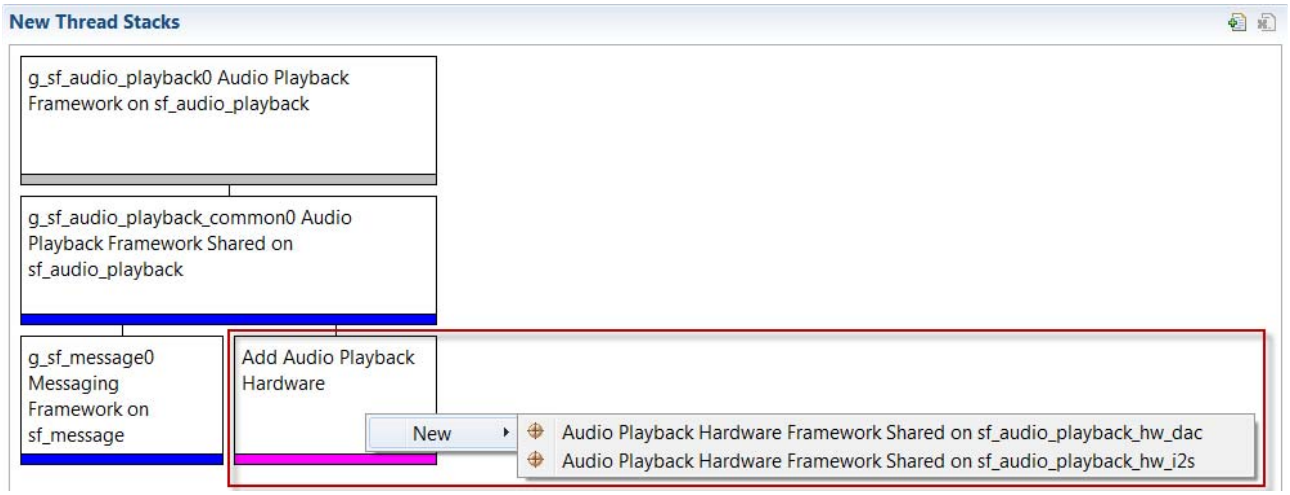


図 32: [Threads] タブでのモジュール依存関係の実現

オーディオ再生フレームワークではまた、タイマドライバーが必要で、AGT、GPT ドライバー実装のいずれかと転送ドライバーを選択しなければなりません。転送ドライバーの実装に選択肢があるかどうかは、この例では、選択された再生ハードウェアによって異なります。統合ソリューション開発環境では次のように表示されます。

- 再生ハードウェアとして DAC を選択した場合、DMAC と DTC の実装のいずれかを転送ドライバーとして選択するオプションがあります。

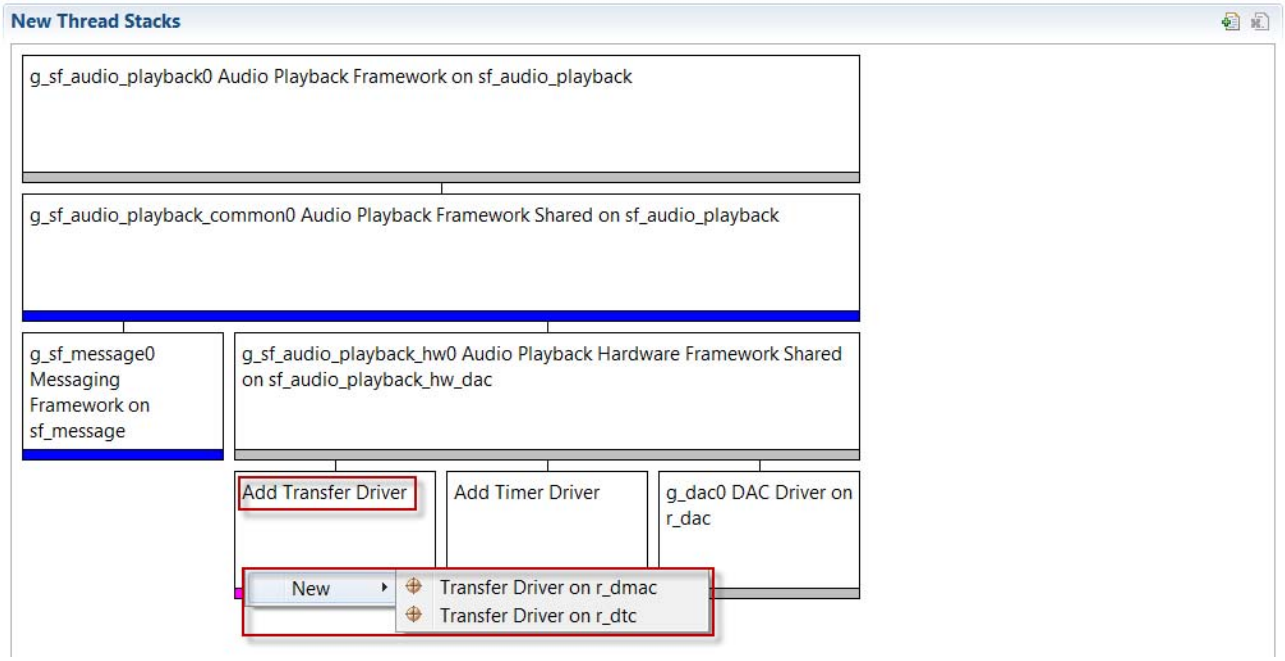


図 33: オーディオフレームワーク用の転送ドライバーの選択

- 再生ハードウェアとして I2S を選択した場合は、 DTC のみがサポートされます。 2 つのインスタンス（チャネルごとに 1 つ）の DTC が必要ですが、これは統合ソリューション開発環境によって追加されます。タイマドライバーを選択する必要もあります。

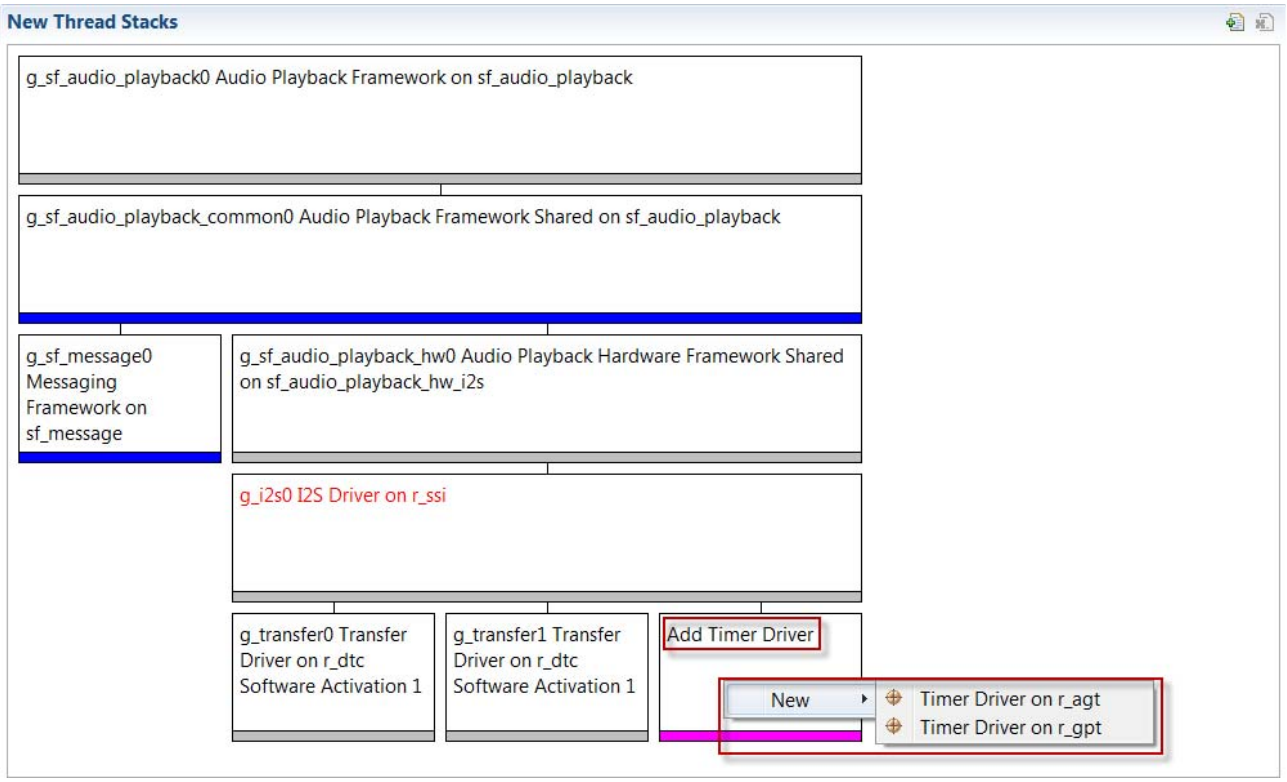


図 34: タイマドライバーの選択

図にあるように、 [Threads] タブではモジュールインスタンスが色分けされた状態で表示されます。

- 灰色のバーで示されるモジュールインスタンスは、それが存在するスレッドに固有です。
- 青色のバーで示されるモジュールインスタンスは、複数のスレッドで共有されている可能性があります。
- ピンクのバーで示されるモジュールインスタンスは、ユーザーからのアクションを必要とする場合があります。

モジュールインスタンス内の赤い文字は、ユーザーが構成する必要のあるプロパティが 1 つ以上あることを示します。

[Threads] タブのデフォルトビューには、[HAL/Common] という共通スレッドが含まれています。このスレッドには、 I/O 制御 （IOPORT）、クロック発生回路 （CGC）、ファクトリー MCU 情報 （FMI）、イベントリンクコントローラ （ELC）のドライバーが付属しています。デフォルトのスタックは [HAL/Common Stacks] ペインに表示されています。HAL/ 共通スレッドに追加されるデフォルトのモジュールは、SSP がそれぞれに単一のインスタンスのみを必要とするという点で特殊で、これは ISDE によってすべてのユーザー定義スレッドにデフォルトで含まれます。

RTOS を使用しないアプリケーション、または RTOS 外で実行されるアプリケーションでは、HAL/ 共通スレッドがアプリケーションにドライバーを追加できるデフォルトの場所となります。

モジュールとスレッドの追加および設定方法についての詳細は、以下のセクションを参照してください。

- HAL ドライバーの追加と設定
- ドライバーのスレッドへの追加とドライバーの設定

モジュールを [HAL/Common] または新しいスレッドに追加したら、[Properties] ビューでドライバー設定オプションにアクセスできます。スレッドオブジェクトを追加したら、同様に [Properties] ビューでオブジェクト構成オプションにアクセスできます。

スレッドの設定方法については、スレッドの設定を参照してください。

注: *ドライバーやモジュールの選択肢や構成オプションは SSP パックで定義され、そのため SSP バージョン変更時に変更することができます。*

注: *SSP のモジュールとドライバーの定義については、「SSP 用語」を参照してください。*

3.1.6.1 HAL ドライバーの追加と設定

RTOS の外または RTOS なしで実行されるアプリケーションの場合、[Threads] タブで [HAL/Common] スレッドを使用してアプリケーションに他の HAL ドライバーを追加できます。ドライバーを追加するには、次の手順を実行します。

1) [Threads] ペインで HAL/Common アイコンをクリックします。[Modules] ペインが [HAL/Common Stacks] に変わります。

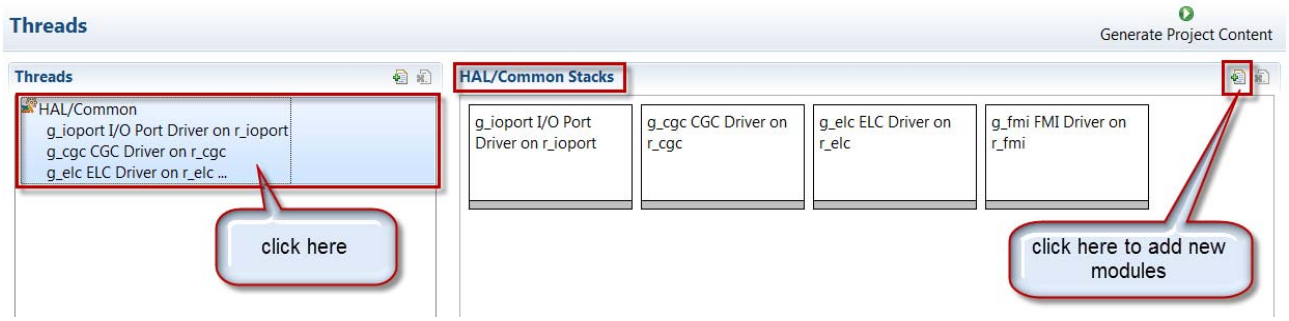


図 35: ISDE プロジェクトコンフィギュレーター - ドライバーの追加

2) [New Stack] をクリックすると、SSP で使用可能な HAL レベルドライバーのドロップダウンリストが表示されます。

3) [New] > [Driver] からドライバーを選択します。また、 [New] > [Framework] > [Service] > [Power Profiles Framework on sf_power_profiles] から RTOS 独立アプリケーションのパワープロファイルを選択することができます。他のすべてのモジュールは、 ThreadX が存在する場合にのみ、スレッドに追加できます。

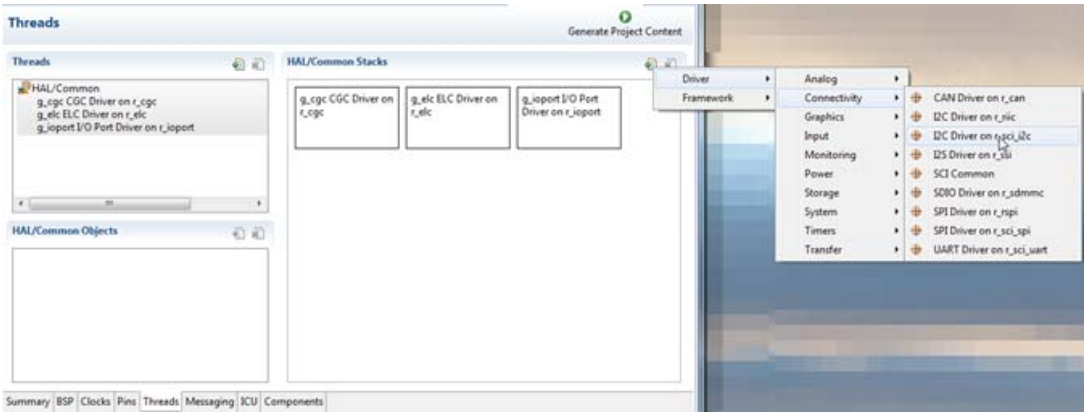


図 36: ドライバーを選択します

ISDE は選択されたドライバーのスタックを作成し、ドライバーで有効化する必要のある追加のリソースがいつ必要かを知らせます。次の例では、[Properties] ビューで、CAN ドライバーに関する割り込みとコールバック関数名を構成できます。

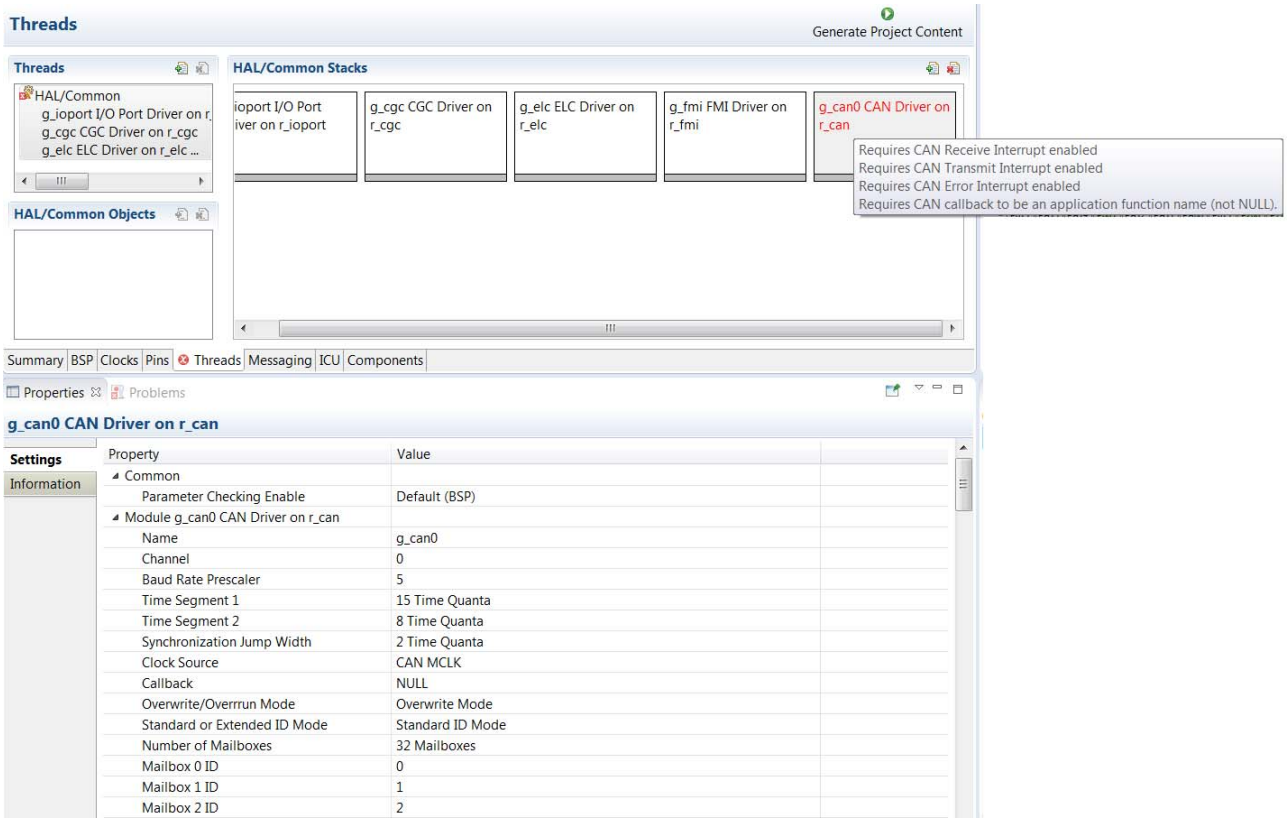


図 37: [Threads] タブでの依存関係の確認

4) ドライバーモジュールを [HAL/Common Modules] ペインで選択し、ドライバーのプロパティを [Properties] ビューで構成します。

[Generate Project Content] ボタンをクリックすると、統合ソリューション開発環境によって以下のファイルが追加されます。

- synergy/ssp ディレクトリ向けに選択したドライバーモジュールとそのファイル。
- main()関数と構成構造体、およびアプリケーション用のヘッダーファイルが下記の表に表示されています。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/main.c	main() 呼び出しとユーザーコードが含まれます。呼び出されたときには、BSP により MCU が初期化されています。	はい
src/synergy_gen/hal_data.c	HAL ドライバーのみのモジュールの設定構造体。	はい
src/synergy_gen/hal_data.h	HAL ドライバーのみのモジュールのヘッダーファイル。	はい
src/hal_entry.c	HAL ドライバーのみのコード用のユーザーエントリポイント。ここにコードを追加します。	いいえ

追加されているすべてのモジュール用の構成ヘッダーファイルは、このフォルダーに作成または上書きされます。

synergy_cfg/ssp_cfg/driver

3.1.6.2 ドライバーのスレッドへの追加とドライバーの設定

ThreadX RTOS で使用されるアプリケーションの場合、1 つ以上のスレッドを追加し、スレッドごとにそのスレッドで実行されるモジュールを追加できます。ドライバーまたはフレームワークのドロップダウンメニューからモジュールを選択できます。スレッドにモジュールを追加するには、次の手順を実行します。

1) [Threads] ペインで、 [New Thread] をクリックし、スレッドを追加します。

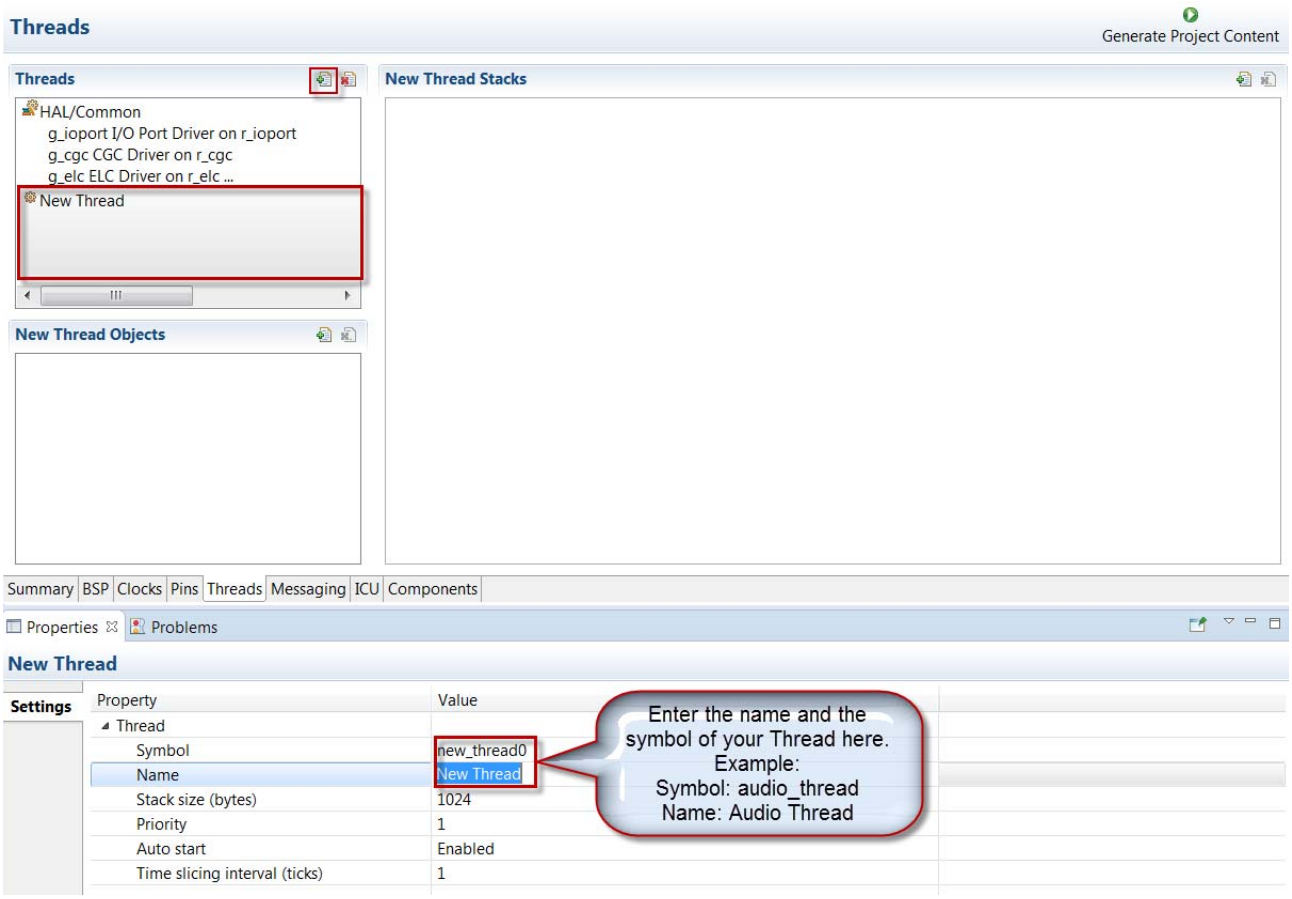


図 38: [Threads] タブでの新規 [RTOS Thread] の追加

2) [Properties] ビューで、[Name] と [Symbol] のエントリーをクリックして、新しいスレッドに固有の名前と記号を入力します。

注 : 統合ソリューション開発環境が、スレッドスタックペインの名前をユーザーがスレッド用に選択した名前に更新します。

3) スレッドスタックペインで [New] をクリックして、モジュールとドライバーのリストを表示します。フレームワークレベルのモジュールと HAL レベルのドライバーの両方をここで追加できます。

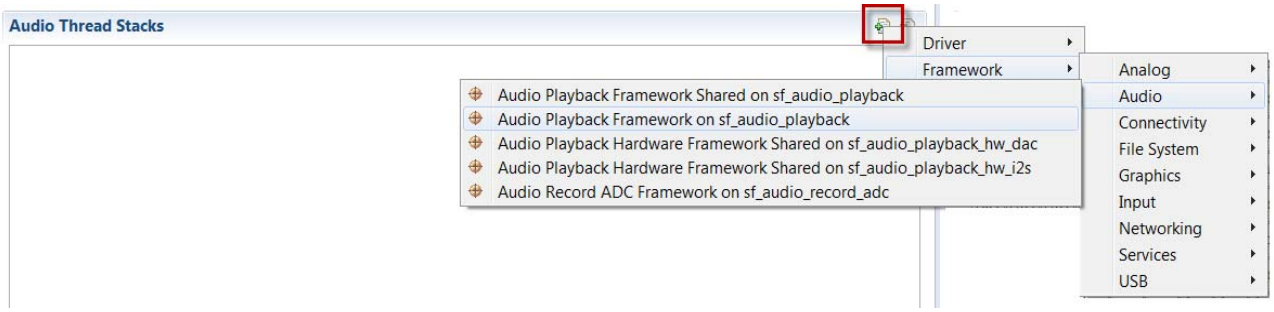


図 39: モジュールとドライバーをスレッドに追加

4) リストからモジュールまたはドライバーを選択します。

5) モジュールまたはドライバーが依存関係を示している場合、不足しているリソースを選択します。

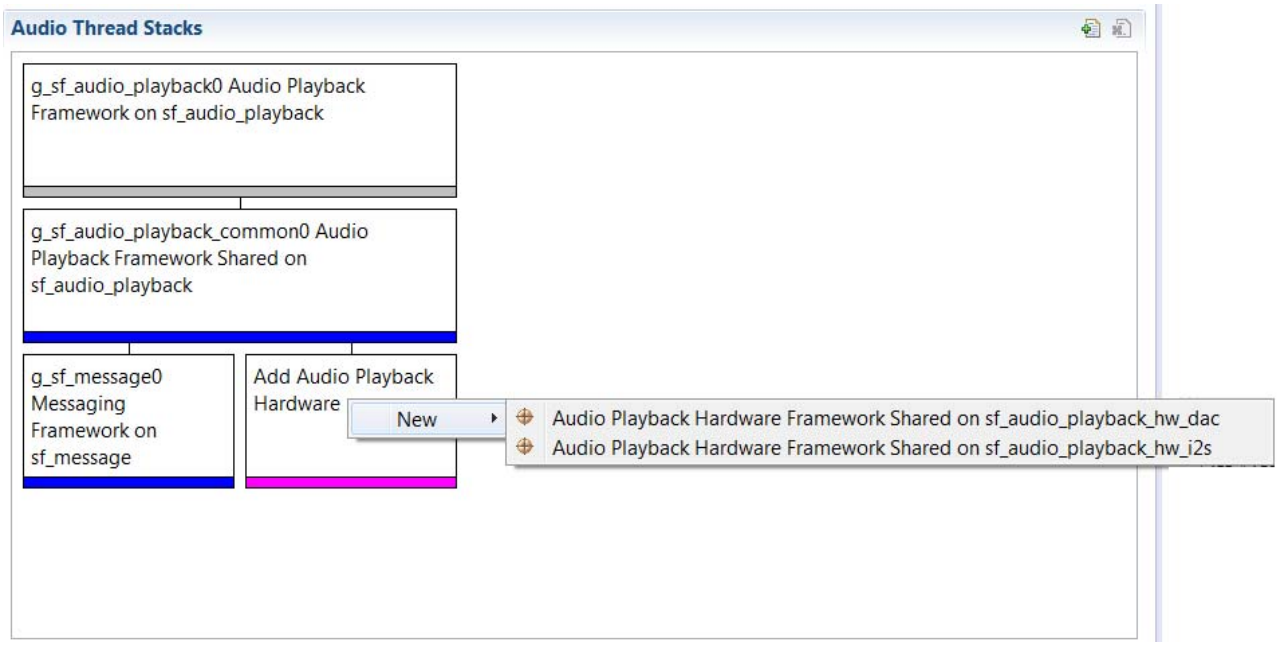


図 40: [Threads] タブでモジュールまたはドライバーの依存関係の特定

6) 追加したドライバーをクリックして、[Properties] ビューの構成パラメータを更新し、必要な依存関係を解決することにより、アプリケーションに必要なドライバーを構成します。選択したモジュールまたはドライバーを表示し、そのプロパティを編集するには、ドライバーを含むスレッドが [Threads] ペインでハイライトされていることを確認します。

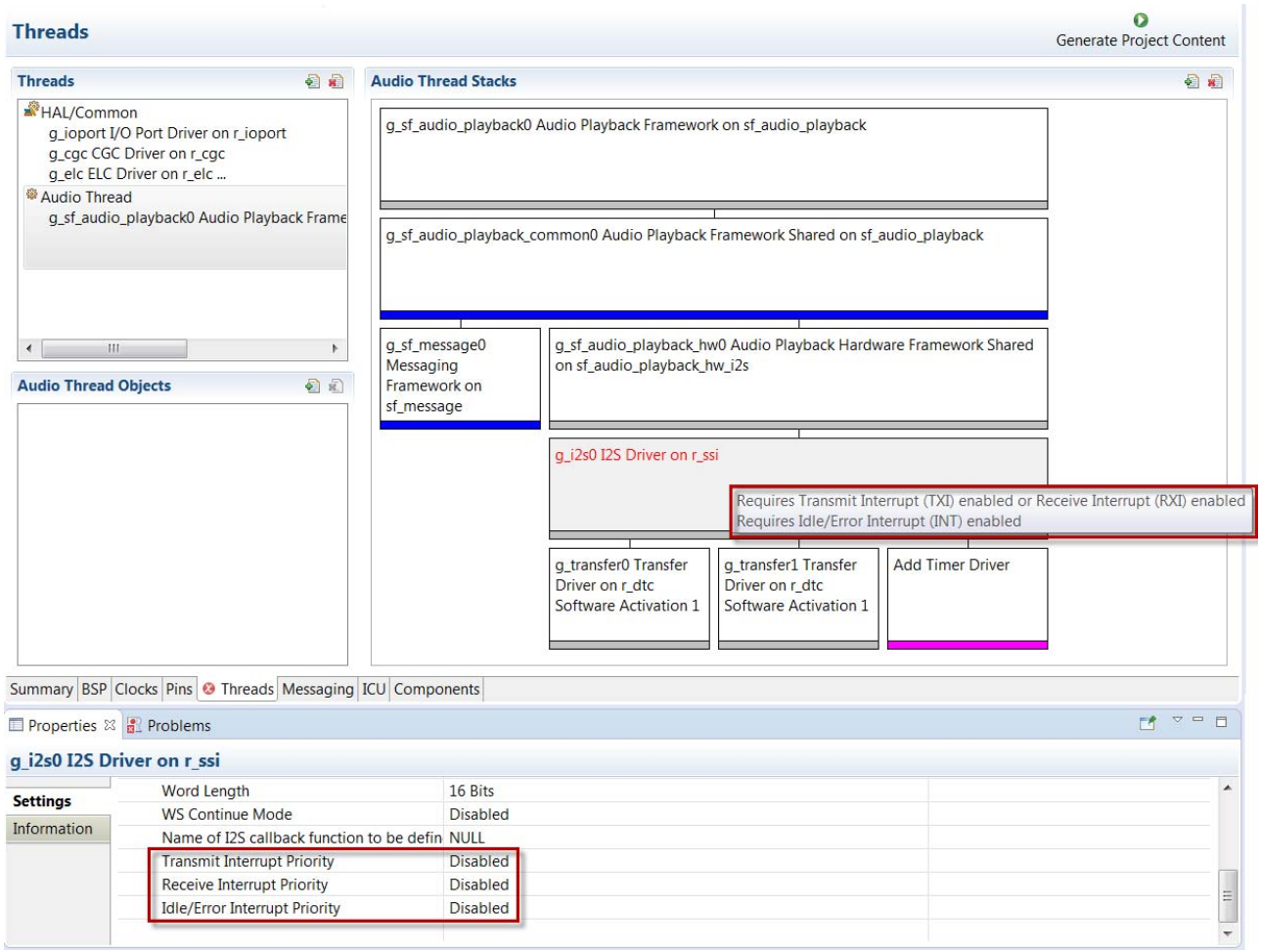


図 41: モジュールまたはドライバーのプロパティの設定

7) 必要に応じて、[Threads] ペインの [New] をクリックして、別のスレッドを追加します。

上記の例で [Generate Project Content] ボタンを押すと、ISDE によって以下の表に示されているファイルが作成されます。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/main.c	main() 呼び出しとユーザーコードが含まれます。呼び出されたときには、BSP により MCU が初期化されています。	はい

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/my_thread.c	生成されたスレッド「my_thread」と、このスレッドに追加されたモジュールの構成構造体。	はい
src/synergy_gen/my_thread.h	スレッド「my_thread」用のヘッダーファイル	はい
src/synergy_gen/hal_data.c	HAL ドライバーのみのモジュールの設定構造体。	はい
src/synergy_gen/hal_data.h	HAL ドライバーのみのモジュールのヘッダーファイル。	はい
src/hal_entry.c	HAL ドライバーのみのコード用のユーザーエントリポイント。ここにコードを追加します。	いいえ
src/my_thread_entry.c	スレッド「my_thread」用のユーザーエントリポイント。ここにコードを追加します。	いいえ

追加されているすべてのモジュールとドライバー用の構成ヘッダーファイルは、次のフォルダーに作成または上書きされます。

synergy_cfg/ssp_cfg/driver

synergy_cfg/ssp_cfg/framework

3.1.6.3 スレッドの設定

アプリケーションで ThreadX RTOS が使用されている場合、[Threads] タブを使用して、ThreadX スレッド、セマフォ、ミューテックス、イベントフラグを簡単に作成できます。

各スレッドのコンポーネントは、([Threads] ペインで選択されていれば) [Properties] ビューで構成できます。

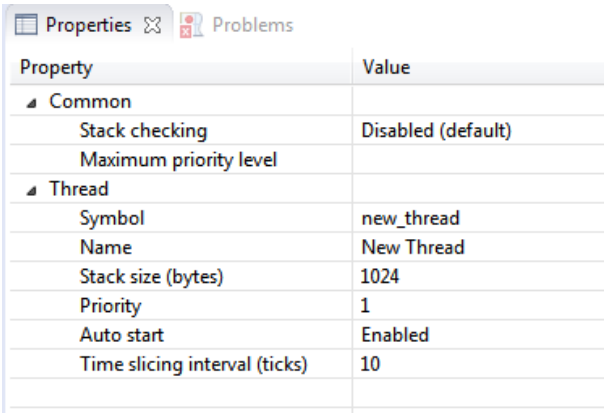


図 42: ISDE スレッドのプロパティ

[Properties] ビューには、すべてのスレッドに共通の設定（[Common]）と、この特定のスレッドの設定（[Threads]）が含まれています。

このスレッドインスタンスについて、スレッドの名前、プライオリティレベルやスタックサイズなどのプロパティを簡単に設定できます。 ISDE は、プロパティフィールドのエントリーが有効であることを確認します。たとえば ISDE は、整数値を必要とする [Property] フィールドなどに 0 ～ 9 の数値のみが含まれていることを確認します。

ThreadX リソースをスレッドに追加するには、[Threads Objects] ペインでスレッドを選択して [New Object] をクリックします。ペインは選択したスレッドの名前、この場合は [Audio Thread Objects] となります。

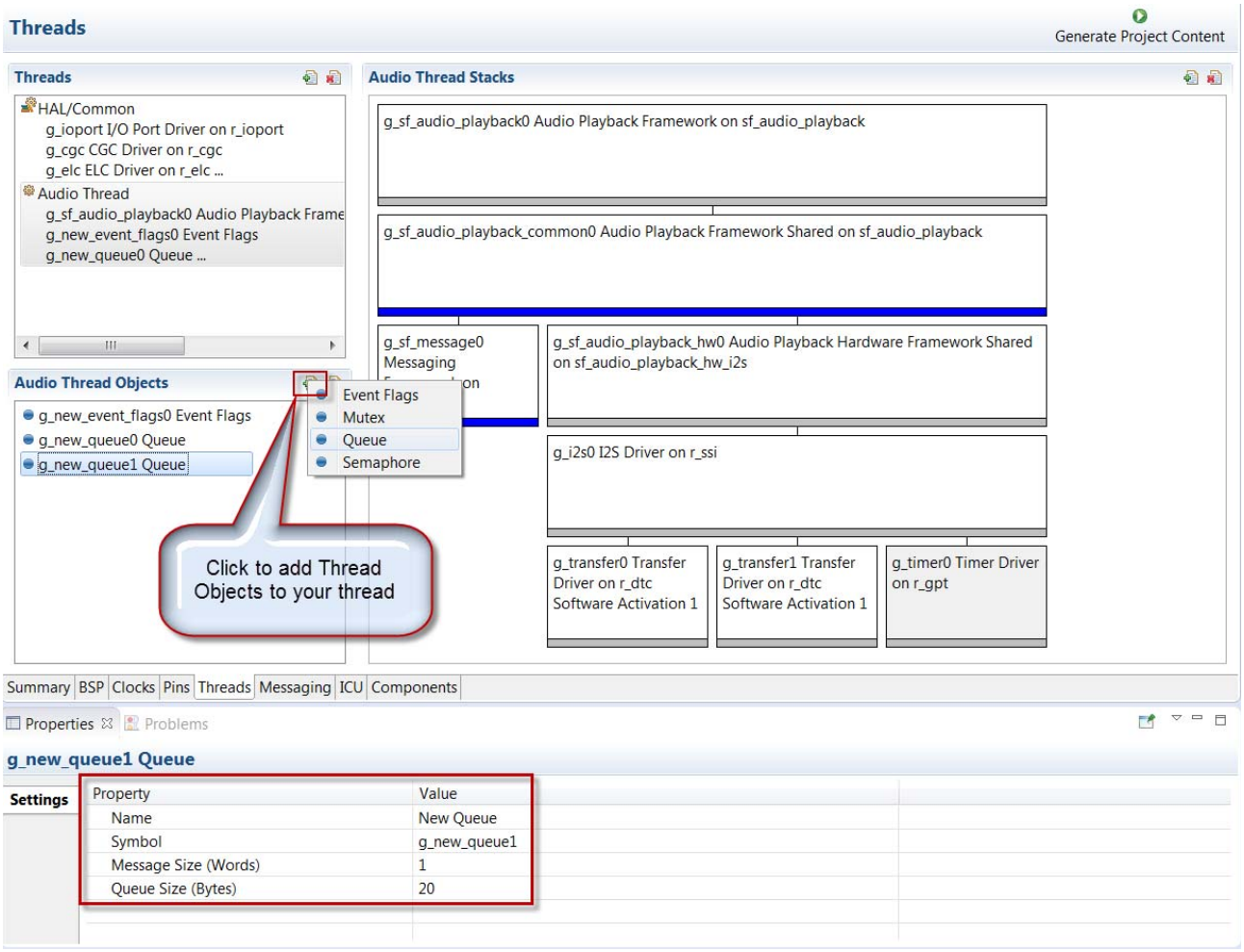


図 43: スレッドプロジェクトのプロパティの設定

それぞれのスレッドオブジェクトに固有の名前と記号が付されるよう、 [Name] と [Symbol] のエントリーを [Properties] ビューで更新します。

3.1.6.4 割り込みの設定

[Threads] タブにある [Properties] ビューを使用して割り込み優先順位を設定すると、割り込みを有効にすることができます。 [Threads] ペインで表示するスレッドを選択し、プロパティを編集します。

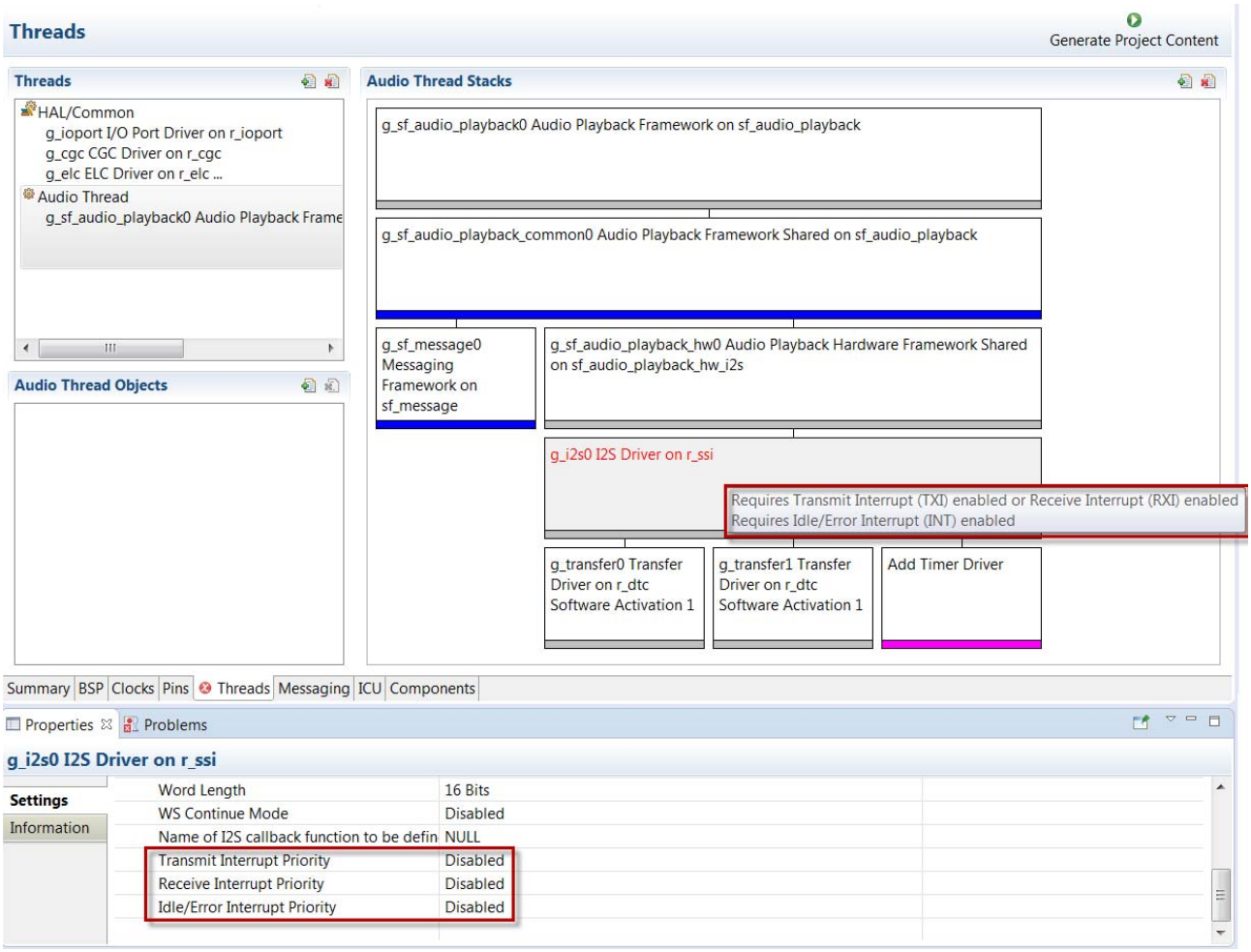


図 44: [Threads] タブで割り込みの設定

3.1.6.5 [Threads] タブでのコピーアンドペーストとカットアンドペースト

e2 studio v5.2.1 以降では、同じプロジェクト内のあるスレッドから他のスレッドに、モジュールスタックをコピーアンドペーストまたはカットアンドペーストすることができます。

以下の機能をサポートしています。

- コンテキストメニューまたは Ctrl-C で、選択したモジュールインスタンスとその下位に存在するすべてのモジュールインスタンスをクリップボードにコピーする
- コンテキストメニューまたは Ctrl-X で、選択したモジュールインスタンスとその下位に存在するすべてのモジュールインスタンスをクリップボードにカットする
- コンテキストメニューまたは Ctrl-V で、同じプロジェクトの任意のスレッド内に、クリップボードの内容を新しいスタックとしてペーストする

注：異なるプロジェクトのスレッドに対してクリップボードの内容を新しいスタックとしてペーストする操作はサポートしていません。

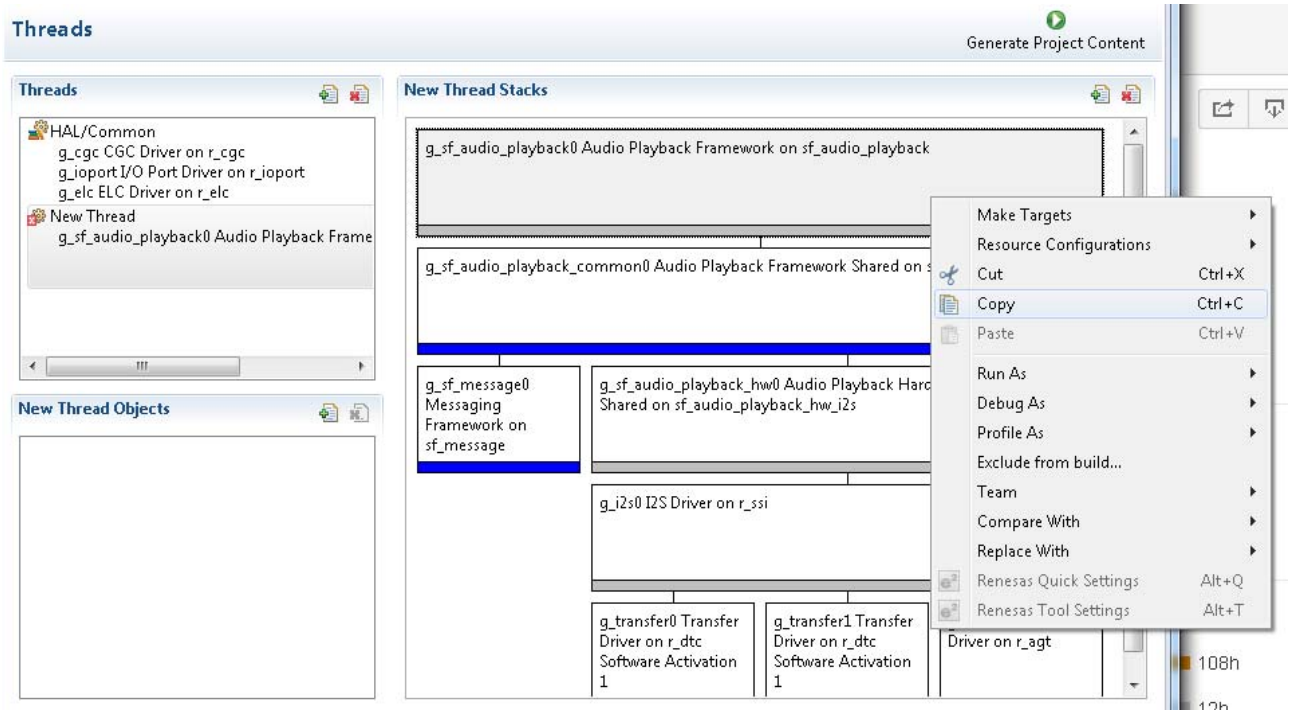


図 45: [Threads] タブでのコピーとペースト

この機能では、クリップボードの内容を既存のスタックに直接ペーストすることはできませんが、間接的にこれを実現する方法なら存在します。クリップボードの内容を受け入れ側のスレッドに新しいスタックとしてペーストし、新しいスタックが配置される（ピンク色のバーが付いた）モジュールインスタンスプレースホルダーの [Use] メニューから、新しいスタックの最上位のモジュールインスタンスを選択します。

3.1.7 SSP メッセージングフレームワークの設定

メッセージングフレームワークは ThreadX メッセージングキュー機能に拡張される、最も重要な SSP モジュールの 1 つです。これは、スレッドが相互にメッセージを交換して通信するメカニズムを提供します。メッセージングフレームワークは、事前設定またはユーザー設定されたイベントが発生した際にスレッドがメッセージを送信（パブリッシュ）または受信待ち（リッスン）できるようにします。スレッドは、特定のイベントクラスをサブスクライブしているすべてのスレッドが受信待ちおよび対応できるイベントクラスを付随させたメッセージをパブリッシュできます。特定のイベントクラスを受信待ちできるスレッドのリストは、そのイベントクラスのサブスクライバーリストと呼ばれます。

メッセージングフレームワークを使用するには、まず [Threads] タブでメッセージングフレームワークインスタンスを 1 つ追加する必要があります。メッセージングフレームワークは、HAL/ 共通スレッド以外のすべてのスレッドに追加できます。プロジェクト内のすべてのスレッドがこのインスタンスを使用して相互に通信できます。タッチパネルフレームワークやオーディオ再生フレームワークなどのモジュールではメッセージングフレームワークが必要で、以下のオーディオ再生フレームワークの例に示されているとおり、自動的に追加されます。プロジェクトにモジュールなどのスレッドが含まれている場合は、アプリケーションにスレ

ッドを追加しても、メッセージングフレームワークの他のインスタンスを追加する必要はありません。すべてのスレッドが 1 つのメッセージングフレームワークインスタンスを共有できます。

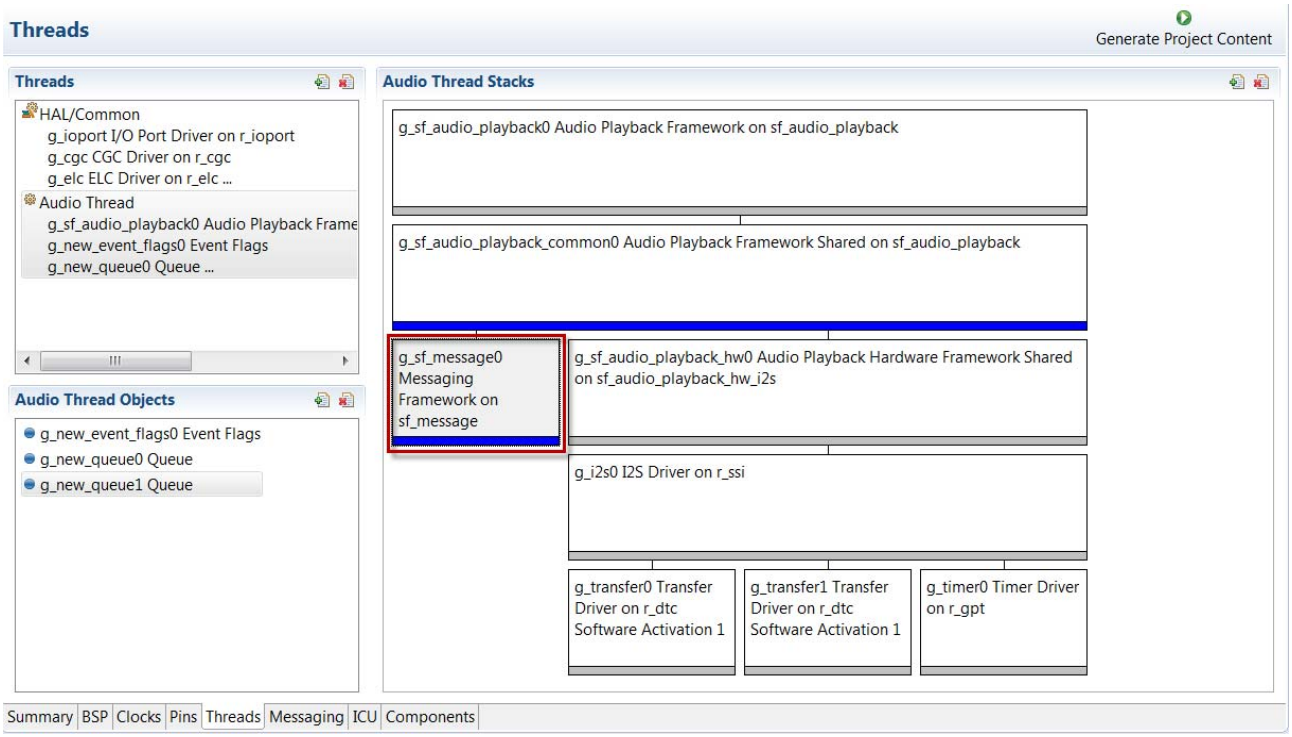


図 46: スレッドへのメッセージングモジュールの追加

[Threads] タブにメッセージングフレームワークインスタンスを追加したら、[Messaging] タブを使用して独自のイベントクラスとイベントを定義し、どのスレッドがどのイベントクラスを受信待ちするかを決定できます。 SSP には、タッチパネルおよびオーディオフレームワークモジュール用に事前定義されたイベントクラスとイベントが含まれています。これらのモジュールを追加した場合、事前定義されたイベントクラスとイベントが [Messaging] タブにも表示されます。オーディオ再生フレームワーク用に事前定義されたイベントクラスとイベントは、以下のとおりです。

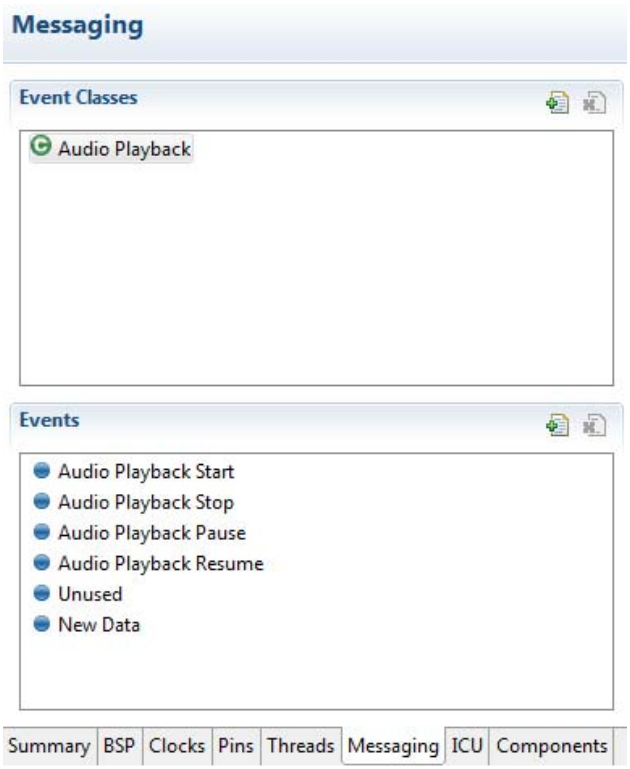


図 47: オーディオ再生フレームワーク用に事前定義されたイベントクラス

3.1.7.1 イベントクラスの追加

独自のユーザー定義イベントクラスをメッセージングシステムに追加するには、次の手順を実行します。

1) [Messaging] タブで、 [Event Class] ペインを選択し、ボタンを追加します。
2) イベントクラスに一意の名前を入力します。
3) [OK] をクリックします。

注 : ユーザー定義のイベントクラスは、イベントクラスアイコン右上の金色の四角形でマークされています。

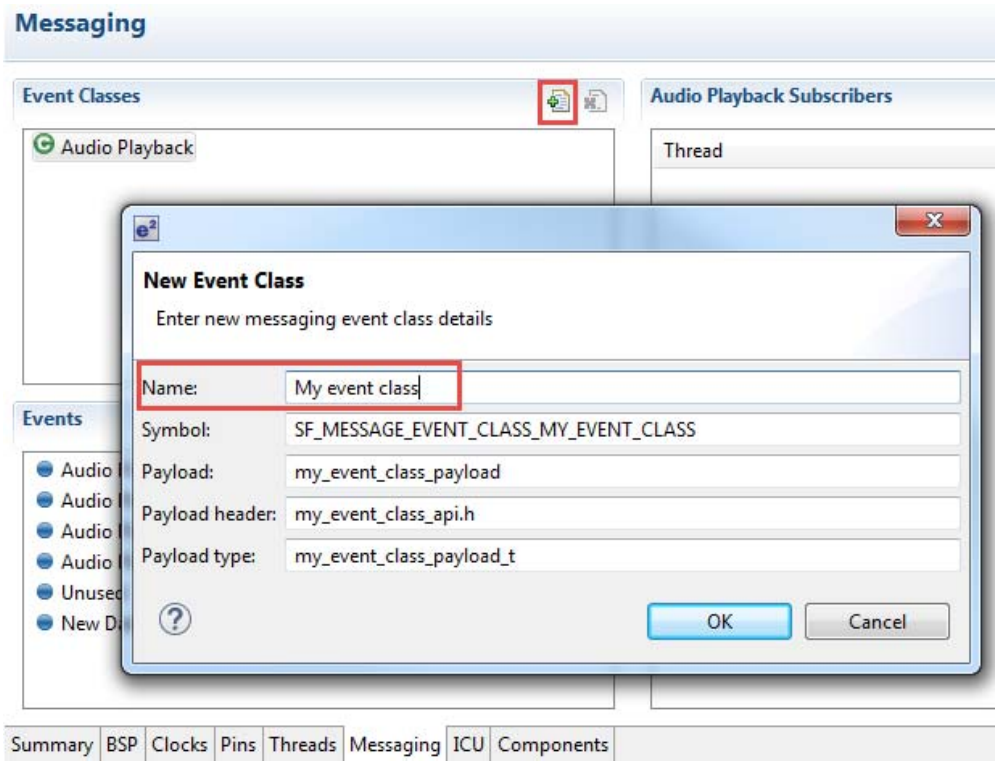


図 48:　メッセージング – イベントクラスの追加

3.1.7.2 イベントの追加

独自のユーザー定義イベントをメッセージングシステムに追加するには、次の手順を実行します。

1) [Messaging] タブで、 [Event] ペインを選択し、ボタンを追加します。
2) イベントクラスに一意の名前を入力します。
3) [OK] をクリックします。

注 : ユーザー定義のイベントは、イベントアイコン右上の金色の四角形でマークされています。

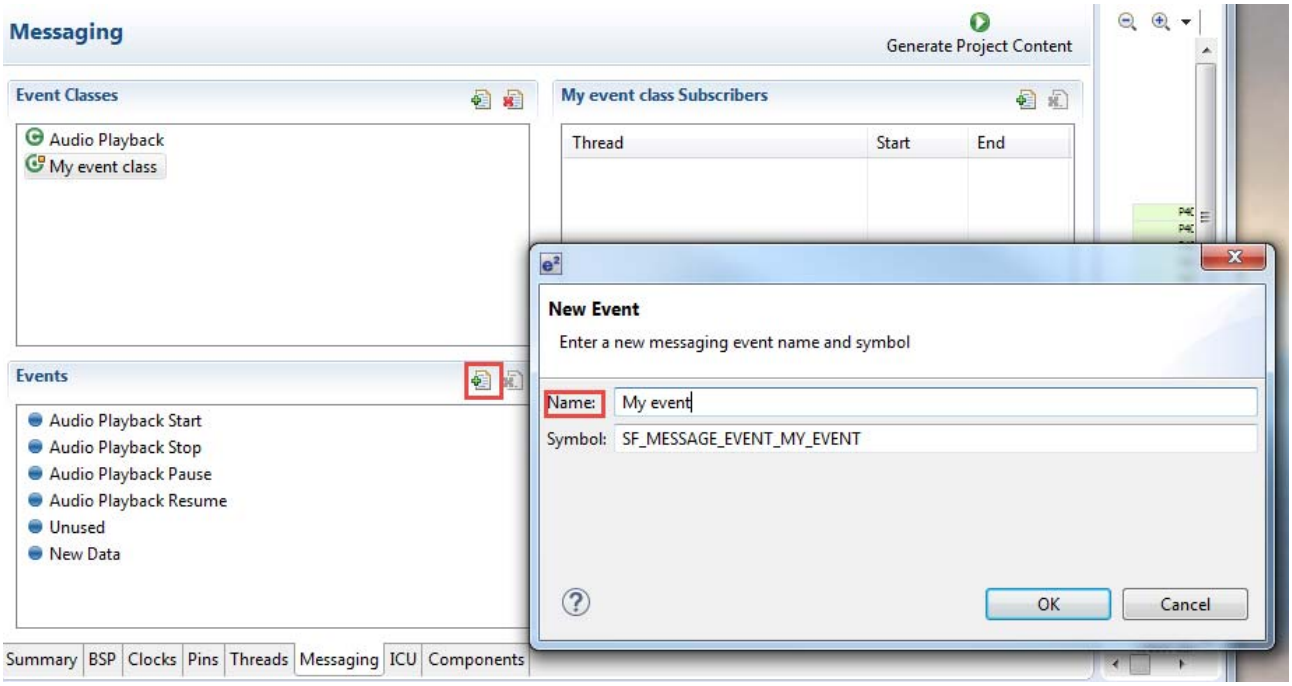


図 49:　メッセージング – イベントの追加

3.1.7.3 メッセージングサブスクライバーリストの設定

サブスクライバーリストでは、メッセージパブリッシャーからのメッセージを受信待ちするスレッドを選択します。パブリッシュスレッドと受信待ちスレッド間の接続は、イベントクラスを通じて確立されます。そのため、プロジェクト内の各イベントクラスについてサブスクライバーリストを定義する必要があります。サブスクライバーリスト内のすべてのスレッドが、選択されたイベントクラスに属するメッセージを受信待ちして対応できます。

以下では、[Threads] タブで 2 つのスレッドが定義され、そのうちの 1 つがオーディオ再生フレームワークを使用することが想定されています。

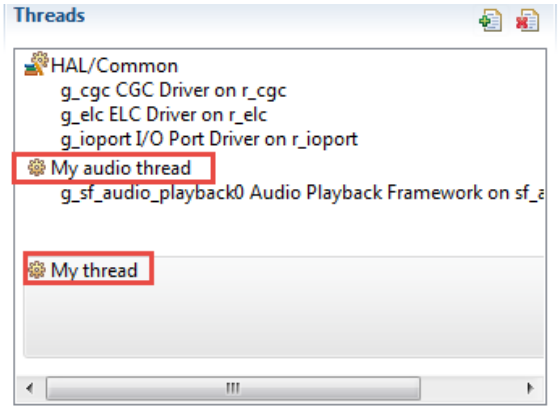


図 50: メッセージング – スレッドの例

イベントクラスのサブスクライバーリストを設定するには、次の手順を実行します。

1) [Messaging] タブの [Event Class] ペインでイベントクラスを選択します。 [Subscriber List] ペインは選択したイベントクラスの名前になります。
2) 追加アイコンをクリックすると、 [New Subscriber] ダイアログボックスが表示されます。
3) [Threads] ドロップダウンメニューから、サブスクライバーリストの追加するスレッドを選択します。
4) 開始点と終了点を選択して、インスタンス範囲を決定します。

   注 : 特定のイベントクラスのインスタンスが 1 つだけの場合は、開始値と終了値をデフォルト値 （0）のままにします。イベントクラスのインスタンスが 2 つ以上ある場合にインスタンス範囲を選択する方法については、メッセージングフレームワークユーザーガイドを参照してください。複数のチャネルでのオーディオストリーミングなど、同じイベントクラスを複数回使用するアプリケーションでは、イベントクラスのインスタンスが複数あると有用です。
5) 選択したイベントクラスのサブスクライバーリストに追加するそれぞれのスレッドについて、手順3と4を繰り返します。

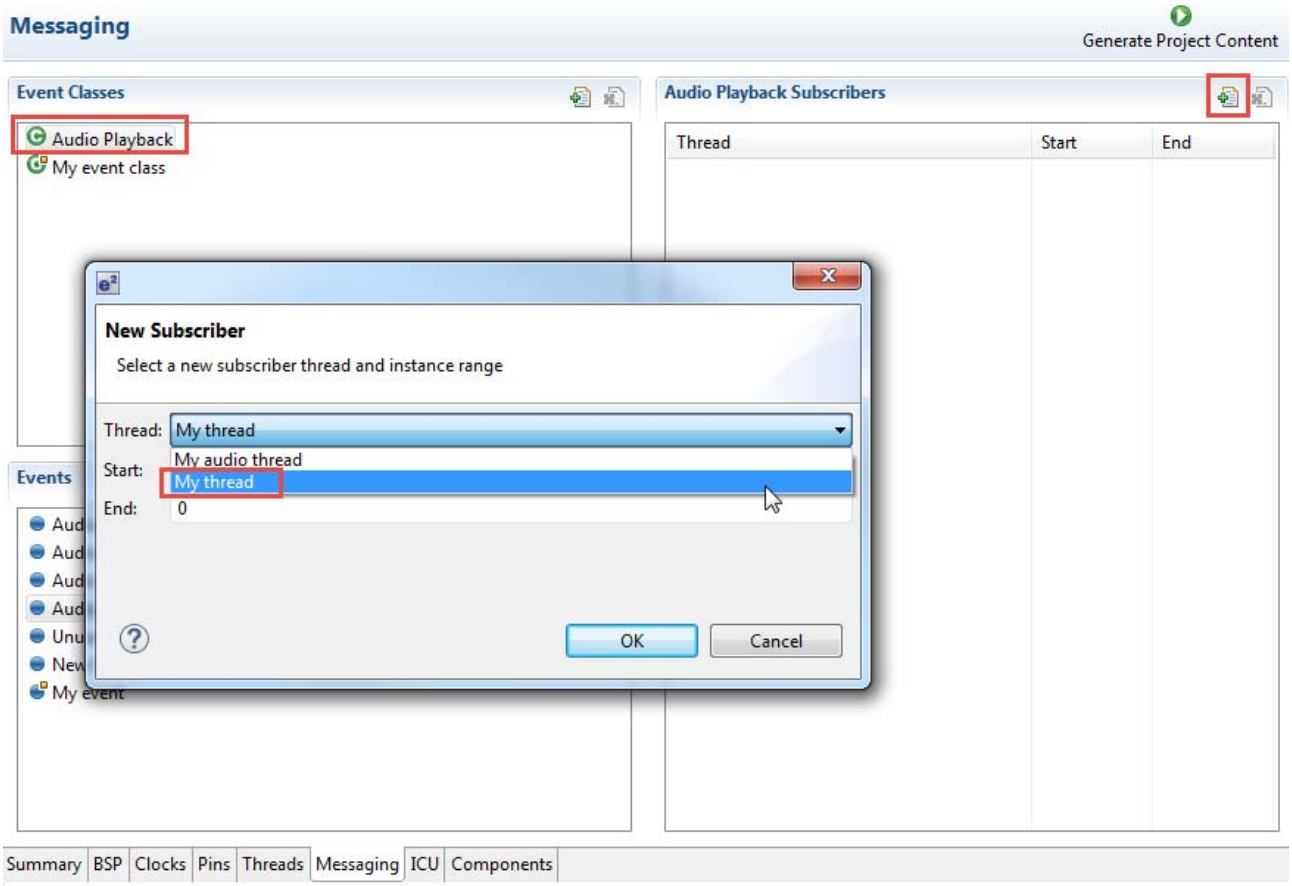


図 51:　メッセージング – サブスクライバーリストの設定

3.1.7.4 メッセージングフレームワーク用ファイルの生成

[Generate Project] ボタンをクリックすると、統合ソリューション開発環境によって以下の構成済みメッセージフレームワーク用ファイルが生成されます。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
synergy_cfg/ssp_cfg/framework/sf_message_port.h	イベントクラスとイベントの列挙を含みます。	はい
synergy_cfg/ssp_cfg/framework/sf_message_payloads.h	イベントクラスペイロードへのポインタを含みます。	はい
synergy_cfg/ssp_cfg/framework/sf_message_payloads.h	メッセージングフレームワークのコンパイラオプション	はい

3.1.8 プロジェクトに含まれるコンポーネントの確認

[Components] タブには、アプリケーションに含まれる個々のコンポーネントのリストが表示されます。すべての Synergy プロジェクトに共通するコンポーネントは、あらかじめ選択されています（たとえば、[BSP]>[Board-specific BSP, and HAL Drivers]>[all]>r_cgc).[Threads] タブで選択されたモジュールに必要なコンポーネントは、すべて自動的に含まれます。コンポーネントを追加するには、必要なコンポーネントの横にあるボックスをオンにし、除外するにはボックスをオフにします。

注: *[Components] タブで SSP モジュールコンポーネントの追加や削除を行うこともできますが、[Components] タブの主な目的は、プロジェクトに含まれているコンポーネントを確認することです。プロジェクト内の SSP モジュールを追加および削除する主な方法は* **[THREADS]** *タブであるため、SSP モジュールはいずれも、[Threads] タブと [Properties] ビューを使用して構成する必要があります。*

注: *[Components] タブで SSP モジュールコンポーネントを削除（選択解除）する場合は特に注意してください。コンポーネントの横にあるボックスのチェックを外すと、関連するモジュールとその構成すべてが [Threads] タブから直ちに削除されます。*

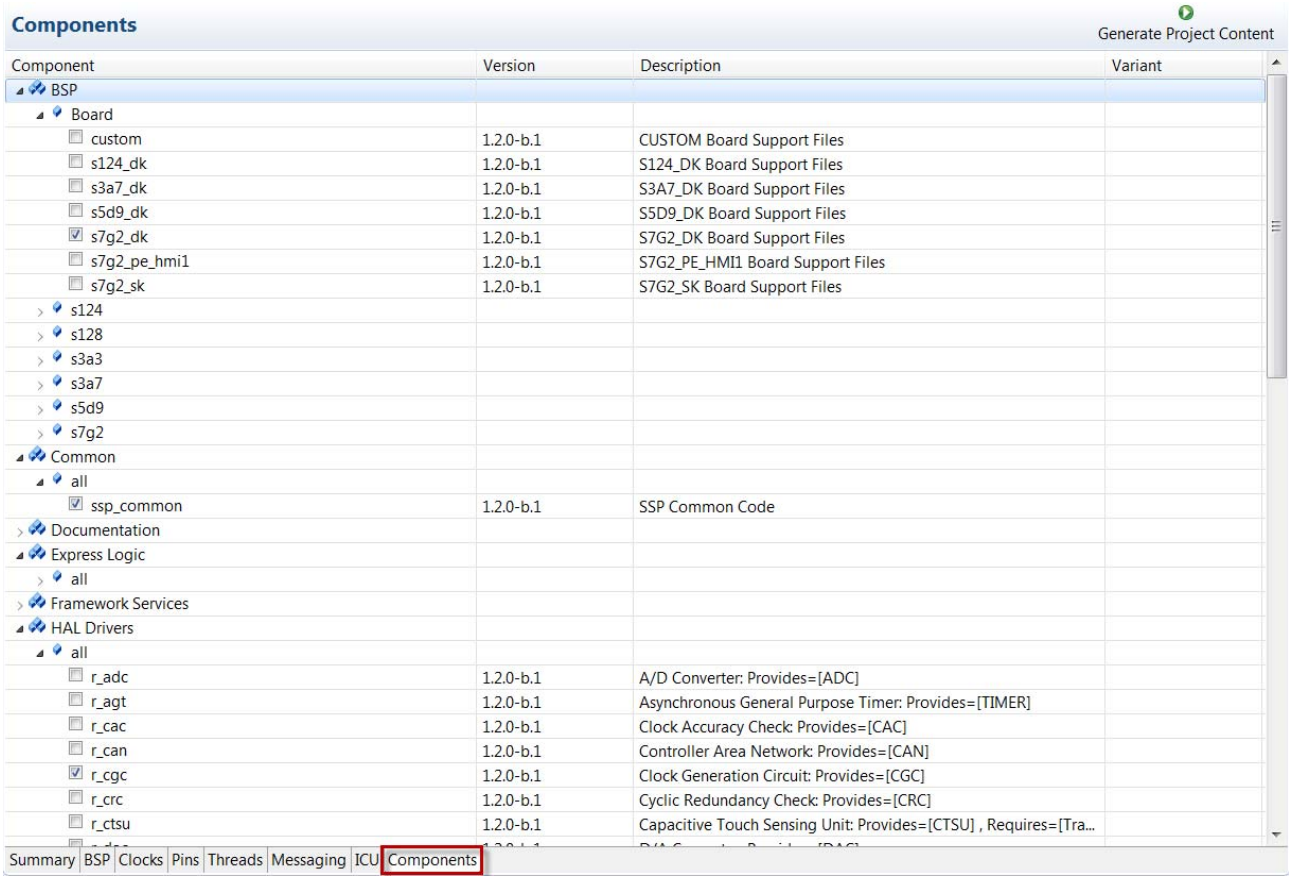


図 52: [Components] タブ

[Generate Project Content] ボタンを押すと、各コンポーネントの .c ファイルと .h ファイルがパックファイルから以下のフォルダにコピーされます。

- synergy/ssp/inc/bsp
- synergy/ssp/inc/driver
- synergy/ssp/inc/framework
- synergy/ssp/src/bsp
- synergy/ssp/src/driver
- synergy/ssp/src/framework

また統合ソリューション開発環境は、[Threads] タブの構成オプションが含まれた構成ファイルを synergy_cfg/ssp_cfg フォルダに作成します。

3.1.9 SSP の C++ サポート

このセクションは、SSP を使用して C++ 開発を行う際にユーザーガイドとして参照してください。このセクションでは、複数のツール構成、技術的な詳細、使用ガイドラインとさまざまなユースケース、SSP の C++ 環境の使用時に必要な考慮事項について説明します。

SSP モジュールは C++ アプリケーション環境をサポートしています。C++ アプリケーションで SSP モジュールを使用できるほか、ツールがサポートしている C++ 言語環境で SSP API を構成し使用することができます。C++ 開発用に、GCC と IAR の両コンパイラをサポートしています。

3.1.9.1 e2 studio の C++ 構成

サポートする e2 studio バージョンは v5.2.1 以降です。

- e2 studio 5.2.1 以降では、「Synergy C プロジェクト」と「Synergy C++ プロジェクト」をサポートする予定です。Synergy C++ プロジェクトは、C++ アプリケーションに必要なすべてのツールチェーンをサポートする予定です。
- GCC コンパイラの場合、ツールのデフォルトの C++ プログラミング言語規格は ISO 2011 C++（-std=c++11）です。
- IAR コンパイラの場合、ツールのデフォルトの C++ プログラミング言語規格は ISO/IEC 14882:2003（C++）です。
- C++ 標準ライブラリの場合、すべての標準関数ライブラリと、ツール（GCC または IAR）が対応している Standard Template Libraries（STL）などの Object Oriented Class ライブラリをサポートする予定です。

3.1.9.2 SSP における C++ の使用上の注意

以下の手順では、 e2 studio を使用して SSP C++ プロジェクトを作成する方法を詳しく説明します。

1) 新しい C++ プロジェクトを作成するには、e2 studio の [Synergy C++ Project] オプションを使用します。

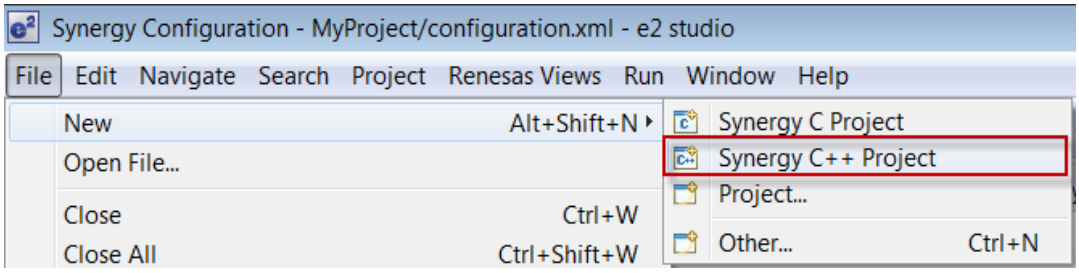


図 53: Synergy C++ プロジェクト

2) Synergy C++ プロジェクトテンプレートは、HAL エントリファイルまたはスレッドエントリファイルを .cpp ファイルとして作成するので、そこに最初の C++ コードを追加できます。また、外部 .cpp クラスファイルをプロジェクトに追加して使用することもできます。

注： v5.3.1 よりも前の e2 studio バージョンでは、生成されるエントリファイルの拡張子は .c であるため、次のように拡張子を 「.cpp」に手動で変更する必要があります。

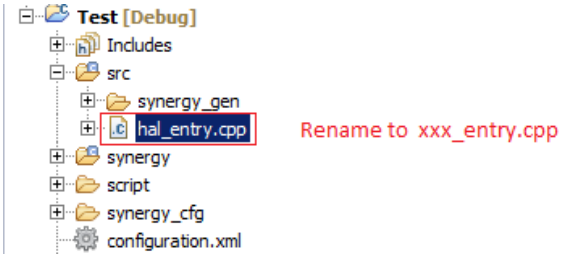


図 54: エントリファイル名の変更

3) ユーザーのアプリケーションに対応させるためにデフォルトの言語規格を任意の C++ 規格に変更する場合は、以下のように [Settings] プロパティから行います。

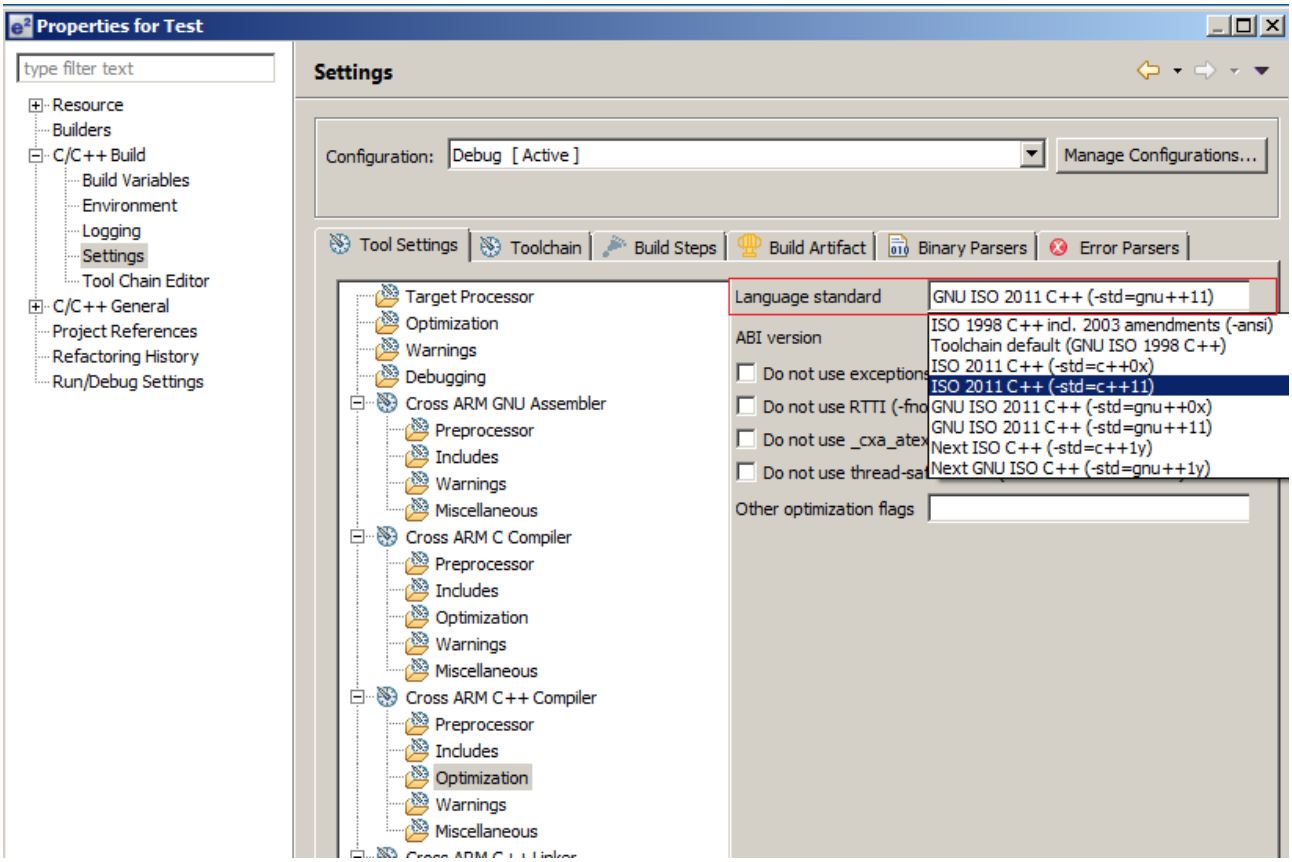


図 55: デフォルト言語の変更

4) このツールでは、.c ファイルであれば C コンパイラ、.cpp ファイルであれば C++ コンパイラで、それぞれコンパイルされます。C99 規格では、すべての SSP ドライバーとフレームワーク C コードは変更されていません。

3.1.9.3 C++ のユースケースと考慮事項

次の表では、C++ の一般的なユースケースと考慮が必要な互換性の問題をいくつか記載しています。

Number	ユースケース	考慮事項
1	SSP v1.2.0 以降と統合ソリューション開発環境 5.3.1 以降で Synergy C++ プロジェクトを作成	可能です。これは通常のユースケースです。

Number	ユースケース	考慮事項
2	SSP v1.1.z 以前で Synergy C++ プロジェクトを作成	このケースはサポートしていません。SSP の C++ サポートは SSP v1.2.0 で実装されました。それ以前のバージョンはいずれも C++ をサポートしておらず、「Synergy C++ プロジェクト」を使用することができません。
3	既存の C プロジェクトを Synergy C++ プロジェクトに移行する	C プロジェクトを C++ プロジェクトに移行させる必要がある場合は、統合ソリューション開発環境で個別の Synergy C++ プロジェクトを作成し、すべての構成とコードを新しいプロジェクトに手動で追加します。 場合によっては、C++ の規格に準拠させるために C コードに変更が必要になります。詳しくは、下記のケース 5 を参照してください。
4	既存の C++ アプリケーションやコードを Synergy C++ プロジェクトに移行または追加。	1. 新しい Synergy C++ プロジェクトを作成し、既存の C++ アプリケーションを追加します。 2.Synergy C++ プロジェクトが存在しており、そこに他の既存の C++ コードを追加する必要がある場合は、.cpp ファイルと .h ファイルをそのプロジェクトに追加します。 この場合、新しいコードが C++ で作成されているため、追加のコード変更は不要です。

Number	ユースケース	考慮事項
5	既存の C アプリケーションやコードを Synergy C++ プロジェクトに移行または追加。	準拠に関してよくみられる問題には、以下のものがあります。 - C++ での構造体の初期化における相違。C コードとは異なり、C++ ではすべてのメンバーが同じ順番で初期化されている必要があります。 - C++ の予約語、たとえば 「class」、「and」、「bool」、「catch」、「delete」、「explicit」、「mutable」、「namespace」、「new」、「operator」、「or」、「private」、「protected」、「friend」など。これらを変数名に使用することはできません。 - void* が他の型のポインタと混在している場合は、キャストを追加します。 - 無名共用体の使用は C++ ではサポートされていません。 ISO C と ISO C++ の間で互換性のない項目の一覧は、以下を参照してください。 http://david.tribble.com/text/cdiffs.htm
6	SSP v1.1.z 以前およびメッセージフレームワークを使用して作成された既存の C プロジェクトを SSP v1.2.0 に移行	メッセージフレームワークを使用した既存プロジェクトの場合、SSP v1.2.0 で使用するためには、マクロ SF_MESSAGE_CLASS をプロジェクトのプリプロセッサ設定 （[Settings] -> [Cross ARM C Compiler] - > [Preprocessor (-D)]）に追加する必要があります。 これは、C++ で作動するように sf_message モジュールが変更されたためです。

3.1.9.4 既知の問題

- 「Blinky」および 「Blinky with ThreadX」プロジェクトテンプレートは、ビルドプロセス時に追加の .c エントリファイルを作成し、C コンパイラを使用してビルドを実行するので、C++ プロジェクトでは使用できません。
- C++ の SSP コードでは現在、デフォルトで C++ の静的コンストラクタが追加されないため、ユーザーが自らアプリケーションコードに追加する必要があります。次のバージョンで SSP コードの一部として追加する予定です。

次のコードは、C++ アプリケーションで静的コンストラクタを使用できるようにするサンプルコードです。

宣言部に以下を追加します。

```
extern void __libc_init_array (void);
void R_BSP_WarmStart (bsp_warm_start_event_t event);
```

関数定義に以下を追加します。

```
void R_BSP_WarmStart (bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_PRE_C == event)
    {
        /* C runtime environment has not been setup so you cannot use globals. S
ystem clocks and pins are not setup. */
    }
    else if (BSP_WARM_START_POST_C == event)
    {
        /* Static Constructor for C++ */
#if defined(__GNUC__)
        __libc_init_array();
#elif defined(__ICCARM__)
          void const * pibase = __section_begin("SHT$$PREINIT_ARRAY");
          void const * ilimit = __section_end("SHT$$INIT_ARRAY");
          __call_ctors((__func_ptr *)pibase, (__func_ptr *)ilimit);
#endif

    }
}
```

3.1.10 アプリケーションの作成

モジュールとドライバーを追加して、[Threads] タブで設定パラメータを設定した後、モジュールおよびドライバーを呼び出すアプリケーションコードを追加できます。

***注：** 構成を確認するには、プロジェクトを一度ビルドして問題がないことを確かめてから、独自のアプリケーションコードを追加します。*

3.1.10.1 RTOS 独立アプリケーション

アプリケーションを作成するには、以下の手順を実行します。

1) [Threads] タブですべてのドライバーとモジュールを追加し、不足している割り込みやドライバーのような、ISDE によってフラグされたすべての依存関係を解決します。
2) [Properties] ビューでドライバーを構成します。
3) [Project Configuration] ビューで [Generate Project Content] ボタンを押します。
4) [Project Explorer] ビューで、src/hal_entry.c ファイルをダブルクリックして、ソースファイルを編集します。

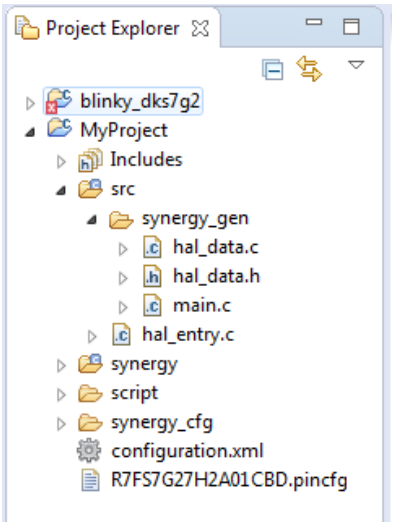


図 56: hal_entry.c

注 : アプリケーションで呼び出すドライバーに必要なすべての構成構造体は、src/synergy_gen/hal_data.c で初期化されます。

注 : ディレクトリ src/synergy_gen のファイルは変更しないでください。これらのファイルは [Generate Project Content] ボタンを押すたびに上書きされます。

5) アプリケーションコードをここに追加します。

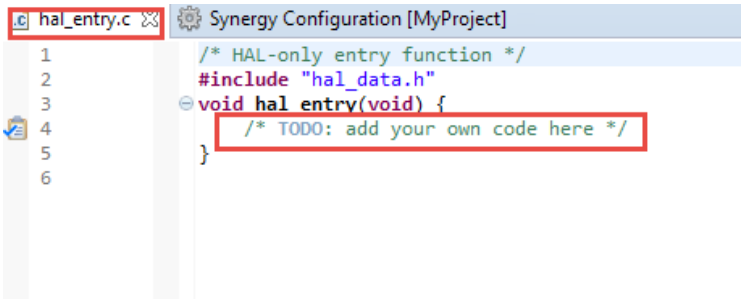


図 57: ユーザーコードを hal_entry.c に追加します。

1) [Project]>[Build Project] をクリックして、プロジェクトをビルドします。発生したビルドエラーを解決します。

次のチュートリアルでは、上記の手順を実行してアプリケーションコードを追加する方法について説明しています : チュートリアル : Using HAL Drivers - Programming the WDT

WDT の例は HAL レベルアプリケーションで、RTOS を使用しません。各モジュールのユーザーガイドには、hal_entry.c に追加できる基本的なアプリケーションコードも含まれます。

3.1.10.2 ThreadX アプリケーション

ThreadX を使用した RTOS 対応アプリケーションコードを記述するには、次の手順を実行します。

1) [Threads] タブを使用してスレッドを追加します。
2) このスレッドの一意の名前を [Properties] ビューで指定します。
3) このスレッドのすべてのドライバーとリソースを構成し、不足している割り込みやドライバーのような、ISDE によってフラグされたすべての依存関係を解決します。
4) スレッドオブジェクトを構成します。
5) 各スレッドオブジェクトの一意の名前を [Properties] ビューで指定します。
6) 必要に応じてさらにスレッドを追加し、手順 1 から 5 を繰り返します。
7) [Synergy Project Editor] で [Generate Project Content] ボタンを押します。
8) [Project Explorer] ビューで、src/my_thread_1_entry.c ファイルをダブルクリックして、ソースファイルを編集します。

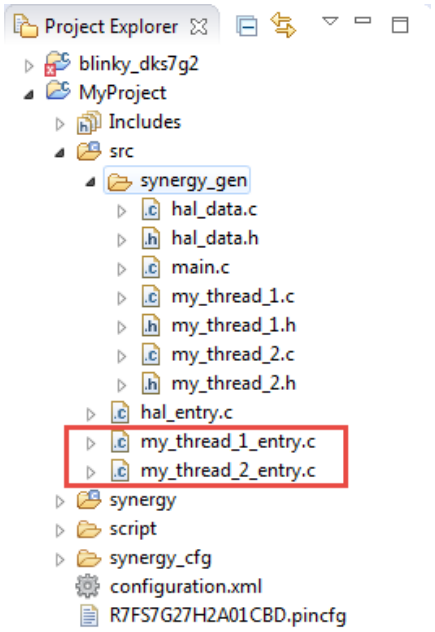


図 58: RTOS アプリケーション用 ISDE 生成ファイル

注:アプリケーションで呼び出すドライバーに必要なすべての構成構造体は、synergy_gen/my_thread_1.c と my_thread_2.c で初期化されます。

注 : ディレクトリ src/synergy_gen のファイルは変更しないでください。これらのファイルは [Generate Project Content] ボタンを押すたびに上書きされます。

9) アプリケーションコードをここに追加します。

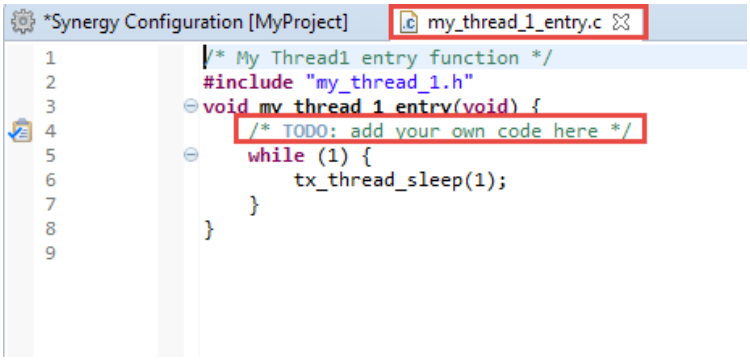


図 59: my_thread_1_entry.c へのユーザーコードの追加

10) 次のスレッドで手順 1 から 9 を繰り返します。

11) [Project]>[Build Project] をクリックして、プロジェクトをビルドします。発生したビルドエラーを解決します。

3.1.11 プロジェクトのデバッグ

プロジェクトの構築がエラーなしで完了すると、デバッガーを使ってアプリケーションをボードにダウンロードして、実行できます。

アプリケーションをデバッグするには、次の手順を実行します。

1) デバッグアイコンの横にあるドロップダウンメニューで **[Debug Configurations]** を選択します。

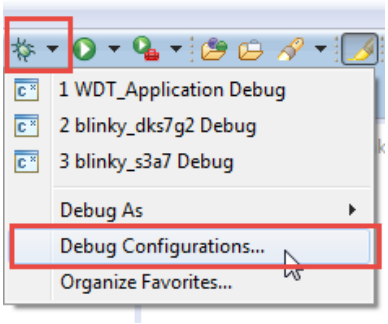


図 60: [Debug Configuration] ダイアログの呼び出し

2) [Debug Configuration] ビューで [MyProject Debug] としてリストされているプロジェクトをクリックします。

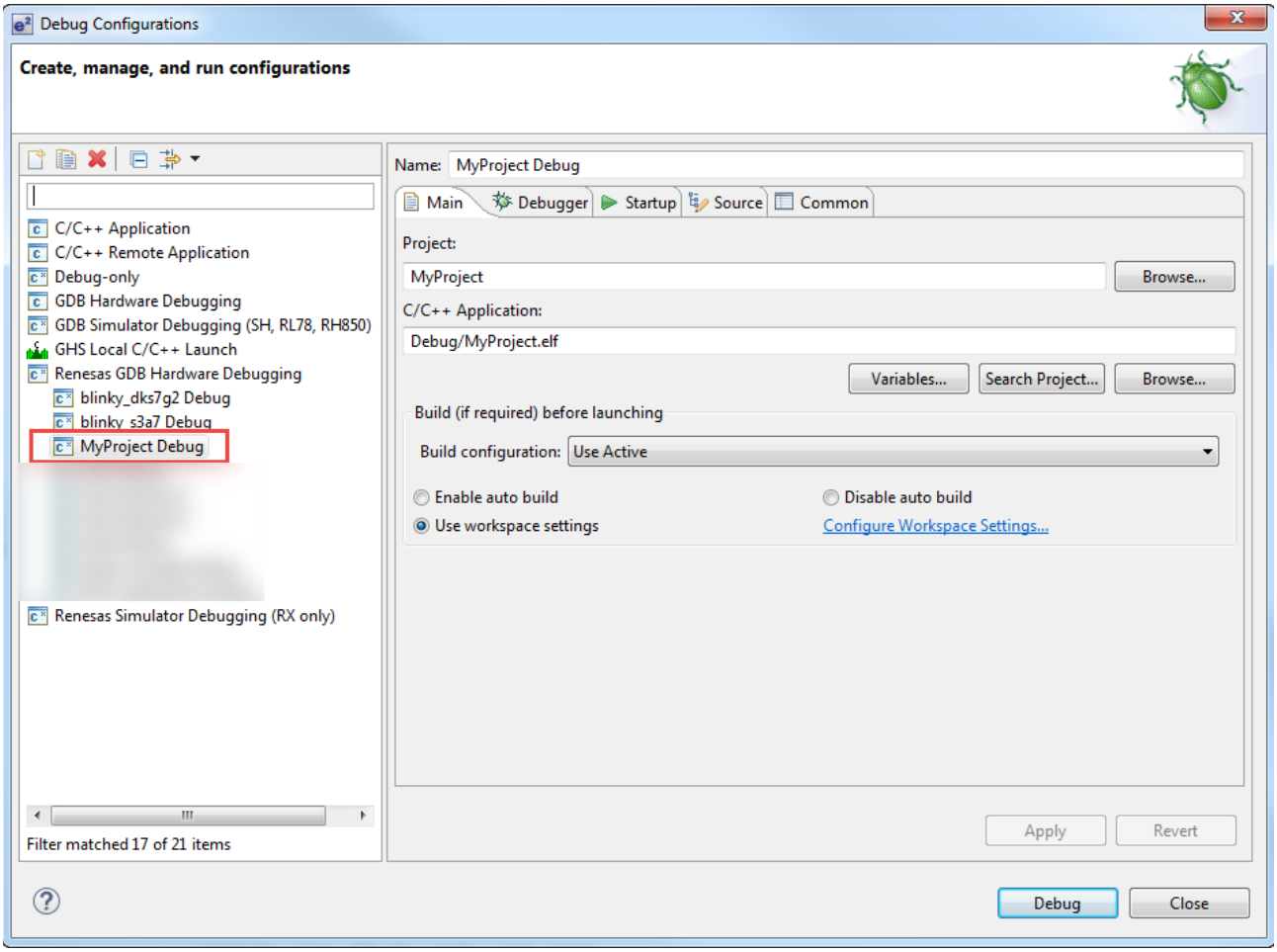


図 61: [Debug Configuration]

3) スタンドアロンの SEGGER J-Link デバッガまたは SEGGER J-Link On-Board（すべての Synergy DK および SK に含まれる）を使ってボードをパソコンに接続し、[Debug] をクリックします。

***注:** J-Link の使用方法とボードをパソコンに接続する方法については、Synergy キットに含まれるクイックスタートガイドを参照してください。*

3.1.11.1 Synergy プロジェクトでの RTOS リソースビューの使用

RTOS リソースビューは、ThreadX によるリソースの使用情報（システム情報やタスクまたはスレッド情報など）を表示します（プロジェクトで ThreadX RTOS を使用している場合）。

RTOS リソースビューを使用するには、デバッグセッションを開始します。次に、[RTOS Resources] ビューを開きます（[Renesas Views]>[Partner OS]>[RTOS Resources]）。

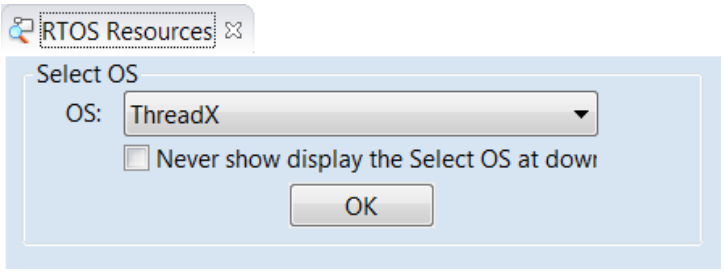


図 62: RTOS リソースビューの呼び出し

注: [Renesas Views]>[Renesas OS] の下にあるビューは使用しないでください。これらのビューでは Synergy はサポートされません。

[OK] をクリックし、OS として ThreadX を確認します。プログラムが実行されるまで、2 ～ 3 回再開します。その後、 [Suspend] をクリックします。 RTOS リソースに、スレッド関連の情報が表示されます。

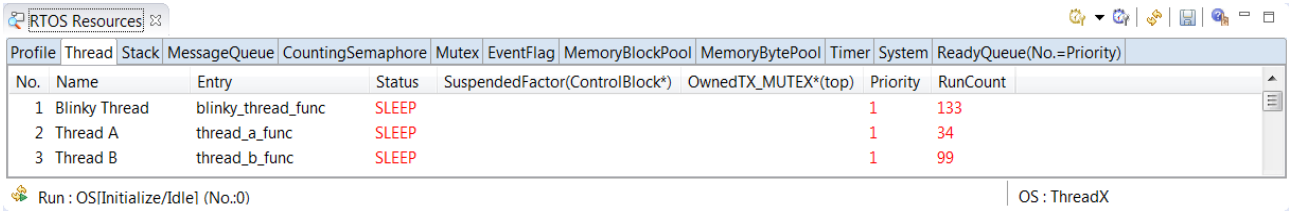


図 63: RTOS リソースビューで表示される RTOS 情報

ビュー内の各種タブをクリックすると、スタックの使用、メッセージキュー、セマフォ、ミューテックスなどの情報を確認できます。

RTOS リソースビューの詳細については、 e2 studio のヘルプコンテンツも参照してください。 [Help]>[Help Contents] をクリックしてアクセスできます。

3.1.11.2 スレッド対応デバッグビューの使用

ThreadX のデバッグ機能は、e2 studio GDB サーバーに完全に統合されていますので、Synergy デバイスでこの RTOS をデバッグする場合には、そちらを使用します。

つまり、対象がサスペンドすると、その ThreadX スレッドが e2 studio のデバッグビューに表示され、 RTOS タスクの 「コンテキスト」デバッグを実行することができます。

デバッグビューで選択内容を変更すると、表示されている他のすべてのデバッグウィンドウのコンテキスト（レジスタの値や変数など）も変化します。

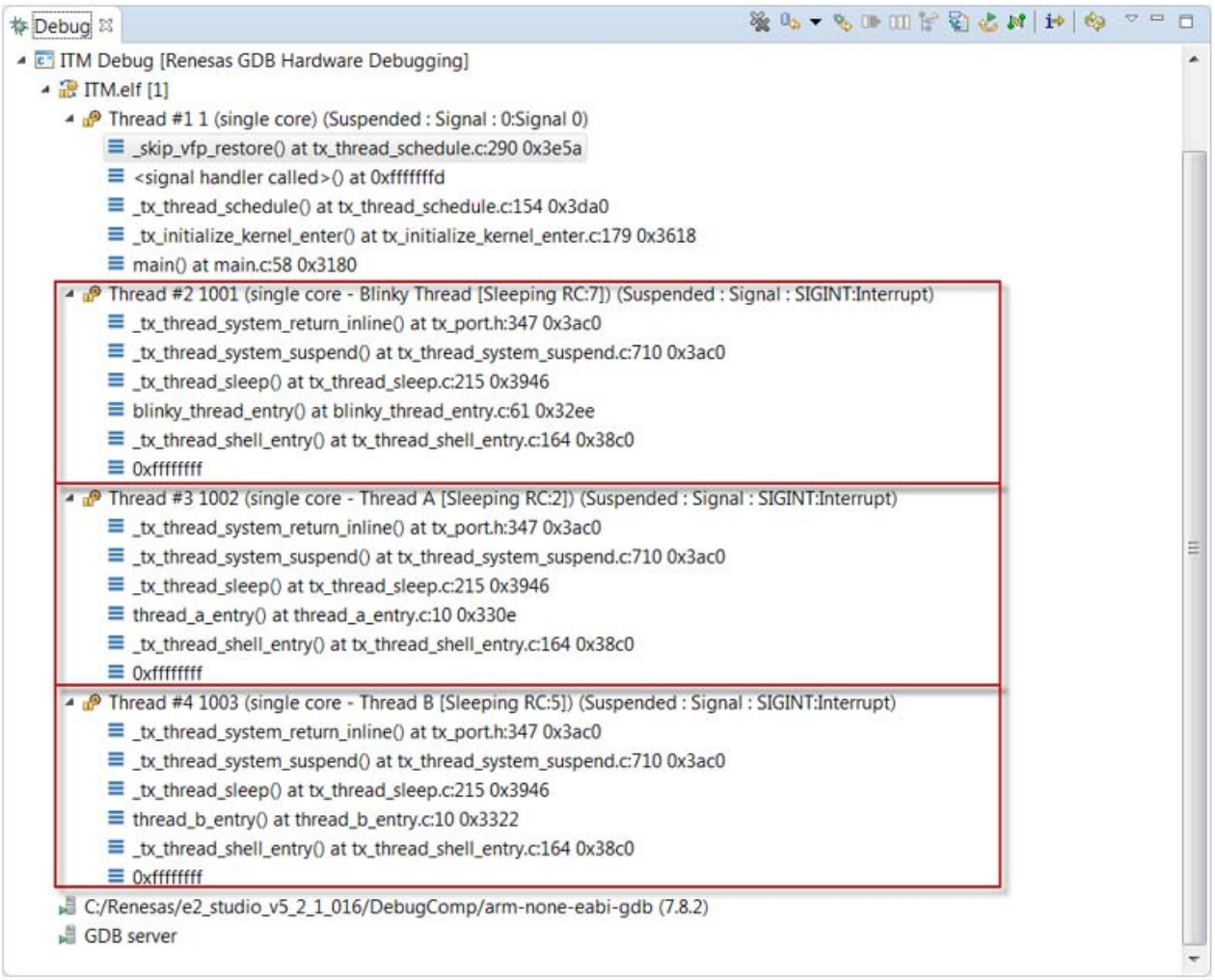


図 64: スレッド対応デバッグビュー

デバッグビューの詳細については、e2 studio のヘルプコンテンツも参照してください。[Help]>[Help Contents] をクリックしてアクセスできます。

デバッグビューのコントロールで問題が発生した場合は、デバッグ構成でこの機能をオフにすることができます。

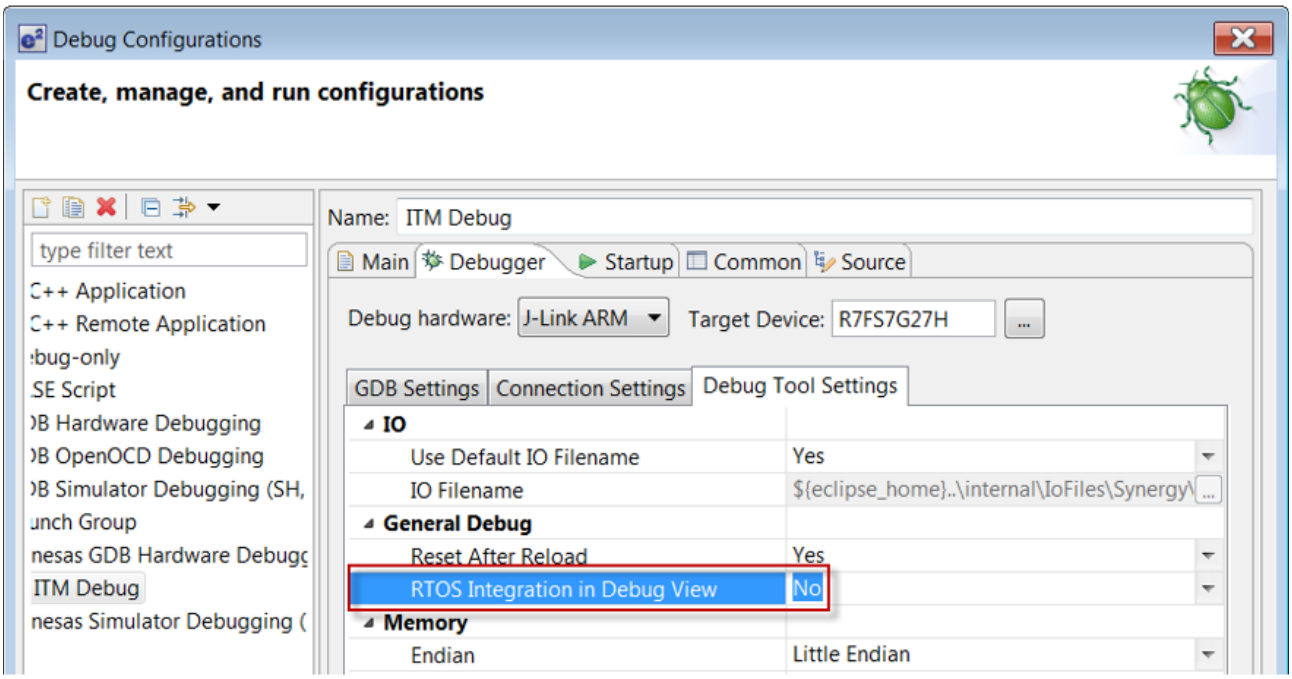


図 65: デバッグ設定におけるスレッド対応デバッグビューの構成

3.1.11.3 障害ステータスビューの使用

ハード障害が原因となってアプリケーションが停止した場合に役立つ障害ステータスデバッグビューが新たに実装されました（e2 studio v5.2.1 より）。

このビューには次の情報が表示されます。

- ハード障害の分類（FORCED、 VECTTBL など）
- 他のレジスタのビットフィールドで示されるさらに詳しい分類（UNALIGNED、 DIVBYZERO など）

図 66: ゼロ除算によるハード障害が発生した場合の障害ステータスビュー

このビューは、ハード障害が発生するまでは 「通常の」ステータスを示します。

***注:** Cortex-M0+ パーツは、ハード障害ステータスレジスタを含まず、表示は簡素です。*

さらに、GDB サーバーにクラッシュレジスタの解析機能が実装されているため、クラッシュが発生する前のコールスタックをデバッグビューに表示することができます。

3.1.11.4 ITM （Instrumentation Trace Macrocell）の使用

「Arm CoreSight ITM Live Trace Console」という新しいデバッグビューが、e2 studio v5.2.1 で実装されました。これにより、 Arm CoreSight ITM （Instrumentation Trace Macrocell）を使用することができます。

ユーザーは個々の有効化または無効化ビットを通じて、 32 ITM 出力ポートを構成できます。 ITM 出力は、Arm CoreSight ITM Live Trace Console の専用ポートタブに表示されます。これらのタブは、ポートを介して出力されるデータに応じて ASCII または未加工の 16 進数値として表示するよう構成できます。タブの表示はリアルタイムです。

ITM トレースの結果は参照用として自動的にトレースファイルに保存され、プログラムの実行がサスペンドされた場合に、「検視」のための完全な表示をユーザーに提供します。トレースファイルは、プロジェクトエクスプローラビューで確認できます。

注：トレースファイルの「タイムスタンプ」は 0 から始まり、実行開始から「CPU サイクル × クロックレート」単位で単純に増加していきます。ホスト PC の日時は反映しません。

ITM をサポートするには、デバッグ構成で接続を SWD モードに設定する必要があります。さらにトレースクロックをデバッガに指定して、通信を同期させる必要もあります（CPU クロックの半分を使用）。

3.1.12 Synergy プロジェクトでの TraceX TM の使用

注：Synergy プロジェクトで TraceX を使用する前に、Synergy Gallery から TraceX ソフトウェアをダウンロードしてインストールする必要があります。

TraceX はホストベースの分析ツールで、リアルタイムのシステムイベントをグラフィカルに表示できます。TraceX はターゲットデバイスのデータを収集して、検査と分析のためにデータを表示します。Synergy デバイスの TraceX バージョンが Synergy Gallery ウェブサイトからダウンロードでき、Synergy ライセンスと共に使用できます。

TraceX を使用するには、次の手順を実行します。

1) e2 studio で、ThreadX ソースコードをプロジェクトに追加するには、[Threads] タブでスレッドを選択し、[Stacks] ペインで緑色の「+」ボタンをクリックして、[Framework]>[RTOS]>[ThreadX Source] の順に選択します。
2) [Threads] タブを使用して、スレッドのプロパティウィンドウの TraceX を有効にします。TraceX バッファーのデフォルト名は g_tx_trace_buffer のままにします。

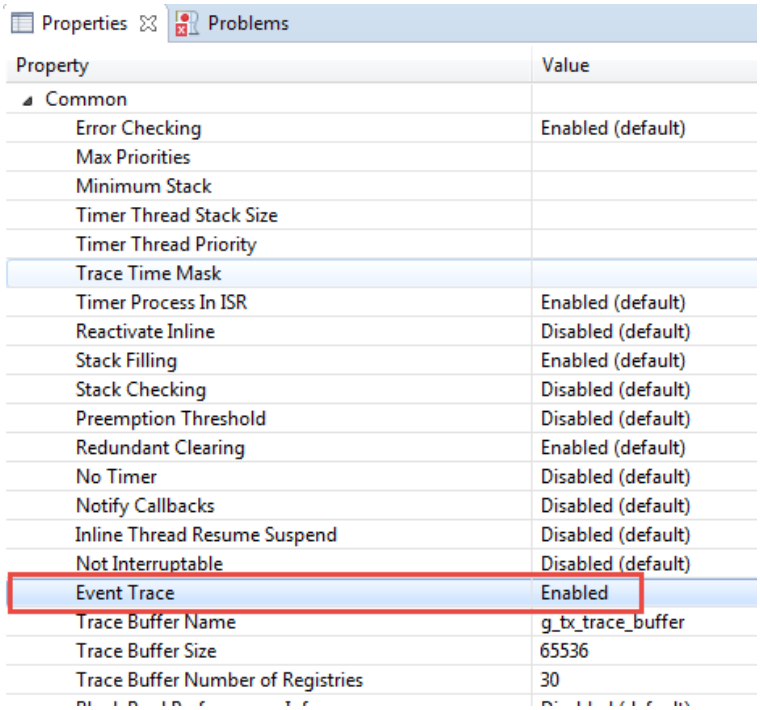


図 67: TraceX の設定

3) [Window]>[Preferences]>[C/C++]>[Renesas]>[TraceX]でTraceXアプリケーションへのパスを設定します。TraceX へのデフォルトパスは C:\Express_Logic\TraceX_5.2.0\TraceX.exe. です。

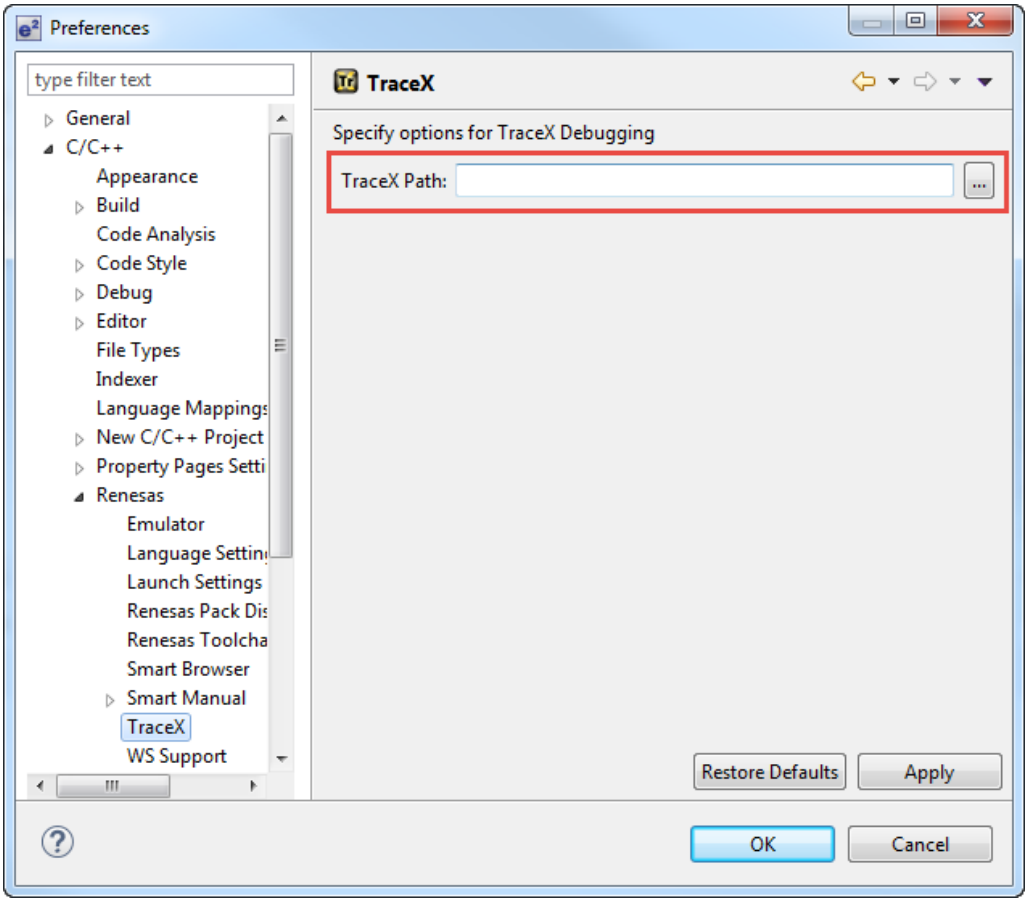


図 68: ISDE TraceX パス

4) プロジェクトをビルドします （[Project]>[Build All]）。
5) Synergy ターゲットボードを接続します。
6) デバッグセッションを開始します （[Run]>[Debug]）。
7) コードを実行して （[Run]>[Resume] （2 回））、 TraceX データの収集を開始します。
8) コードの実行をサスペンドします （[Run]>[Suspend]）。
9) [Run]>[TraceX] で [Launch TraceX Debugging] を選択します。

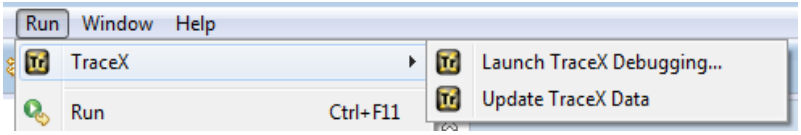


図 69: ISDE TraceX 起動

10) [TraceX Debugging] ウィンドウで、 [Buffer Start Address] に &g_tx_trace_buffer. を設定します。

11) [TraceX Debugging] ウィンドウで、 [Buffer Size (bytes)] を手順 1 のプロパティウィンドウで選択したバッファサイズに設定します。デフォルトは 65536 です。

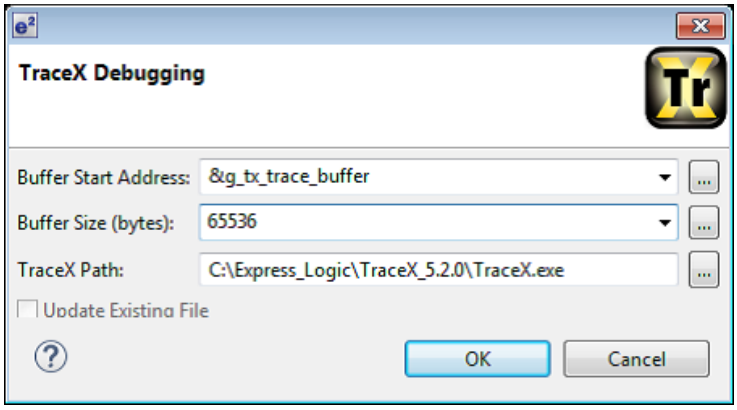


図 70: ISDE TraceX デバッグ

12) [OK] をクリックします。

13) TraceX で収集データを観察します。

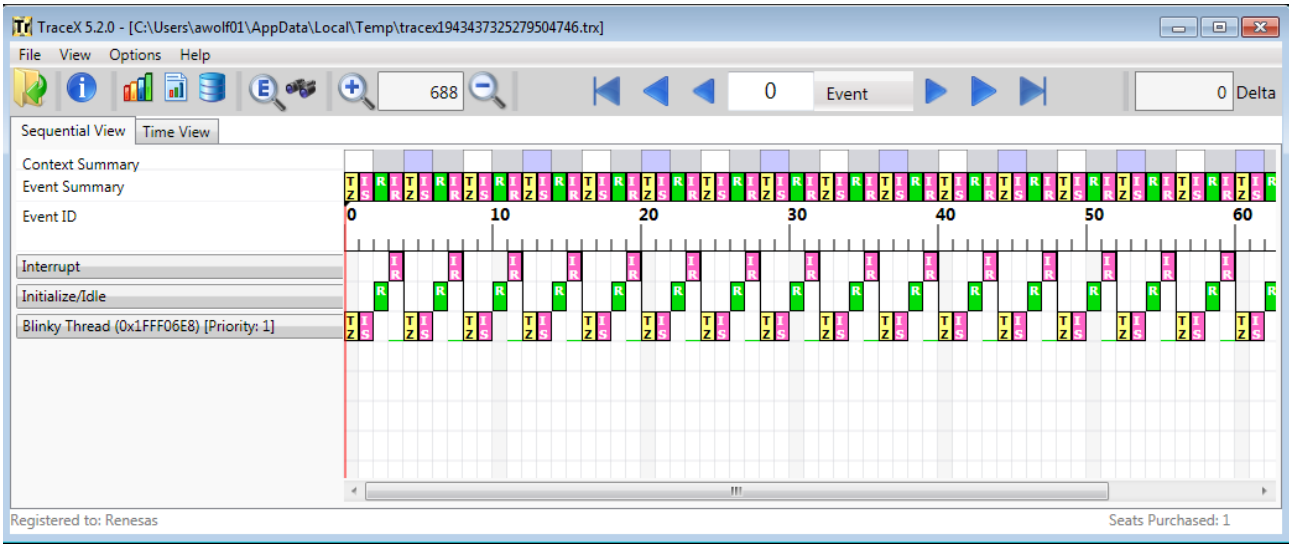


図 71: 統合ソリューション開発環境の TraceX デバッグ出力

14) 再度コードを実行し （[Run]>[Resume]）より多くの TraceX データを収集した後、コードの実行をサスペンドします （[Run]>[Suspend]）。

15) [Run]>[TraceX] で [Update TraceX Data] を選択します。

16) TraceX で収集された追加データを観察します。

Synergy Gallery にある TraceX ユーザーマニュアル （[Development Tools]>[TraceX]>[Documentation]>[TraceX User's Manual]）には、 TraceX の詳しい使用方法が記載されています。

3.1.13 ツールチェーン設定の変更

使用するツールチェーンを変更することが必要な場合があります。（たとえば、コンパイラの最適化レベルを変更したり、リンカーにライブラリを追加する場合などです。）そのような変更を行うには、ISDE で、プロジェクトが選択された状態でメニューから [Project] > [Renesas Tool Settings] を選択します。次のスクリーンショットは、GNU Arm ツールチェーンの設定ダイアログを示しています。このダイアログの外観は、使用するツールチェーンによって若干異なります。

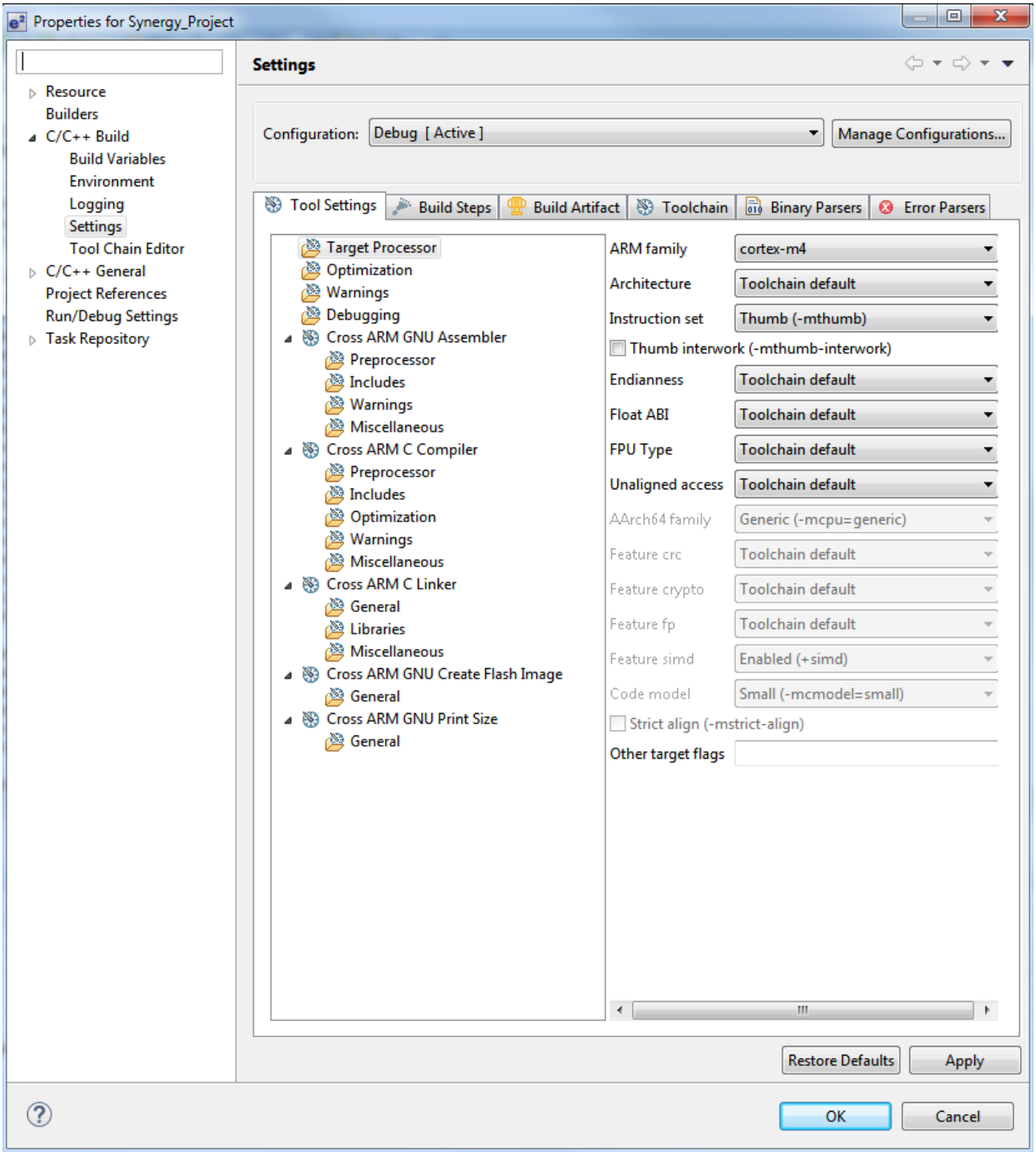


図 72: ISDE プロジェクトツールチェーン設定

設定のスコープはプロジェクトのスコープで、変更対象のプロジェクトのみで有効です。

リンカーの、各種メモリセクションの場所を制御する設定は、使用するデバイスに固有のスクリプトファイルに格納されます。作成されたスクリプトファイルはプロジェクトに含まれ、スクリプトフォルダーに格納されます （たとえば、 /script/S7G2.ld)。

3.1.14 カスタムボードパック （「カスタム BSP」）の作成

ほとんどの Synergy ユーザーは、標準の Renesas Synergy ボードのいずれか（たとえば、DK、SK）で開発を開始します。しかし、（全員ではないにしても）そのほとんどが、いずれは Synergy デバイスを基に独自のカスタムボードを開発するようになります。そのときには、自身のカスタムボードに基づいて新しい Synergy プロジェクトを作成できるよう、独自のカスタムボードパックを作成したいと思うことも出てくるでしょう。ここではその方法を説明します。

カスタムボードパックの作成に必要な手順は、次のとおりです。

1) 新しい Synergy プロジェクトを作成します （カスタムボードパックを作成するボードに最も近いボードを基にします）。
2) 自身のカスタムボードに準じて、 BSP、クロック、ピンの設定をカスタマイズします。
3) いくつかの主要ファイルを編集して、ボード名をカスタマイズします。
4) プロジェクトをビルドします （ビルドするカスタムボードに基づいたプロジェクトであることを確認します）。
5) User Pack Exporter を使用して、カスタムボードパックを作成します。

この時点で、新しいカスタムボードパックのテストとして、自身のカスタムボードを基に新しいプロジェクトを作成してみます。

詳しい手順

特定バージョンの SSP（SSP v1.2.0-b.1 など）のカスタムボードパックを作成するには、以下の手順に従います。

注: *本稿執筆時点では、 IAR Embedded Workbench® for Renesas Synergy™ にこの機能は存在しません。*

1) e2 studio v5.2.1（またはそれ以降）で、BSP プロジェクトテンプレートを使用して、ユーザーがインストールした SSP バージョン （SSP v1.2.0-b.1 など）のための新しい Synergy C プロジェクトを作成します。ボードには、利用可能なボードのうち、新しいカスタムボードに最も近いものを選択します。
2) プロジェクトを作成したら、プロジェクト内の既存のピン構成ファイルをコピーして、'MyCustomBoard.pincfg' などの任意の名前に変更し、カスタムピン構成ファイルの名前とします。
3) 次に、以下の手順に従い、[Pins] タブのピンコンフィギュレータを、新しいカスタムピン構成ファイルに切り替えます。
   a) 現在アクティブな .pincfg ファイルの [Generate Data] ボックスをオフにします。
   b) プルダウンメニューでカスタム pincfg ファイルに切り替え、[Generate Data] ボックスをオンにして、データ構造体名 「'g_bsp_pin_cfg'.」を入力します。
4) BSP、クロック、ピンのそれぞれのタブで、カスタムボードに合わせて設定をカスタマイズします。
5) [Pins]、[Clocks]、[BSP] の各タブでのカスタマイズを完了した後、[Generate Project Content] をクリックしてそれぞれのファイルをアップデートします。 [Save All] をクリックします。

6) ボードのフォルダ名 'synergy/board/<xxx>' を編集し、カスタムボード名に合わせます（'synergy/board/my_custom_board'. など）。

7) ファイル 'synergy_cfg/ssp_cfg/bsp/bsp_board_cfg.h' を編集して、次のように 4 行目をボードパスに置き換えます。

```
#include "../../../synergy/board/<xxx>/bsp.h"
```

上記を以下のように変更します。

```
#include "../../../synergy/board/my_custom_board/bsp.h"
```

[Save All] をクリックします。

8) 次の手順で、ファイル bsp_board_cfg.h のプロパティを「リード専用」に変更し、プロジェクトをビルドするときに上書きされないようにします。

a) bsp_board_cfg.h を右クリックして、[Properties] を選択します。

b) [Resource] をクリックします。

c) [Read-only] ボックスをオンにします。

d) [Apply]、[OK] の順にクリックします。

9) プロジェクトをビルドします。（この手順で、新しく作成したボードパックに基づくプロジェクトが正常にビルドされることを確認します。）

10) 新しいカスタムボードパックのエクスポート処理を開始するには、プロジェクトを右クリックして、**[Export Synergy User Pack]** を選択します。

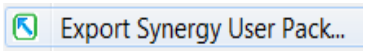


図 73: [Export Synergy User Pack] ウィザードの起動

[Export Synergy User Pack] ウィザードが起動します。

11) [Create a board pack] を選択します。

12) 新しいパックの場所を設定します。

- e2 studio を使用する場合は、場所として <e2_studio_base_dir>/internal/projectgen/arm/Packs/ を指定します。
- IAR Embedded Workbench for Synergy を使用する場合は、場所として <SSC_base_dir>/internal/projectgen/arm/Packs/ を指定します。

13) パックの詳細を指定するには、[PackVendor]、[Pack Name]、[Pack Version] の情報を入力します。パックのバージョンは、このパックとともに使用する SSP バージョンに一致させる必要があります。その結果、次のような画面になります。

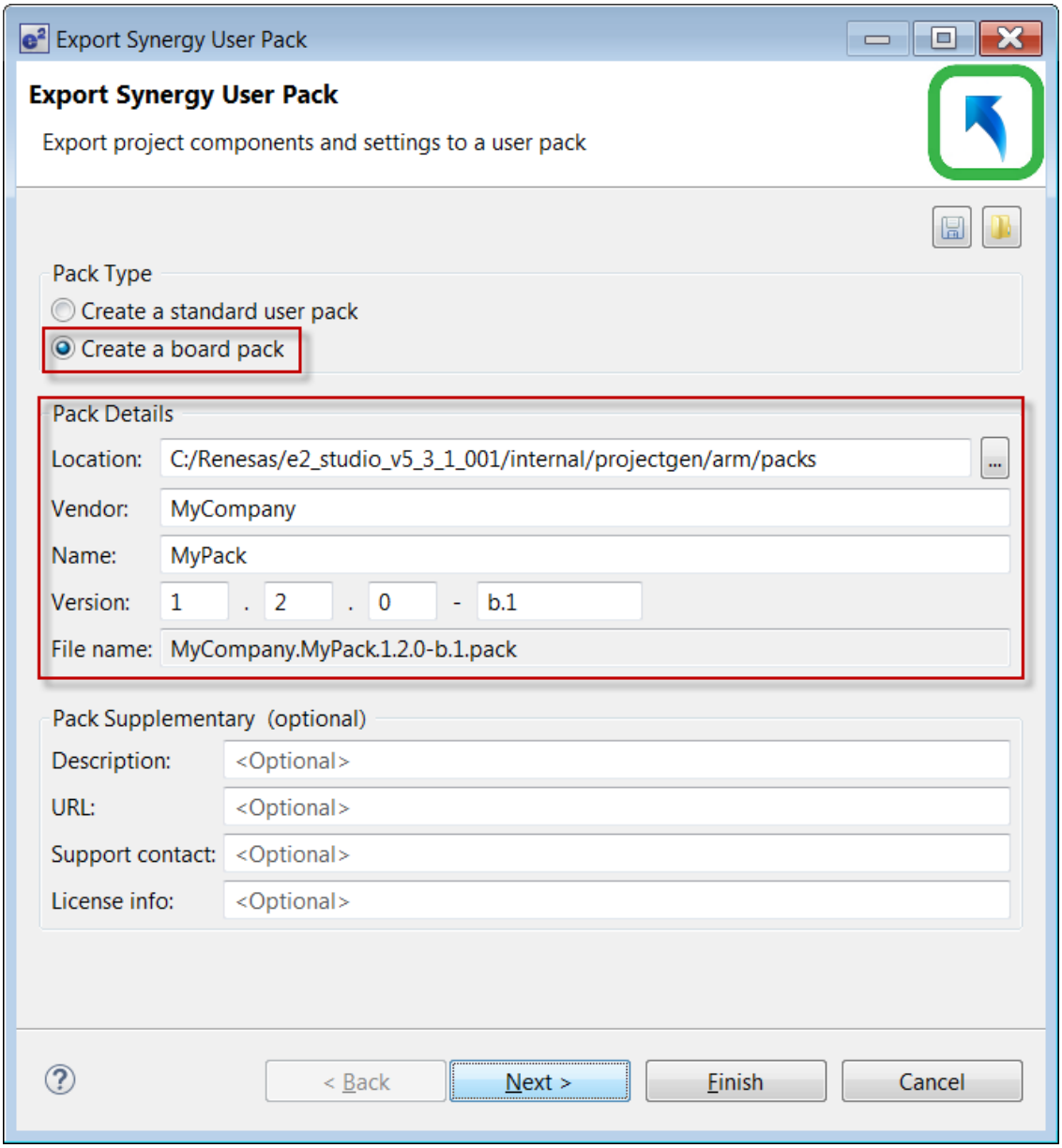


図 74: Synergy ユーザーパックのエクスポート画面 1

[Next] をクリックします。

14) 次の画面で、ボードコンポーネントの詳細を指定します。[Board Component Name]（[Sub-group] フィールド）、[Board Component Variant]（オプション）、短い [Board Component Description]（オプション）、[Board Component Version]（パックのバージョンと同じバージョンを使用）を入力します。

15) [Board Name] を入力します。

注：このボード名は、 synergy/board/<board_folder_name> にあるボードフォルダ名に一致させる必要があります。

その結果、次のような画面になります。

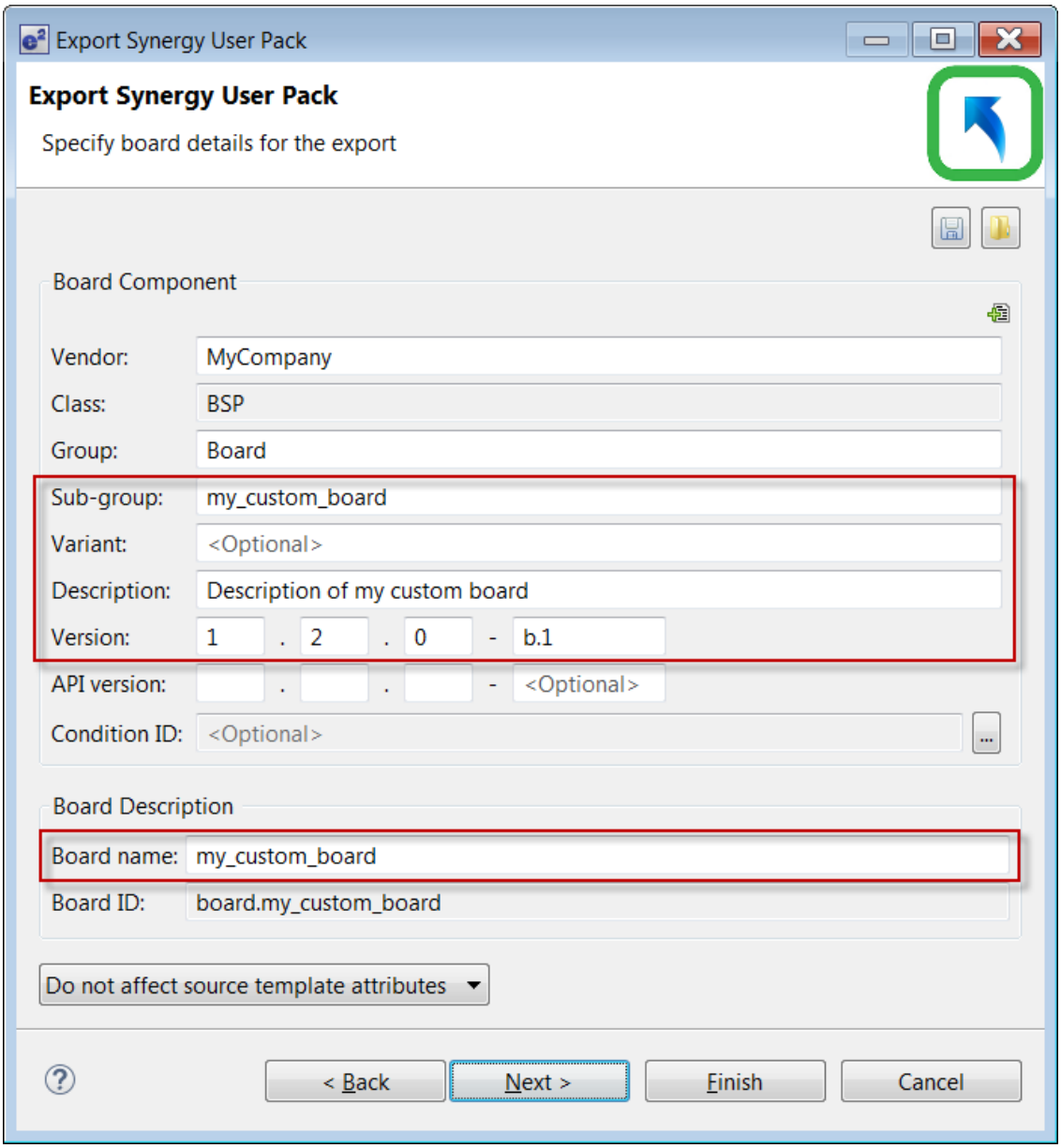


図 75: Synergy ユーザーパックのエクスポート画面 2

16) ウィザード画面右上の「+」アイコンを使用して、パックに含めるファイルを選択します。自身のカスタムピン構成ファイルとカスタムボードソースファイルをもれなく選択し、[Add] をクリックします。

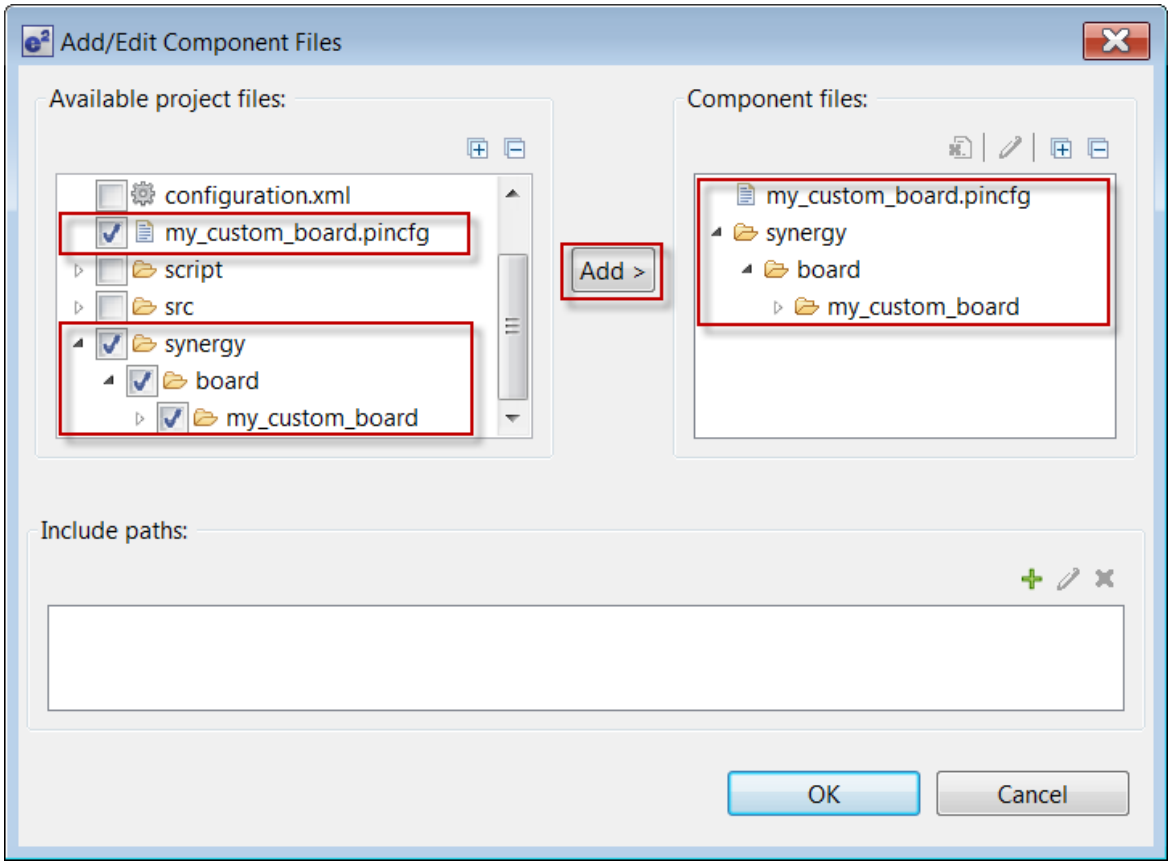


図 76:　コンポーネントファイルの追加と編集

17) [Component files] ペインで *.pincfg ファイルを選択し、鉛筆アイコンをクリックしてファイルプロパティを次のように編集します。

- [Category] プロパティを source に変更します。
- [Attribute] プロパティを template に変更します。

[OK] をクリックします。

18) [OK]、[Finish] をクリックすると、指定した場所にカスタムボードパックが作成されます。

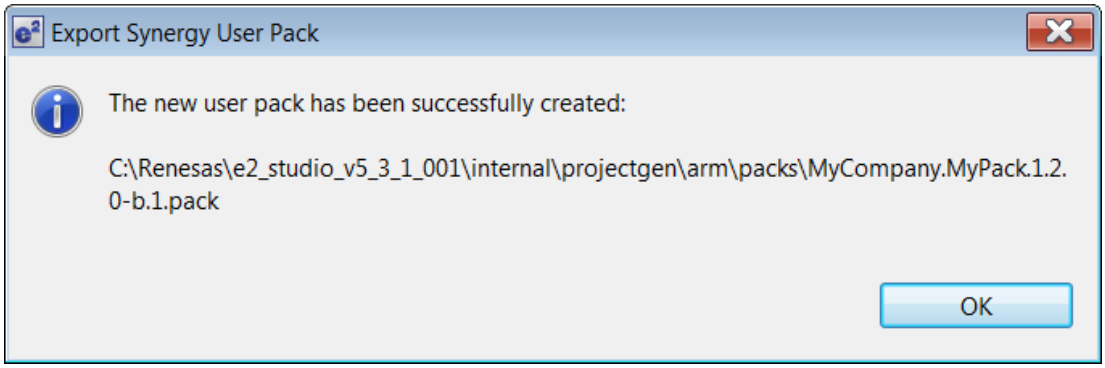


図 77:　正常に作成されたカスタムボードパック

新しいカスタムボードパックをテストするには、新しい Synergy C プロジェクトを作成し、ボードを選択する際に、リストから自身のカスタムボードを選択します。

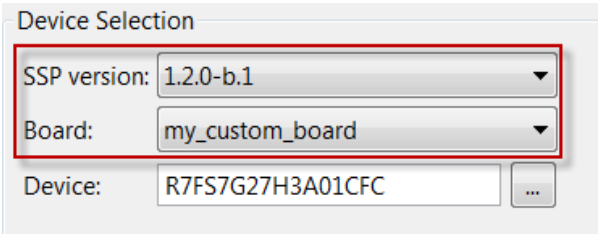


図 78:　カスタムボードに基づいた新しいプロジェクトの作成

プロジェクトを作成したら、Synergy Project Editor を開き、BSP、クロック、ピンの各タブでカスタマイズの内容を確認します。また、コンポーネントのタブも確認します。自身のカスタムボードが、BSP のボードコンポーネントとして選択されているはずです。コンポーネントの上にマウスカーソルを合わせると、カスタムボードパックファイルが表示されます。

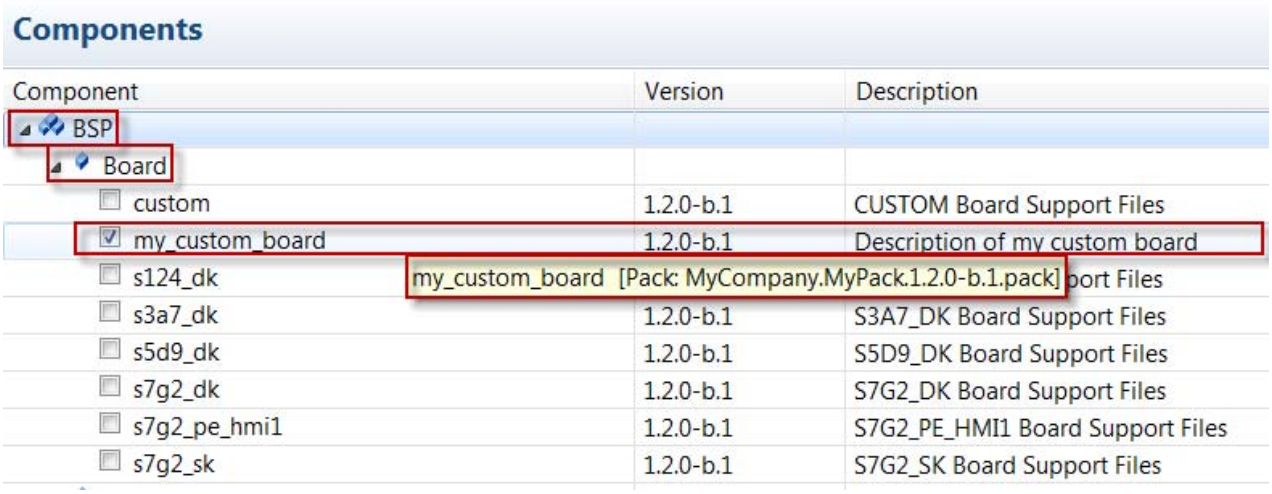


図 79: [Components] タブでのカスタムボードパックの表示

アプリケーションコードを追加し、カスタムボード上でプロジェクトのビルドとデバッグを行うことができるようになりました。

3.1.15 カスタムユーザーパックの作成

[Export Synergy User Pack] ウィザードでは、非 SSP（つまりカスタム）コンポーネント、RTOS スレッド、SSP コンポーネント設定のような項目を含むカスタムユーザーパックを作成することもできます。

次の要素はエクスポートできません。

- SSP コンポーネントと SSP ソースファイル（たとえば、<Project>\synergy\ssp）
- SSP 生成ファイル（たとえば、<Project>\synergy_cfg\ssp_cfg)

カスタムユーザーパックを作成するには、カスタムコンポーネントを含むプロジェクトで右クリックし、Synergy User Pack Exporter を呼び出します。

最初の画面で、パックの種類として [Create a standard user pack] を選択し、パックの詳細を入力します。[Next] をクリックします。

2 番目の画面で、エクスポートするプロジェクトコンポーネントを選択（または新規作成）し、コンポーネントごとにエクスポートするスレッド構成を選択します。 [Next] をクリックします。

3 番目の画面で、エクスポートする既存のパック条件を選択するか（オプション）、新しい条件を作成するか（オプション）、それらの両方を行います。[Finish] をクリックすると、指定の場所にカスタムユーザーパックが作成されます。

e2 studio はカスタムユーザーパックを読み取り、パック内のコンポーネントを抽出して、 [Components] タブにコンポーネントを表示します。 [Components] タブでいずれかのコンポーネントを選択すると、それがプロジェクトに追加されます。

3.1.16 e2 studio ISDE 使用上の注意

3.1.16.1 SSP リリースの処理

Synergy Gallery ウェブサイトでは SSP のさまざまなリリースパックを公開しています。ダウンロードしたリリースパックは、 e2 studio の特定のインスタンスにインストールすることができます。

***注：** 原則として、複数の SSP バージョンをインストールするときは、リリースされた順にインストールしてください。*

SSP パックには 2 種類あります。

1) X.Y.0 Gold – 完全な SQA を含む生産リリース。 X.Y.0 パックだけが含まれます。例 : SSP v1.1.0
2) X.Y.Z– リリースされた X.Y.0 バージョン （X= メジャー、 Y= マイナー、 Z= パッチ）に基づいたパッチリリース。 X.Y.Z パックで追加または更新されたモジュールだけが含まれます （完全な X.Y.0 パックは含まれません）。

X.Y.0 生産リリースに加えて、複数の X.Y.Z パッチリリースをインストールすることができます。

たとえば、次の SSP バージョンをインストールすることができます。

- SSP v1.1.0: 生産リリース
- SSP v1.1.1: 最初のパッチリリース
- SSP v1.1.2: 2 回目のパッチリリース
- SSP v1.1.3: 3 回目のパッチリリース

***注：** SSP パッチリリースインストーラにより以前のパッチリリースや生産リリースがインストールされることもあります。詳細については、SSP インストールログを参照してください。*

Synergy Project Generator または Synergy Project Editor の [BSP] タブで SSP バージョン（SSP v1.1.3 など）を選択すると、 e2 studio 統合ソリューション開発環境は、プロジェクトに含めるために対応する SSP コンポーネント（入手できる場合）を自動的に選択します。この結果は、[Components] タブでモジュールをルックアップすると確認できます。

同一プロジェクトで使用できるのは、メジャーバージョンとマイナーバージョンが同じコンポーネントのみです。たとえば、次の SSP バージョンをインストールしたとします。

- SSP v1.1.3
- SSP v1.2.0
- SSP v1.2.1

このシナリオでは、SSP v1.2.1 プロジェクトを使用している場合、SSP v1.2.1 と SSP v1.2.0 両方のコンポーネントが [Components] タブに表示され、そのプロジェクトで使用できます。ただし、 SSP v1.1.3 コンポーネントはフィルタ処理によって除外され、 [Components] タブには表示されません。

選択した SSP バージョンで必要な SSP コンポーネントが利用できない場合は、 e2 studio 統合ソリューション開発環境により、利用可能なもののなかから最もバージョンの新しい SSP コンポーネントが選択されます。

[Components] タブの SSP コンポーネントが、選択された SSP バージョン （「パッチ」バージョンを含む）と正確に一致しない場合、 [BSP] タブの [SSP Version] フィールドの横に警告アイコンが表示されます。

次の例では、SSP v1.1.3 が選択されています。しかし、SSP v1.1.3 はパッチリリースです。このため、プロジェクトには以前のパッチリリース（SSP v1.1.1 など）や生産リリース（SSP v1.1.0）に由来し、SSP v1.1.3 には存在しないモジュールが含まれている可能性があります。この警告は情報提供のみを目的としており、ユーザーのプロジェクトがビルドされない、または正常に実行されないことを示すものではありません。

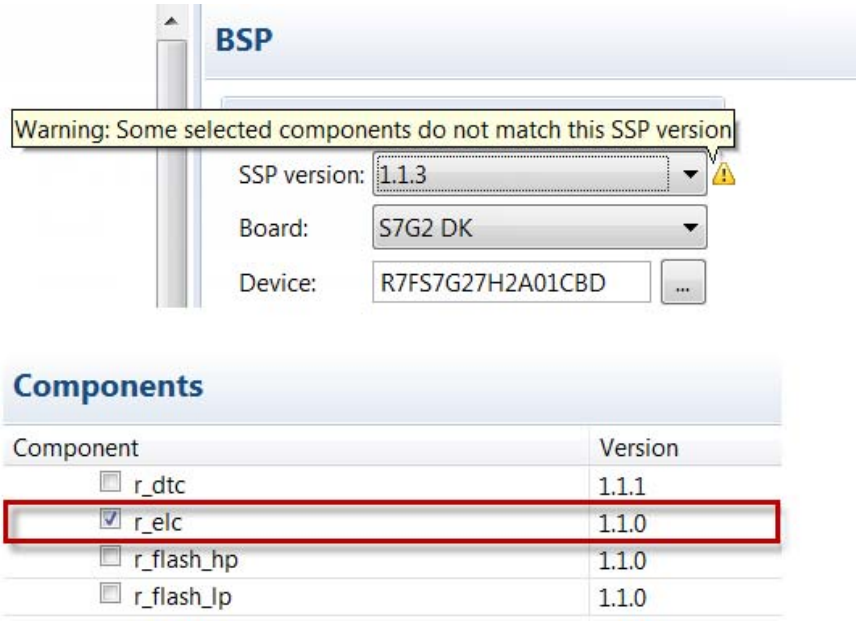


図 80: e2 studio コンポーネントで複数の SSP バージョンをインストールすると不一致が発生します。

最後にインストールされたパッチリリースより前の SSP バージョンを選択すると（たとえば、SSP v1.1.3 もインストールしている状況で SSP v1.1.1 を選択）、e2 studio 統合ソリューション開発環境は、たとえ特定の SSP コンポーネントの新しいバージョン（つまり SSP v1.1.3）が利用できる場合でも、利用可能な SSP v1.1.1 コンポーネントをプロジェクトに含めます。この挙動は、以前の SSP バージョンをユーザーが明確に指定したことによるものです。

3.1.16.2 X-Ware ソースの追加 （SSP v1.2.0 以降）

[Threads] タブを使ってプロジェクトに ThreadX などの X-Ware ソースコードを追加する手順は、次のとおりです。

1) [Threads] ペインでスレッドをクリックします。
2) [Stacks] ペインで、[New]>[X-Ware] を選択し、ソースコードを追加する X-Ware コンポーネントを選択します。次にメニューに従って、X-Ware ソースモジュールまで進みます。

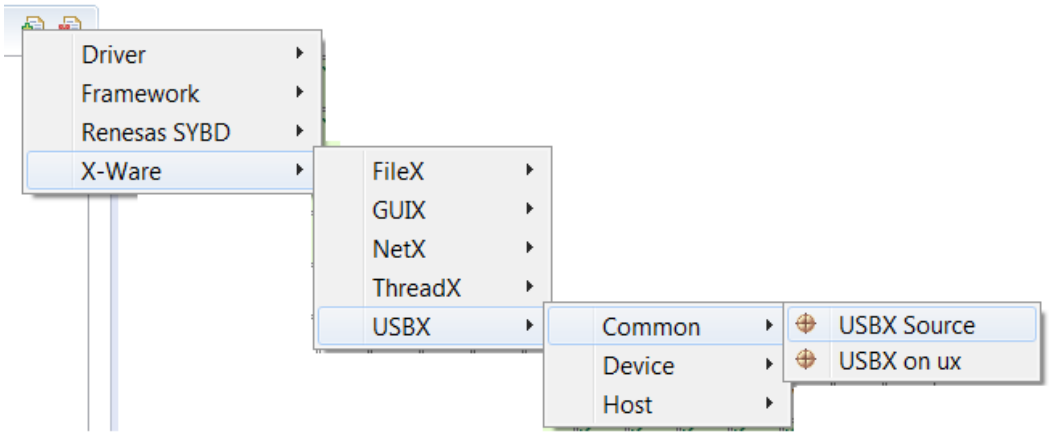


図 81: プロジェクトへの X-Ware ソースの追加

3) [Properties] ビューを使用して、 X-Ware プロパティを構成します。

4) [Generate Project] をクリックします。

- e2 studio 統合ソリューション開発環境により、X-Ware ソースコードが synergy/ssp/src/framework/el/… または synergy/ssp/src/framework/sf_el_*. ディレクトリに抽出されます。

  注 : 適切なライセンスがある場合にのみ、 X-Ware ソースコードの表示と変更を行うことができます。

  注 : X-Ware ソースを抽出すると、プロジェクトのコンパイル時間は増加します。

  特に留意したいのは、プロジェクトに ThreadX/NetX/FileX/GUIX/USBX ソースコンポーネントを追加しても、これらのコンポーネントのビルド済みライブラリは自動で削除されるわけではないということです。これが、構造体の定義の競合やライブラリまたはソースファイルのエラーをもたらし、ビルドエラーにつながることがあります。このため、（SSP v1.2.0 で） X-Ware ソースを追加する場合は、警告メッセージを確認するようにしてください （Synergy Project Editor [Threads] タブで確認できます）。

ビルドエラーが何も出ない場合は、 [Properties] ビューで警告メッセージを無効化することができます（X-Ware ソースモジュールがスコープ内にある間に限ります）。

これらのビルドエラーが出た場合は、以下の手順でエラーを解決してください。

1) e2 studio でリンカ設定にアクセスします （[Project]>[Properties]>[C/C++ Build]>[Settings]>[Tool Settings]>[Cross ARM C Linker]>[Libraries]）。

2) [Libraries] ペインで、リストからソースコードを追加したライブラリを削除して、リンクされないようにします。

3) [Apply]、 [OK] の順にクリックします。

4) プロジェクトを再ビルドします。定義が複数存在することによるビルドエラーが解消しているはずです。

   注 : クリーンビルドを試みるたびに、またはプロジェクトコンテンツが再生成されるときは常に、この手順を行う必要があります。

3.1.16.3 ヘッドレスビルドを使用した Synergy プロジェクトのビルド

e2 studio GUI を使用して Synergy プロジェクトをビルドする他に、ヘッドレスビルドを実行してプロジェクトをビルドすることもできます。これは、基本的には、e2 studio UI を使用せずに、コマンドラインから、またはスクリプトを使用してプロジェクトをビルドする方法を意味します。これは Hudson や Jenkins などの継続的インテグレーションツールを使用してビルドを自動化する場合に役立ちます。

e2 studio を使用してヘッドレスビルドを実行する方法の詳細については、e2 studio ヘルプコンテンツの「How to build e2 studio generated projects in Hudson/Jenkins」を参照してください。

3.2 チュートリアル : Your First Synergy Project - Blinky

このチュートリアルの目的は、 e2 studio を使用した単純なアプリケーションの作成と、そのアプリケーションの Synergy ボード上での実行手順を通じて、 Synergy プラットフォームに素早く精通することです。

3.2.1 前提条件

このチュートリアルを実行するには、以下のものが必要です。

- Windows ベースの PC
- e2 studio
- Synergy Software Package
- Synergy ボードキット

3.2.2 Blinky 用の新規プロジェクトの作成

Synergy プロジェクトの作成と設定は、アプリケーションを作成するための最初のステップです。ベース SSP パックには、シンプルですべての Renesas Synergy ボードで動作する、作成済みの Blinky サンプルアプリケーションが含まれています。

Synergy プロジェクトを作成するには以下の手順に従います。

1) e2 studio ISDE で、 [File] > [New] > [Synergy Project] の順にクリックします。
2) この新しいプロジェクトに名前を付けます。このチュートリアルでは Blinky という名前を使用します。
3) ライセンスウィンドウが空白の場合はライセンスファイルを特定します。このバージョンのプラットフォームのライセンスファイルは、 <ISDE ベースディレクトリ > 内の ISDE\internal\projectgen\arm\Licenses\SSP_License_Example_EvalLicense_<rev>.xml にあります。
4) [Next] をクリックします。 [Project Configuration] ウィンドウに選択内容が表示されます。

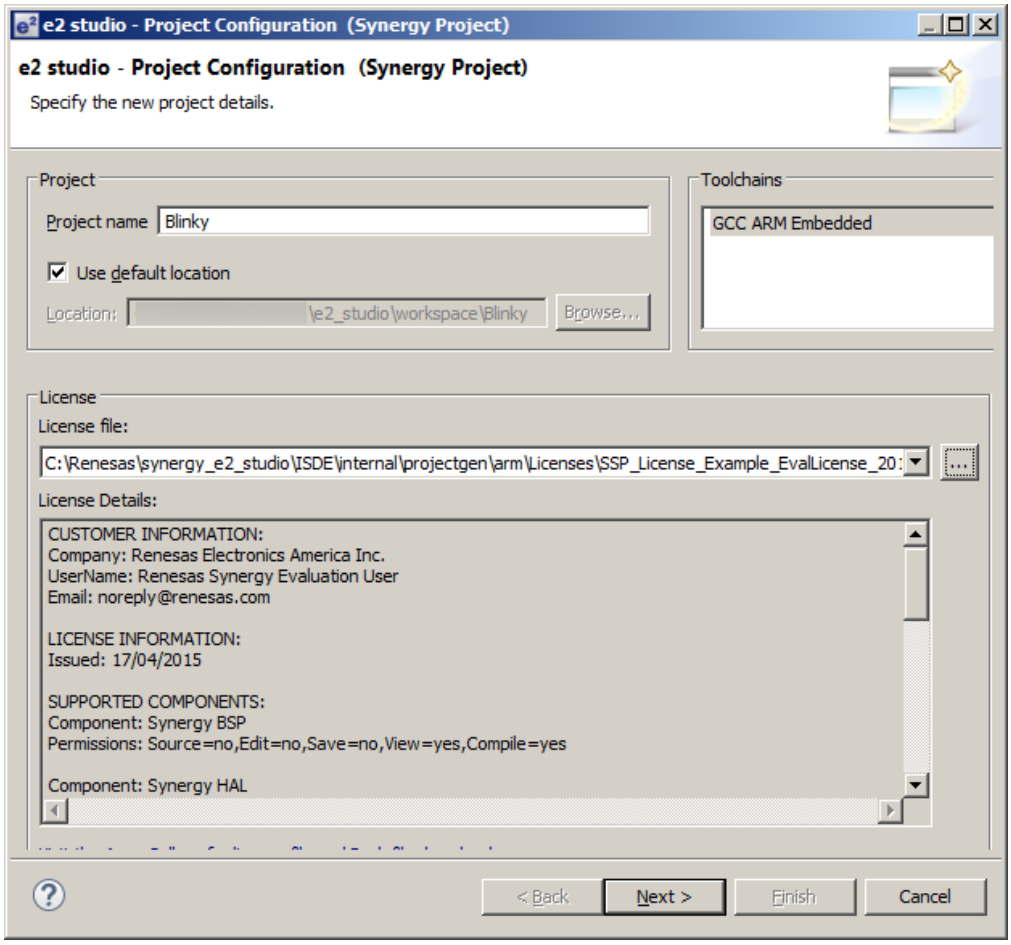


図 82: e² studio ISDE の [Project Configuration] ウィンドウ （パート 1）

5) ボードサポートパッケージを選択するため、[Device Selection] ドロップダウンリストから使用するボード名を選択し、 [Next] をクリックします。

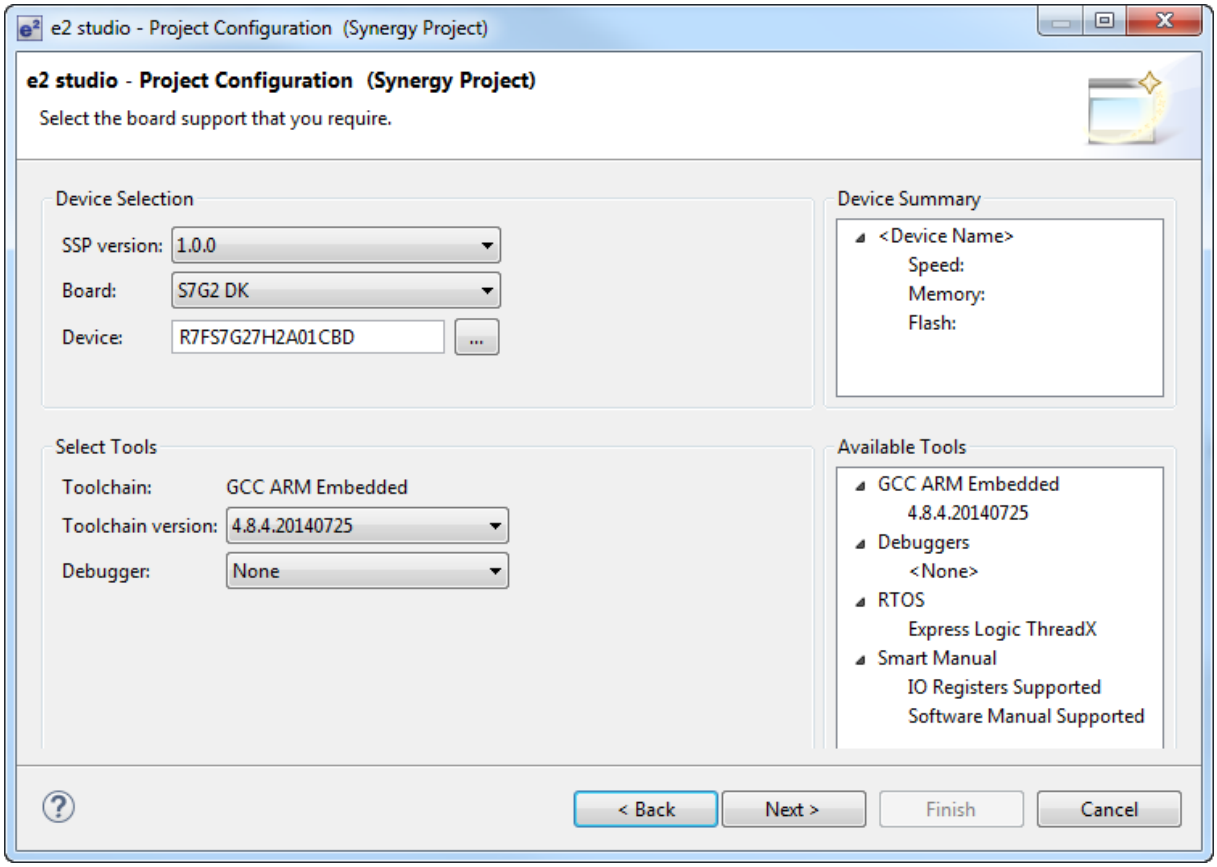


図 83: e² studio ISDE の [Project Configuration] ウィンドウ （パート 2）

6) 使用するボード用の Blinky テンプレートを選択し、 [Finish] をクリックします。

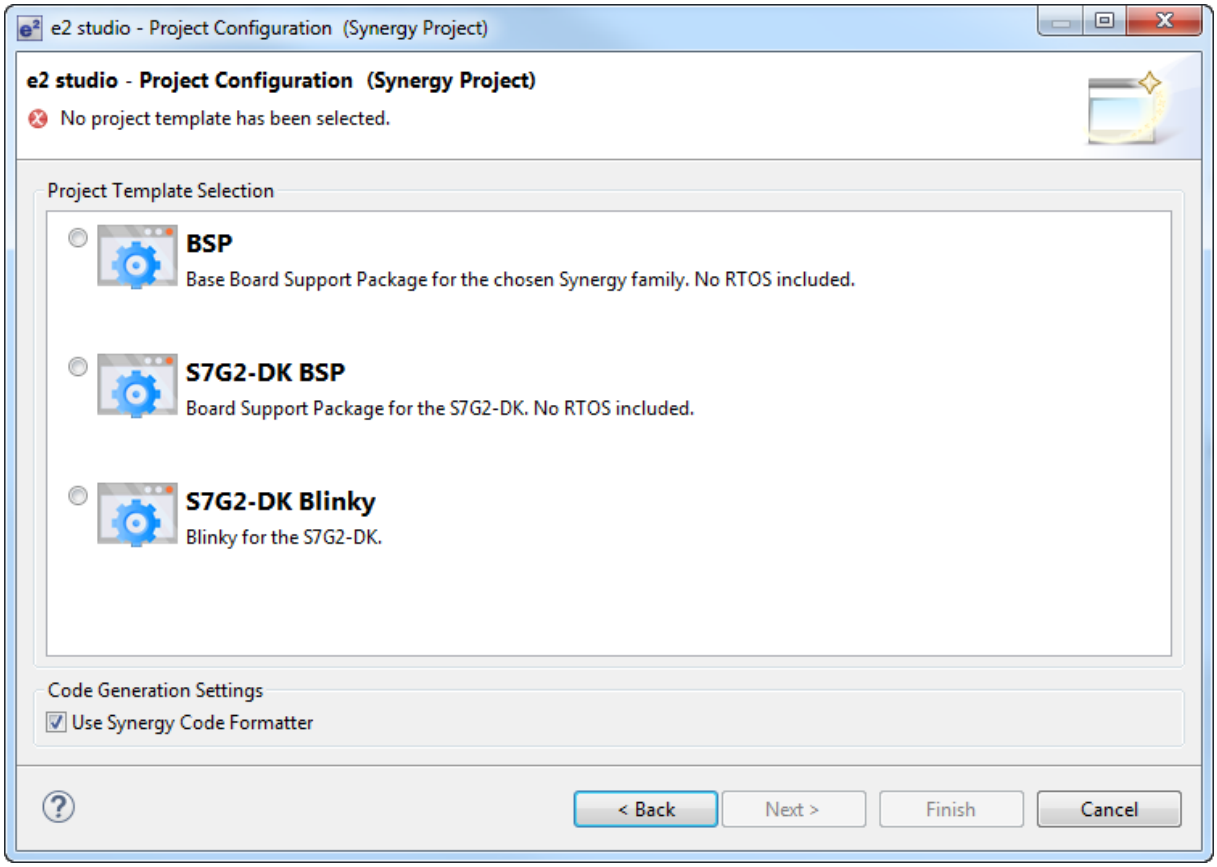


図 84: e² studio ISDE の [Project Configuration] ウィンドウ （パート 3）

プロジェクトが作成されると、プロジェクトの名前が ISDE の [Project Explorer] ウィンドウに表示されます。 [Project Configuration] ウィンドウの右上隅にある [Generate Project Content] ボタンを押し、ボード固有のファイルを生成します。

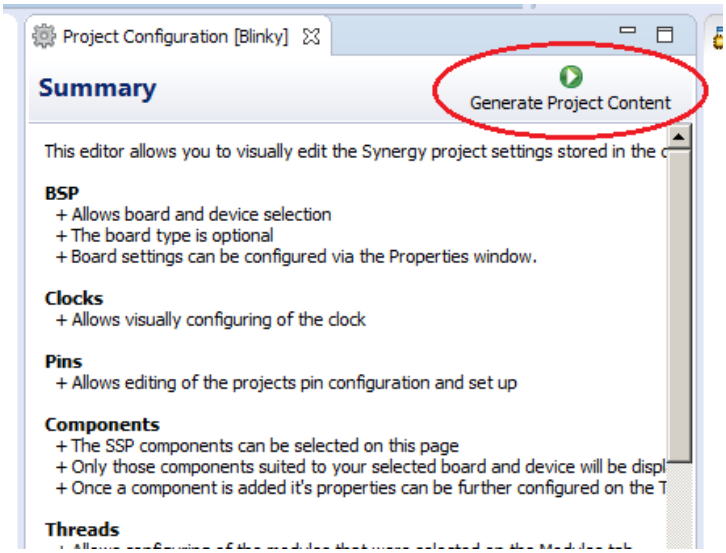


図 85: e² studio ISDE の [Project Configuration] タブ

これで新しいプロジェクトが作成、設定され、ビルドの準備ができました。

3.2.2.1 Blinky の設定に関する詳細

[Generate Project Content] ボタンは、構成ヘッダーファイルを作成し、テンプレートからソースファイルをコピーして、通常は [Project Configuration] 画面の状態に基づいてプロジェクトを構成します。

たとえば、[Components] タブのモジュールの横にあるチェックボックスをオンにし、[Generate Project Content] ボタンを押すと、そのモジュールをプロジェクトに追加するために必要なすべてのファイルがコピーまたは作成されます。その同じチェックボックスをオフにすると、それらのファイルが削除されます。

3.2.2.2 Blinky のクロックの設定

Blinky テンプレートを選択することで、ISDE により Blinky アプリケーション用にクロックが設定されます。Blinky のクロック構成は、ISDE クロック構成タブに表示されます（クロックの設定を参照）。Blinky のクロック構成は、 BSP クロック構成ファイルに保存されます （BSP のクロック設定を参照）。

3.2.2.3 Blinky のピン設定

Blinky テンプレートを選択することで、LED1 を切り替えるために使用する GPIO ピンが、ISDE により Blinky アプリケーション用に設定されます。ISDE のピン構成タブには、Blinky アプリケーション用のピン構成が表示されます（ピンの設定を参照）。 Blinky のピン構成は、 BSP 構成ファイルに保存されます（BSP のピン設定を参照）。

3.2.2.4 Blinky コンポーネント用のパラメータの設定

Blinky プロジェクトは、 ISDE コンポーネント内の次の HAL コンポーネントを自動的に選択します。

- r_cgc
- r_elc
- r_ioport

いずれかのコンポーネントの構成パラメータを表示するには、それぞれのドライバーの [HAL] ウィンドウの [Properties] タブを確認します （HAL ドライバーの追加と設定を参照）。

3.2.2.5 main() の場所

main 関数は <プロジェクト >/src/synergy_gen/main.c にあります。これは、プロジェクト作成段階で生成されるファイルの 1 つであり、 hal_entry() の呼び出しのみを含んでいます。生成されるファイルの詳細については、 HAL ドライバーの追加と設定を参照してください。

3.2.2.6 Blinky のサンプルコード

Blinky アプリケーションは、hal_entry.c ファイルに格納されます。このファイルは、Blinky プロジェクトテンプレートを選択したときに ISDE によって生成され、プロジェクトの src/ フォルダー内に配置されます。

アプリケーションは次の手順を実行します。

1) BSP HAL 関数を呼び出すことにより、選択されたボードの LED 情報を取得します (R_BSP_LedsGet)。
2) 選択されたボードの LED を制御する GPIO ピンの出力レベル HIGH を定義します。
3) 選択されたシステムクロック速度を取得し、 LED のトグルを観察できるようにクロックをスケールダウンします。
4) 次のものを使用して GPIO ピンに書き込むことにより、 LED をトグルします。

```
g_ioport.p_api->pinWrite()
```

3.2.3 Blinky プロジェクトのビルド

新しいプロジェクトを [Project Explorer] ウィンドウでハイライトし、ビルドします。

プロジェクトをビルドするには、以下の 3 つの方法があります。

a. メニューバーの [Project] をクリックし、 [Build Project] を選択します。

b. ハンマーアイコンをクリックします。

c. プロジェクトを右クリックし、 [Build Project] を選択します。

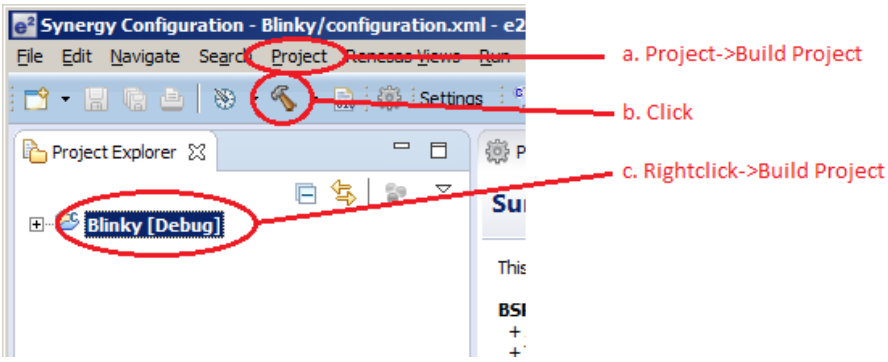


図 86: e² studio ISDE の [Project Explorer] ウィンドウ

ビルドが完了すると、メッセージがビルドの [Console] ウィンドウに表示され、最終的なイメージファイル名と、そのイメージ内のセクションサイズが表示されます。

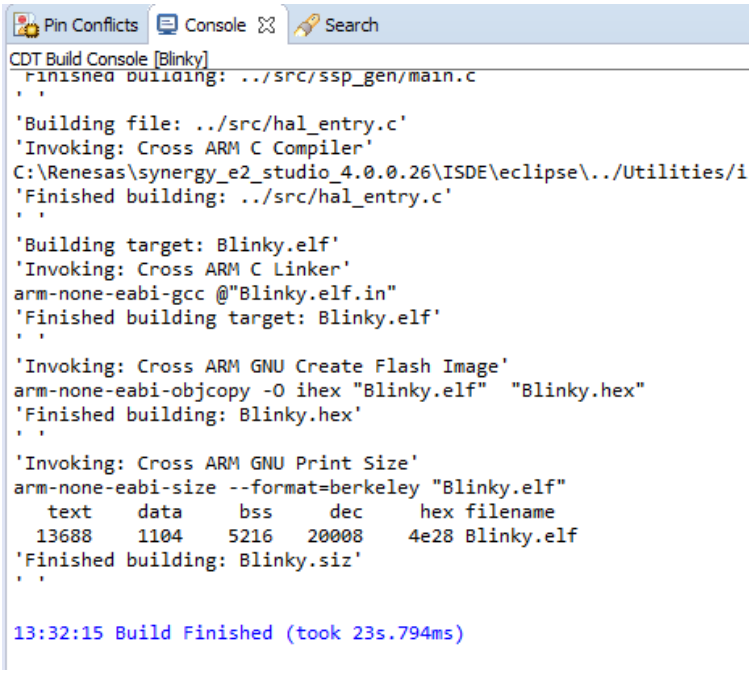


図 87: e² studio ISDE のプロジェクトビルドコンソール

3.2.4 Blinky プロジェクトのデバッグ

3.2.4.1 デバッグの前提条件

ボード上でプロジェクトをデバッグするには、以下のものが必要です。

- ISDE に接続するボード

- ボードと通信するように設定されるデバッガー
- マイクロコントローラにプログラムされるアプリケーション

アプリケーションは、マイクロコントローラの内蔵フラッシュから実行します。アプリケーションを実行またはデバッグするには、まずアプリケーションがマイクロコントローラのフラッシュにプログラムされている必要があります。そのための方法としては、以下の 2 つの方法があります。

- JTAG デバッガー
- UART または USB を通じた組み込みブートローダー

一部のボードにはオンボード JTAG デバッガーが搭載されていますが、他のボードでは、ボード上のヘッダーに接続された外部 JTAG デバッガーが必要です。

ボードのユーザーマニュアルで、JTAG デバッガーを ISDE に接続する方法を確認してください。

3.2.4.2 デバッグ手順

Blinky アプリケーションをデバッグするには、以下の手順を実行します。

1) プロジェクト用にデバッガを設定するため、[Run] > [Debugger Configurations] をクリックします。

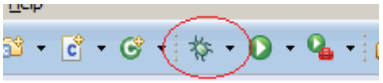


図 88: e² studio ISDE のデバッグアイコン

または、バグアイコンの横にあるドロップダウンメニューで [Debugger Configurations] を選択します。

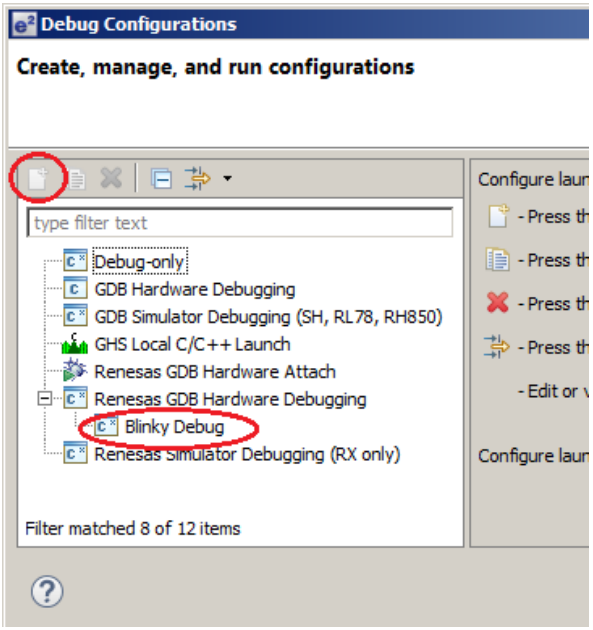


図 89: e² studio ISDE の [Debugger Configuration] ウィンドウ

2) ウィンドウでデバッガー設定を選択します。デバッガー設定が表示されていない場合は、ウィンドウの左上隅にある [New] アイコンをクリックして作成する必要があります。選択すると、[Debug Configuration] ウィンドウに、 Blinky プロジェクトのデバッグ設定が表示されます。

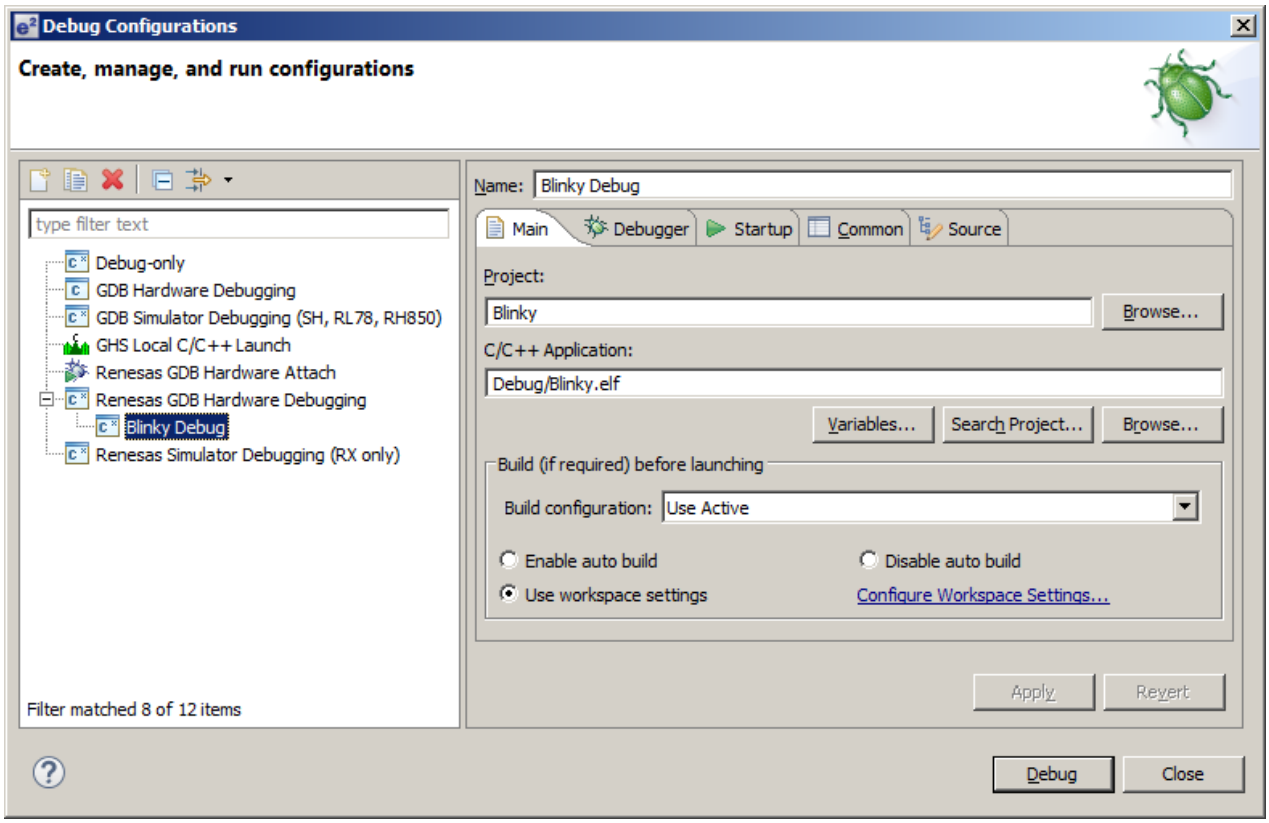


図 90: e² studio ISDE の [Debugger Configuration] ウィンドウと Blinky プロジェクト

3) [Debug] を押してアプリケーションのデバッグを開始します。

3.2.4.3 デバッグ手順の詳細

デバッグモードでは、 ISDE は次のタスクを実行します。

1) アプリケーションイメージをマイクロコントローラにダウンロードし、イメージを内蔵フラッシュメモリにプログラミングします。
2) main() にブレークポイントを設定します。
3) スタックにスタックポインタレジスタを設定します。
4) プログラムカウンタレジスタに、リセットベクトルのアドレスをロードします。
5) プログラムカウンタが指しているスタートアップコードを表示します。

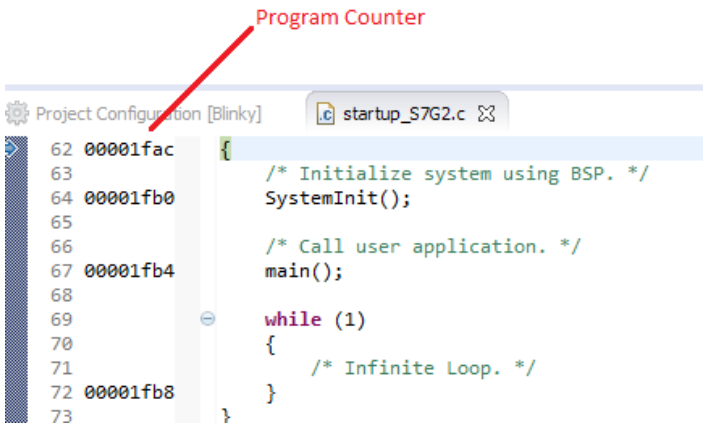


図 91: e² studio ISDE のデバッガメモリウィンドウ

3.2.5 Blinky プロジェクトの実行

デバッグモードで、[Run] > **[Resume]** をクリックするか、[Play] アイコンを 2 回クリックします。

図 92: e² studio ISDE デバッガの [Play] アイコン

ボード上の LED1 とマークされた LED が点滅します。

3.3 チュートリアル : Using HAL Drivers - Programming the WDT

このアプリケーションは、WDT HAL ドライバーの WDT によって実装されている WDT Interface を使用します。このドキュメントでは、ISDE と SSP を使用して、Synergy MCU のウォッチドッグタイマ (WDT) 周辺デバイス用のアプリケーションを作成する方法について説明します。このアプリケーションは、以下の SSP モジュールを利用します。

- Board Support Package（ボードサポートパッケージ）
- CGC（クロック生成回路）
- WDT（ウォッチドッグタイマ）
- IOPORT(GPIO)

3.3.1 Synergy SSP と ISDE を使用した WDT アプリケーションの作成

3.3.1.1 SSP および e^2 studio ISDE の使用方法

Renesas 製の Synergy Software Package (SSP) は、 Synergy アプリケーションを開発するためのドライバーライブラリ一式を提供します。 SSP は、ハードウェア抽象化レイヤー (HAL) ドライバー、ボードサポートパッケージ (BSP) ドライバー、および開発者がアプリケーションを作成するために使用する上位のフレームワークアプリケーションを提供します。SSP は、eclipse ベースの Renesas e^2 studio 統合ソリューション開発環境 (ISDE) に統合されており、ビルド（エディター、コンパイラ、リンカー）およびデバッグフェーズと、拡張された GNU Debug (GDB) インタフェースを提供します。

3.3.1.2 WDT アプリケーション

WDT アプリケーションのフローチャートを以下に示します。

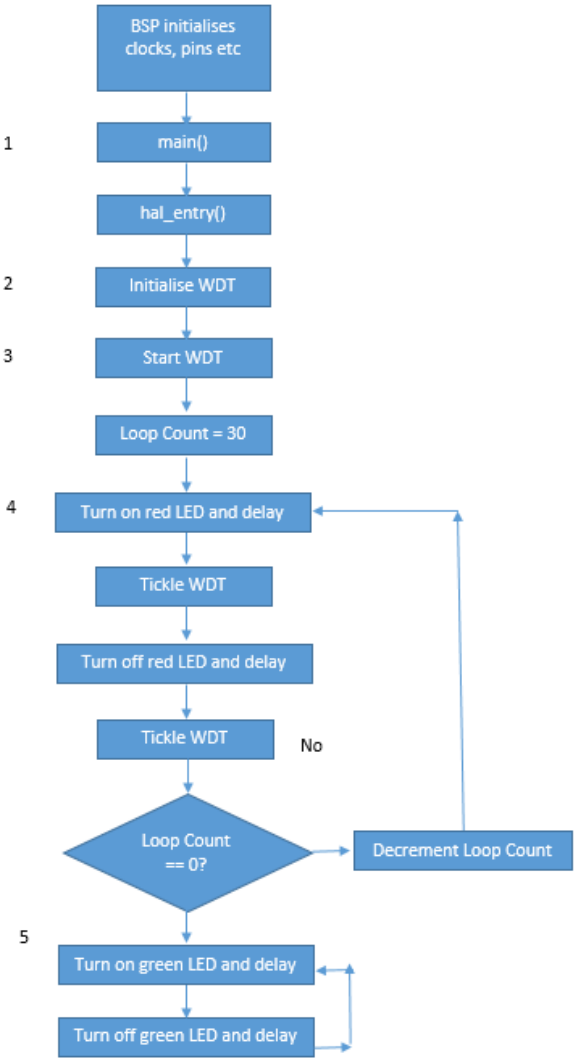


図 93: WDT アプリケーションのフロー図

3.3.1.3 WDT アプリケーションのフロー

WDT アプリケーションの主な部分は以下のとおりです。

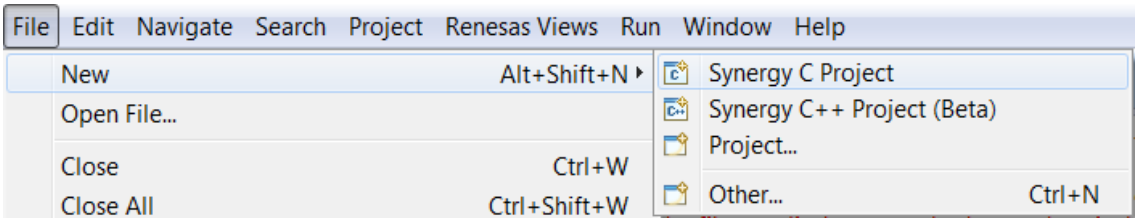
1) main() が hal_entry() を呼び出します。関数 hal_entry() は SSP によって作成され、ユーザーコードのプレースホルダが含まれます。 WDT 用のコードをこの関数に追加します。
2) WDT を初期化します。ただし、まだ開始しません。
3) WDT をリフレッシュすることによって、 WDT を開始します。
4) 赤い LED が 30 回点滅し、 LED の状態が変化するたびにウォッチドッグがリフレッシュされます。

5) 緑色の LED が点滅しますが、ウォッチドッグはリフレッシュされません。ウォッチドッグのタイムアウト期間の後、デバイスがリセットされます。その様子は、シーケンスが繰り返されるときに赤い LED が再度点滅することで確認できます。

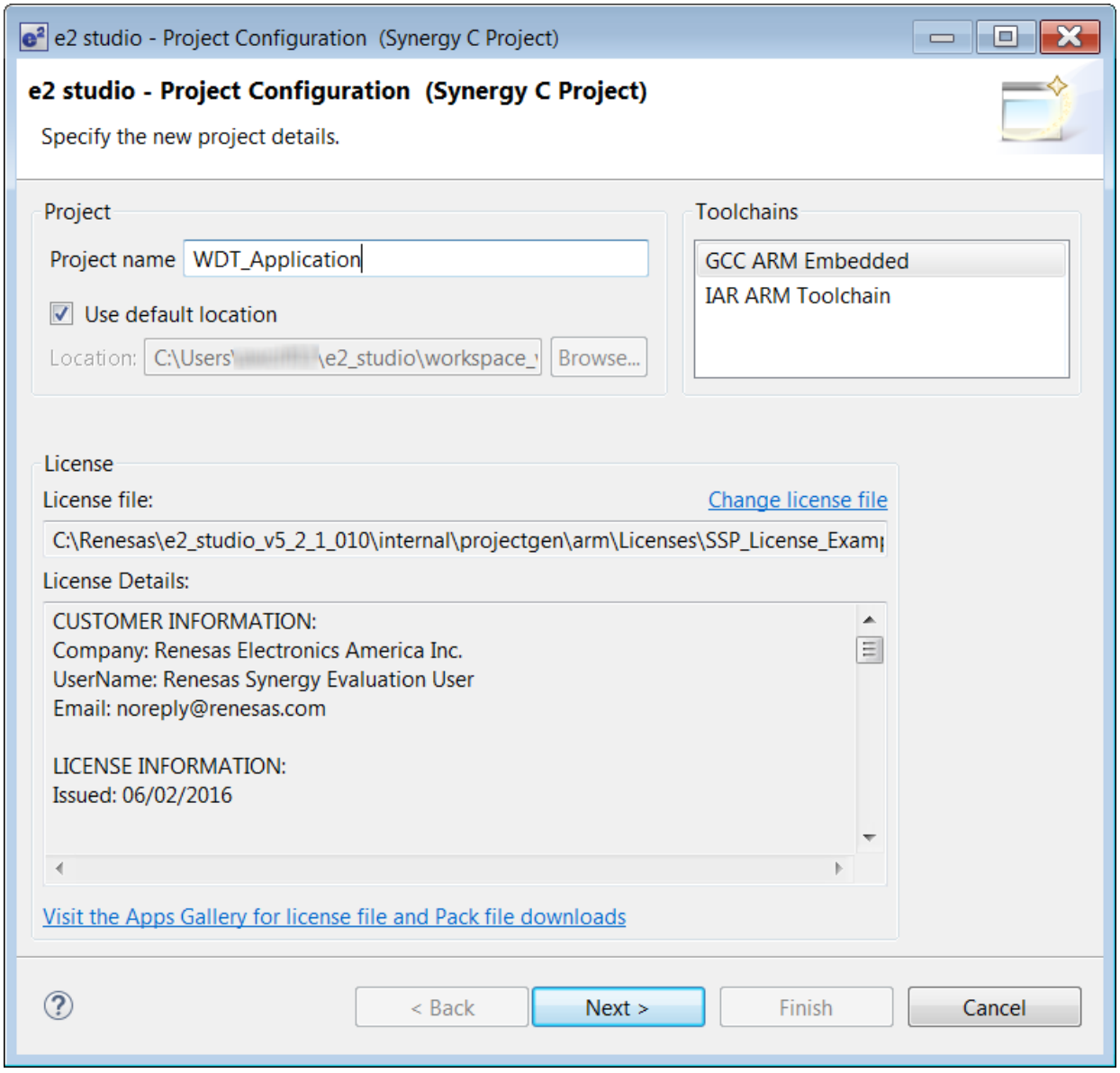
3.3.2 ISDE でのプロジェクトの作成

ISDE を起動し、ワークスペースラウンチャーでワークスペースフォルダーを選択します。以下のようにして新しい Synergy プロジェクトを設定します。

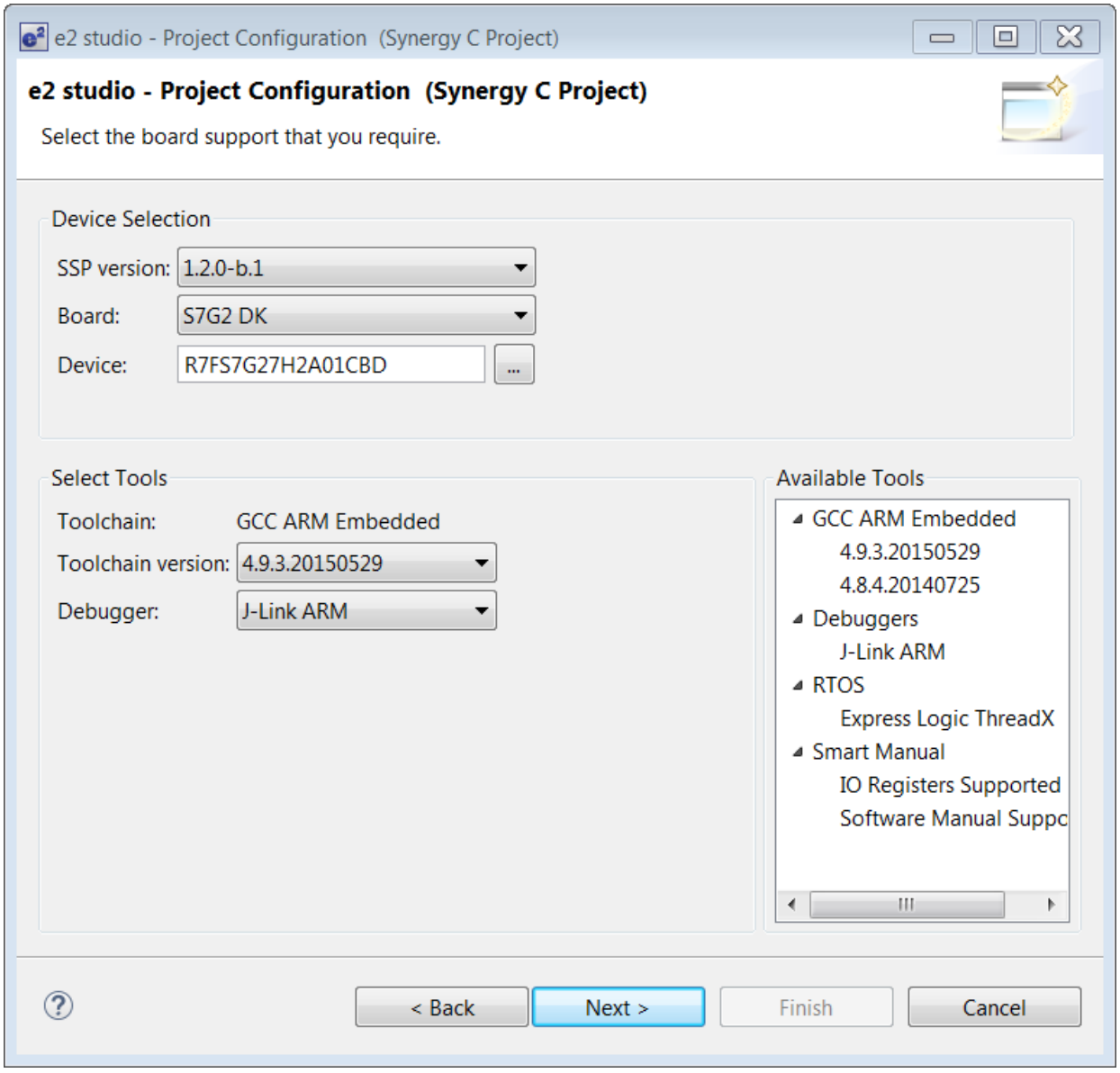
1) [File]>[New]>[Synergy C Project] の順に選択します。



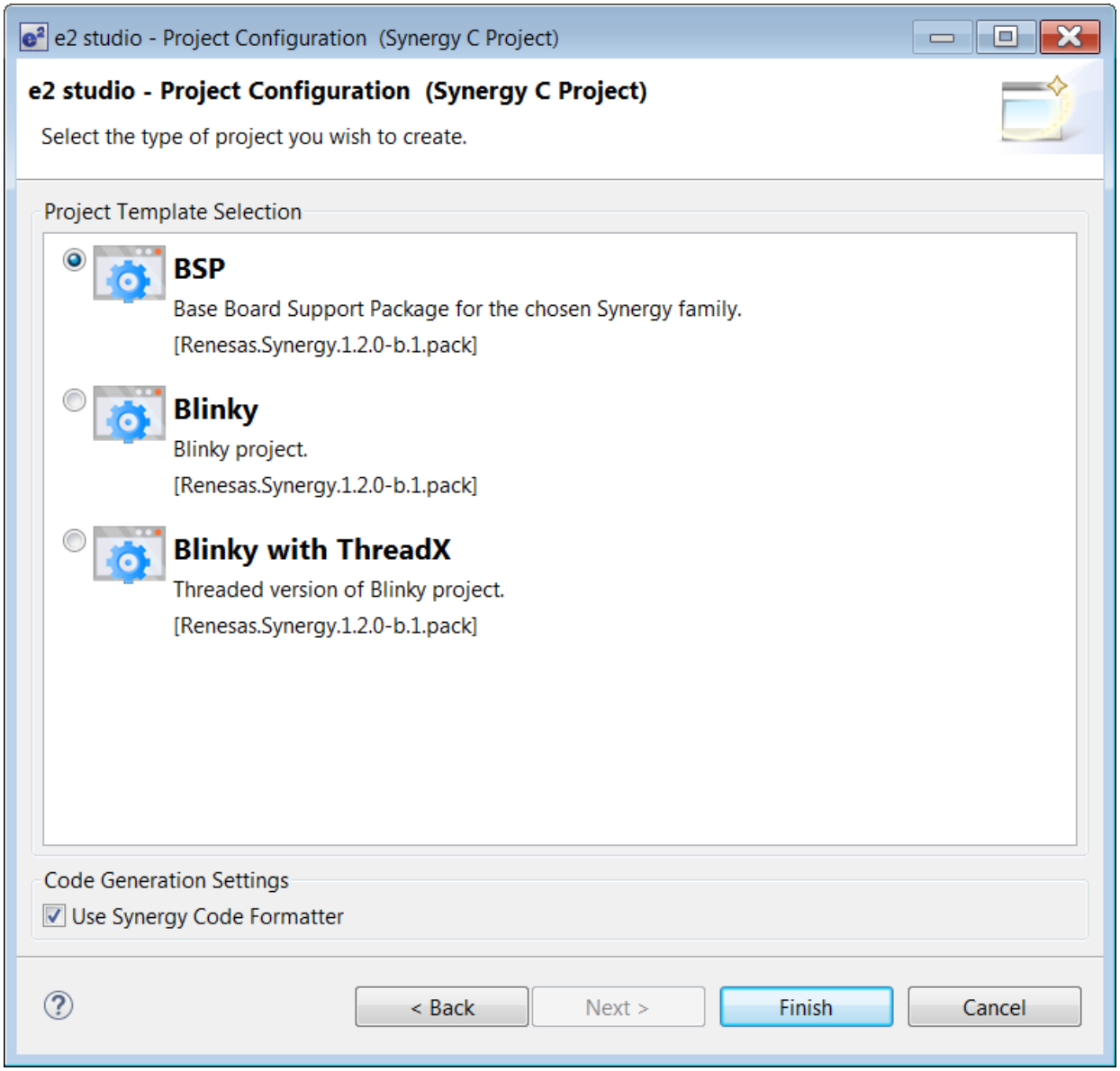
2) ISDE の [SynergyProject Generator (Synergy C Project)] ウィンドウにプロジェクト名（たとえば、WDT_Application). を入力します。また、ツールチェーンとライセンスファイルを選択します。プロジェクトの新しい場所を選択するには、[Use default location] チェックボックスをオフにします。[Next] をクリックします。



3) このアプリケーションは、Synergy S7G2 ベースの DK-S7G2 ボードで動作します。そのため、[Board] には [S7G2 DK] を選択します。これにより、[Device] ドロップダウンに、このボードで使用される正しいデバイスが自動的に設定されます。ツールチェーンのバージョンを選択します。J-Link Arm をデバッガとして選択します。このアプリケーションでは RTOS は使用されていませんが、デフォルトの Express Logic ThreadX のままで構いません。 [Next] をクリックしてプロジェクトを構成します。



今度はプロジェクトテンプレートを選択します。 RTOS は必要ないため、 **BSP** を選択します。



4) [Finish] をクリックします。

プロジェクトが作成され、[Project Explorer] ビューと [Synergy Project Editor] ビューが開き、後者の [Summary] タブにプロジェクト構成の概要が示されます。

3.3.3 ISDE でのプロジェクトの設定

e^2 studio ISDE は、プロジェクトを設定するためのオプションを選択する GUI インタフェースを備えているため、プロジェクト構成手順が簡略化され、時間も短縮されます。

ISDE には、進行中の操作に応じて各種のウィンドウを表示する、いくつかのパースペクティブが備わっています。デフォルトのパースペクティブは、[C/C++]、[Synergy Configuration]、[Debug] です。パースペクティブを変更するには、 ISDE の右上にあるボタンから新しいパースペクティブを選択します。

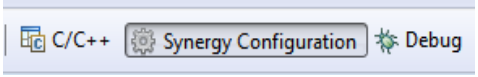
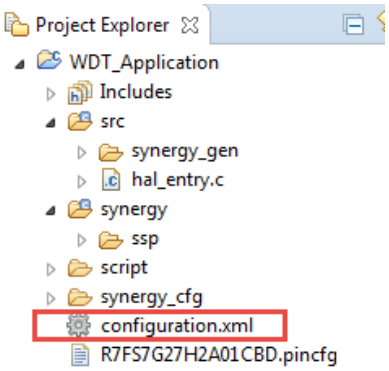


図 94: パースペクティブの選択

C/C++ パースペクティブは、コード編集用に選択されたレイアウトを提供します。[Synergy Configuration] パースペクティブには Synergy プロジェクトを設定するための要素があり、[Debug] パースペクティブはデバッグに適したビューを備えています。

1) プロジェクトを設定するには、 [Synergy Configuration] パースペクティブが選択されていることを確認します。
2) [[WDT_Application]Synergy Configuration] が開いていることを確認します。要約情報が表示されている場合はすでに開かれています。ここで、または任意のタイミングで [Synergy Configuration] を開くには、[Synergy Configuration] パースペクティブが選択されていることを確認し、 ISDE の右側にある [Project Explorer] ペインで configuration.xml ファイルをダブルクリックします。



[Synergy Project Editor] ビューの下部には、プロジェクトを構成するためのタブがいくつかあります。プロジェクトでは、これらのタブの一部または全部に対する変更が必要になる可能性があります。これらのタブについて以下に説明します。

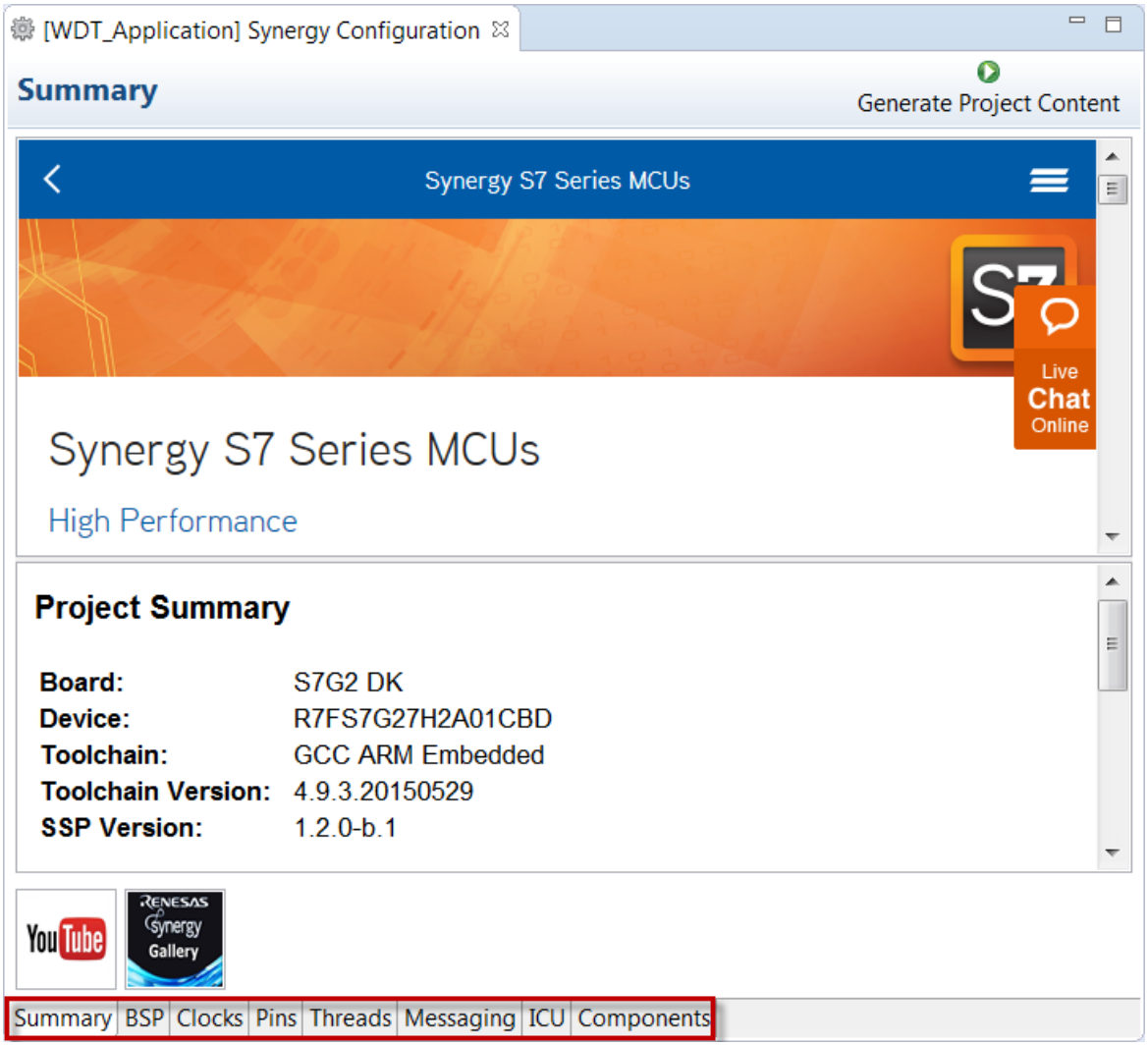
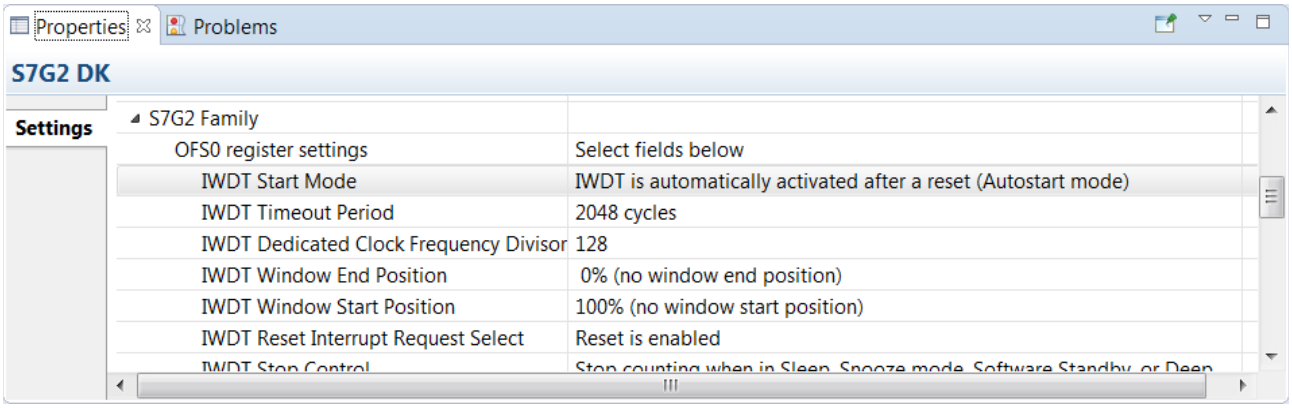


図 95: [Synergy Project Editor] タブ

3.3.3.1 BSP タブ

[BSP] タブでは、ボードサポートパッケージ (BSP) オプションをデフォルト値から変更できます。この特定の WDT プロジェクトについては、変更は不要です。ただし、WDT をオートスタートモードで使用する場合は、[BSP] タブで OFS0（オプション機能選択レジスタ 0) レジスタを設定できます。WDT のオートスタートモードの詳細については、 Synergy ハードウェアユーザーズマニュアルを参照してください。



3.3.3.2 [Clocks] タブ

[Clocks] タブには、デバイスのクロックツリーがグラフィカルに表示されます。 GUI のドロップダウンボックスを使用して、各種クロックを設定できます。 WDT は PCLCKB を使用します。このクロックのデフォルト出力周波数は 60 MHz です。クロックがこの値を出力していることを確認してください。

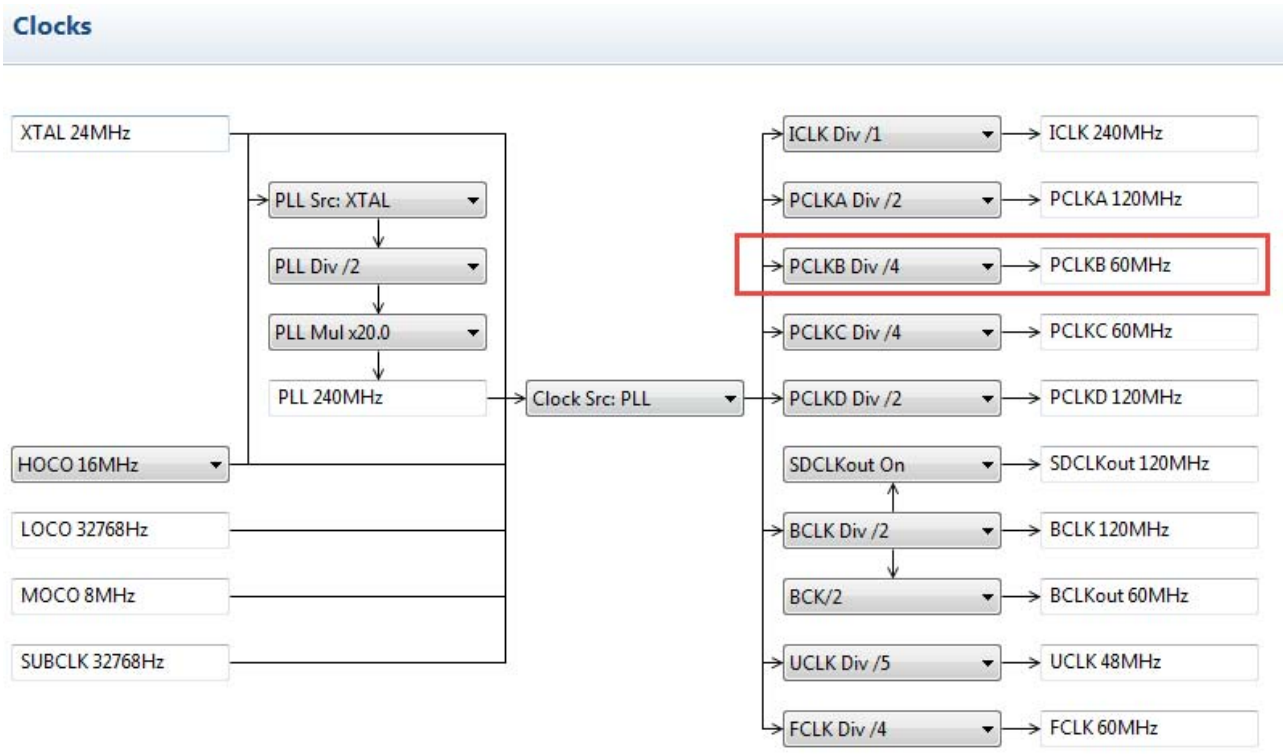


図 96: クロック設定

3.3.3.3 [Pins] タブ

[Pins] タブでは、デバイスのピンの機能を設定するためのグラフィカルツールを利用できます。WDT プロジェクトでは、ピンの設定は不要です。プロジェクトではデバイス上のピンに接続された 2 個の LED を使用しますが、これらのピンは出力 GPIO ピンとして BSP によりあらかじめ設定されています。

3.3.3.4 [Threads] タブ

[Threads] タブでは、任意のドライバーをプロジェクトに追加できます。クロック生成回路、イベントリンクコントローラ、および IO ポートピン用の HAL ドライバーは、プロジェクトの設定時に ISDE によって自動的に追加されます。WDT アプリケーションは ThreadX リソースを使用しないため、追加するのは HAL WDT ドライバーだけにしてください。

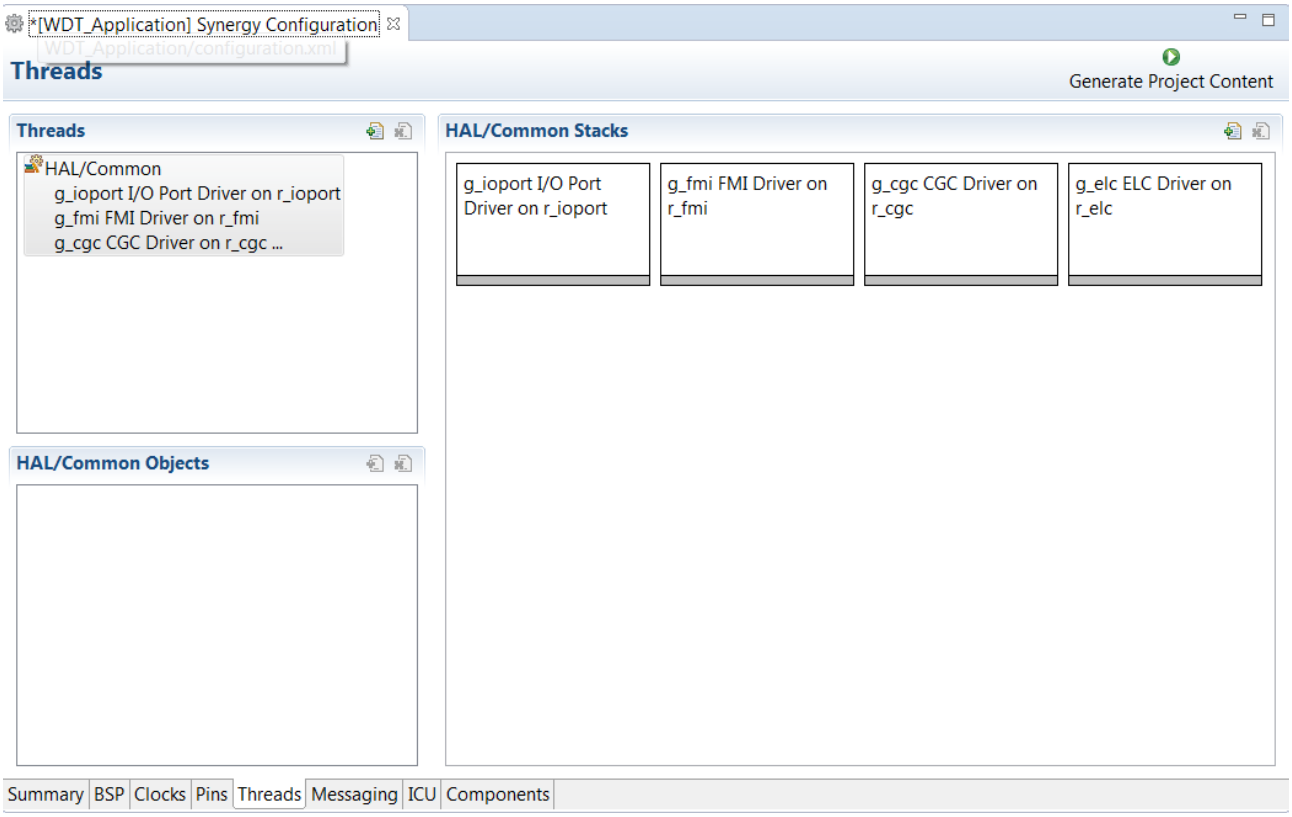
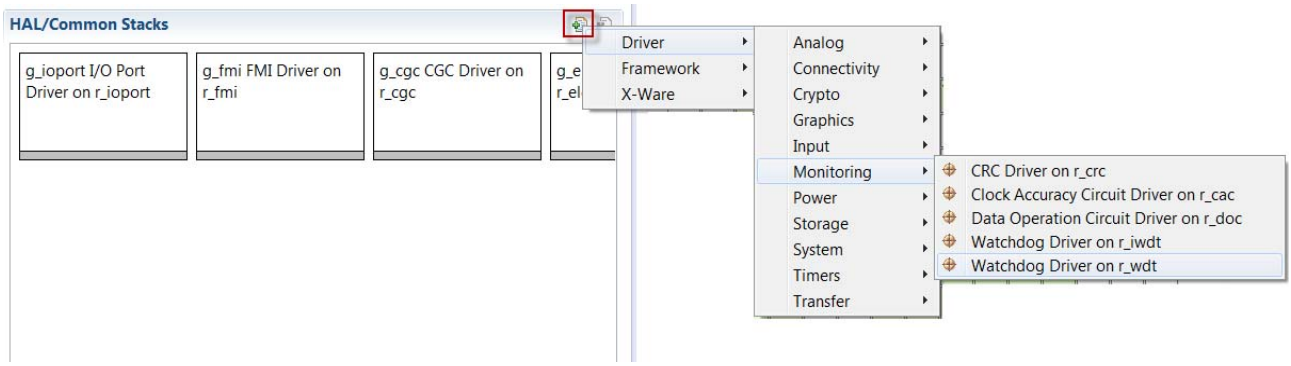


図 97: [Threads] タブ

1) 上の図に示すように、[Threads] ウィンドウで [HAL/Common] パネルをクリックします。[Modules Stacks] ペインが [HAL/Common Stacks] ペインになり、統合ソリューション開発環境によってあらかじめ選択されたモジュールが表示されます。
2) [New]>[Driver]>[Monitoring] をクリックして、 [WATCHDOG Driver on WDT] を選択します。



選択した HAL WDT ドライバーが [HAL/Common Modules] ペインに追加され、選択されたモジュールのすべての構成オプションが [Property] ビューに表示されます。 [Property] ビューが画面の左下に表示されます。表示されない場合は、 [Synergy Configuration] パースペクティブが選択されていることを確認します。

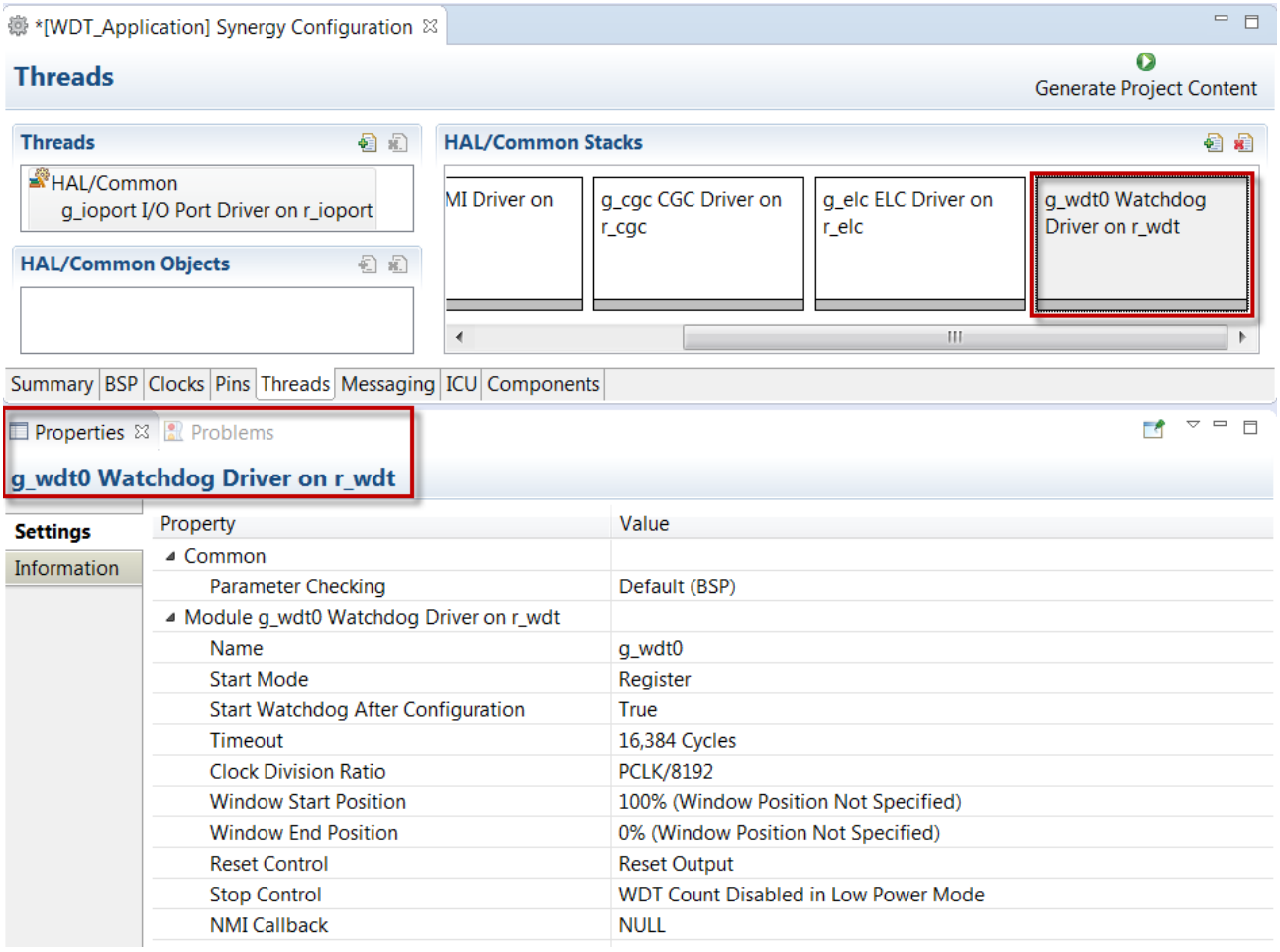


図 98: [Properties] ビューのモジュールプロパティ

パラメータ **[Start Watchdog After Configuration]** を **[True]** から **[False]** に変更します。他のパラメータはデフォルト値のままで構いません。**[Start Watchdog After Configuration]** を **[False]** に設定すると、WDT ドライバーに対し （その open API コールを通じて）、 WDT を設定するものの、開始しないように指示します。後で更新することで開始されます。

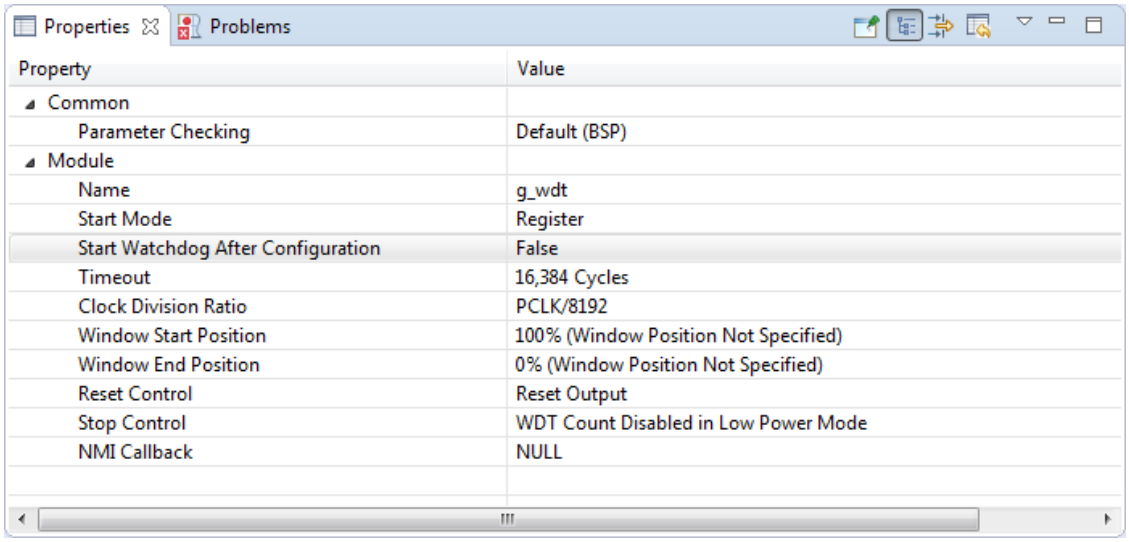


図 99: WDT プロパティ上の g_wdt ウォッチドッグドライバー

PCLKB が 60 MHz で動作している状態で、 WDT は最後の更新から 2.23 秒後にデバイスをリセットします。

WDT クロック = 60 MHz / 8192 = 7.32 kHz

サイクル時間 = 1 / 7.324 kHz = 136.53 マイクロ秒

タイムアウト = 136.53 マイクロ秒 x 16384 = 2.23 秒

プロジェクト構成ファイルを保存して、[Synergy Project Editor] の右上隅にある **[Generate Project Content]** ボタンをクリックします。

図 100: [Generate Project Content] ボタン

ISDE はプロジェクトファイルを生成します。

3.3.3.5 [Components] タブ

[Components] タブは参照用に存在し、ここでプロジェクトに含まれているモジュールを確認できます。[Threads] タブでモジュールを追加すると、それらのモジュールが [Components] ビューで自動的に選択されます。

WDT プロジェクトでは、以下のモジュールが選択されていることを確認してください。

1) HAL_Drivers -> r_cgc
2) HAL_Drivers -> r_elc
3) HAL_Drivers -> r_ioport
4) HAL Drivers -> r_fmi
5) HAL_Drivers -> r_wdt

注 :[Components] タブに表示されるモジュールの一覧は、インストールされている SSP のバージョンによって異なります。

3.3.4 WDT の生成されるプロジェクトファイル

[Generate Project Content] ボタンを押すと以下のタスクが実行されます。

- r_wdt フォルダーと WDT ドライバーの内容が次の場所に作成されます。
  - synergy/ssp/src/driver/
- r_wdt_api.h が次の場所に作成されます。
  - synergy/ssp/inc/driver/api
- r_wdt.h が次の場所に作成されます。
  - synergy/ssp/inc/driver/instances

上記のファイルは、 WDT HAL モジュールの標準的なファイルです。これらのファイルには、プロジェクト固有の内容は含まれていません。これは WDT のドライバーファイルです。これらのファイルの内容の詳細については、 WDT HAL モジュールのドキュメントを参照してください。

WDT プロジェクトでの WDT HAL モジュールの設定情報は次の場所にあります。

- synergy_cfg/ssp_cfg/driver/r_wdt_cfg.h

上記のファイルの内容は、**[g_wdt0 Watchdog Driver on r_wdt Properties]** ビューの **[Common]** 設定に基づいています。

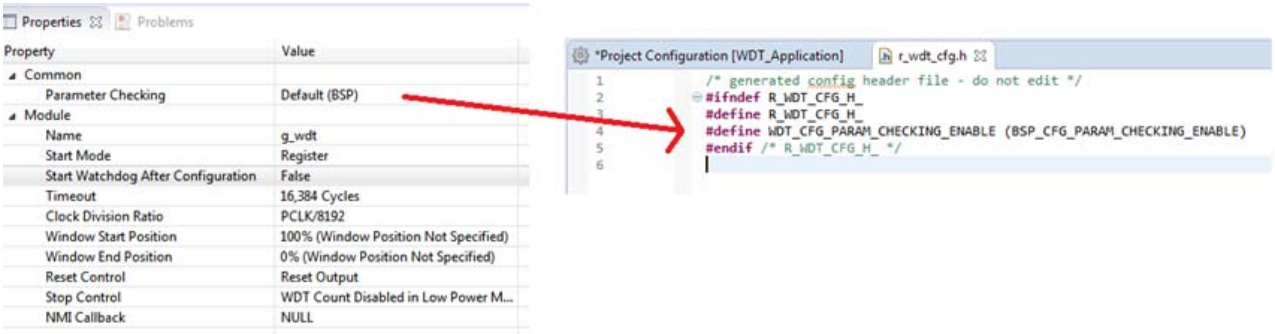


図 101: r_wdt_cfg.h の内容

注: これらのファイルは、[Generate Project Content] ボタンを押すたびに再作成され、変更内容が上書きされるため、編集しないでください。

ISDE は、WDT 用の HAL ドライバーファイルを生成するのに加えて、WDT 用の構成データが格納されたファイルと、ユーザーコードを安全に追加できるファイルも生成します。これらのファイルを以下に示します。

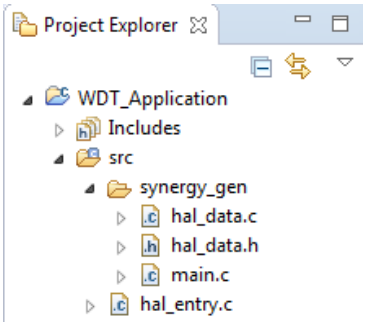


図 102: WDT プロジェクトファイル

3.3.4.1 これは、正しいヘッダーファイルを src/synergy_gen/hal_data.h でインクルードするために、WDT プロジェクトに含まれています。

hal_data.h をダブルクリックし、ファイルを開いて確認します。

hal_data.h には、ISDE で生成されたプロジェクトに必要なヘッダーファイルが格納されています。また、WDT HAL ドライバーに使用される構成、制御、 API の各構造体へのポインタを含む **g_wdt0** インスタンス構造体への外部参照も含まれます。

注: このファイルは、[Generate Project Content] ボタンを押すたびに再生成されるため、編集しないでください。

3.3.4.2 WDT プロジェクトファイル

hal_data.c をダブルクリックし、ファイルを開いて確認します。

hal_data.c には、WDT HAL ドライバーのこのインスタンス用の制御構造体である g_wdt0_ctrl が含まれています。この構造体は、ドライバーがオープンされたときに初期化されるため、初期化しないでください。

g_wdt0_cfg の内容は、 [Properties] ビューの **[g_wdt0 Watchdog Driver on r_wdt Properties]** を使用して、このファイルに設定されます。この構造体の内容に、 ISDE で行った設定が反映されていない場合は、 **[Project Configuration]** の設定が ISDE で保存されているのを確認してから、**[Generate Project Content]** ボタンを押してください。

注: このファイルは、[Generate Project Content] ボタンを押すたびに再生成されるため、編集しないでください。

3.3.4.3 この構造体は、ドライバーがオープンされたときに初期化されるため、初期化しないでください。

BSP のスタートアップコードで呼び出される main() が格納されています。 main() は、ユーザーが開発したコードが含まれている hal_entry() を呼び出します（次のファイルを参照）。main.c の内容は以下のとおりです。

```
/* generated main source file - do not edit
```

3.4 IAR Embedded Workbench for Renesas Synergy

3.4.1 IAR Embedded Workbench for Renesas Synergy の使用

このセクションでは、IAR Embedded Workbench for Renesas Synergy（IAR EW for Synergy）と Renesas Synergy Standalone Configurator（SSC）を組み合わせて使用し、 Renesas Synergy Software Package（SSP）を用いたアプリケーションを開発する方法について説明します。 SSP のアーキテクチャは、 IAR EW for Synergy と SSC をどのように使用して Synergy アプリケーションを開発するかを直接決定します。このマニュアルに含まれる SSP アーキテクチャの詳細については、次のドキュメントを参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

3.4.2 IAR EW for Synergy とは

IAR Embedded Workbench は現在、 Renesas Synergy プラットフォームに完全に統合されています。新製品の IAR EW for Synergy は、アドオン機能を提供してソフトウェア開発を簡略化し加速するほか、最適なパフォーマンスを実現し、コードサイズを最小限に抑えます。

e^2 studio と同じく、IAR EW for Synergy は安全なソースレベルの可視性を Synergy Software Package（SSP）にもたらし、さらに安全なソースレベルのデバッグを可能にします。開発者は保護されたソースコードを見ることはできますが、変更したり保存したりすることはできません。アプリケーションコードの開発後に、IAR EW for Synergy が IAR C-STAT® アナライザと C-RUN® アナライザを組み入れるので、ユーザーはこれらのツールを利用してアプリケーションコードの品質改善を進めることができます。

3.4.3 主な特長

- プロジェクト管理ツールとエディタを備えた統合開発環境
- Renesas Synergy デバイス用に高度な最適化を実現する C および C++ コンパイラとリンカ
- Renesas Synergy Standalone Configurator （SSC）に対する統合サポート
- コード分析ツール C-STAT および C-RUN を内蔵
- 広範な HW ターゲットシステムをサポート
- ソースコードと電力消費の相関関係を視覚化するパワーデバッグ
- JTAG/SWD をサポートし、ハードウェア上の RTOS 対応デバッグもサポートする C-SPY® デバッガ
- ETM トレースのサポート
- 包括的なユーザーガイドやリファレンスガイドと状況に応じたヘルプ機能
- Arm 組み込みアプリケーションバイナリインタフェース（EABI）と Arm Cortex® マイクロコントローラソフトウェアインタフェース標準 （CMSIS）に準拠

IAR EW for Synergy のダウンロードとインストールの詳しい方法については、 Synergy Gallery の IAR EW for Synergy リリースノートを参照してください。

3.4.4 Synergy Standalone Configurator（SSC）とは

Synergy Standalone Configurator （SSC）は、 Renesas e^2 studio 統合ソリューション開発環境に実装されている Eclipse Rich Client Platform（RCP）アプリケーションで、Synergy Project Generator と Synergy Project Editor が含まれています。 SSC は、クロックコンフィギュレータ、ピンコンフィギュレータ、RTOS コンフィギュレータ、SSP モジュールセレクタ / コンフィギュレータ、割り込みコントローラユニット（ICU）コンフィギュレータといった各種のコンフィギュレータを備えており、それらを IAR EW for Synergy などのサードパーティ IDE で使用することができます。

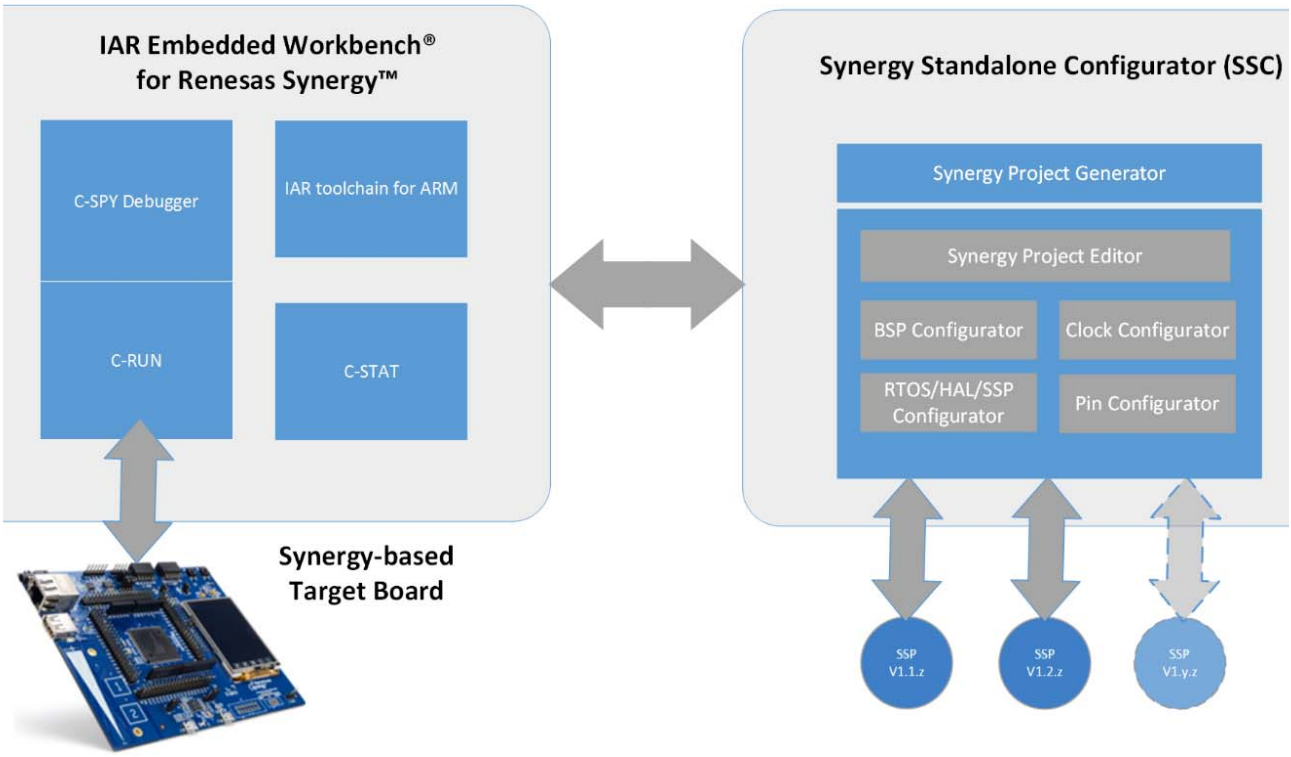


図 103: IAR EW for Synergy と SSC の機能ブロック図

SSC の機能は Renesas e^2 studio 統合ソリューション開発環境に実装されている Synergy Project Generator および Synergy Project Editor と同様であるため、使用法については e^2 studio ISDE ユーザーガイドを参照してください。

IAR EW for Synergy とともに使用する SSC と SSP をダウンロードおよびインストールする方法については、Synergy Gallery の SSC リリースノートと SSP リリースノートを参照してください。

3.4.5 ツールのインストール

ツールをインストールするには、次の手順を実行します。

1) Renesas Synergy Gallery から Renesas Synergy Standalone Configurator（SSC）をダウンロードして、インストールします。[Development Tools] の下に配置されています。デフォルトのインストールディレクトリは C:\Renesas\Synergy\SSC_<SSCversion> です。
2) Renesas Synergy Gallery から Renesas Synergy Software Package（SSP）をダウンロードして、インストールします。インストール中に、SSP のインストールディレクトリを指定するよう求められます。SSP インストーラに対し、直前に SSC をインストールしたディレクトリを指定します（たとえば、C:\Renesas\Synergy\SSC_<SSCversion>）。
3) Renesas Synergy Gallery から IAR Embedded Workbench for Renesas Synergy をダウンロードして、インストールします。次の手順で IAR Embedded Workbench をインストールします。
   a) Web ブラウザに URL https://synergygallery.renesas.com を入力し、Renesas Synergy Gallery から IAR Embedded Workbench for Renesas Synergy をダウンロードします。ライセンスを入手してライセンス番号を取得する方法についても、同じ場所に情報があります。
   b) ダウンロードしたファイルに含まれているインストーラを実行します。
   c) IAR License Manager の指示に従ってライセンス番号を入力します。

      注 : IAR EW for Synergy ライセンスにより、ユーザーは IAR Embedded Workbench のこのエディションを使用する権利を付与されますが、Synergy Standalone Configurator の使用にはまた別のライセンスが必要です。

3.4.6 IAR EW for Synergy と SSC での Renesas Synergy プロジェクトの作成

IAR EW for Synergy と SSC を使用して Synergy プロジェクトを作成するには、次の手順に従います。

1) IAR Embedded Workbench IDE で、**[Project]>[Create New Project]** を選択します。
2) **[Create New Project]** ダイアログボックスで、**[Renesas Synergy Project]** を選択し、**[OK]** をクリックします。

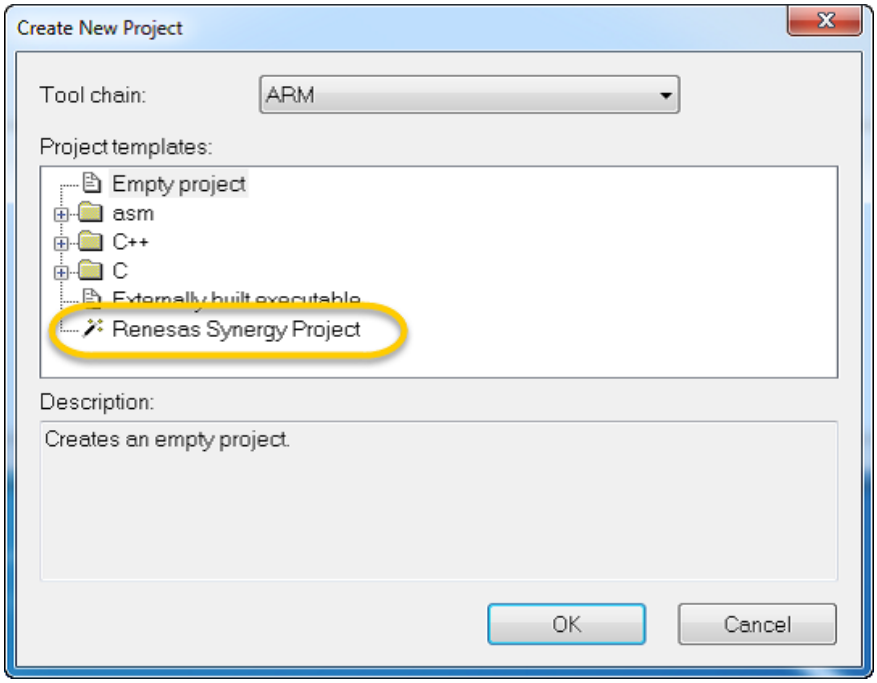


図 104: IAR EW for Synergy での新しい Synergy プロジェクトの作成

3) **[Save As]** ダイアログボックスで、ワークスペース（プロジェクトを格納するコンテナ）の適切な保存先ディレクトリ（「MyWorkspace」など）を選択し、**[Save]** をクリックします。

   注： ワークスペースは、オペレーティングシステムのルートディレクトリ（C:）に保存しないでください。

4) **[Renesas Synergy Setting]** ダイアログボックスで、次のように指定します。

   - インストールされた Synergy Standalone Configurator（SSC）の場所。デフォルトでは、C:\Renesas\Synergy\SSC_<SSCversion> にインストールされます。
   - Synergy ライセンスファイル。これは、SSC のインストールディレクトリにあります（たとえば、C:\Renesas\Synergy\SSC_<SSCversion>\internal\projectgen\arm\Licenses）。
   - 暗号化されたソースファイルをすべて復号されたソースファイルに置き換えるかどうか。この機能には、Renesas Synergy ソースライセンスが必要であることに注意してください。

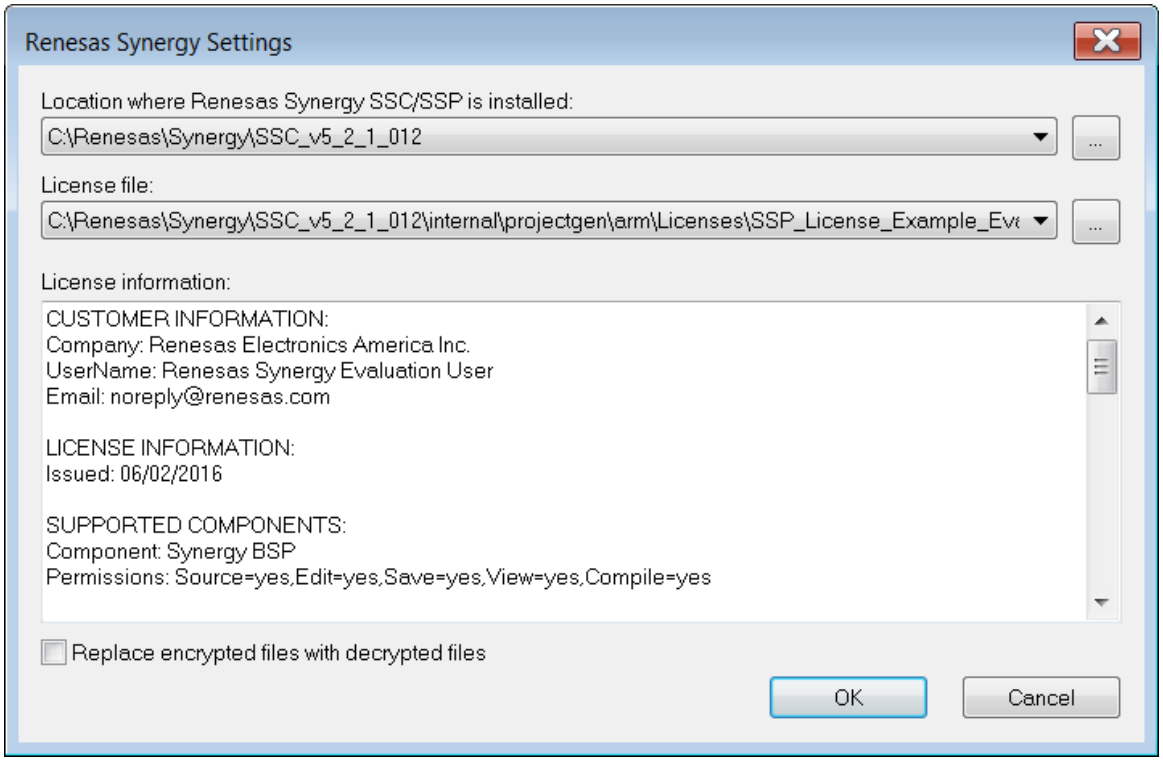


図 105: Renesas Synergy 設定 （IAR EW for Synergy）

5) **[OK]** をクリックします。

6) **[Save As]** ダイアログボックスで、プロジェクトの名前 （「MyProject」など）を指定します。

   注: プロジェクトは、オペレーティングシステムのルートディレクトリ（C:）に保存しないでください。

7) ここで IAR Embedded Workbench IDE は Renesas Synergy Standalone Configurator （SSC）に接続します。必要なボードサポートを指定します。

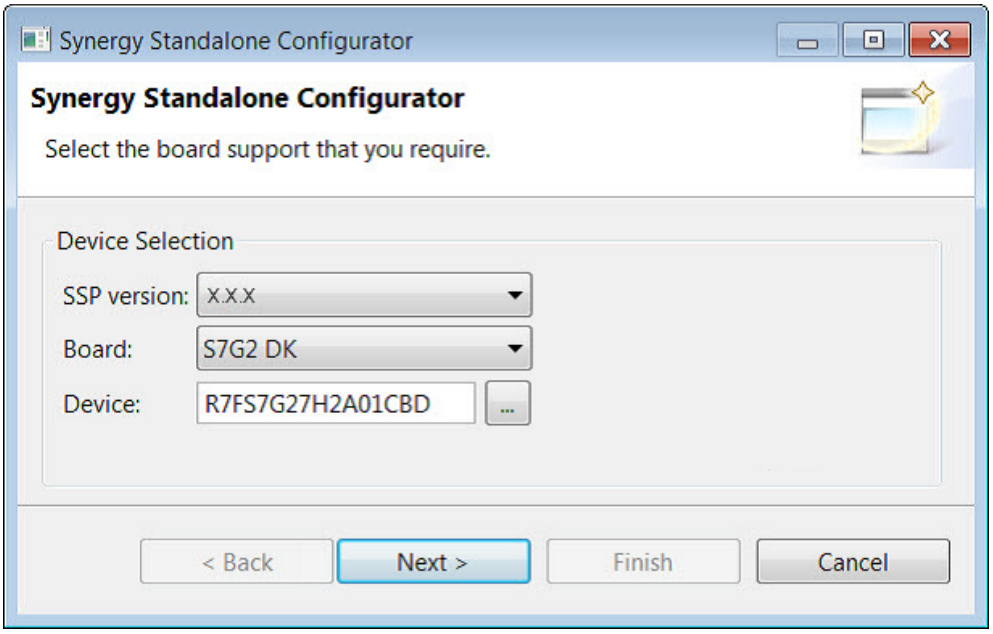


図 106: 使用する SSP、ボード、 MCU に関する SSC 選択

ダイアログ

注：ドロップダウンリストには、ご使用のコンピュータにインストール済みの SSP バージョンが表示されます。

[Next] をクリックします。

8) Synergy Software Package にはいくつかのサンプルプロジェクトが用意されており、使用するデバイスに応じたソースコードファイル、ヘッダーファイル、リンカ構成ファイルなどが含まれています。プロジェクトに追加したいサンプルパッケージを選択します。

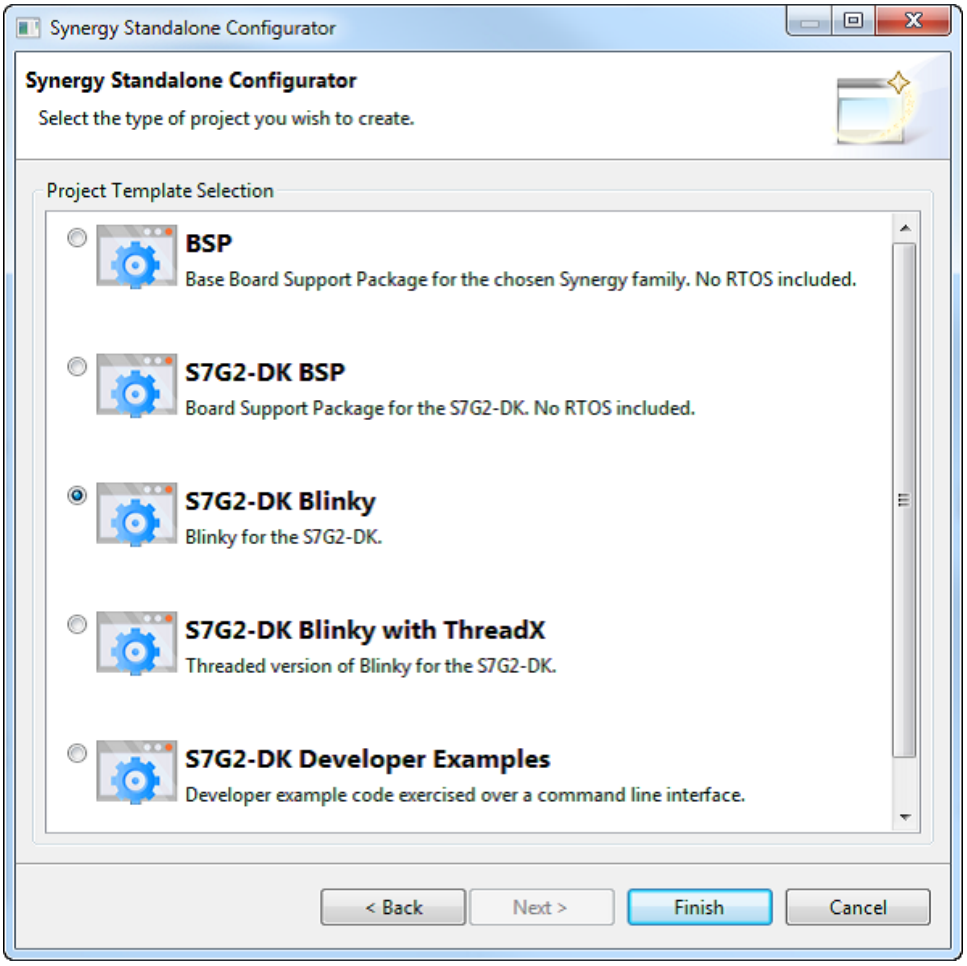


図 107: [Project Template Selection] ダイアログ

[Finish] をクリックします。

9) Synergy Project Editor が表示され、MCU ピン機能の割り当て、クロックとペリフェラルの設定、割り込み要因の割り当てを構成することができます。構成が完了したら、**[Generate Project Content]** ボタンをクリックします。するとソースコードが生成されます。

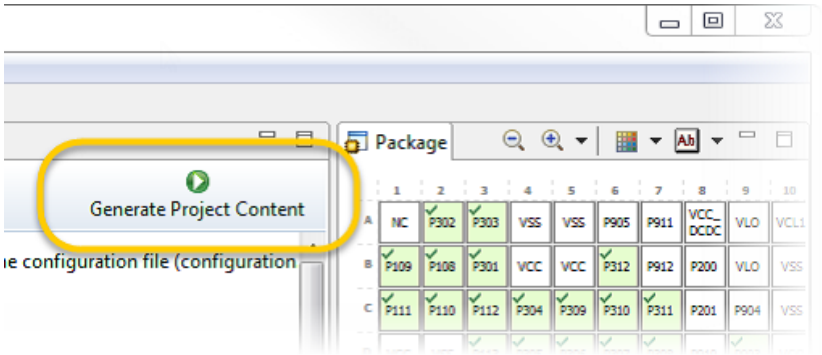


図 108: [Generate Project Content] ボタン

注 : Synergy プロジェクトの構成は、後からいつでも追加したり変更したりすることができます。

10) 数秒後、IAR EW の [Workspace] ウィンドウに Renesas Synergy プロジェクトが表示されます。

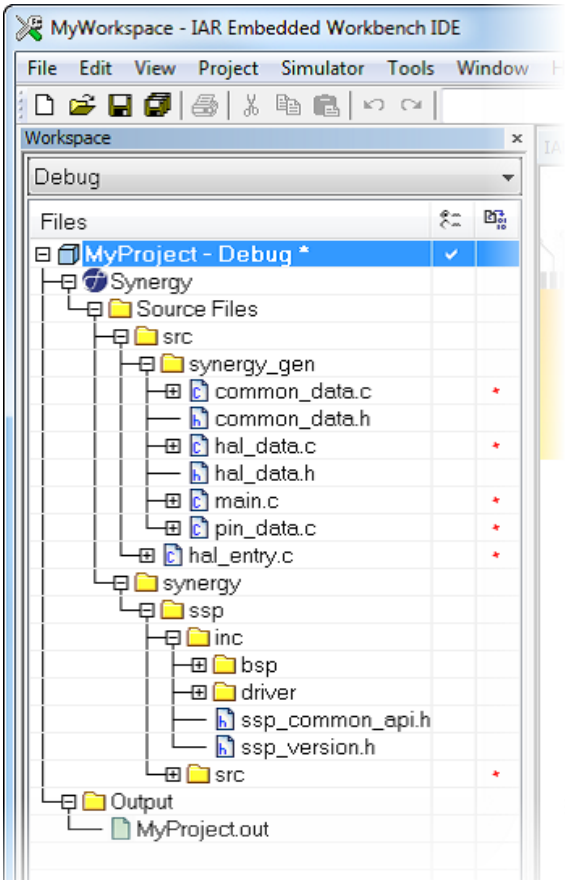


図 109: IAR EW for Synergy のワークスペースとプロジェクトの表示ウィンドウ

11) SSC を閉じた場合は、ツールバーの [Synergy Configuration] ボタンをクリックするか、

メニューから **[Renesas Synergy]>[Configurator]** を選択して開くことができます。

12) 構成設定を変更するために SSC に切り替える場合は常に、完了時に **[Generate Project Content]** ボタンをクリックします。該当するソースコードファイルが再生成されます。

13) この後は IAR Embedded Workbench IDE の標準の操作に従って、ビルドとデバッグを実行できます。IAR ウェブサイトの『IAR Embedded Workbench®IDE User Guide』を参照してください。SSC は e^2 の Synergy Project Editor と同様に動作するので、詳しい使用法については、『e^2 studio ISDE ユーザーガイド』を参照してください。 Synergy Project はデフォルトで J-Link Debug Probe 用に構成されていることに注意する必要があります。 I-jet または I-jet Trace などの他のデバッグプローブを使用する場合は、 IDE で **[Project]>[Options]>[Debugger]** を選択し、 [Driver] ドロップダウンリストから 「I-jet/JTAGjet」を選択します。

第 4 章　SSP v1.3.0 への移行

4.1 SSP v1.3.z へのアップグレード

このドキュメントでは、既存のプロジェクトを SSP v1.1.z および SSP v1.2.z から SSP v1.3.z にアップグレードする手順を説明します。

SSP v1.1.z から SSP v1.3.z へプロジェクトをアップグレードするには 2 段階で行う必要があります。最初は、プロジェクトを SSP v1.1.z から SSP v1.2.z へアップグレードします。次に、プロジェクトを SSP v1.2.z から SSP v1.3.z へアップグレードします。

4.1.1 SSP v1.1.z から SSP v1.2.z へのプロジェクトのアップグレード

次の手順に従って、プロジェクトを SSP v1.1.z から SSP v1.2.z へアップグレードします。

1) **Upgrade Documents for SSP v1.2.0.zip** を Renesas Synergy Gallery （https://synergygallery.renesas.com/）からダウンロードします。これは Gallery の [SSP] セクションの [Release Archive] タブにあります。
2) **Upgrade Documents for SSP v1.2.0.zip** を圧縮解除します。
3) アプリケーションでカスタム BSP を使用する場合は、**r11an0071eu0110-synergy-bsp-upgrade-sspv120.pdf** の説明に従って BSP パックファイルをアップグレードしてから、次の手順に進みます。
4) **r11an0072eu0110-synergy-upgrade-sspv120.pdf** に従って、プロジェクトを SSP v1.1.z から SSP v1.2.z にアップグレードします。

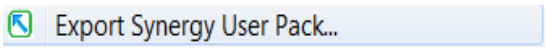
4.1.2 SSP v1.2.z から SSP v1.3.z へのプロジェクトのアップグレード

このセクションの手順に従って、プロジェクトを SSP v1.2.z から SSP v1.3.z へアップグレードします。

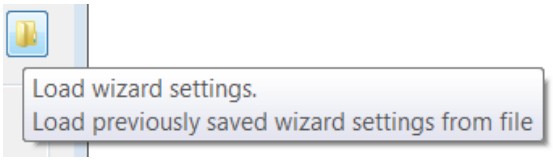
1) SSP v1.3.z で使用予定の開発環境をダウンロードしてインストールします。
   - e^2 studio v5.4.0.023 以降
   - IAR EW for Synergy v7.71.3 以降および SSC v5.4.0.023 以降
2) 元の SSP バージョン （SSP v1.2.z）と一致する SSP バージョンを開発環境にインストールします。
3) SSP v1.3.z を開発環境にインストールします。
4) プロジェクトでカスタム BSP を使用する場合は、次の手順を実行して SSP v1.3.z 互換カスタム BSP を作成してから、手順 5 に進みます。
   a) カスタムボードパックを作成するために、SSP v1.2.z を使用して作成した Synergy e^2 studio プロジェクトを開きます。
   b) このプロジェクトで、[BSP] タブの SSP バージョンを「1.3.z」に変更し、プロジェクトコンテンツを再生成します。

図 110: ->

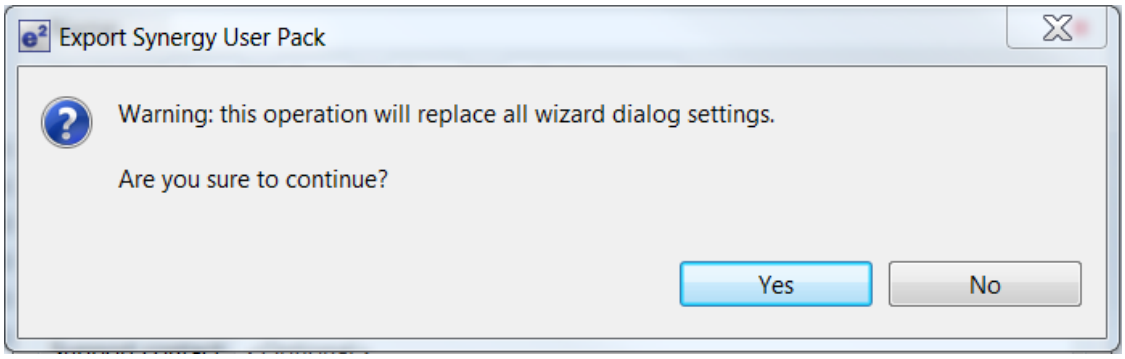
c) プロジェクトを右クリックして、[Synergy User Pack Exporter] ウィザードを開きます。



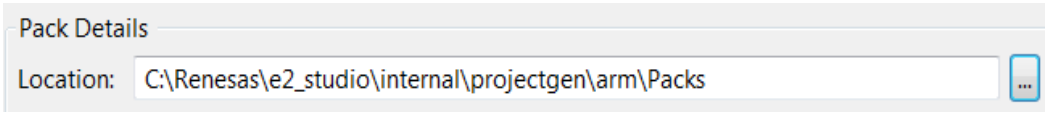
d) SSP v1.2.z 用に最初にボードパックを作成したときに保存した設定を読み込みます（これは元のボードパック内の **userpackexportsettings.xml** ファイルに格納されています。元のパックファイルを圧縮解除して userpackexportsettings.xml を取得します）。



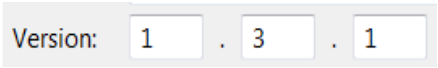
プロンプトが表示されたら [Yes] をクリックし、ダイアログのすべての設定を更新します。



e) [Location]プロパティを再確認して調整し、生成される新しいパックファイルを保存する場所を選択します。

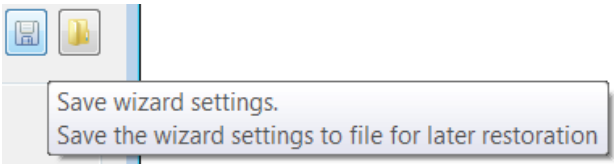


f) パックファイルのバージョンを 1.3.z に変更します（2 箇所）。



変更する 1 箇所目は [Pack Details] セクションの下です。変更する 2 箇所目は [Board Component] セクションの下です。

g) XML 設定を保存します。

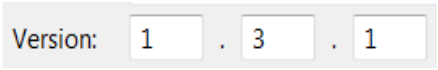


h) [Finish] をクリックすると、 SSP v1.3.z のカスタムボードパックが生成されます。

i) [Export Synergy User Pack] で自動的に選択されない場合は、新たに生成されたパックファイルを開発環境フォルダ \（e2_studio の場合は e2_studio インストールフォルダ、 IAR の場合は SSC フォルダ） \internal\projectgen\arm\Packs\ にコピーします。

5) SSP v1.2.z ベースのアプリケーションプロジェクトをインポートし、 Synergy コンテンツを生成し、ビルドと実行を行って機能を確認します。

6) Synergy コンフィギュレータの [BSP] タブに移動し、SSP バージョンのタブを元の SSP バージョン番号（1.2.z）から 1.3.z に切り替えます。次の図では例として PK-S5D9 プロジェクトを使用しています。



カスタム BSP を使用するアプリケーションの場合は、正しいボードファイルを選択するように確認してください。

7) 構成に関する問題がある場合には解決します。SSP v1.3.z のリリースノートを見直して、コンフィギュレータの変更の可能性について理解します。

8) [Generate Project Content] をクリックしてプロジェクトをビルドします。

9) コンパイルとリンクのプロセスで問題が発生した場合は解決します。SSP v1.3.z のリリースノートを見直して、リリースノートに記載されている既知の問題や使用上の注意がプロジェクトに影響を与える可能性がないかどうかを確認します。

10) アプリケーションをテストして機能を確認します。

第 5 章　モジュールの概要

各 SSP モジュールの「モジュールの概要」は、前のリリースから大幅に変更されています。新しい「モジュールの概要」では、ターゲットアプリケーションでの使用について特定のモジュールの適合性を評価するために開発者が必要なすべての情報が提供され、開発プロセスに非常に役立ちます。このような説明の目的は、ターゲットモジュールを使用して開発を始めるために必要なすべての情報を、わかりやすい場所にまとめて記載することです。

それぞれの「モジュールの概要」には次のセクションが含まれます。

1) モジュールの簡単な紹介には、短い説明、主要モジュールコンポーネントのブロック図、機能リストが含まれます。
2) API 表には、使用可能なすべての API、API の使用例、API 機能の簡単な説明が含まれます。主要なステータス戻り値のリストもあり、API コールの結果を判別する際に役立ちます。
3) 機能の概要では、主要なモジュールの動作について説明します。モジュールの重要な制限事項のリストも含まれます。
4) ISDE の [Threads] タブやスタック選択プロセスを使用してモジュールをアプリケーションに組み込む詳しい手順。
5) モジュールおよび主要な下位レベルモジュールの構成パラメータを説明する一連の表が提供され、開発者がモジュールの重要機能をすぐに把握できます。構成可能なこれらのプロパティは、MCU シリーズや SSP リリースごとに異なることに注意してください。これらの表の説明を例として参照しながら、ターゲット MCU および選択した SSP リリースの ISDE で設定できる実際のパラメータを確認してください。ピンとクロックの構成例と選択に関する情報も提供され、開発を進める際に役立ちます。
6) ターゲットモジュールを使用する単純な実装について説明し、典型的なアプリケーションで使用される手順、関連するフロー図、各手順での API の使用を示します。これによって、API の一般的な使用方法がわかり、開発者が効率よく実装を始めることができます。

モジュールガイドのアプリケーションの注意事項

これら 6 個のセクションはそれぞれのモジュールの「モジュールガイドのアプリケーションの注意事項」にも含まれます。アプリケーションの注意事項には、関連するアプリケーションプロジェクトの詳しい説明（実際の設計におけるモジュールの仕組み）が含まれるその他のセクションがあります。新しい設計の出発点または参考としてアプリケーションプロジェクトを使用すると、開発を大幅に簡略化することができます。モジュールガイドのアプリケーションの注意事項は、[Renesas Synergy Tools & Kits] ページの [Sample Code] タブの下にあります（https://www.renesas.com/en-us/products/synergy/tools-kits.html#sampleCodes モジュールガイドのアプリケーションの注意事項は随時追加されているため、新しいノートがリリースされているかどうか適宜確認してください。

モジュールガイドのモジュールの概要の使用

「モジュールの概要」によって開発を始めるために十分な詳細情報が提供されますが、設計を実装する際にはさらに多くの情報が役立つケースがあります。『SSP ユーザーズマニュアル』には、API の実装、構造、列挙などの詳細が豊富に提供されています。API リファレンスのセクションに移動し、関心があるモジュールを見つけて必要な追加情報を探します。一般的な項目の例を次に示し、それらがどこに記載されているかを説明します。

- API フレームワークの関数の詳細 -API リファレンス : フレームワークインタフェースの例 -ADC 周期フレームワーク。6.2.1.2 章に使用可能な API のリストがあります。表内の API をクリックすると、API のテンプレート、説明、パラメータ、インスタンス構造体を確認できます。
- API フレームワークの関数戻り値 -API リファレンス : フレームワークの例 -ADC 周期フレームワーク。7.2.1 章に、定義、戻り値、フレームワークで使用される内部の関数手順の詳細があります。HAL モジュールに対しても上記のトピックについて説明する章があります。時間をとってすべてのリファレンスの章にある参照文献に目を通すことを強く推奨します。そうすることで、API 実装に関する疑問の回答が、モジュールガイドの使用上の注意または関連するモジュールガイドのアプリケーション注意事項で見つからない場合に、何を参照すればよいかがわかります。

「モジュールの概要」のリストは以下のページで見つけることができます。

- フレームワークレイヤー
- HAL レイヤー

5.1 フレームワークレイヤー

ADC 周期フレームワーク

オーディオ再生フレームワークモジュール

オーディオ再生 DAC フレームワーク

オーディオ記録 ADC フレームワーク

オーディオ再生 I2S フレームワーク

BLE フレームワーク

セルラーフレームワーク

UART 通信フレームワーク

USBX™ 上の通信フレームワークモジュール

コンソールフレームワーク

暗号化フレームワーク

静電容量式タッチフレームワーク

静電容量式タッチボタンフレームワーク

静電容量式タッチスライダフレームワーク

外部 IRQ フレームワーク

FileX ポートブロックメディアフレームワーク

I2C フレームワーク

JPEG デコードフレームワーク

メッセージングフレームワーク

パワープロファイルフレームワーク

パワープロファイル V2 フレームワークモジュール

SPI フレームワーク

スレッド監視フレームワーク

タッチパネルフレームワーク

WiFi フレームワークモジュール

GUIX Synergy ポートフレームワークモジュール

Express Logic 社の NetX ポート ETHER モジュール

Express Logic 社の USBX Synergy ポートフレームワーク

NetX Duo MQTT クライアントモジュールガイド

NetX Duo TLS セッションモジュールガイド

5.1.1 ADC 周期フレームワーク

- Synergy MCU グループ用の 14 ビット A/D コンバータ （S3A7、 S124）または 12 ビット A/D コンバータ （S7G2、 S5D9）
- 複数の動作モード
- シングルスキャン
- グループスキャン
- 連続スキャン
- 複数チャネル
- S7G2 および S5D9 用の 13 チャネル （ユニット 0）、 12 チャネル （ユニット 1）
- S124 用の 18 チャネル
- S3A7 用の 28 チャネル

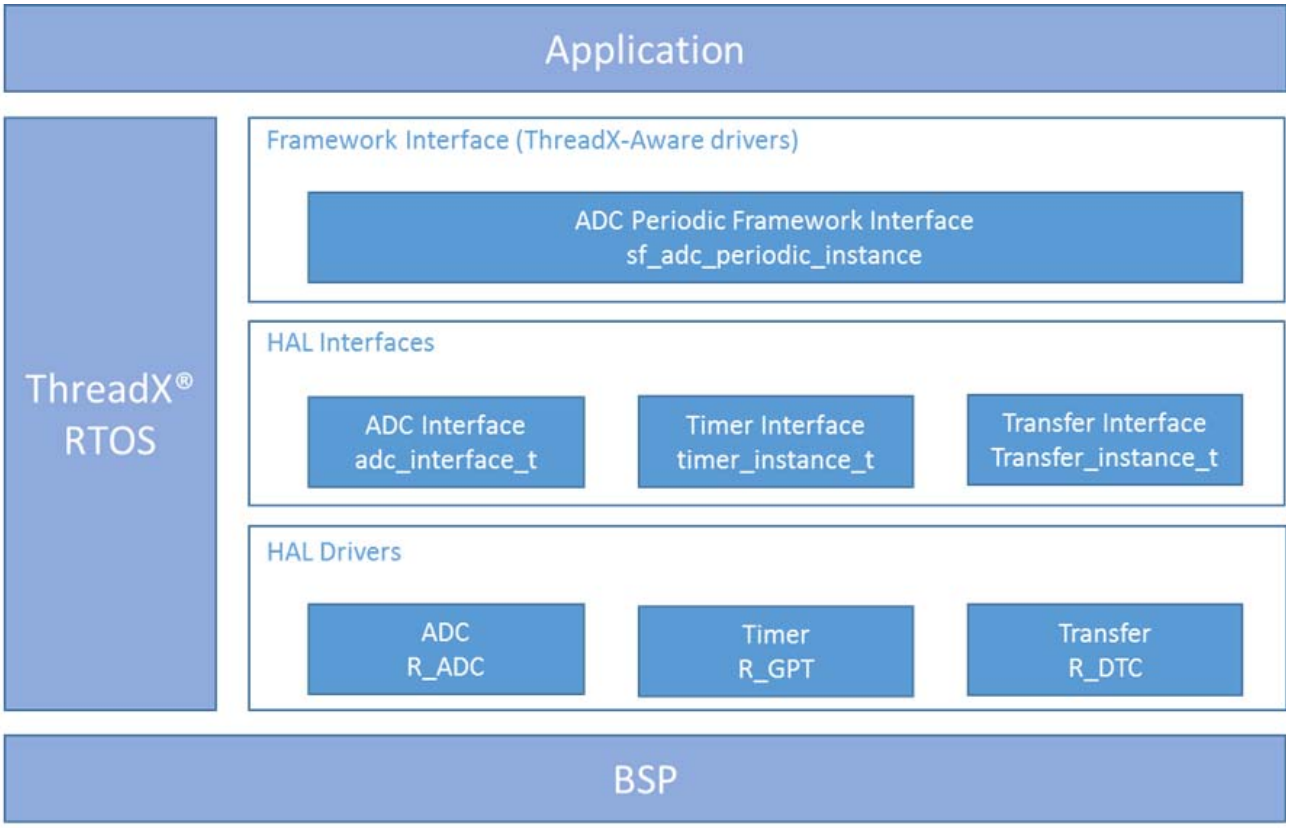


図 111: ADC 周期フレームワークモジュールのブロック図

5.1.1.1 ADC 周期フレームワークモジュール API の概要

ADC 周期フレームワークは、ADC スキャンのオープン、クローズ、開始、停止の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ADC 周期フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_adc_periodic.p_api->open(g_sf_adc_periodic.p_ctrl, g_sf_adc_periodic.p_cfg); ミューテックスを取得した後、HAL レイヤーでモジュールを初期化します。

Function Name	API の呼び出し例と説明
start	g_sf_adc_periodic.p_api->start(g_sf_adc_periodic.p_ctrl); タイマイベントを介した ADC のトリガが、ドライバーにより有効化されます。
stop	g_sf_adc_periodic.p_api->stop(g_sf_adc_periodic.p_ctrl); ハードウェアトリガ（タイマ）が、それ以上の ADC スキャンをトリガしないように停止します。
close	g_sf_adc_periodic.p_api->close(g_sf_adc_periodic.p_ctrl); チャネルミューテックスを解放し、 HAL レイヤーでチャネルを閉じます。
versionGet	g_sf_adc_periodic.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_UNSUPPORTED	コマンドが現在のメニューで見つかりませんでした。
SSP_ERR_NOT_OPEN	ドライバー制御ブロックが無効です。 SF_ADC_PERIODIC_Open を呼び出して構成します。
SSP_ERR_ASSERTION	バージョン取得エラー :p_version が NULL でした。
SSP_ERR_INTERNAL	内部 ThreadX® エラーが発生しました。これは通常、ミューテックスの作成 / 使用または内部スレッドの作成の失敗を意味します。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.1.2 ADC 周期フレームワークモジュール動作の概要

ADC 周期フレームワークモジュールは、 ADC データをサンプリングしてバッファに格納します。設定されたサンプル数のバッファリングが完了すると、アプリケーションに通知されます。ADC 周期フレームワークは以下のように機能します。

- 初期構成とスキャンプロセスの開始後、フレームワークはハードウェアタイマを使用してワンショットモードで ADC スキャンをトリガします。各スキャンは 1 つ以上のチャネルで構成できます。各スキャンが完了すると、DTC によって ADC 割り込みが捕捉され、スキャン結果がユーザーバッファに移動します。
- 各スキャンは 「サンプリングの一定回数の繰り返し」と定義され、各スキャンで行われるサンプリング回数はチャネルと等しくなります。チャネルが連続している場合（たとえば、チャネル 1、2、3、4 の場合）、データは順序どおりに取得されます。チャネルが連続していない場合（たとえば、チャネル 1、3、4、5 の場合）は、スキャンごとに、間にある未使用のチャネルからもデータがサンプリングされます。したがって、2 つ目の例では、毎回 5 個のサンプルがユーザーバッファに格納されます。
- ユーザーは、その回数のサンプリングが完了した後に通知を受ける、サンプリング繰り返し回数の総数を指定します。指定されたサンプリング繰り返し回数が実行され、各回のデータがユーザーバッファに格納されたら、バッファ内の有効なデータのインデックスを含むコールバックと、指定された繰り返し回数のサンプリングが完了したことを示すイベントが発生して、ユーザーに通知されます。
- ユーザーがスキャンプロセスを停止しない限り、引き続きタイマによってスキャンがトリガされ（AGT または GPT を使用）、データがユーザーバッファに書き込まれます。ユーザーバッファはフレームワークによって循環バッファとして扱われます。バッファの名前と長さは ISDE コンフィギュレータによって指定します。

ADC 周期フレームワークモジュールの動作に関する注意事項

1) API によってエラーが返されないようにするには、ADC HAL モジュールを構成するときに少なくとも 1 つのチャネルを選択する必要があります。
2) ADC 周期フレームワークのスキャンレート （GPT または AGT タイマ周期）を構成するとき、選択したすべてのチャネルをスキャンできる十分な長さの周期を構成してください（Synergy S7G2 MCU グループデバイスでは、各チャネル変換に約 2 マイクロ秒かかります）。
3) ADC 周期フレームワークは、各スキャンで収集したすべてのチャネルのデータをユーザー指定のバッファに格納します。指定されたサンプリング繰り返し回数が完了すると、ユーザーに通知されます。5 つのチャネルを選択していて（チャネル 1、2、3、4、5）、サンプルカウントを 3 に設定している場合は、 15 （5 x 3）のサンプルが使用可能になるとユーザーに通知されます。サンプルの順序は次のようになります。

図 112: ADC 周期動作 – サンプル順序

ADC 周期フレームワーク構成でデータバッファ長を選択するときは、バッファの長さが、生成されるサンプル数の長さの 2 倍以上（この例では 15 x 2 = 30 以上）になるようにしてください。そのようにする理由は、ユーザーアプリケーションがデータ使用可能の通知を受けた後も、引き続き新しいデータがサンプルレートでバッファリングされるためです。このバッファは循環バッファとして扱われるため、データが意図せず上書きされる可能性があります。つまり、バッファサイズが生成されるサンプル数以下の場合は、アプリケーションでデータが使用可能になる前にデータが上書きされてしまいます。

アプリケーションコールバックによって、有効なデータが存在するバッファ内の適切な位置を指すインデックスが通知されます。

ADC 周期フレームワークモジュールの制限事項

ADC 周期フレームワークでは現在、以下の機能はサポートされていません。

- グループスキャンモードの使用
- DMA の使用
- このフレームワークで使用する ADC チャネルを構成するとき、他の使用可能なチャネルも合わせて選択する場合は、温度センサーと電圧センサーを選択しないでください。温度センサーだけ、電圧センサーだけ、または任意の数の通常 ADC チャネルを使用できます。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

5.1.1.3 アプリケーションへの ADC 周期フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ADC 周期フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

ADC 周期フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。（ADC 周期フレームワークのデフォルト名は g_sf_adc_periodic0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

ADC 周期フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_adc_periodic0 ADC Periopdic Framework	**Threads**	**New Stack> Framework> Analog> ADC Periodic Framework on sf_sdc_periodic**

次の図に示すように、sf_adc_periodic の ADC 周期フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

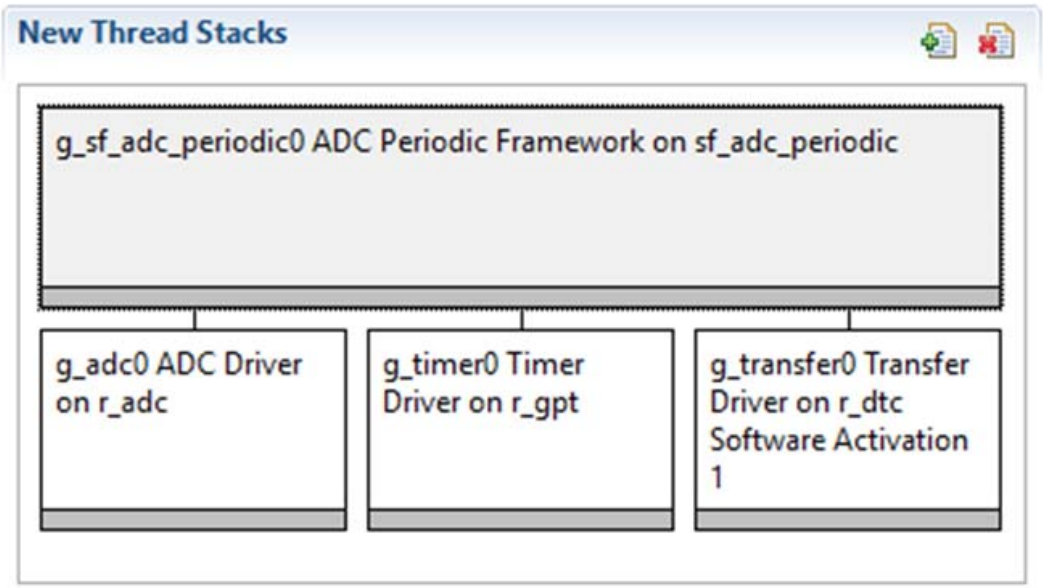


図 113: ADC 周期フレームワークモジュールのスタック

5.1.1.4 ADC 周期フレームワークモジュールの構成

ユーザーは必要な動作に合わせて ADC 周期フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは**ロック**アイコンで識別され、ISDE 内の **[Properties]** ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、

[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることに注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_adc_periodic の ADC 周期フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_sf_adc0_periodic0	ADC 周期フレームワークモジュール名
Name of the data-buffer to store samples	Default: g_user_buffer	サンプルを格納する 16 ビットデータバッファの名前
Length of the data-buffer	Default: 128	データが格納されるバッファの長さ
Number of sampling iterations	Default: 10	繰り返しごとに取得されるサンプル数
GPT Timer channel used to trigger the scan	Default: Channel 0	ELC イベントエントリ scan_trigger の生成に使用されるチャネル
Callback	Default: g_adc_framework_user_callback	サンプリング繰り返し回数のデータがバッファに格納された後に呼び出されるユーザー関数

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、異なる ADC ユニットの選択や平均機能の有効化が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *低レベルモジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する* **[PROPERTIES]** *ウィンドウを調べることで決定されます。*

ADC HAL モジュール (r_adc) は ADC ユニットを構成するために使用されます。ほとんどのデフォルトプロパティが許容されます。割り込みを有効にし、名前をモジュールに割り当てる必要があります。この名前はアプリケーションコードがユニットを参照するために使用されます。

ADC HAL モジュールの構成設定

r_adc の ADC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_adc0	モジュール名
Unit	Default: 0 (1 for S7G2 MCU)	使用する ADC ユニットを指定します。S7G2 MCU には、0 と 1 の 2 つのユニットがあります。
Resolution	8-bit, 10-bit, 12-bit, 14-bit (Select as per the board)	このユニットの変換解像度を指定します。
Alignment	Right, Left (Default: Right)	変換結果のアラインメントを指定します。
Clear After read	On, Off (Default: On)	変換結果を読み取った後で、結果レジスタを自動的にクリアするかどうかを指定します。
Mode	Single Scan, Continuous scan, group scan (Default: Single Scan)	この ADC ユニットを使用するモードを指定します。
Channel Scan Mask Channel 0-27, Temperature sensor, Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、 0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。

ISDE Property	Value	説明
Normal/Group A trigger	None, Asynchronous external trigger, ELC event, Software	このユニットに使用するトリガタイプを指定します。使用モードがグループモードの場合、このフィールドは、グループ A トリガを設定するために使用されます。***注:*** *グループモードで有効な唯一のオプションは ELC トリガです。*
Group B trigger	Default: ELC event	グループ B トリガを指定します。このオプションは、モードとしてグループモードが選択されている場合のみ有効です。
Add Average Count	Disabled, Enabled Default: Disabled	このユニット内のいずれかのチャネルに対して、加算または平均化を行う必要があるかどうかを指定します。実際のチャネルはチャネルマスク add_mask. を使用して指定します。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ADC HAL モジュールの構成設定 （続き）

ISDE Property	Value	説明
Group Priority	Default: Group A cannot interrupt Group B, Group A can interrupt group B; Group B scan restarts at next trigger, Group A cannot interrupt Group B, Group A can interrupt group B; Group B scan restarts immediately Group A cannot interrupt Group B, Group A can interrupt group B; Group B scan restarts and scans continuously	進行中のグループ B のスキャンをグループ A のトリガで割り込むことができるかどうか、グループ A のトリガで中断するかどうか、または一時停止してグループ A のスキャンを許可し、グループ A のスキャンが完了したらすぐに再開するかどうかを決定します。***注:*** *このフィールドは、グループモードでのみ有効です。*

ISDE Property	Value	説明
Addition/ Average mask Channel 0-27, Temperature sensor, Voltage Sensor	Disabled, Enabled Default: Disabled	このフィールドは、add_average_count が有効になっている場合にのみ有効です。このフィールドは、平均化または合計するチャネル結果を決定するために使用します。
Sample Hold Mask Channel 0-2	Disabled, Enabled Default: Disabled	チャネル 0、1、および 2 のどれが、sample_hold_states. で指定される更新されたサンプルアンドホールド状態値を使用するかを決定します。このフィールドは、チャネル 0、1、および 2 のデフォルトのサンプルアンドホールドカウント値を変更する場合にのみ設定する必要があります。
Sample Hold States	4-255	チャネル専用のサンプルアンドホールド回路用の、更新されたサンプルアンドホールドカウントを指定します。このフィールドは、sample_hold_mask が 0 でない場合にのみ有効です。チャネル 0、1、2 のみに専用のサンプルアンドホールド回路があります。 ***注：****値をサンプリングするデフォルトの状態数 (24) を変更するにはこのフィールドを使用します。各状態は、1/ADCLK 時間に等しくなります。*
Callback	NULL	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。プロトタイプ sf_adc_periodic_cb を使用してユーザーコールバック関数を定義し、上書きする必要があります。
Scan End Interrupt Priority	Disabled, Priority 0-15 (Default: BSP)	adc モジュールを有効にするために割り込みを指定します。

ISDE Property	Value	説明
Scan End Group B interrupt priority.	Disabled, Priority 0-15 (Default: BSP)	グループ B で adc モジュールを有効にするために割り込みを指定します。

GPT HAL モジュールの構成設定

ADC 周期フレームワークモジュールで使用される GPT タイマドライバーの構成パラメータは、多くの場合、デフォルト設定で使用できます。通常は、少数の設定のみをデフォルト設定から変更する必要があります。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、GPT HAL モジュールを構成するために設定しなければならないオプションの一覧を示します。

GPT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
GPT0 COUNTER OVERFLOW	Disabled, Priority 0-15 (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_timer0	モジュール名
Channel	0	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Mode	Periodic	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Period Value	Default: 10	ADC スキャンをトリガするタイマ周期を設定します。
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz (Default: Milliseconds)	期間を設定する単位

ISDE Property	Value	説明
Duty Cycle Value	Default: 50	デューティサイクル設定の値
Duty Cycle Unit	Raw count, unit %, Unit % x 1000 (Default: Raw Counts)	デューティサイクル値で使用される単位
Autostart	False	このフィールドはADCフレームワークによって事前設定されており、ロックされています。
GTIOCA Output Enabled	True, False (Default: False)	この設定により出力が有効化または無効化されます。
GTIOCA Stop Level	Pin level High, Low or Retained (Default: Low)	遷移後の出力ピンレベルを決定します。
GTIOCB Output Enabled	True, False (Default: False)	この設定により出力が有効化または無効化されます。
GTIOCB Stop Level	Pin level High, Low or Retained (Default: Low)	遷移後の出力ピンレベルを決定します。
Callback	NULL	このフィールドはADCフレームワークによって事前設定されており、ロックされています。
Interrupt Priority	Priority 0:15	割り込み優先順位

***注:** 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

DTC HAL モジュール構成設定

DTC HAL モジュールは、 ADC 周期フレームワークによって内部で構成されます。このため、ユーザーはモジュール名の指定を除いて、構成を行う必要はありません。詳細については、[SSP configuration] ウィンドウ内の対応するプロパティ設定を参照してください。

ADC 周期フレームワークモジュールのピン構成

チャネルにアクセスするには、 ADC チャネルを ISDE の **[Pins]** タブで設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示しています。

ADC HAL モジュールでのピンの選択

Resource	ISDE Tab	Pin selection Sequence
ADC	**Pins**	Select **Peripherals > Analog ADC > ADC0\1 > AN_XX**

注: *内部温度センサーおよび内部電圧センサーの場合、ピン構成は必要ありません。*

5.1.1.5 アプリケーションでの ADC 周期フレームワークモジュールの使用

シンプルな ADC 周期フレームワークモジュールを構築する際の重要な要素は、スタックの選択と構成、タイマ（GPT または AGT）の選択、コールバック関数 p_callback の本文の定義、フレームワークの初期化、定義された ADC チャネルのスキャン、アプリケーションで必要な取得済みデータに対する操作、定期的な再スキャン、 ADC 測定が完了した場合の停止です。アプリケーションで外部 ADC 周期フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して ADC を初期化します。
2) start API を使用してチャネルのスキャンを開始します。
3) stop API を使用してスキャンを停止します。
4) read API を使用して変換結果を読み取ります。
5) close API を使用してインスタンスを閉じます。

多くの場合、スキャンは周期的に繰り返されたり、各スキャン操作の合間に停止されたりします。これらの一般的な手順を、次の図の通常の動作フロー図に示します。

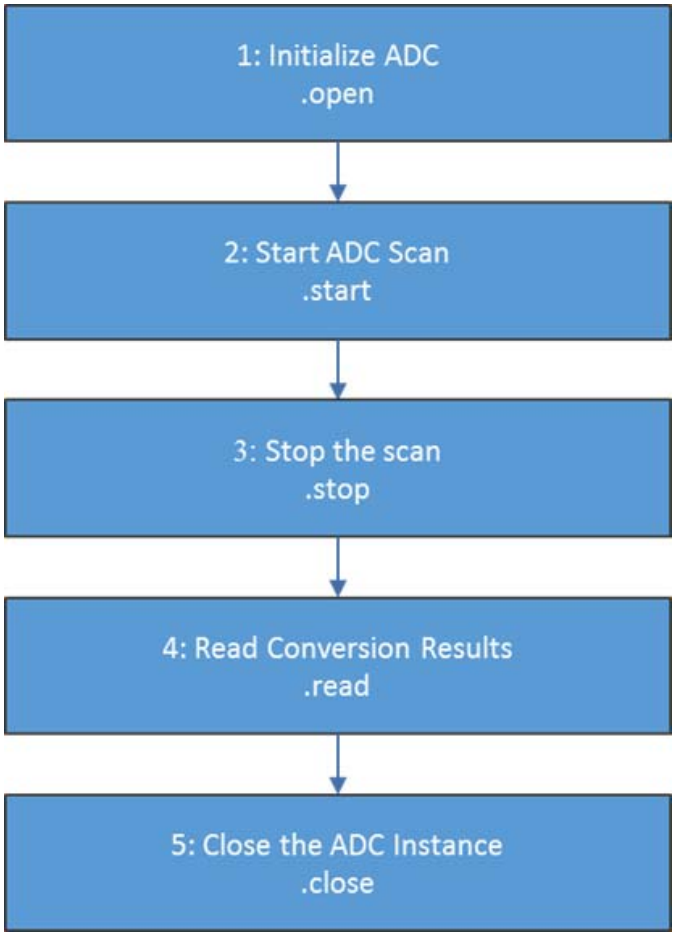


図 114: ADC 周期フレームワークモジュールのアプリケーションプロジェクトフロー図

5.1.2 オーディオ再生フレームワークモジュール

オーディオ再生フレームワークモジュールは、以下の機能に対応します。

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。
- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。
- ソフトウェア音量制御。
- 一時停止と再開機能。
- スケーリング。たとえば、符号付き 16 ビット PCM データを符号なし 12 ビット DAC の範囲にシフトします。
- 複数のストリームについての基本的なミキシング。

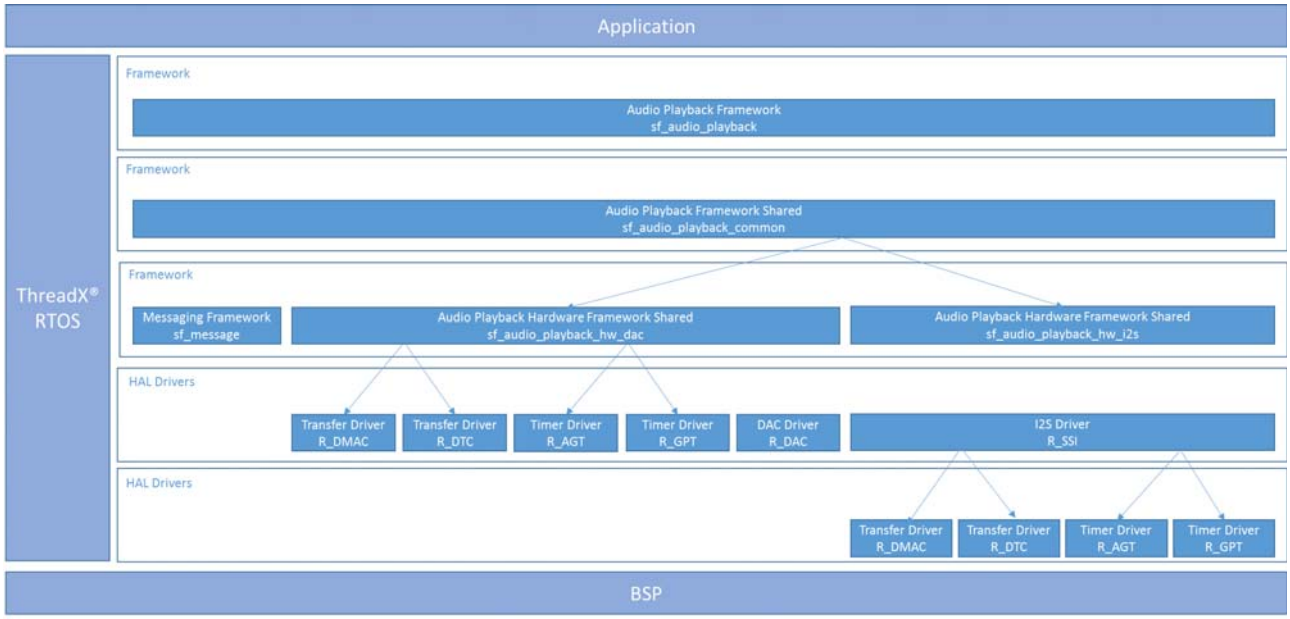


図 115:　再生フレームワークのスタックオプション

5.1.2.1 オーディオ再生フレームワークモジュールの API の概要

オーディオ再生フレームワークモジュールは、オープン、開始、再生、停止などの操作の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ再生フレームワークの API 要約

Function Name	API の呼び出し例と説明
.open	g_sf_audio_playback0.p_api->open(g_sf_audio_playback0.p_ctrl, g_sf_audio_playback0.p_cfg); 読み書きおよび制御のためにデバイスチャネルを開きます。
.start	g_sf_audio_playback0.p_api->start(g_sf_audio_playback0.p_ctrl); オーディオ再生ハードウェアを開始します。

Function Name	API の呼び出し例と説明
.stop	g_sf_audio_playback0.p_api->stop(g_sf_audio_playback0.p_ctrl); オーディオ再生ハードウェアを停止します。
.play	g_sf_audio_playback0.p_api->play(g_sf_audio_playback0.p_ctrl, &buffer, length); オーディオバッファを再生します。
.dataTypeGet	g_sf_audio_playback0.p_api->dataTypeGet(g_sf_audio_playback0.p_ctrl, &data_type); 指定されたポインタ data_type. に、指定されたデータ型を格納します。
.close	g_sf_audio_playback0.p_api->close(g_sf_audio_playback0.p_ctrl); オーディオモジュールを閉じます。
.versionGet	g_sf_audio_playback0.p_api->versionGet(&version); ドライバーのバージョンを返します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、SSP ユーザーズマニュアルで関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	ポインタが NULL またはパラメータが無効です。
SSP_ERR_OUT_OF_MEMORY	一度に開くストリームの数は、SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS に制限されています。この数値を超えると、メモリ不足のエラーが発生します。

Name	説明
SSP_ERR_TIMEOUT	再生が終了する前にタイムアウトが発生しました。
SSP_ERR_NOT_OPEN	ストリーム制御ブロック p_ctrl が初期化されていません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.2.2 オーディオ再生フレームワークモジュールの動作の概要

オーディオ再生フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生フレームワークのスレッドとパブリックオーディオ再生フレームワーク API とのやり取りを表すフローチャートを示します。

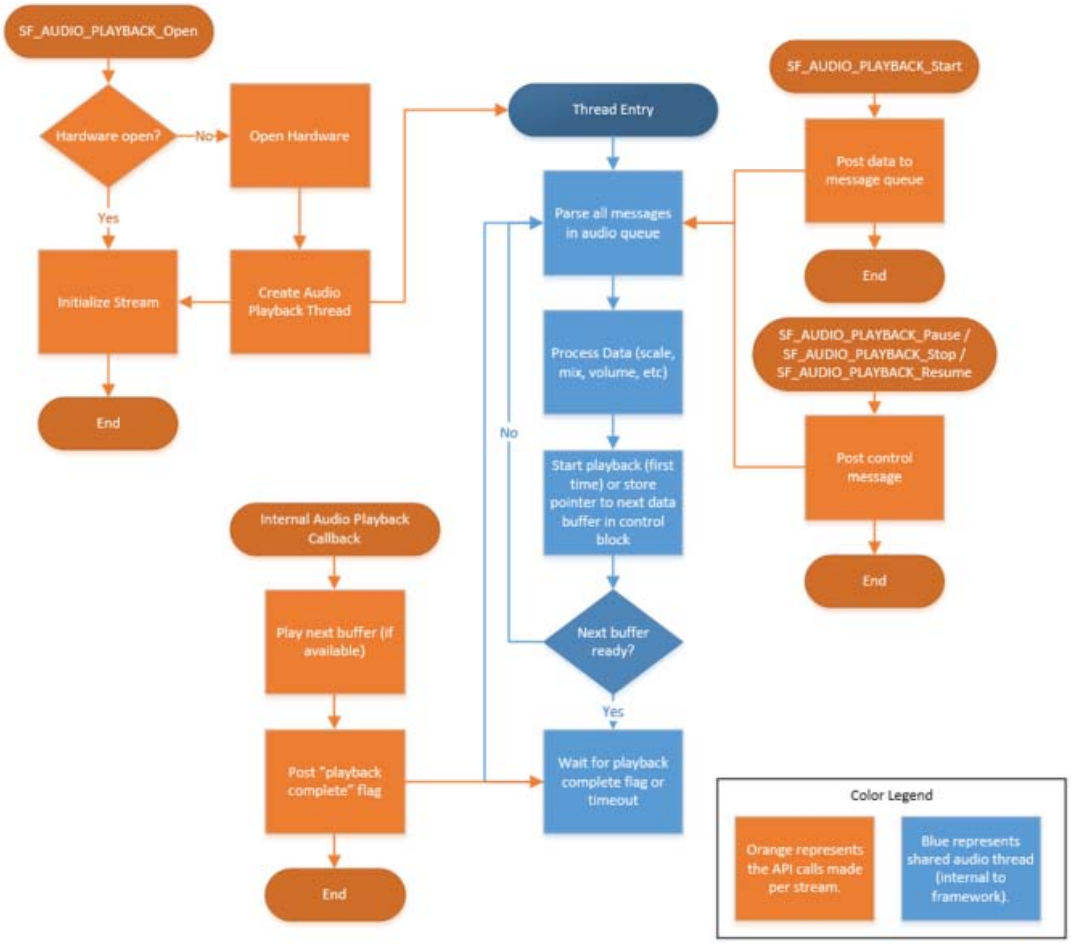


図 116:　オーディオ再生フレームワークフローチャート

オーディオ再生フレームワークの推奨される使用方法を以下に示します。

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore). これは **[Threads]** タブで行うことができます。初期値を 2 に設定します（オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータメッセージを保持できます）。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback). オーディオ再生フレームワークインスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。
- メインループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージングフレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。

オーディオ再生フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。 2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

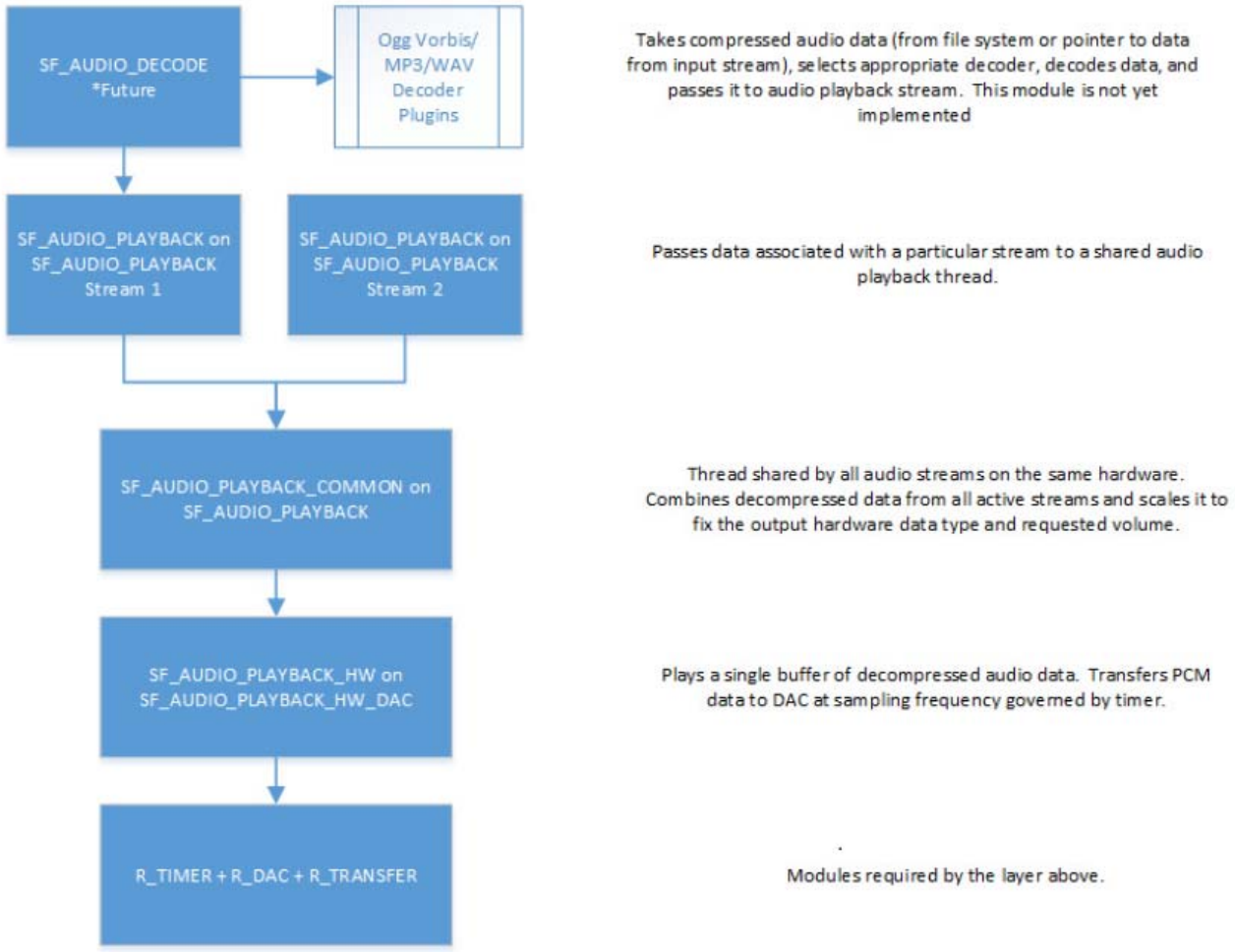


図 117: 複数オーディオストリームの実装

オーディオ再生フレームワークモジュールの動作に関する注意事項

メッセージの構成

[Messaging] タブのメッセージフレームワークコンフィギュレータを使用して、メッセージフレームワークを設定します。

1) オーディオ再生イベントクラスをハイライトします。
2) 新しいサブスクライバを追加します。以下の構成を選択して、sf_audio_playback モジュール上のオーディオ再生フレームワークの [Properties] タブでメッセージクラスインスタンスが開始インスタンスと終了インスタンスの間に適切に設定されていることを確認します。
   - スレッド : アプリケーション内の任意のスレッド。
   - 開始 : アプリケーションで使用されている最初のオーディオインスタンス。
   - 終了 : アプリケーションで使用されている最後のオーディオインスタンス。

3) オーディオ再生サブスクライバーで新しいサブスクライバーをハイライトします。シンボル名を記録します。
4) **[Threads]** タブに戻ります。
5) HAL/ 共通のオーディオ再生フレームワーク共有モジュールをハイライトして、オーディオメッセージキュー名をオーディオ再生サブスクライバーのシンボル名に設定します。

I2S 実装の使用

オーディオフレームワーク I2S ハードウェアポートには、I2S ドライバーモジュールに対する依存関係があります。 I2S ドライバーモジュールは DTC を使用して高速化できます （推奨）。

- I2S ドライバーモジュールの統合ソリューション開発環境プロパティを設定します。
  - オーディオクロック周波数（ヘルツ単位）を、使用されている入力オーディオクロックの周波数に設定します。
  - サンプリング周波数（ヘルツ単位）をオーディオデータのサンプリング周波数に設定します。
  - データビットとワード長を 16 ビットに設定します （オーディオフレームワークでは 16 ビットのみが受け入れられます）。
  - SSIn TXI および SSIn INT 割り込みを有効にします。
- （推奨） r_dtc 上の転送モジュールは自動で追加されます。

DAC 実装の使用

オーディオフレームワーク DAC ハードウェアポートには、タイマ、 DAC、および転送 API モジュールに対する依存関係があります。

- タイマモジュールを追加します。
  - 周波数 （Hz 単位）をオーディオデータのサンプリング周波数に設定します。
  - DTC を転送モジュールとして使用する場合は （推奨）、割り込みを有効にします。
- DAC モジュールを追加します。
- r_dtc 上の転送モジュールを追加します。
  - DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC->DADRn[0] に設定し、 DAC チャネル 1 を使用する場合は &R_DAC->DADRn[1] に設定します。
  - アクティベーションソースを上で選択したタイマ割り込みに設定します。

動作に関するその他の注意事項

- 使用されるキューは、[Audio Playback Framework Shared on sf_audio_playback] の [Properties] で指定された名前と一致する必要があります （デフォルトは g_sf_audio_playback_queue).

オーディオ再生フレームワークモジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.2.3 アプリケーションへのオーディオ再生フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ再生フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』の最初のいくつかのセクションを参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

オーディオ再生をアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（オーディオ再生のデフォルト名は g_sf_audio_playback. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

オーディオ再生の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback Audio Playback Framework on g_sf_audio_playback	**Threads**	**New Stack> Framework> Audio> Audio Playback Framework on g_sf_audio_playback**

次の図に示すように、sf_audio_playback のオーディオ再生フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

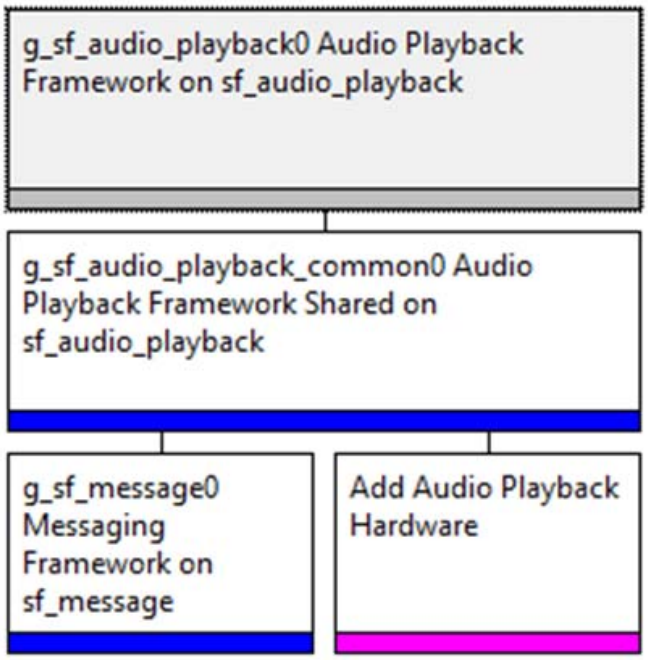


図 118:　オーディオ再生 DAC フレームワークのスタック

5.1.2.4 オーディオ再生フレームワークモジュールの構成

オーディオ再生フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_audio_playback 上のオーディオ再生フレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します。
Buffer Size Bytes	512	バッファサイズのバイトの選択。
Maximum Number of Streams	1	最大ストリーム数の選択。
Thread Stack Size	512	スレッドスタックサイズの選択。
Name	g_sf_audio_playback0	モジュール名。
Message Class Instance	0	メッセージクラスインスタンスの選択。
Callback	NULL	コールバックの選択。

注: *例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *低レベルモジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。*

オーディオ再生フレームワークの低レベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

sf_audio_playback_common 上のオーディオ再生フレームワーク共有の構成

ISDE Property	Value	説明
Name	g_sf_audio_playback_common0	モジュール名
Thread Priority	3	スレッドのプライオリティ（編集しない）
Audio Message Queue Name	g_sf_audio_playback_queue	メッセージキュー名

sf_message 上のメッセージフレームワークの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します。
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	メッセージサブスクライバーの ThreadX メッセージキューのサイズをバイト単位で指定します。この値は、すべてのメッセージキューに適用されます。
Name	g_sf_message0	メッセージングフレームワークモジュール制御ブロックインスタンスの名前。
Work memory size in bytes	2048	作業メモリサイズをバイト単位で指定します。小さい数を選択すると、同時に割り当てることができるバッファの数が少なくなります（総バッファ数は sf_message_ctrl_t::number_of_buffers). で確認できます）。この値が、同時にポストできるピークメッセージ数よりも小さい場合、フレームワークでバッファ割り当てが失敗し、システムのパフォーマンスに影響を与えます。
Pointer to subscriber list array	p_subscriber_lists	サブスクライバーリスト配列へのポインタ名を指定します。
Name of the block pool internally used in the messaging framework	sf_msg_blk_pool	フーレムワークが制御ブロック内に作成するメモリブロックメモリの名前です。このパラメータは、デバッグ目的で有用な可能性がありますが、メモリを節約するために NULL を指定できます。

sf_audio_playback_hw_dac 上のオーディオ再生ハードウェアフレームワーク共有

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します。

ISDE Property	Value	説明
DMAC Support	Disabled, Enabled (Default: Disabled)	DMAC サポートの選択。
Name	g_sf_audio_playback_hw0	モジュール名。

r_dmac 上の転送ドライバーの構成 （転送ドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer0	モジュール名
Channel	0	チャネルの選択
Mode	Normal	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source	Software Activation, Peripheral Events (Default: Software Activation)	アクティベーションソースの選択
Auto Enable	False	自動有効の選択

ISDE Property	Value	説明
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択

r_dtc 上の転送ドライバーの構成設定 （転送ドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択

ISDE Property	Value	説明
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events (Default: Software Activation 1)	アクティベーションソースの選択
Auto Enable	True, False (Default: True)	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

r_agt 上のタイマドライバーの構成設定 （タイマドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled(Default: BSP)	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	物理ハードウェア チャネル。

ISDE Property	Value	説明
Mode	Periodic	警告：ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。
Period Value	10	「タイマ期間の計算」を参照
Period Unit	Hertz	「タイマ期間の計算」を参照
Auto Start	FALSE	設定後にタイマを起動するには、true に設定します。start を呼び出すまで測定を無効状態にするには false に設定します。
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub(Default: PCLKB)	AGT カウンタのクロックソース。
AGTO Output Enabled	True, False(Default: False)	AGT 用に設定されたポートピン（AGTO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
AGTIO Output Enabled	True, False(Default: False)	AGT 用に設定されたポートピン（AGTIO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
Output Inverted	True, False(Default: False)	出力信号 low を開始するには false を設定します。出力信号 high を開始するには true を設定します。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid)(Default: Disabled)	タイマ割り込み優先順位。0が最高のプライオリティです。

r_gpt 上のタイマドライバーの構成設定 （タイマドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	チャネルの選択。
Mode	Periodic	警告：ワンショット機能は、 GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。
Period Value	10	「タイマ期間の計算」を参照
Period Unit	Hertz	「タイマ期間の計算」を参照

ISDE Property	Value	説明
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 (Default: Unit Raw Counts)	デューティサイクル単位の選択
Auto Start	FALSE	設定後にタイマを起動するには、true に設定します。start を呼び出すまで測定を無効状態にするには false に設定します。
GTIOCA Output Enabled	True, False (Default: False)	GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	タイマが停止したときの出力ピンレベルを制御します。
GTIOCB Output Enabled	True, False (Default: False)	GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	タイマが停止したときの出力ピン レベルを制御します。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択

r_dac 上の DAC ドライバーの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータ エラー チェックを有効または無効にします。
Name	g_dac0	モジュール名。
Channel	0	出力 DA0 の場合は 0、出力 DA1 の場合 1 を設定します。
Synchronize with ADC	Enabled, Disabled (Default: Disabled)	A/D コンバータ (ADC) モジュールを使用した干渉防止同期の場合は true を設定します。アナログモジュール間の電源干渉が問題にならないか、DAC モジュールによる非同期変換が望ましい場合は false を設定します。

Parameter	Value	説明
Data Format	Right Justified	12 ビットデータ値をビット 11 から 0 にロードする（右詰め）場合は 0 を設定します。12 ビットデータ値をビット 15 から 4 にロードする（左詰め）場合は 1 を設定します。
Output Amplifier	Enable, Disable (Default: Disable)	出力アンプハードウェア機能が望ましい場合は true を設定します。出力アンプハードウェア機能をバイパスする場合は false を設定します。

sf_audio_playback_hw_i2s 上のオーディオ再生フレームワーク共有の構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_sf_audio_playback_hw0	モジュール名。

r_ssi 上の I2S HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_i2s0	モジュール名。
Channel	0	物理ハードウェアチャネル。
Audio Clock Frequency (Hertz)	2,822,400	I2S クロックの作成に用いられる入力オーディオクロック周波数。1 ～ 128(sampling_freq_hz * word_length_in_bits)の範囲で設定する必要があります。
Sampling Frequecy (Hertz)	44,100	オーディオデータのサンプリング周波数。

Parameter	Value	説明
Data Bits	8, 16, 18, 20, 22, 24 bits (Default: 16 bits)	オーディオデータのビット深度（オーディオデータのサンプル 1 点のビットサイズ）。
Word Length	8 bits, 16, 24, 32 (Default: 16 bits)	オーディオデータのワード長、少なくともビット深度と同じサイズである必要があります（データビットフィールド）。
WS Continue Mode	Enabled, Disabled (Default: Disabled)	WS 続行モードを有効化すると、ペリフェラルがアイドル状態になった場合でも、ワード選択ラインの出力が継続して行われます。ペリフェラルがアイドル状態になった場合にワード選択ラインの出力を停止するには、これを無効化します。
Name of I2S callback function to be defined by user	NULL	ユーザーコールバック関数はopenで登録する必要があります。全送信データの送信後に送信FIFO最高値に達した場合、あるいは受信が完了した（要求されたバイト数が受信された）場合に、割り込みサービスルーチン(ISR)からコールバックが呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	送信割り込み優先順位の選択

Parameter	Value	説明
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	受信割り込み優先順位の選択
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	アイドル/エラー割り込み優先順位の選択

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択。
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Fixed	宛先アドレスモードの選択。
Source Address Mode	Incremented	ソースアドレスモードの選択。

Parameter	Value	説明
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events (Default: Software Activation 1)	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択。

Parameter	Value	説明
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Incremented	宛先アドレスモードの選択。
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events (Default: Software Activation 1)	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。

Parameter	Value	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択

r_agt 上の AGT HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択
Mode	Periodic	モードの選択
Period Value	2,822,400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Auto Start	FALSE	オートスタートの選択
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub (Default: PCLKB)	クロックソースの選択
AGTO Output Enabled	True, False (Default: False)	AGTO 出力の選択
AGTIO Output Enabled	True, False (Default: False)	AGTIO 出力の選択

Parameter	Value	説明
Output Inverted	True, False (Default: False)	出力の反転の選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択

r_gpt 上の GPT HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択
Mode	Periodic	モードの選択
Period Value	2,822,400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 (Default: Unit Raw Counts)	デューティサイクル単位の選択
Auto Start	FALSE	オートスタートの選択

Parameter	Value	説明
GTIOCA Output Enabled	True, False (Default: False)	GTIOCA 出力有効の選択
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	GTIOCA 停止レベルの選択
GTIOCB Output Enabled	True, False (Default: False)	GTIOCB 出力有効の選択
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	GTIOCB 停止レベルの選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *例の値とデフォルトは、Synergy S7G2 グループ MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

オーディオ再生クロック構成

オーディオ再生フレームワークハードウェアモジュールは、 [Clock configuration] ウィンドウにある周辺クロックを使用します。

ピン設定

DAC または SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。この後で、[SSP configuration] ウィンドウ内でのピンの選択方法と、関連するピンの選択例を示します。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

DAC のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select **Peripherals > Analog:ADC > ADC0/ADC1**

注: *この選択シーケンスでは、ADC0 または ADC1 か SSI0 または SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。*

DAC 上の DAC ドライバーのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Enabled, Disabled	動作の選択
DA	None, P014 (Default: P014)	DAC ピンの選択

注: *例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.2.5 アプリケーションでのオーディオ再生フレームワークモジュールの使用

アプリケーションでオーディオ再生フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API

1) を使用してオーディオストリームを初期化します。
2) コールバック関数を使用して、初期数がアプリケーションで使用されるバッファ数と等しいセマフォにポストします。アプリケーションスレッドでこのセマフォを取得してから start を呼び出します。
3) bufferAcquire を使用してメッセージフレームワークからバッファを取得します。
4) バッファ内にオーディオフレームワークデータ構造体 sf_audio_playback_data_t を作成します。
5) start API を使用してオーディオ再生フレームワークを開始します。
6) 複数のストリームが必要な場合は、追加のオーディオストリームのために手順 1 ～ 5 を繰り返します。個別のオーディオストリームを使用する必要があるのは、ミキシングを使用してストリームを同時に再生する必要がある場合のみです。オーディオのサウンドが常に順番に再生され、重ならない場合には、ストリームを再利用できます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

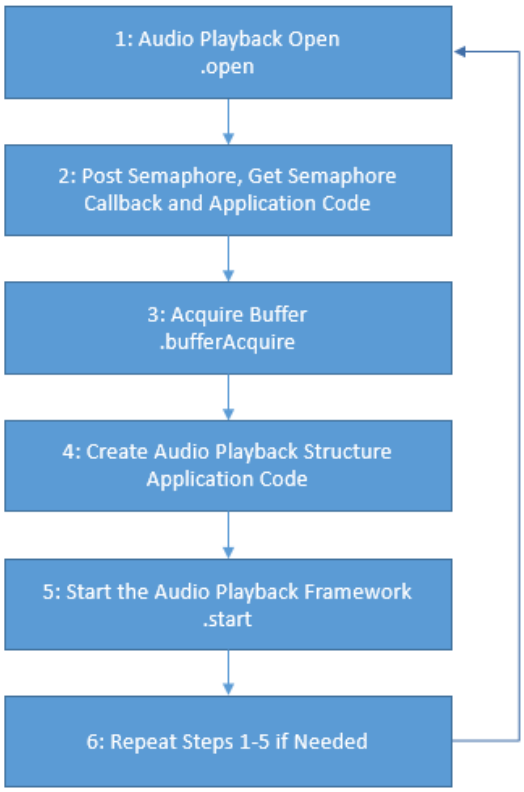


図 119: 一般的なオーディオ再生アプリケーションのフロー図

5.1.3 オーディオ再生 DAC フレームワーク

オーディオ再生ハードウェア DAC フレームワークモジュールは、以下の機能に対応します。

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。
- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。
- ソフトウェア音量制御。
- 一時停止と再開の機能。
- スケーリング。たとえば、符号付き 16 ビット PCM データを符号なし 12 ビット DAC の範囲にシフトします。
- 複数のストリームについての基本的なミキシング。

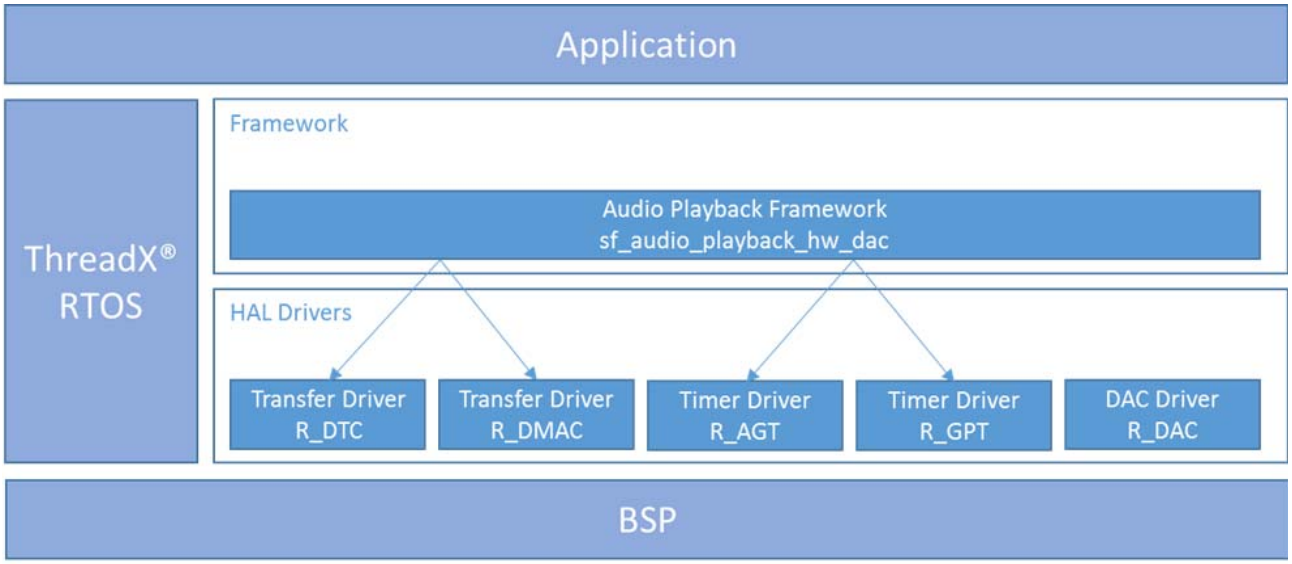


図 120: オーディオ再生ハードウェア DAC フレームワークモジュールのブロック図

5.1.3.1 オーディオ再生ハードウェア DAC フレームワークモジュールの API の概要

オーディオ再生 DAC モジュールは、オープン、開始、再生、停止などの操作の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ再生ハードウェア DAC フレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
open	g_sf_audio_playback0.p_api->open(g_sf_audio_playback0.p_ctrl, g_sf_audio_playback0.p_cfg); 読み書きおよび制御のためにデバイスチャネルを開きます。
start	g_sf_audio_playback0.p_api->start(g_sf_audio_playback0.p_ctrl, &p_data, 100); オーディオ再生ハードウェアを開始します。

Function Name	API の呼び出し例と説明
stop	g_sf_audio_playback0.p_api->stop(g_sf_audio_playback0.p_ctrl); オーディオ再生ハードウェアを停止します。
resume	g_sf_audio_playback0.p_api->resume(g_sf_audio_playback0.p_ctrl); オーディオバッファを再生します。
volumeSet	g_sf_audio_playback0.p_api->volumeSet(g_sf_audio_playback0.p_ctrl, p_data_type); 指定されたポインタ p_data_type に、指定されたデータ型を格納します。
close	g_sf_audio_playback0.p_api->close(g_sf_audio_playback0.p_ctrl); オーディオドライバーを閉じます。
versionGet	g_sf_audio_playback0.p_api->versionGet(&version); ドライバーのバージョンを返します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	ポインタが NULL またはパラメータが無効です。
SSP_ERR_OUT_OF_MEMORY	一度に開くストリームの数は、SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS に制限されています。この数値を超えると、メモリ不足のエラーが発生します。

Name	説明
SSP_ERR_TIMEOUT	再生が終了する前にタイムアウトが発生しました。
SSP_ERR_NOT_OPEN	ストリーム制御ブロック p_ctrl が初期化されていません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.3.2 オーディオ再生ハードウェア DAC フレームワークモジュールの動作の概要

オーディオ再生フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生フレームワークのスレッドとパブリックオーディオ再生フレームワーク API とのやり取りを表すフローチャートを示します。

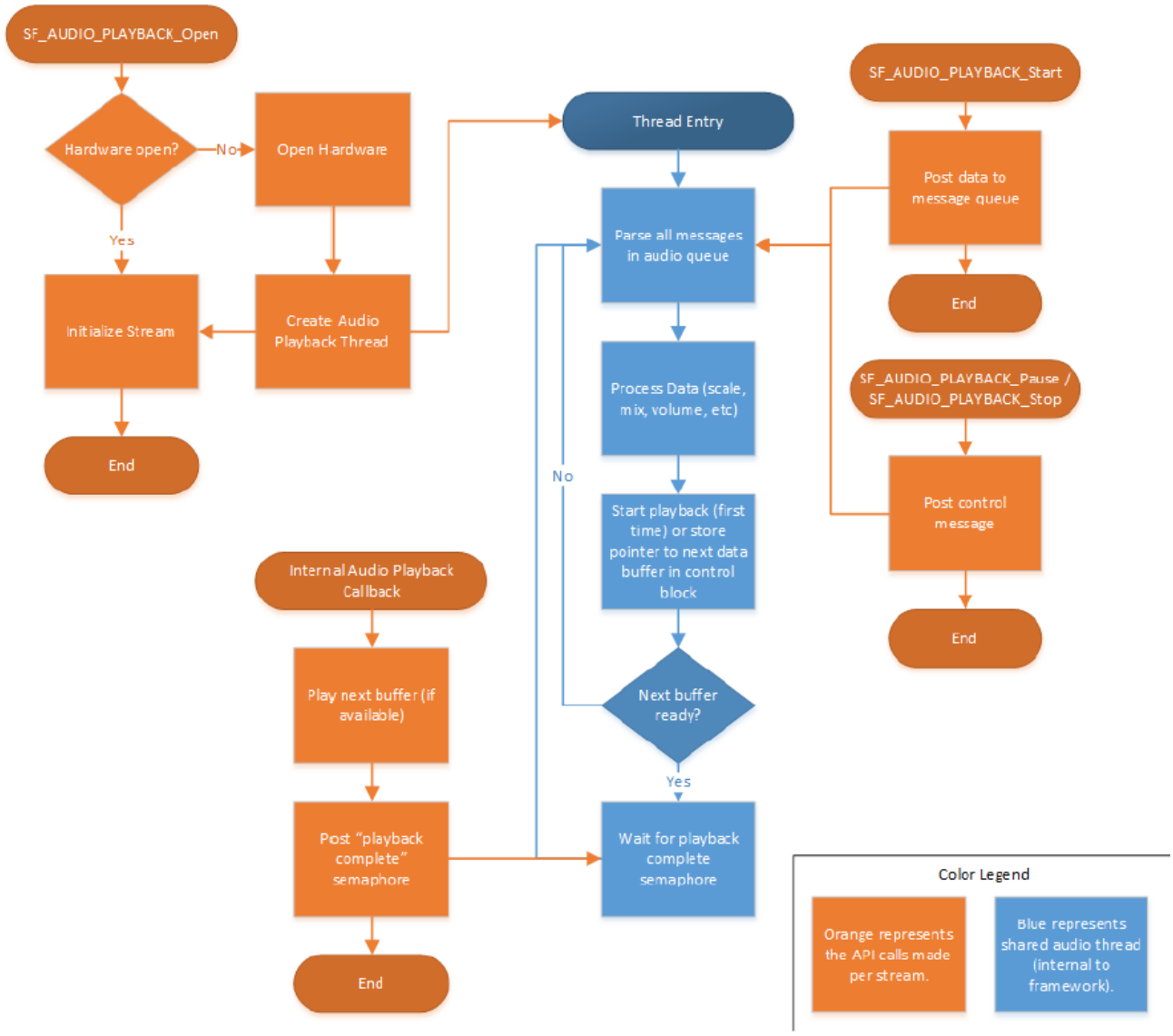


図 121: オーディオ再生フレームワークフローチャート

オーディオ再生フレームワークの推奨される使用方法を以下に示します。

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore). これは **[Threads]** タブで行うことができます。初期値を 2 に設定します（オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータメッセージを保持できます）。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback). オーディオ再生フレームワークインスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。

- メインループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージングフレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。

オーディオ再生フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

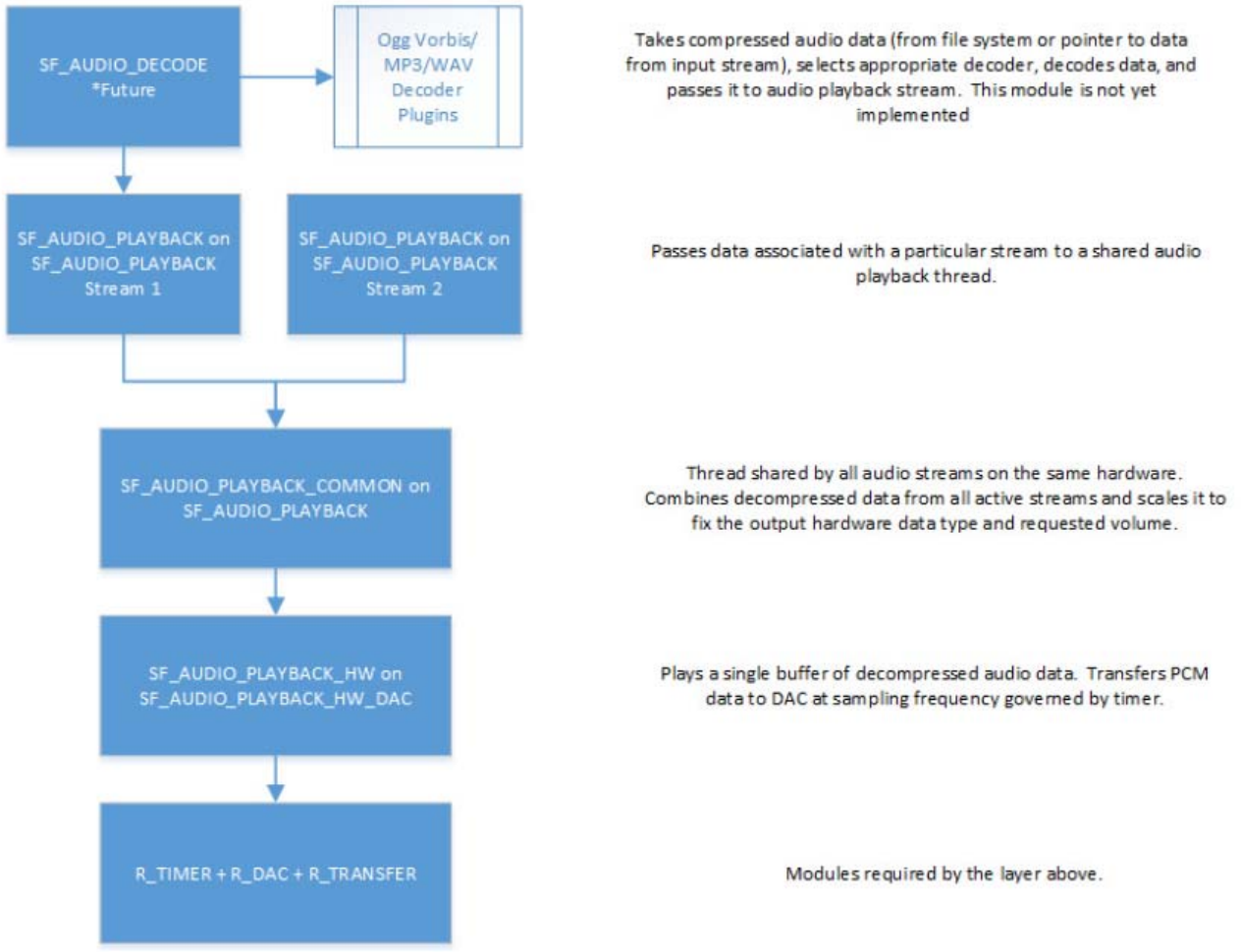


図 122: 複数オーディオストリームの実装

オーディオ再生ハードウェア DAC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- オーディオフレームワークの DAC ハードウェアポートは、転送 API を使用して、内蔵タイマで定義されたサンプリング周波数で再生バッファから DAC にオーディオデータを転送します。
- オーディオフレームワーク DAC ハードウェアポートには、タイマ、DAC、および転送 API モジュールに対する依存関係があります。
  - タイマモジュールを追加します。

  - **周波数（Hz 単位）をオーディオデータのサンプリング周波数に設定します。**
  - **DTC を転送モジュールとして使用する場合は （推奨）、割り込みを有効にします。**
  - r_dtc または r_dmac. 上で転送モジュールを追加します。
  - **DTC の場合**
  - **DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC->DADRn[0] に設定し、 DAC チャネル 1 を使用する場合は &R_DAC->DADRn[1] に設定します。**
  - **アクティベーションソースを上で選択したタイマ割り込みに設定します。**
  - **DMAC の場合**
  - **dmac サポートを有効にします。**
  - **DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC->DADRn[0] に設定し、 DAC チャネル 1 を使用する場合は &R_DAC->DADRn[1] に設定します。**
  - **アクティベーションソースを上で選択したタイマ割り込みに設定します。**
- オーディオ再生フレームワークは、API への変更なしに、以下の MCU ファミリをサポートするように設計されています。
  - S7G2
  - S3A3、 S3A7、 S3D9
  - S124
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.3.3 アプリケーションへのオーディオ再生ハードウェア DAC フレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションにオーディオ再生ハードウェア DAC フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

オーディオ再生ハードウェア DAC フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。（オーディオ再生ハードウェア DAC フレームワークモジュールのデフォルト名は g_sf_audio_playback_hw0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

オーディオ再生ハードウェア DAC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback_hw0 Audio Playback Hardware Framework on g_sf_audio_playback_hw0	**Threads**	**New Stack> Framework> Audio> Audio Playback Hardware Framework on g_sf_audio_playback_hw_dac**

次の図に示すように、sf_audio_playback_hw_dac のオーディオ再生ハードウェア DAC フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。

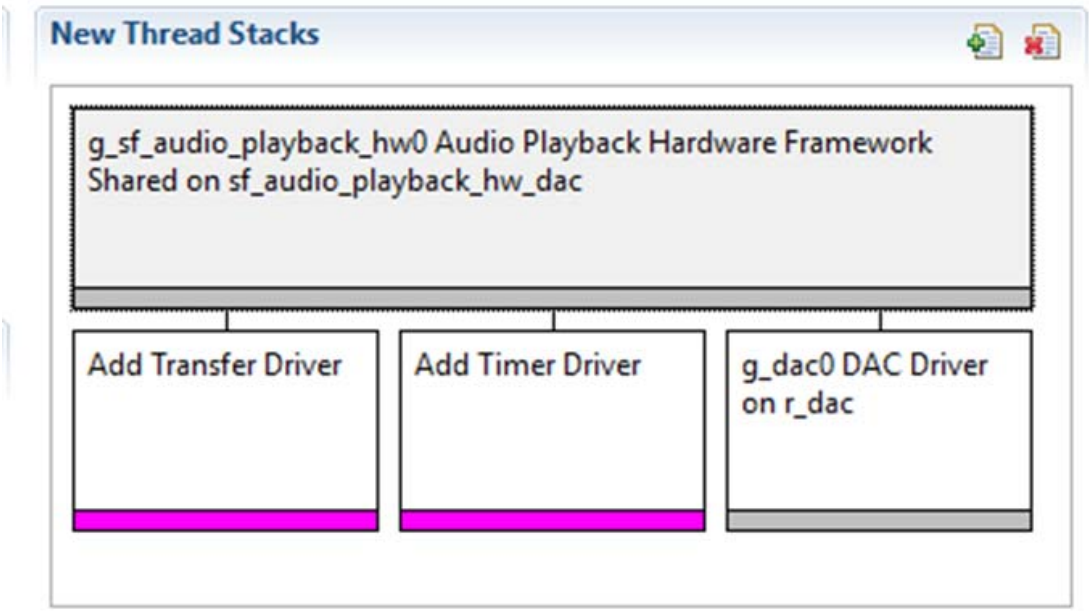


図 123: オーディオ再生ハードウェア DAC フレームワークモジュールのスタック

5.1.3.4 オーディオ再生ハードウェア DAC フレームワークモジュールの構成

オーディオ再生ハードウェア DAC フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の **[Properties]** ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_audio_playback_hw_dac 上のオーディオ再生ハードウェア DAC フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
DMAC Support	Enabled, Disabled Default: Disabled	DMAC サポートの有効化または無効化。
Name	g_sf_audio_playback_hw0	モジュール名。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

オーディオ再生 DAC フレームワークの低レベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_dmac Software Activation 上の DMAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer0	モジュール名
Channel	0	チャネルの選択
Mode	Normal	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source	Software Activation, Peripheral Events Default: Software Activation	アクティベーションソースの選択
Auto Enable	False	自動有効の選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: 設定例とデフォルトは、Synergy SK-S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	アクティベーションソースの選択
Auto Enable	False	自動有効の選択

ISDE Property	Value	説明
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	物理ハードウェア チャネル。
Mode	Periodic	***注:*** *ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。*
Period Value	10	「タイマ期間の計算」を参照。
Period Unit	Hertz	「タイマ期間の計算」を参照。
Auto Start	FALSE	設定後にタイマを起動するには、 true に設定します。 start を呼び出すまで測定を無効状態にするには false に設定します。

ISDE Property	Value	説明
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	AGT カウンタのクロックソース。
AGTO Output Enabled	True, False Default: False	AGT 用に設定されたポートピン（AGTO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
AGTIO Output Enabled	True, False Default: False	AGT 用に設定されたポートピン（AGTIO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
Output Inverted	True, False Default: False	出力信号 low を開始するには false を設定します。出力信号 high を開始するには true を設定します。
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 ***注：****コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	タイマ割り込み優先順位。0が最高のプライオリティです。

注: 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	チャネルの選択。
Mode	Periodic	*注: ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。*
Period Value	10	「タイマ期間の計算」を参照
Period Unit	Hertz	「タイマ期間の計算」を参照
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	デューティサイクル単位の選択
Auto Start	FALSE	設定後にタイマを起動するには、 true に設定します。 start を呼び出すまで測定を無効状態にするには false に設定します。
GTIOCA Output Enabled	True, False Default: False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、 true を設定します。タイマ信号を出力しない場合は false を設定します。

ISDE Property	Value	説明
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	タイマが停止したときの出力ピンレベルを制御します。
GTIOCB Output Enabled	True, False Default: False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	タイマが停止したときの出力ピンレベルを制御します。
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 ***注:*** *コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dac 上の DAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_dac0	モジュール名。
Channel	0	出力 DA0 の場合は 0、出力 DA1 の場合 1 を設定します。
Synchronize with ADC	Enabled, Disabled Default: Disabled	A/D コンバータ (ADC) モジュールを使用した干渉防止同期の場合は true を設定します。アナログモジュール間の電源干渉が問題にならないか、DAC モジュールによる非同期変換が望ましい場合は false を設定します。
Data Format	Right Justified	12 ビットデータ値をビット 11 から 0 にロードする（右詰め）場合は 0 を設定します。12 ビットデータ値をビット 15 から 4 にロードする（左詰め）場合は 1 を設定します。
Output Amplifier	Enable, Disable Default: Disable	出力アンプハードウェア機能が望ましい場合は true を設定します。出力アンプハードウェア機能をバイパスする場合は false を設定します。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

オーディオ再生ハードウェア DAC フレームワークモジュールのクロック構成

オーディオ再生フレームワークハードウェアモジュール（DAC）は、**[Clocks]** 構成ウィンドウにある周辺クロックを使用します。

オーディオ再生ハードウェア DAC フレームワークモジュールのピン構成

DAC または SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は関連するピンの選択例を示しています。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

DAC でのピンの選択

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select **Peripherals > Analog:ADC > ADC0/ADC1**

注: *この選択シーケンスでは、ADC0 または ADC1 か SSI0 または SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。*

r_dac 上の DAC HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Enabled, Disabled	動作の選択
DA	None, P014 (Default: P014)	DAC ピンの選択

注: *前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.3.5 アプリケーションでのオーディオ再生ハードウェア DAC フレームワークモジュールの使用

アプリケーションでオーディオ再生ハードウェアDACフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用してオーディオ再生 DAC フレームワークを初期化します。
2) start API を使用してオーディオ再生 DAC フレームワークの低レベルハードウェアを開始します。
3) play API を使用して PCM オーディオサンプルを再生します。
4) ハードウェアでサポートされる PCM オーディオサンプルは dataTypeGet API によって提供されます。
5) stop API を使用してオーディオ再生 DAC フレームワークの低レベルハードウェアを停止します。
6) close API を使用してオーディオ再生 DAC フレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

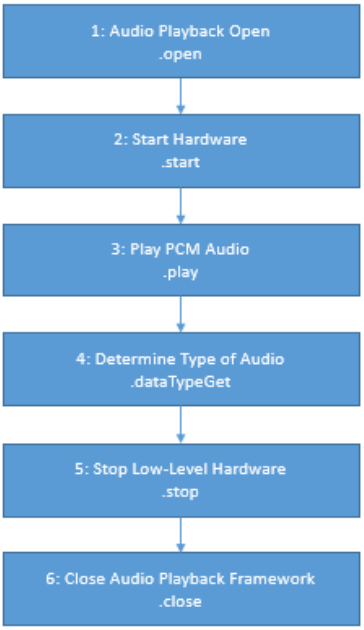


図 124: 通常のオーディオ再生ハードウェア DAC フレームワークモジュールアプリケーションのフロー図

5.1.4 オーディオ記録 ADC フレームワーク

- 8 ビットまたは 12 ビット PCM でデータを記録します
- ADC 周期フレームワークを使用して構成と統合を単純化します
- ThreadX オブジェクト（ミューテックスなど）を使用して不適切なアクセスからハードウェアを保護します
- 高レベル関数の API によってコーディングが簡略化されます
  - open、 start
  - stop、 infoGet
  - close

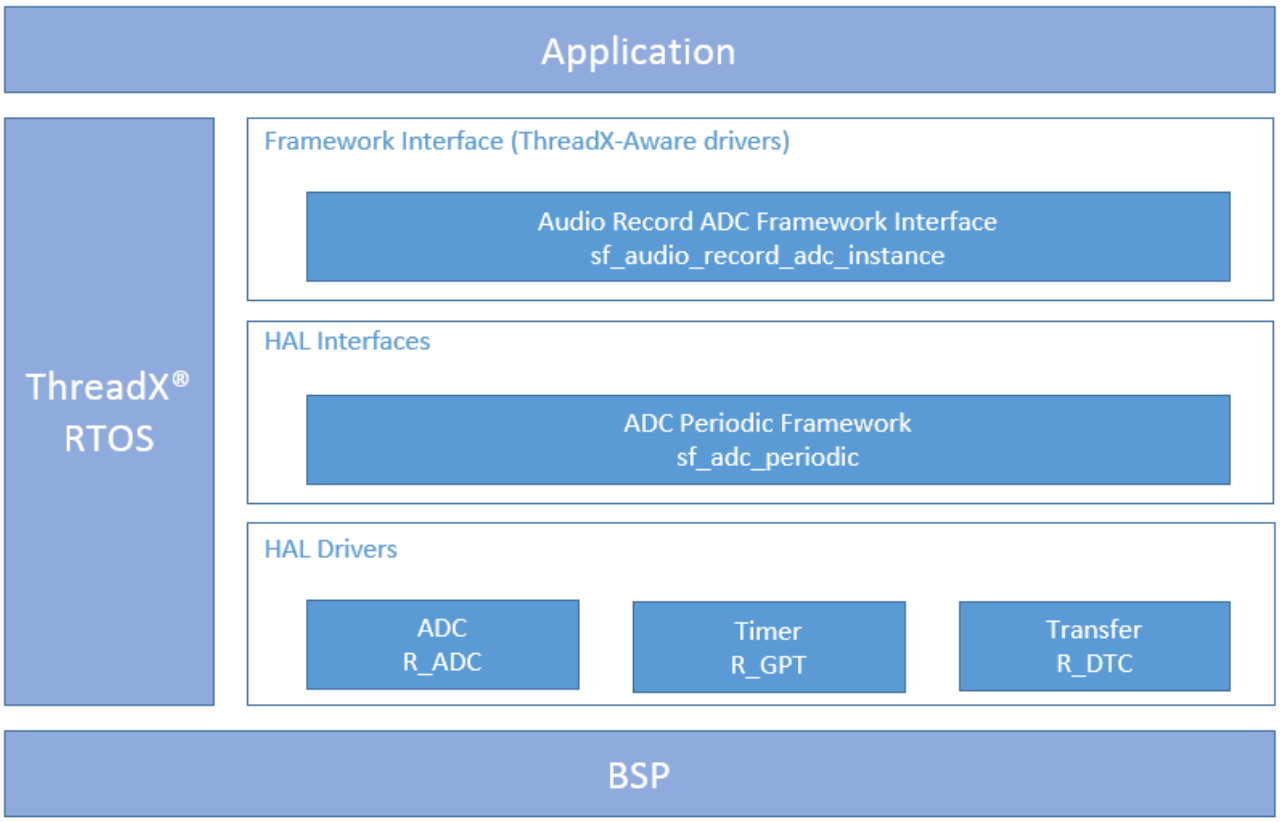


図 125: オーディオ記録 ADC フレームワークモジュールの編成、オプション、スタックの実装

5.1.4.1 オーディオ記録 ADC フレームワークモジュールの API の概要

オーディオ記録 ADC フレームワークモジュールは、記録プロセスのオープン、クローズ、開始、停止などの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

オーディオ記録 ADC フレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
.open	g_sf_audio_record_adc.p_api->open(g_sf_audio_record_adc.p_ctrl, g_sf_audio_record_adc.p_cfg); モジュールを初期化します。

Function Name	API の呼び出し例と説明
.start	g_sf_audio_record_adc.p_api->start(g_sf_audio_record_adc.p_ctrl); オーディオ記録を開始します。
.stop	g_sf_audio_record_adc.p_api->stop(g_sf_audio_record_adc.p_ctrl); オーディオ記録を停止します。
.infoGet	g_sf_audio_record_adc.p_api->infoGet(g_sf_audio_record_adc.p_api.*p_ctrl*); チャネル情報を取得します（モノまたはステレオ）。
.close	g_sf_audio_record_adc.p_api->close(g_sf_audio_record_adc.p_ctrl); モジュールを閉じます。
.versionGet	g_sf_audio_record_adc.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、SSP ユーザーズマニュアルで関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です。
SSP_ERR_IN_USE	ADC 周期フレームワークミューテックスが、要求されたユニットで使用できない可能性があります。考えられる他の原因については、HAL ドライバーを参照してください。

Name	説明
SSP_ERR_INTERNAL	内部 ThreadX エラーが発生しました。これは通常、ミューテックスの作成/使用または内部スレッドの作成の失敗を意味します。
SSP_ERR_NOT_OPEN	ユニットが開いていません。
SSP_ERR_ASSERTION	パラメータ p_ctrl または p_sample が NULL です。
SSP_ERR_UNSUPPORTED	この機能は HAL ドライバーでサポートされません （p_ctrl->p_api->close は NULL です）。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.4.2 オーディオ記録 ADC フレームワークモジュールの動作の概要

オーディオ記録 ADC フレームワークモジュールは、ADC 周期フレームワークを使用してオーディオアナログデータをサンプリングします。取得されたデータサンプルはユーザーバッファに保存されます。このデータは、アプリケーションの必要に応じてさらに処理することができます。オーディオレコード ADC フレームワークには、フレームワークの初期化中に初期化される構成パラメータが存在します。フレームワークの初期化では他にも、データキャプチャ用に基礎となる ADC 周期フレームワークが初期化されます。

取得されたデータはユーザー定義バッファに格納されますが、これは次に示すようにコールバック関数内で実行されます。

コールバックの名前が sf_audio_record_user_callback. と構成されているとします。

```
uint16_t * audio_record_buffer;
void sf_audio_record_user_callback (sf_audio_record_callback_args_t *p_args)
{
audio_record_buffer = ((uint16_t *)g_sf_audio_record_adc.p_cfg->
    p_capture_data_buffer + (p_args->buffer_index/2)); }
```

オーディオ記録 ADC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- オーディオ記録 ADC フレームワークモジュールの構成データで、データバッファの長さ、データ幅、サンプリングレート、サンプリング繰り返し回数を指定できます。
- 現在のオーディオ記録 ADC は、下位のフレームワークとして ADC 周期フレームワークのみをサポートしているため、I2S を介した記録には対応していません。
- 現状では、8 ビットまたは 12 ビットの PCM データの記録をサポートしています。
- 現在、オーディオ記録 ADC ではモノチャネルしかサポートされません。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.4.3 アプリケーションへのオーディオ記録 ADC フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ記録 ADC フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

オーディオ記録 ADC フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（オーディオ記録 ADC フレームワークモジュールのデフォルト名は g_audio_record_adc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

オーディオ記録 ADC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_audio_record_adc0 Audio Record ADC Framework on sf_audio_record_adc	Threads	New Stack> Driver> Audio> Audio Record ADC Framework on sf_audio_record_adc

次の図に示すように、sf_audio_record_adc のオーディオ記録 ADC がスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。

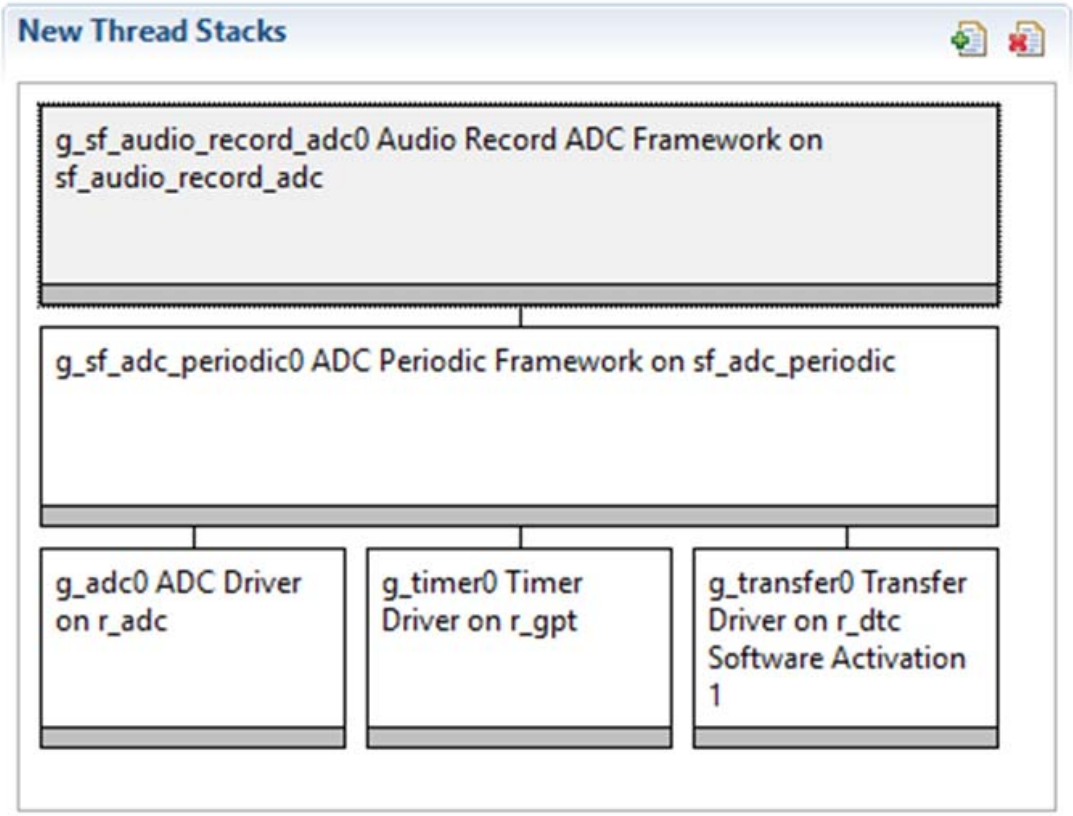


図 126:　オーディオ記録 ADC フレームワークモジュールのスタック

5.1.4.4 オーディオ記録 ADC フレームワークモジュールの構成

オーディオ記録 ADC フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

sf_audio_record_adc 上のオーディオ記録 ADC フレームワークモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_sf_audio_record_adc0	モジュール名
Name of the data-buffer to store samples	P_capture_data_buffer	サンプルを格納するデータバッファの名前
Length of the data-buffer	2048	データが格納されるバッファの長さ
Audio Record Data Size	8-bit, 16-bit Default: 8-bit	データ幅
Sampling Rate in HZ	8000	サンプリングレート
Number of sampling iterations	Default: 256	繰り返しごとに取得されるサンプル数
Callback	g_audio_redord_framework_user_callback	サンプリング繰り返し回数のデータがバッファに格納された後に呼び出されるユーザー関数。

注: 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なるバッファサイズやサンプリングレートの選択が役立つ場合です。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

オーディオ記録 ADC フレームワークモジュールのローレベルドライバーの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

sf_adc_periodic 上の ADC 周期フレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_sf_adc_periodic0	モジュール名
Name of the data-buffer to store samples	g_user_buffer	サンプルを格納するデータバッファの名前
Length of the data-buffer	2048	データが格納されるバッファの長さ
Number of sampling iterations	256	繰り返しごとに取得されるサンプル数
GPT Timer channel used to trigger scan	Channel 0-12	チャネル番号
Callback	g_audio_redord_framework_user_callback	サンプリング繰り返し回数のデータがバッファに格納された後に呼び出されるユーザー関数。

r_adc 上の ADC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	選択するとパラメータチェック用のコードがビルドに含まれます。
Name	g_adc0	モジュール名
Unit	0, 1 (S7G2 Only) Default: 0	使用する ADC ユニットを指定します。S7G2 には、0 と 1 の 2 つのユニットがあります。
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2 Only), 8-Bit (S7G2 Only) Default: 8-Bit (S7G2 Only)	このユニットの変換解像度を指定します。

ISDE Property	Value	説明
Alignment	Right, Left Default: Right	変換結果のアラインメントを指定します。
Clear after read	Off, On Default: On	変換結果を読み取った後で、結果レジスタを自動的にクリアするかどうかを指定します。 ***注:*** *これが有効になっている場合、デバッガーを使用して結果レジスタをウォッチすると、常に 0 になります。*
Mode	Single Scan	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。

ISDE Property	Value	説明
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B (Default: Unused)	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	温度センサーはチャネルスキャンマスクの選択肢を使用します。
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	電圧センサーはチャネルスキャンマスクの選択肢を使用します。

ISDE Property	Value	説明
Scan Mask Group B	Use #define ADC_MASK_xxx which are defined in r_adc.h.Use (ADC_MASK_xxx / ADC_MASK_xxx) for multiple channels.	このモードはシングルスキャンモードに固定されているため、ADC フレームワークでは使用しないでください。
Normal/Group A Trigger	ELC Event	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously (Default: Group A cannot interrupt Group B)	このモードはシングルスキャンモードに固定されているため、ADC フレームワークでは使用しないでください。
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	このユニット内のいずれかのチャネルに対して、加算または平均化を行う必要があるかどうかを指定します。実際のチャネルはチャネルマスク add_mask を使用して指定します。
Channels 0-27	Disabled, Enabled Default: Disabled	このフィールドは、add_average_count が有効になっている場合にのみ有効です。平均化または合計するチャネル結果を決定するために使用します。
Temperature Sensor	Disabled, Enabled Default: Disabled	温度センサーは追加マスクまたは平均マスクの選択肢を使用します。

ISDE Property	Value	説明
Voltage Sensor	Disabled, Enabled Default: Disabled	電圧センサーは追加マスクまたは平均マスクの選択肢を使用します。
Channels 0-2	Disabled, Enabled Default: Disabled	チャネル 0、1、2 のどれが、sample_hold_states で指定されたサンプルアンドホールド状態の更新値を使用するかを決定します。このフィールドは、チャネル 0、1、および 2 のデフォルトのサンプルアンドホールドカウント値を変更する場合にのみ設定する必要があります。
Sample Hold States (Applies only to the 3 channels selected above)	24	チャネル専用のサンプルアンドホールド回路用の、更新されたサンプルアンドホールドカウントを指定します。このフィールドは、sample_hold_mask が 0 でない場合にのみ有効です。チャネル 0、1、2 のみに専用のサンプルアンドホールド回路があります。 ***注：****値をサンプリングするデフォルトの状態数 (24) を変更するにはこのフィールドを使用します。各状態は、1/ADCLK 時間に等しくなります。*
Callback	NULL	ADC フレームワークはこのコールバックを内部で使用します。
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	スキャン終了割り込み優先順位の選択

ISDE Property	Value	説明
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	スキャン終了グループ B 割り込み優先順位の選択

r_gpt 上の GPT HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	このフィールドは、ADC フレームワークで選択されたチャネルに基づいて事前設定されており、ロックされています。
Mode	Periodic	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Period Value	10	ADC スキャンをトリガするタイマ周期を構成します。
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	上で指定したタイマ周期の単位を構成します。
Duty Cycle Value	50	デューティサイクル値の選択

ISDE Property	Value	説明
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	デューティサイクル単位の選択
Auto Start	False	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
GTIOCA Output Enabled	True, False Default: False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	タイマが停止したときの出力ピン レベルを制御します。
GTIOCB Output Enabled	True, False Default: False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	タイマが停止したときの出力ピンレベルを制御します。
Callback	NULL	このフィールドは ADC フレームワークによって事前設定されており、ロックされています。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

オーディオ記録 ADC フレームワークモジュールのクロック構成

ADC 周辺モジュールは PCLKC をそのクロックソースとして使用します。

オーディオ記録 ADC フレームワークモジュールのピン構成

ADC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。 ADC ピンはアナログピンとして構成する必要があります。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

ADC のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog:ADC > ADC0

注: *上記の選択シーケンスでは、ADC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

ADC のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Custom (Default: Custom)	ADC の動作モードを選択します。
ADTRG	None, P407, P102 (Default: None)	ADTRG ピン
AN00-19	None, Pnnn, Pmmm (Default: None)	アナログ入力ピン
PGAVSS0	None, P003 (Default: None)	PGAVSS ピン

注: *前の設定例は、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.4.5 アプリケーションでのオーディオ記録 ADC フレームワークモジュールの使用

アプリケーションでオーディオ記録ADCフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用してモジュールをオープンします。
2) start API を使用して記録を開始します。
3) コールバックを使用してユーザーバッファにデータを保存します。
4) 必要に応じてデータを処理します。
5) close API を使用してモジュールを閉じます。

上記の一般的な手順を、次の図の通常の動作フロー図に示します。

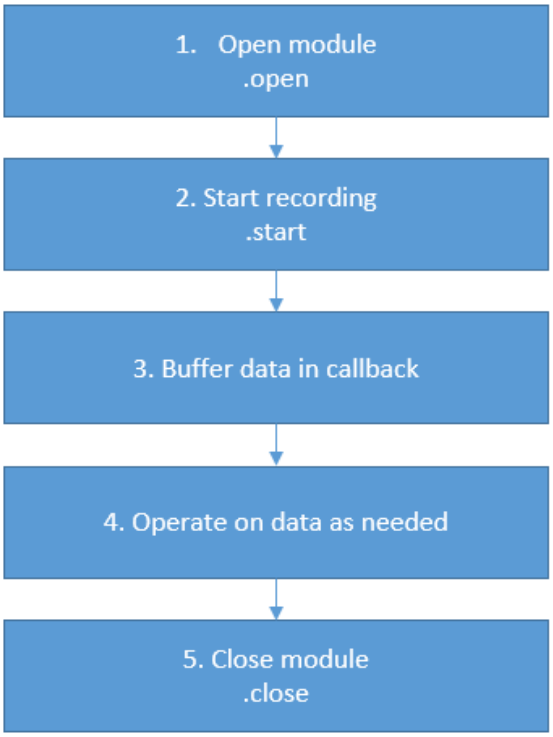


図 127: 通常のオーディオ記録 ADC フレームワークモジュールアプリケーションのフロー図

5.1.5 オーディオ再生 I2S フレームワーク

オーディオ再生 I2S ハードウェアフレームワークモジュールは、以下の機能に対応します。

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。

- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。
- ソフトウェア音量制御。
- 一時停止と再開機能。
- 複数のストリームについての基本的なミキシング。

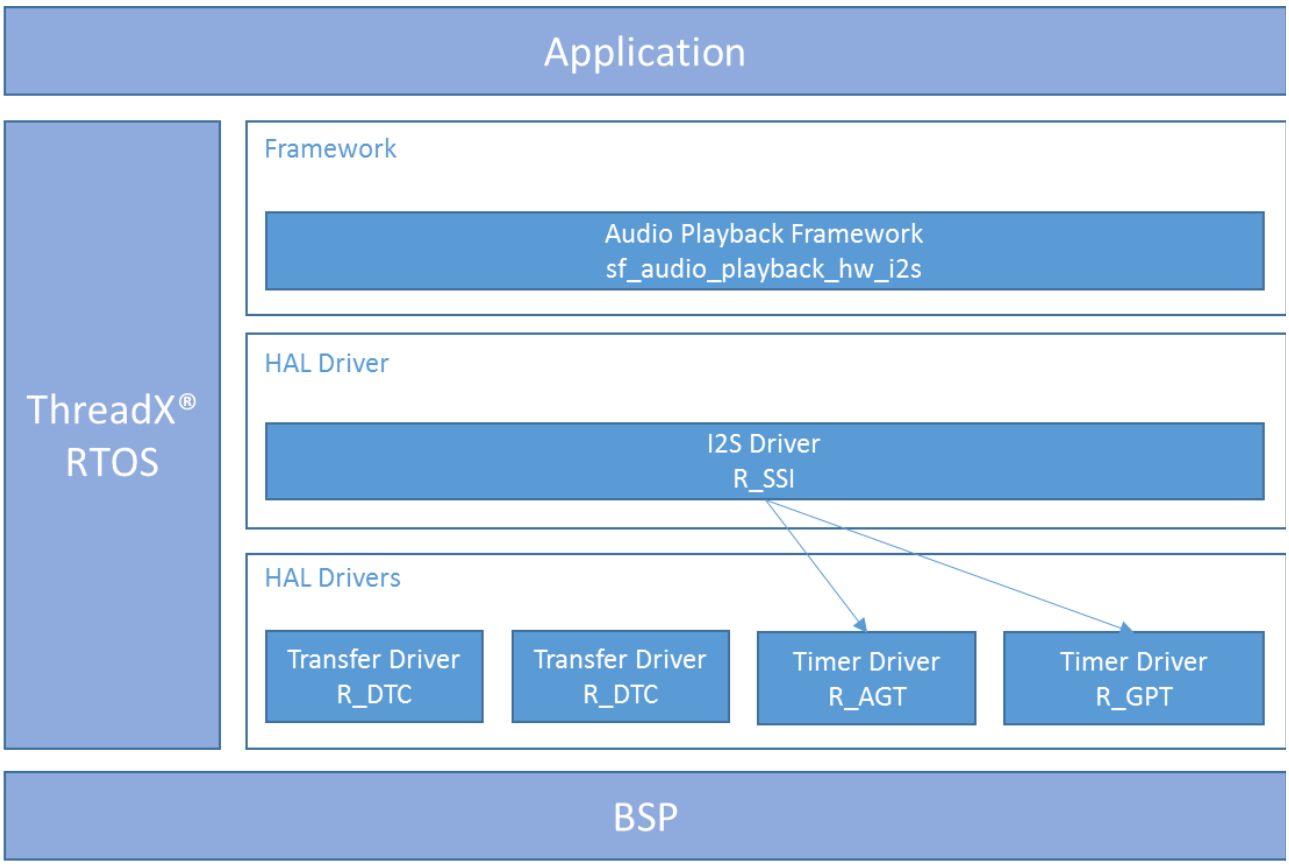


図 128: オーディオ再生フレームワークのスタックオプション

5.1.5.1 オーディオ再生 I2S フレームワークの API の概要

オーディオ再生 I2S フレームワークモジュールは、オープン、開始、再生、停止などの操作の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

オーディオ再生 I2S フレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
open	g_sf_audio_playback_hw_i2s0.p_api->open(g_sf_audio_playback_hw_i2s0.p_ctrl, g_sf_audio_playback_hw_i2s0.p_cfg); 読み書きおよび制御のためにデバイスチャネルを開きます。
start	g_sf_audio_playback_hw_i2s0.p_api->start(g_sf_audio_playback_hw_i2s0.p_ctrl); オーディオ再生ハードウェアを開始します。
stop	g_sf_audio_playback_hw_i2s0.p_api->stop(g_sf_audio_playback_hw_i2s0.p_ctrl); オーディオ再生ハードウェアを停止します。
play	g_sf_audio_playback_hw_i2s0.p_api->stop(g_sf_audio_playback_hw_i2s0.p_ctrl, p_buffer, length); オーディオバッファを再生します。
dataTypeGet	g_sf_audio_playback_hw_i2s0.p_api->dataTypeGet(g_sf_audio_playback_hw_i2s0.p_ctrl, p_data_type); 指定されたポインタに、予期するデータ型を格納します。
close	g_sf_audio_playback_hw_i2s0.p_api->close(g_sf_audio_playback_hw_i2s0.p_ctrl); オーディオドライバーを閉じます。
versionGet	g_sf_audio_playback_hw_i2s0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを返します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、SSP ユーザーズマニュアルで関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_OUT_OF_MEMORY	一度に開くストリームの数は、SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS に制限されています。この数値を超えると、メモリ不足のエラーが発生します。
SSP_ERR_TIMEOUT	再生が終了する前にタイムアウトが発生しました。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.5.2 オーディオ再生 I2S フレームワークモジュールの動作の概要

オーディオ再生 I2S フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生フレームワークのスレッドとパブリックオーディオ再生フレームワーク API とのやり取りを表すフローチャートを示します。

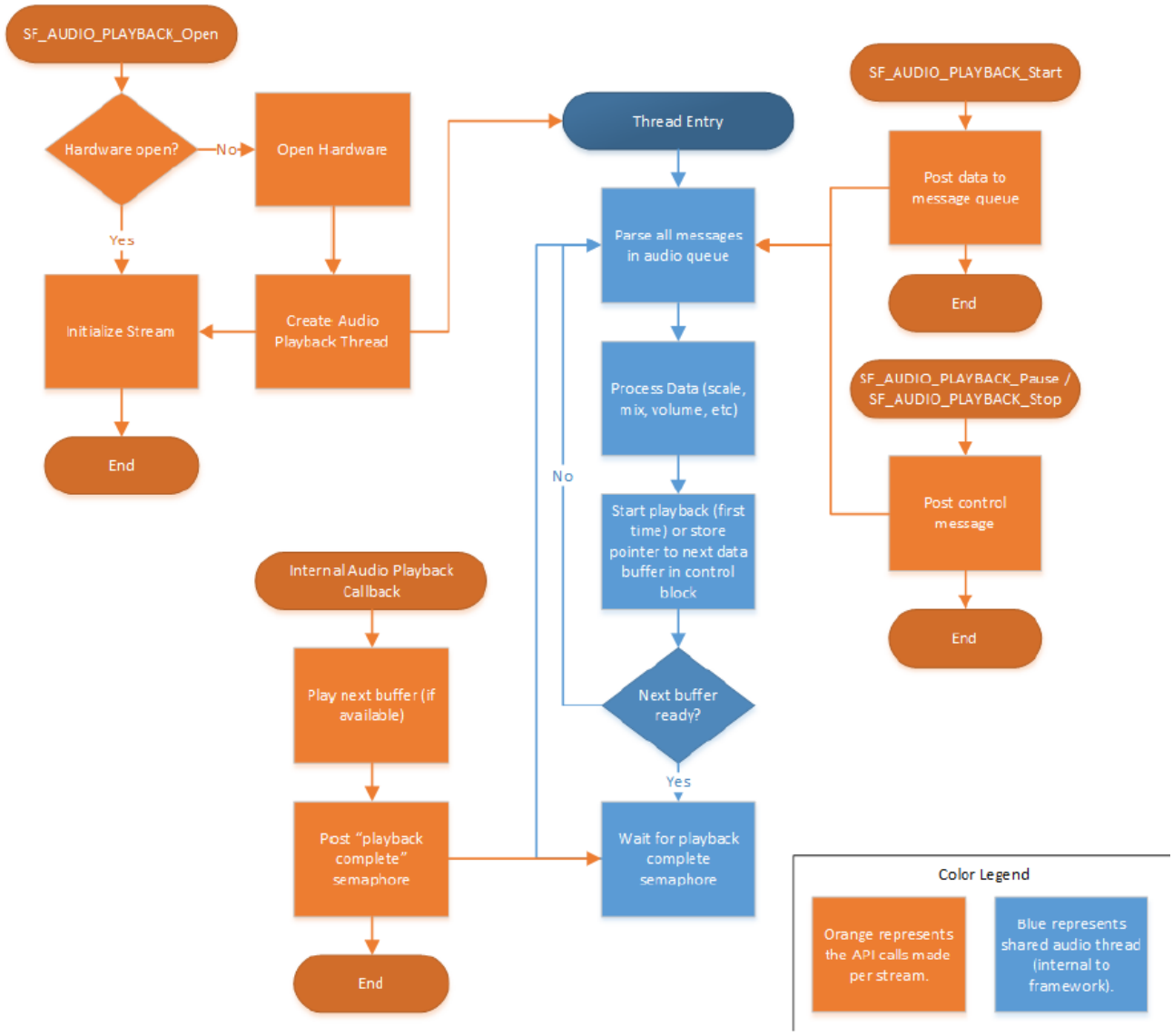


図 129: オーディオ再生フレームワークフローチャート

オーディオ再生フレームワークの推奨される使用方法を以下に示します。

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore). これは **[Threads]** タブで行うことができます。初期値を 2 に設定します（オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータメッセージを保持できます）。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback). オーディオ再生フレームワークインスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。

- メインループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージフレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。オーディオ再生フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

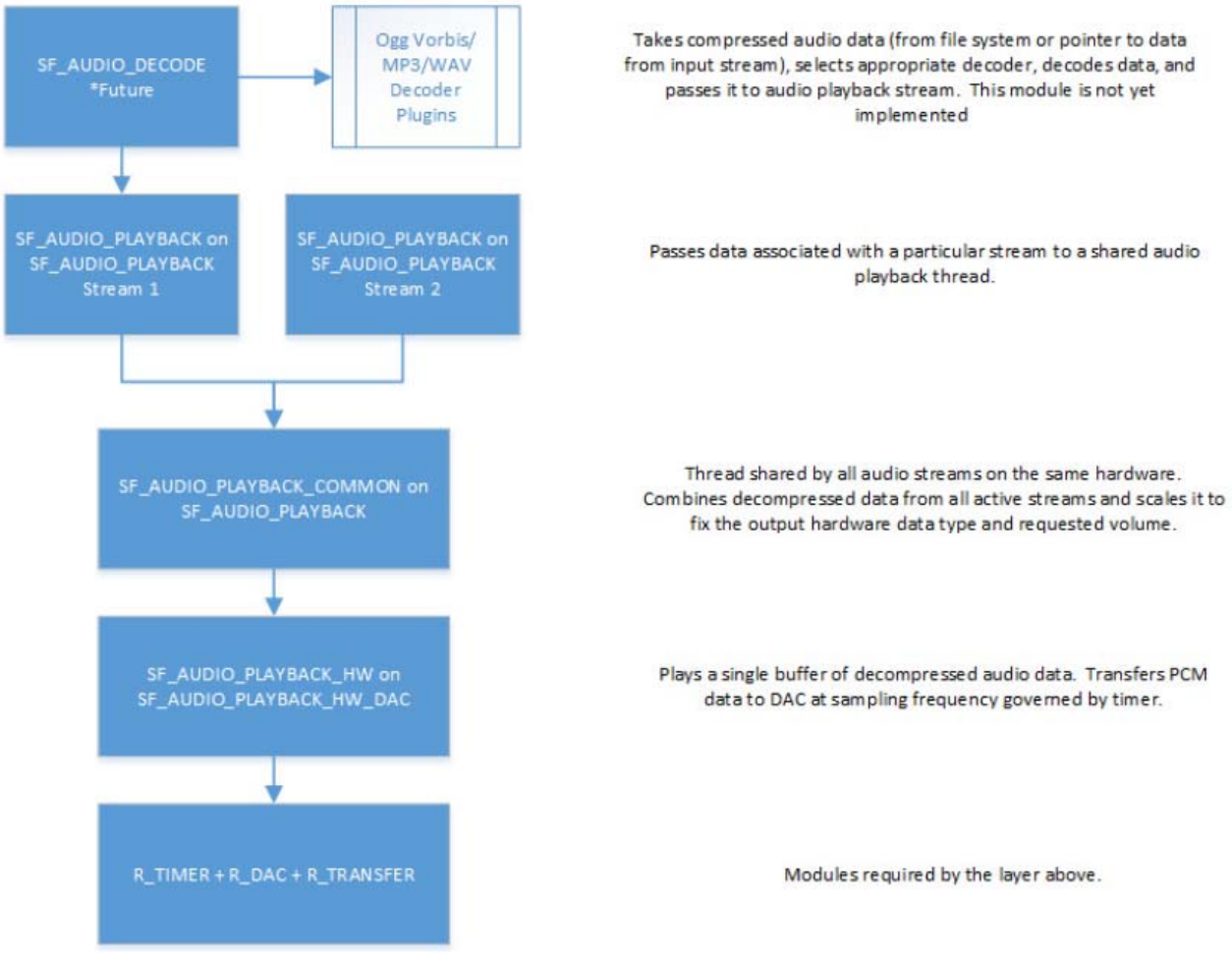


図 130: 複数オーディオストリームの実装

オーディオ再生 I2S フレームワークモジュールの動作に関する注意事項

- 使用されるキューは、[Audio Playback Framework Shared on sf_audio_playback] の **[Properties]** で指定された名前と一致する必要があります（デフォルトは g_sf_audio_playback_queue）。
- オーディオフレームワーク I2S ハードウェアポートには、I2S モジュールに対する依存関係があります。I2S ドライバーモジュールは DTC を使用して高速化できます（推奨）。
- I2S ドライバーモジュール。

  - オーディオクロック周波数（ヘルツ単位）を、使用されている入力オーディオクロックの周波数に設定します。
  - サンプリング周波数（ヘルツ単位）をオーディオデータのサンプリング周波数に設定します。
  - データビットとワード長を 16 ビットに設定します（オーディオフレームワークでは 16 ビットのみが受け入れられます）。
  - SSIn TXI および SSIn INT 割り込みを有効にします。
- r_dtc 上の転送モジュール（推奨）。
- アクティベーションソースを SSIn TXI 割り込みに設定します。
- オーディオ再生 I2S フレームワークは、API への変更なしに、以下の MCU ファミリをサポートするように設計されています。
  - S7G2
  - S3A7、S3D9、S3A3

オーディオ再生 I2S フレームワークモジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.5.3 アプリケーションへのオーディオ再生 I2S フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ再生 I2S フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

オーディオ再生 I2S フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（オーディオ再生のデフォルト名は g_sf_audio_playback_hw0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

オーディオ再生 I2S フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback_hw0 Audio Playback Hardware Framework on g_sf_audio_playback_hw0	**Threads**	**New Stack> Framework> Audio> Audio Playback Hardware Framework on g_sf_audio_playback**

次の図に示すように、sf_audio_playback_hw_i2s のオーディオ再生 I2S フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、テキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバ

一の選択が必要です。これらはオプションまたは推奨であり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

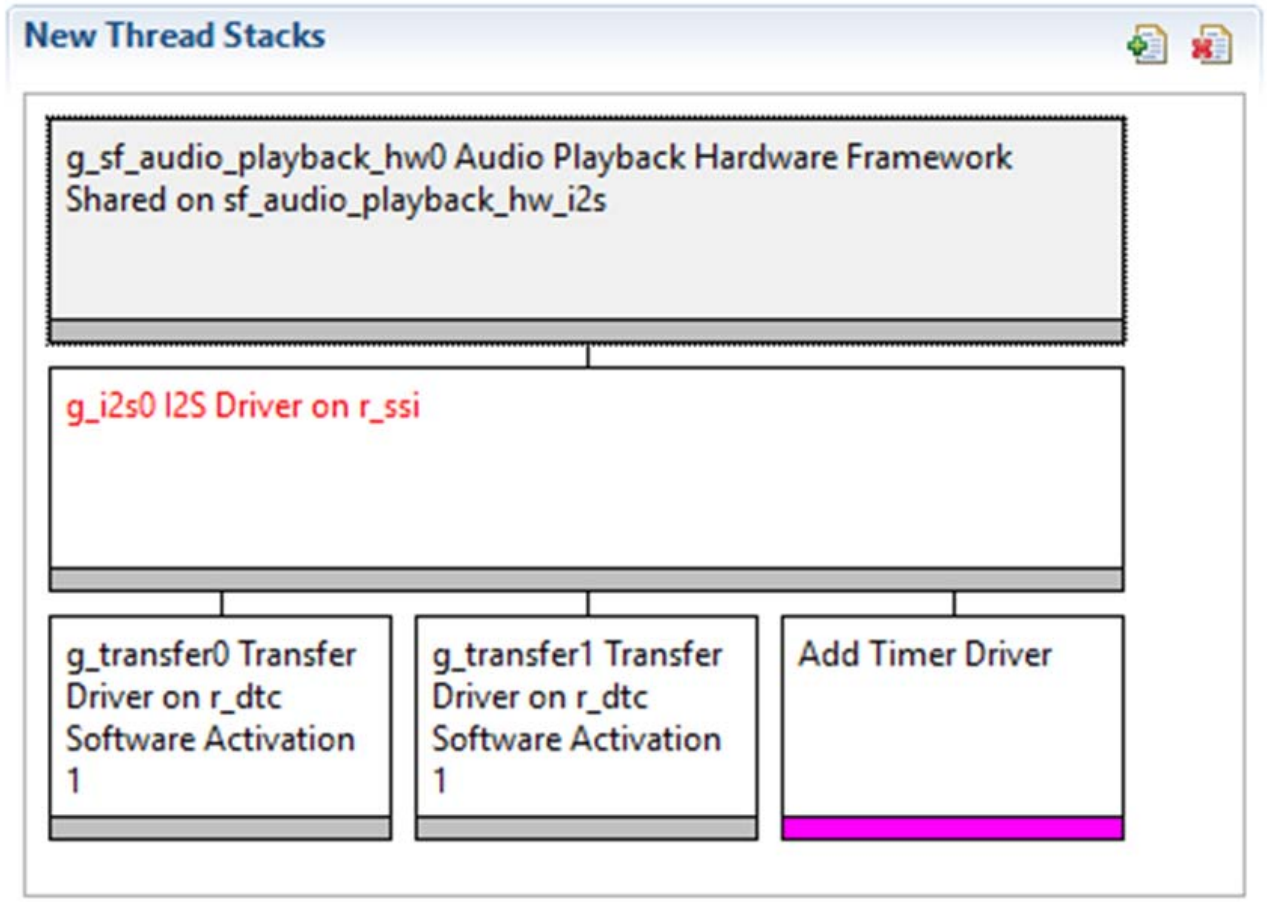


図 131: オーディオ再生 I2S フレームワークのスタック

5.1.5.4 オーディオ再生 I2S フレームワークモジュールの構成

オーディオ再生 I2S フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_audio_playback_hw_i2s 上のオーディオ再生 I2S ハードウェアフレームワークモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_sf_audio_playback_hw0	モジュール名。

注: *上記の設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルまたは I2S チャネルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する* **[PROPERTIES]** *ウィンドウを調べることで決定されます。*

5.1.5.5 オーディオ再生フレームワークの低レベルモジュールの構成設定

通常は、下位レベルモジュールの少数の設定のみをデフォルトから変更する必要があります。（これらは [thread stack] ブロックに赤いテキストで示されます。）一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_ssi 上の I2S HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_i2s0	モジュール名。
Channel	0	物理ハードウェアチャネル。
Audio Clock Frequency (Hertz)	2,822,400	I2S クロックの作成に用いられる入力オーディオクロック周波数。1 ～ 128(sampling_freq_hz*word_length_in_bits) の範囲で設定する必要があります。
Sampling Frequecy (Hertz)	44,100	オーディオデータのサンプリング周波数。
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	オーディオデータのビット深度（オーディオデータのサンプル 1 点のビットサイズ）。
Word Length	8 bits, 16, 24, 32 Default: 16 bits	オーディオデータのワード長、少なくともビット深度と同じサイズである必要があります（データビットフィールド）。
WS Continue Mode	Enabled, Disabled Default: Disabled	WS 続行モードを有効化すると、ペリフェラルがアイドル状態になった場合でも、ワード選択ラインの出力が継続して行われます。ペリフェラルがアイドル状態になった場合にワード選択ラインの出力を停止するには、これを無効化します。

ISDE Property	Value	説明
Name of I2S callback function to be defined by user	NULL	ユーザーコールバック関数はopenで登録する必要があります。全送信データの送信後に送信FIFO最高値に達した場合、あるいは受信が完了した（要求されたバイト数が受信された）場合に、割り込みサービスルーチン(ISR)からコールバックが呼び出されます。 ***注：*** *コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信割り込み優先順位の選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	受信割り込み優先順位の選択
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	アイドル/エラー割り込み優先順位の選択

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択。
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Fixed	宛先アドレスモードの選択。
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。

ISDE Property	Value	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択。
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Incremented	宛先アドレスモードの選択。
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。

ISDE Property	Value	説明
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

r_agt 上の AGT HAL モジュールの構成設定

ISE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択
Mode	Periodic	モードの選択
Period Value	2,822,400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Auto Start	FALSE	オートスタートの選択

ISE Property	Value	説明
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	クロックソースの選択
AGTO Output Enabled	True, False Default: False	AGTO 出力の選択
AGTIO Output Enabled	True, False Default: False	AGTIO 出力の選択
Output Inverted	True, False Default: False	出力の反転の選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

r_gpt 上の GPT HAL モジュールの構成設定

ISE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択

ISE Property	Value	説明
Mode	Periodic	モードの選択
Period Value	2,822,400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	デューティサイクル単位の選択
Auto Start	FALSE	オートスタートの選択
GTIOCA Output Enabled	True, False Default: False	GTIOCA 出力有効の選択
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA 停止レベルの選択
GTIOCB Output Enabled	True, False Default: False	GTIOCB 出力有効の選択
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB 停止レベルの選択
Callback	NULL	コールバックの選択

ISE Property	Value	説明
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *上記の設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

5.1.5.6 オーディオ再生 I2S フレームワークモジュールのクロック構成

オーディオ再生フレームワークハードウェアモジュール（I2S）は、[Clock configuration] ウィンドウにある周辺クロックを使用します。

5.1.5.7 オーディオ再生 I2S フレームワークモジュールのピン構成

SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は対応するピンの選択例を示しています。

ADC または I2S のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
I2S	Pins	Select **Peripherals > Connectivity:SSI > SSI0/SSI1**

注: *上記の選択シーケンスでは、ADC0 または ADC1 か SSI0 または SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。*

SSI での I2S ドライバーのピン構成設定

Pin Configuration Property	Value	説明
Pin Group Selection	_A only, _B only, Mixed	I2S ポートのピングループ
Operation Mode	Enabled, Custom, Disabled	動作の選択

Pin Configuration Property	Value	説明
SSISCK	None, P204 (Default: None)	SSI シリアルクロック
SSIWS	None, P205 (Default: None)	SSI ステレオピンの選択
SSIDATA	None, P206 (Default: None)	SSI データピンの選択

***注：** 前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.5.8 オーディオ再生 I2S フレームワークモジュールのメッセージ構成

[Messaging] タブのメッセージフレームワークコンフィギュレータを使用して、メッセージフレームワークを設定します。

1) オーディオ再生イベントクラスをハイライトします。
2) 新しいサブスクライバを追加します。
3) 以下の構成を選択して、sf_audio_playback モジュール上のオーディオ再生フレームワークの **[Properties]** タブでメッセージクラスインスタンスが開始インスタンスと終了インスタンスの間に適切に設定されていることを確認します。
   a) スレッド : アプリケーション内の任意のスレッド。
   b) 開始 : アプリケーションで使用されている最初のオーディオインスタンス。
   c) 終了 : アプリケーションで使用されている最後のオーディオインスタンス。
4) オーディオ再生サブスクライバーで新しいサブスクライバーをハイライトします。シンボル名を記録します。
5) **[Threads]** タブに戻ります。
6) HAL/ 共通のオーディオ再生フレームワーク共有モジュールをハイライトして、オーディオメッセージキュー名をオーディオ再生サブスクライバーのシンボル名に設定します。

5.1.5.9 アプリケーションでのオーディオ再生 I2S フレームワークモジュールの使用

アプリケーションでオーディオ再生 I2S フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用してオーディオ再生 I2S フレームワークを初期化します。
2) start API を使用してオーディオ再生 I2S フレームワークの低レベルハードウェアを開始します。
3) play API を使用して PCM オーディオサンプルを再生します。
4) ハードウェアでサポートされる PCM オーディオサンプルは dataTypeGet API によって提供されます。
5) stop API を使用してオーディオ再生 I2S フレームワークの低レベルハードウェアを停止します。
6) close API を使用してオーディオ再生 I2S フレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

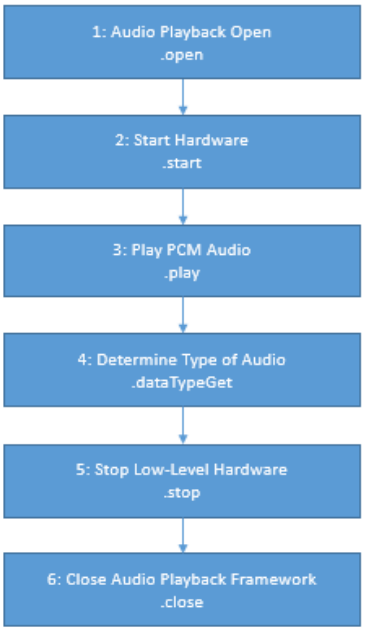


図 132: 通常のオーディオ再生フレームワークモジュールアプリケーションのフロー図

5.1.6 BLE フレームワーク

Bluetooth Smart とも呼ばれる Bluetooth® Low Energy（BLE）はクラシック Bluetooth の軽量サブセットであり、Bluetooth 4.0 コア仕様に導入されました。BLE はクラシック Bluetooth とは対照的に、消費電力を非常に低く抑えるように設計されています。これにより、電力容量が限られている Internet of Thing（IoT）デバイスが、近接するデバイス間で小容量のデータを送信できるようになります。

アプリケーション開発者は、BLEスタックによって提供される機能を利用するためにそのAPIを使用します。さまざまなベンダによって提供される BLE スタック API は標準化されていません。したがって、アプリケーション開発者は異なる BLE スタックにコードを移植する際にコードを更新する必要があります。

Synergy BLE フレームワークは、さまざまなベンダによって提供される基礎となる BLE スタック用の汎用インタフェースを提供することでこの問題に対処し、アプリケーションとベンダ固有 BLE スタックコードが対になることを防ぎます。汎用 API の使用により、アプリケーション開発が単純かつ移植可能になります。

BLE フレームワークは BLE アプリケーション用の高レベル API を提供し、sf_ble_rl78g1d. として実装されます。 BLE フレームワークは Synergy ソフトウェアパッケージ （SSP）通信フレームワークを使用し、このフレームワークが、基礎となる BLE モジュールと通信するための UART ドライバーを有効にします。これによって汎用 BLE プロファイルフレームワーク (g_sf_ble_onboard_profile) も生成され、BLE プロファイルに対して統一されたインタフェースが提供されます。 RL78G1D BLE ハードウェアモジュールの場合、汎用 BLE プロファイルは BLE モジュールファームウェアによって実装されます。

5.1.6.1 機能

Synergy BLE フレームワークは以下の機能に対応します。

- ThreadX® RTOS 対応およびスレッドセーフ
- Bluetooth v4.2 準拠フレームワーク。
- Generic Access Profile （GAP）機能
  - ユーザー定義アドバタイジングデータ
  - セキュリティモード 1 および 2
  - ペリフェラルロールおよびセントラルロール
  - 最大 6 デバイスに対応するホワイトリスト
  - ボンディングサポート
- Generic Attribute Profile （GATT）機能
  - GATT クライアントおよびサーバー
- Generic Attribute Profile （GATT） API
- Generic Access Profile （GAP） API
- 汎用オンボードプロファイル API

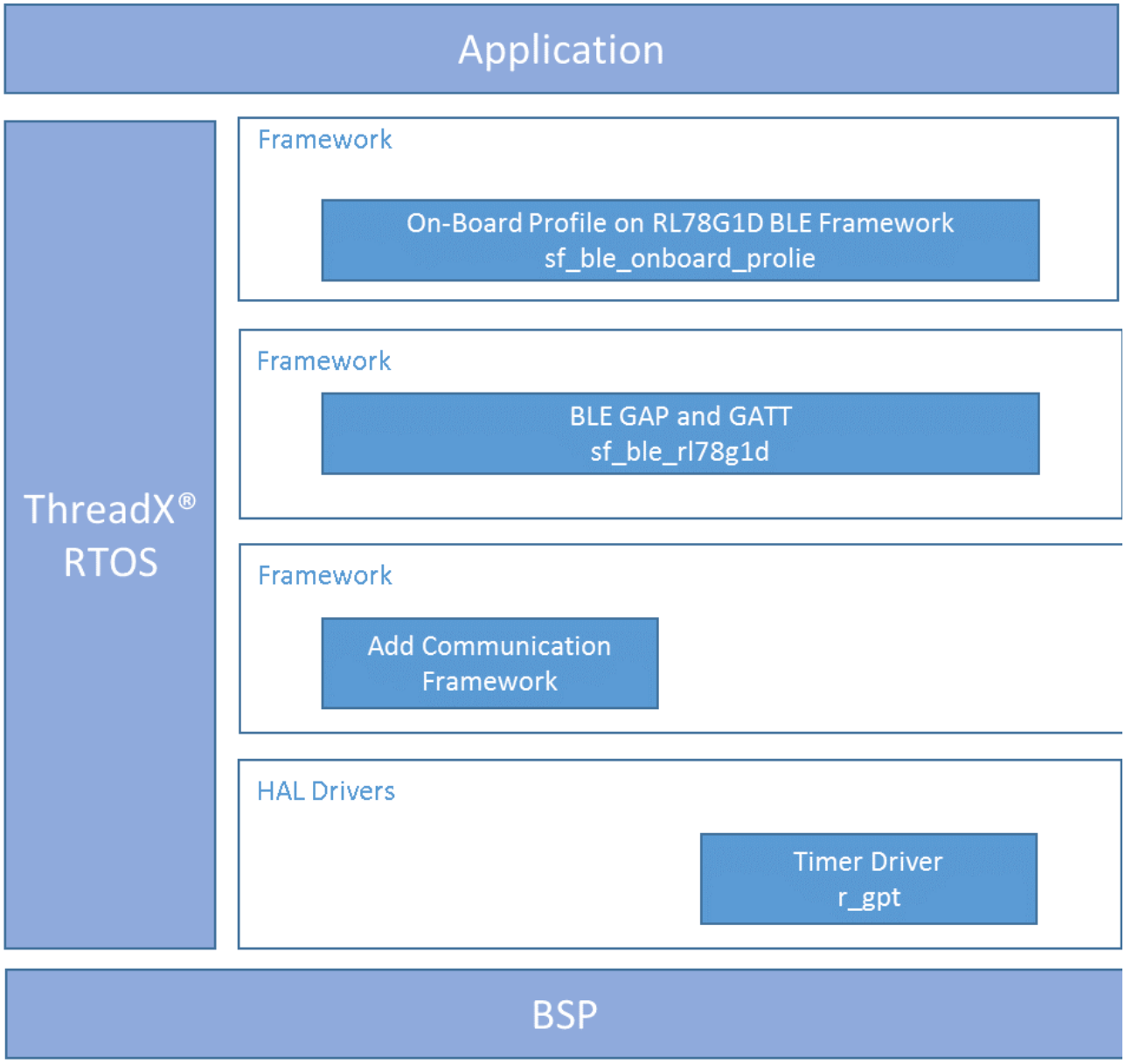


図 133: BLE フレームワークモジュールの編成、オプション、スタックの実装

5.1.6.2 BLE フレームワークモジュールの API の概要

BLE フレームワークは、初期化、値の設定と取得、モジュールの停止の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

BLE フレームワークモジュールのオンボードプロファイル API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_ble_onboard_profile0.p_api->open (g_sf_ble_onboard_profile0.p_cfg); この API はデータ転送のインタフェースを初期化します。
.close	g_sf_ble_onboard_profile0.p_api->close (g_sf_ble_onboard_profile0.p_cfg); この API はインタフェースの初期化を解除し、ローパワーモードにするか電源をオフにします。 API はドライバーを閉じ、ドライバーのリンクと割り込みを無効にします。
.onbpEnable	g_sf_ble_onboard_profile0.p_api->onbpEnable (sf_ble_onboard_profile0.p_cfg, p_handle, profile, p_prf_cb, sec); サーバーモードまたはクライアントモードでプロファイルを有効にします。
.onbpServerWriteData	g_sf_ble_onboard_profile0.p_api->onbpServerWriteData (sf_ble_onboard_profile0.p_cfg,p_handle, profile, characteristics, p_data); ローカルデータベースで特性の値を更新します。
.onbpServerSendNotification	g_sf_ble_onboard_profile0.p_api->onbpServerSendNotification (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics, p_data); 通知を送信します。
.onbpServerSendIndication	g_sf_ble_onboard_profile0.p_api->onbpServerSendIndication (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics, p_data); 指示を送信します。
.onbpClientWriteCCCD	g_sf_ble_onboard_profile0.p_api->onbpClientWriteCCCD (sf_ble_onboard_profile0.p_cfg, p_handle, profile, cccd_char, cccd_val); リモートデバイスでクライアント構成制御記述子を設定します。
.onbpDisable	g_sf_ble_onboard_profile0.p_api->onbpDisable (sf_ble_onboard_profile0.p_cfg, p_handle, profile); サーバーモードまたはクライアントモードでプロファイルを無効にします。
.onbpClientReadChar	g_sf_ble_onboard_profile0.p_api->onbpClientReadChar (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics); プロファイルまたはサービスに関連付けられたGATT特性を読み取ります。

Function Name	API の呼び出し例と説明
.onbpClientWriteChar	g_sf_ble_onboard_profile0.p_api->onbpClientWriteChar (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics); プロファイルまたはサービスに関連付けられたGATT特性を書き込みます。
.versionGet	g_sf_ble_onboard_profile0.p_api-> versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している SSP リファレンスマニュアルを参照してください。*

BLE フレームワークモジュールの GAP API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_ble0.p_api->open(g_ sf_ble_0.p_cfg); この関数は、BLE モジュールを初期化して有効にします。これは BLE モジュール構成を引数として受け取ります。引数には次のパラメータがあります。 - 48 ビット Bluetooth アドレス - デバイススキャンインターバル - デバイススキャンウィンドウ - デバイス検出可能時間 - デバイス接続インターバル - スレーブレイテンシ - スーパービジョンタイムアウト - 独自アドレスタイプ - 接続できる最大スレーブ数
.close	g_sf_ble0.p_api->close (g_ sf_ble0.p_cfg); この API はインタフェースの初期化を解除し、BLE モジュールをローパワーモードにするか電源をオフにします。また、ドライバーを閉じ、ドライバーのリンクと BLE モジュールドライバーの割り込みを無効にします。
.infoGet	g_sf_ble0.p_api->infoGet (g_sf_ble_0.p_cfg, p_handle, p_ble_info); infoGet API は BLE 制御構造体を引数として取ります。BLE モジュールから取得した次の情報を返します。 - チップセットまたはドライバーの情報文字列 - RSSI 値（無符号整数 16 ビット）

Function Name	API の呼び出し例と説明
.provisionGet	g_sf_ble0.p_api->provisionGet (g_sf_ble_0.p_cfg, p_ble_provisioning); provisionGet API は BLE GAP プロビジョニング情報を取得し、BLE 制御構造体を引数として取ります。これは次のパラメータを返します。 - ボンディングモード - セキュリティモード - GAP ロール（セントラルかマスタまたはペリフェラルかスレーブ） - セキュリティキー
.provisionSet	g_sf_ble0.p_api->provisionSet (g_sf_ble_0.p_cfg, p_ble_provisioning); provisionSet() 関数は BLE モジュールをプロビジョニングします。これは BLE 制御構造体とプロビジョニング構造体を引数として取ります。 - ボンディングモード - セキュリティモード - GAP ロール（セントラルかマスタまたはペリフェラルかスレーブ） - セキュリティキー - GAP ユーザーイベントコールバック
.scan	g_sf_ble0.p_api->scan (g_sf_ble_0.p_cfg, p_scan, p_cnt, p_scan_info); scan() 関数は BLE 制御構造体を引数として取ります。scan() 関数は、BLE モジュールがスキャンした BLE デバイスのリストを以下のパラメータを使用して返します。 - Bluetooth アドレス（48 ビット） - RSSI - スキャンデータ scan() 関数はデバイス数を引数として取り、この引数は入力パラメータおよび出力パラメータとして作動します。これはスキャン結果配列のサイズを指定し、BLE フレームワークが、配列に格納されるスキャン結果の数を示す値に設定します。この関数はスキャンタイプを argument(Active/Passive) として受け取ります。

Function Name	API の呼び出し例と説明
.advertisementStart	g_sf_ble0.p_api->advertisementStart (g_sf_ble_0.p_cfg, p_advt_info); advertisementStart() 関数は以下のパラメータを取ります。 - 検出モード （一般または制限） - フィルタポリシー – スキャン要求および接続要求のフィルタの組み合わせをサポート - アドバタイズメントデータ - 接続モード - アドバタイズメントインターバル - チャネルマップ - アドレスタイプ
.advertisementStop	g_sf_ble0.p_api->advertisementStop (g_sf_ble_0.p_cfg); アドバタイズメントを停止します。
.whitelistAdd	g_sf_ble0.p_api->whitelistAdd (g_sf_ble_0.p_cfg, p_bp_addr); whitelistAdd() 関数は、アドバタイズメント、スキャン要求、接続要求のホワイトリストにデバイスを追加します。
.whitelistDel	g_sf_ble0.p_api->whitelistDel (g_sf_ble_0.p_cfg, p_bp_addr); whitelistDel() 関数は、アドバタイズメント、スキャン要求、接続要求のホワイトリストからデバイスを削除します。
.bondingStart	g_sf_ble0.p_api->bondingStart (g_sf_ble_0.p_cfg, p_handle, p_bp_addr, p_bonding_start); bondingStart() 関数はリモートデバイスとのボンディングを開始します。
.bondingResponse	g_sf_ble0.p_api->bondingResponse (g_sf_ble_0.p_cfg, p_handle, p_bp_addr, p_bonding_resp); bondingResponse() 関数はボンディング要求に応答します。
.startEncryption	g_sf_ble0.p_api->startEncryption (g_sf_ble_0.p_cfg, p_enc_info); startEncryption() 関数は暗号化処理を開始します。
.connect	g_sf_ble0.p_api->connect (g_sf_ble_0.p_cfg, p_handle, p_conn);connect 関数はリモートデバイスに接続します。
.disconnect	g_sf_ble0.p_api->disconnect (g_sf_ble_0.p_cfg, p_handle); disconnect() 関数はリモートデバイスから切断します。
.listen	g_sf_ble0.p_api->listen (g_sf_ble_0.p_cfg);listen 関数は、リモートデバイスからの接続要求の受信をリスニングします。

注: 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している SSP リファレンスマニュアルを参照してください。

BLE フレームワークモジュールの GATT API の要約

Function Name	API の呼び出し例と説明
.gattAddCustomProfiles	g_sf_ble0.p_api->gattAddCustomProfiles (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); この関数はカスタムプロファイルをGATTデータベースに追加します。
.gattServiceDiscovery	g_sf_ble0.p_api->gattServiceDiscovery (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); gattServiceDiscovery() 関数はサービス検出を実行します。
.gattCharDiscovery	g_sf_ble0.p_api->gattCharDiscovery (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); gattCharDiscovery() 関数は特性検出処理を実行します。
.gattCharDescDiscovery	g_sf_ble0.p_api->gattCharDescDiscovery (g_sf_ble_0.p_cfg, p_handle, start_handle, end_handle, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); リモートデバイスで GATT 特性記述子を検出します。
.gattCharRead	g_sf_ble0.p_api->gattCharRead (g_sf_ble_0.p_cfg, p_handle, start_handle, p_char_read_req, p_char_read_rsp); リモートデバイスの GATT 特性記述子を読み取ります。
.gattCharWrite	g_sf_ble0.p_api->gattCharWrite (g_sf_ble_0.p_cfg, p_handle, p_char_read_req); リモートデバイスに GATT 特性記述子を書き込みます。
.gattCharExecuteWrite	g_sf_ble0.p_api->gattCharExecuteWrite (g_sf_ble_0.p_cfg, p_handle, execute_flag); リモートデバイスに GATT 特性のライト（コミット）を実行します。
.gattCharWriteLocal	g_sf_ble0.p_api->gattCharWriteLocal (g_sf_ble_0.p_cfg, char_handle, data_length); ローカル GATT データベースを更新します。

Function Name	API の呼び出し例と説明
.gattSendNotify	g_sf_ble0.p_api->gattSendNotify (g_sf_ble_0.p_cfg, p_handle, char_handle); ローカル GATT サーバーからリモート GATT クライアントに通知を送信します。
.gattSendIndicate	g_sf_ble0.p_api->gattSendIndicate (g_sf_ble_0.p_cfg, p_handle, char_handle); ローカル GATT サーバーからリモート GATT クライアントに指示を送信します。
.gattWriteResponse	g_sf_ble0.p_api->gattWriteResponse (g_sf_ble_0.p_cfg, p_handle, char_handle); リモートGATTクライアントからの特性値のライト要求に応答します。
.versionGet	g_sf_ble0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している SSP リファレンスマニュアルを参照してください。*

エラーステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。
SSP_ERR_INVALID_PTR	p_version が NULL です。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。関連するすべてのエラーコードの定義については、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』を参照してください。*

5.1.6.3 BLE フレームワークモジュールの動作の概要

このセクションでは、Synergy BLE フレームワークソフトウェアアーキテクチャの概要を示し、ユーザーのアプリケーションレベルの動作フローシーケンスに沿って、BLE フレームワークの一部として使用される主な SSP モジュールについて詳しく説明します。

注: BLE フレームワークモジュールの総合的な説明は、『BLE Framework- Application Project』にあります。プロジェクトと関連するアプリケーションに関する完全な説明は、 http://www.renesas.com ホームページの上部の検索バーで「r30qan0309eu」を検索すると見つかります。

BLE フレームワークモジュールの動作の説明

BLE フレームワークはアプリケーションの共通インタフェースを提供します。このインタフェースの実装はモジュールごとに固有です。現在、Synergy BLE フレームワークは、RL78G1D BLE モジュールに対して実装されるインタフェースを定義します。各実装は対応する BLE デバイスドライバーとやりとりします。BLE デバイスドライバーは基礎となる SSP 通信フレームワーク (g_sf_comms) を使用し、このフレームワークが汎用非同期送受信機（UART）、データトランスファコントローラ（DTC）、汎用 PWM タイマ（GPT）ドライバーなど SSP HAL コンポーネントとやりとりして、BLE モジュールと通信します。次の図に、BLE フレームワークモジュールのアーキテクチャ概要を示します。

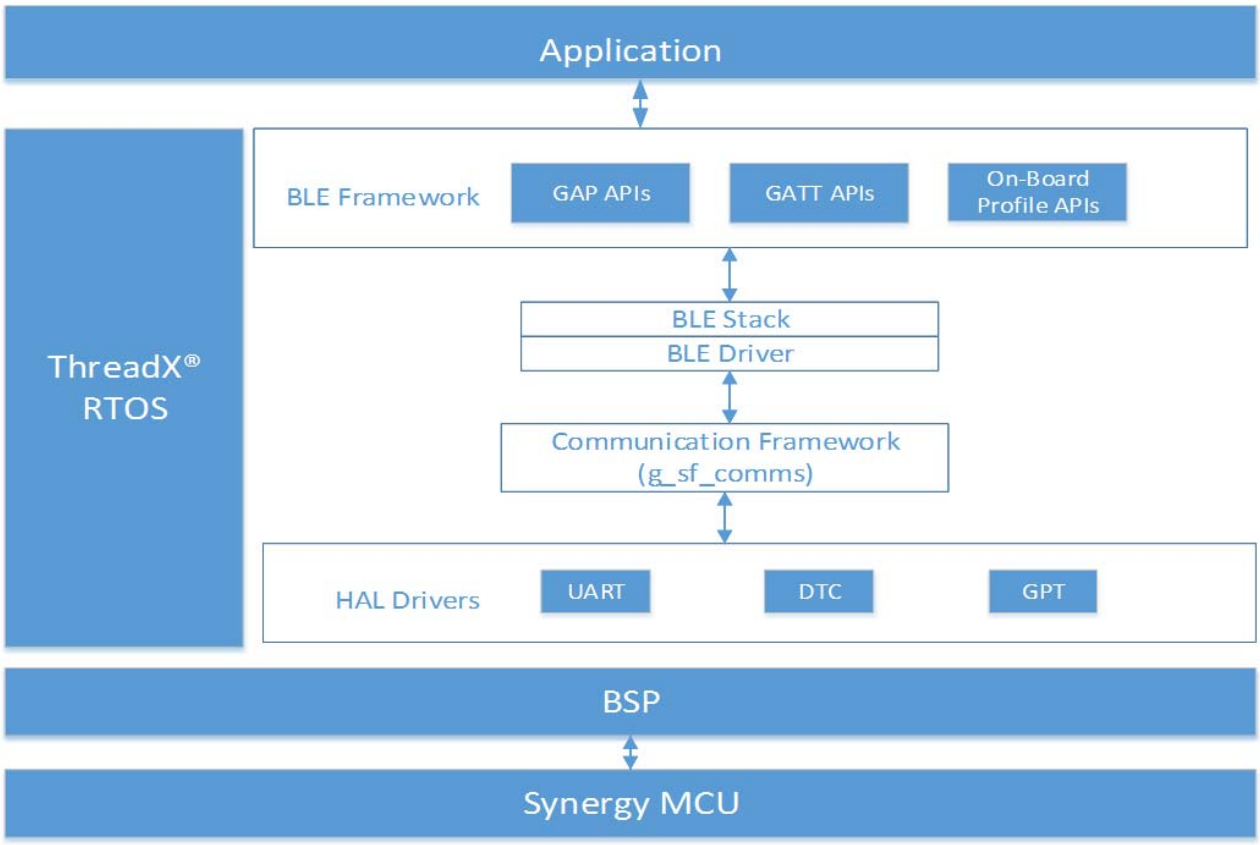


図 134: 一般的な BLE モジュールのアーキテクチャタイプ

GAP API および GATT API

BLE フレームワークは、BLE モジュールの構成とプロビジョニングを行うアプリケーションのために汎用インタフェースを提供します。BLE モジュールには、一連の Bluetooth Smart 規格によって指定されたさまざまな構成パラメータがあります。個別のデバイスドライバーまたは BLE モジュール（あるいは両方）ですべての構成パラメータがサポートされるとは限りません。プロビジョニング API で最小限提供されるのは、BLE

インタフェースの動作モード、セキュリティモード、セキュリティキー、ボンディングモードを設定するメカニズムです。 GAP および GATT レイヤーの API も提供されます。

オンボードプロファイル API

オンボードプロファイル API では、BLE モジュールファームウェアによって実装される BLE プロファイルに統一インタフェースが提供されます。

BLE スタック

通常、 BLE モジュールホストスタックは BLE モジュールベンダによって提供されます。一般的に BLE モジュールは、ホスト MCU と BLE モジュールの間の HW/SW パーティションに応じて 3 種類あります。RL78G1D BLE モジュールはネットワークコントローラ実装アーキテクチャに含まれ、このアーキテクチャでは、 BLE チップセットに BLE リンクレイヤー、GAP、GATT、オンボードプロファイルのすべての実装が含まれます。このモジュールは、SSP によって提供される sf_comms フレームワークを介して MCU とのインタフェースになります。

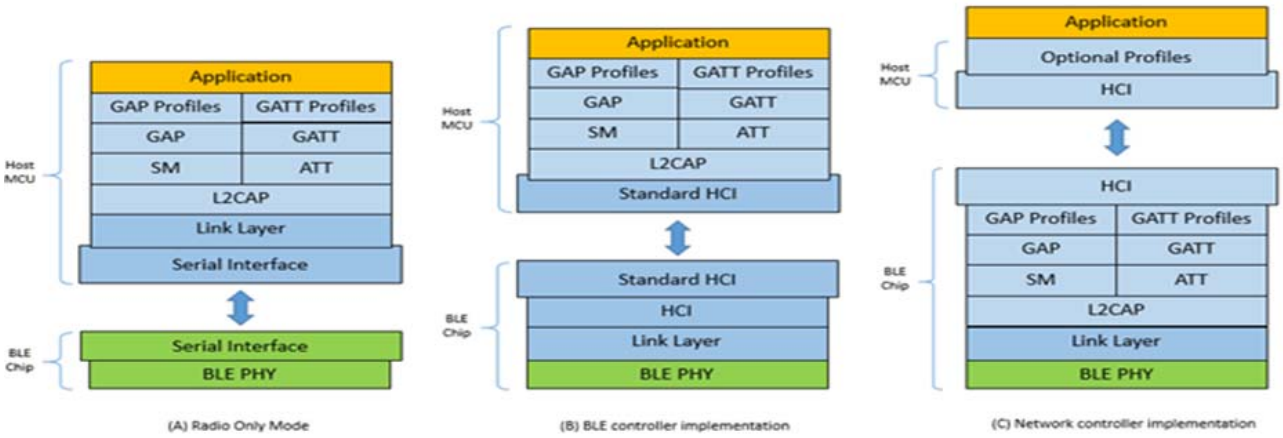


図 135: BLE モジュールのアーキテクチャタイプ

A: BLE 無線専用モード

リンクレイヤー、L2CAP、GATT、GAP レイヤー、プロファイル、アプリケーションはホスト MCU で実行します。物理レイヤーは BLE チップセットで実行します。

B: BLE コントローラ実装

リンクレイヤーは BLE チップセットで実行し、L2CAP レイヤーと上位 BLE プロトコル（GATT、GAP）レイヤー、プロファイル、アプリケーションはホスト MCU で実行します。

C: ネットワークコントローラ実装

リンクレイヤー、L2CAP、GATT、GAP レイヤー、汎用プロファイルは、BLE チップセットで実行します。オプションのプロファイルとアプリケーションはホストプロセッサで実行します。

5.1.6.4 BLE フレームワークインスタンス

アプリケーションは、BLE フレームワークインスタンスを使用する前に定義する必要があります。インスタンスは次のいずれかへのポインタを含む構造体です。

BLE フレームワーク制御構造体

この構造体はすべての BLE フレームワーク API で使用されます。この構造体にはドライバーハンドルへのポインタが含まれます。ドライバーハンドルは、BLE デバイスドライバーで必要な情報を格納するためにフレームワークが使用します。

BLE フレームワーク構成構造体

この構造体は open() API に渡されます。この構造体を使用して BLE モジュールを構成できます。この構成は、初期化中（open など）にもプロビジョニング中（provisioningSet など）にも適用されます。BLE モジュールでサポートされない構成パラメータはフレームワークによって無視されます。

BLE フレームワーク API 構造体

この構造体には、所定のモジュール固有の BLE フレームワーク API へのポインタが含まれます。これらの API の詳細については、「BLE フレームワークモジュールの API の概要」を参照してください。

BLE フレームワークモジュールの動作フロー

アプリケーションで BLE フレームワークモジュールを使用する際の手順は次のとおりです。

1) BLE ハードウェアモジュールを初期化します。
2) GATT レイヤーのロール（GATT クライアントまたは GATT サーバー）を選択します。スレーブ（ペリフェラル）デバイスを GATT サーバーに、マスタ（セントラル）デバイスを GATT クライアントにするのが最も一般的です。
3) アプリケーションが汎用（オンボード）プロファイル API または GAP API か GATT API を使用して動作を制御します。

次の BLE モジュール初期化シーケンスは Synergy 自動生成コードの一部です。

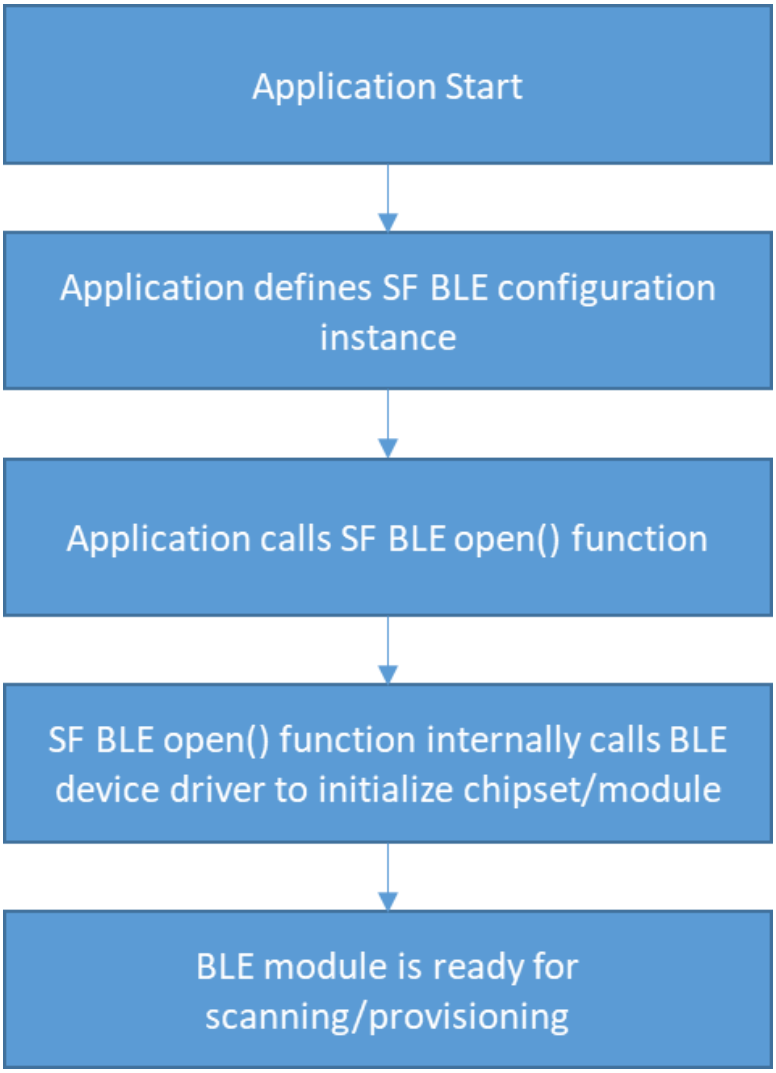


図 136:　自動生成初期化シーケンスフローチャート

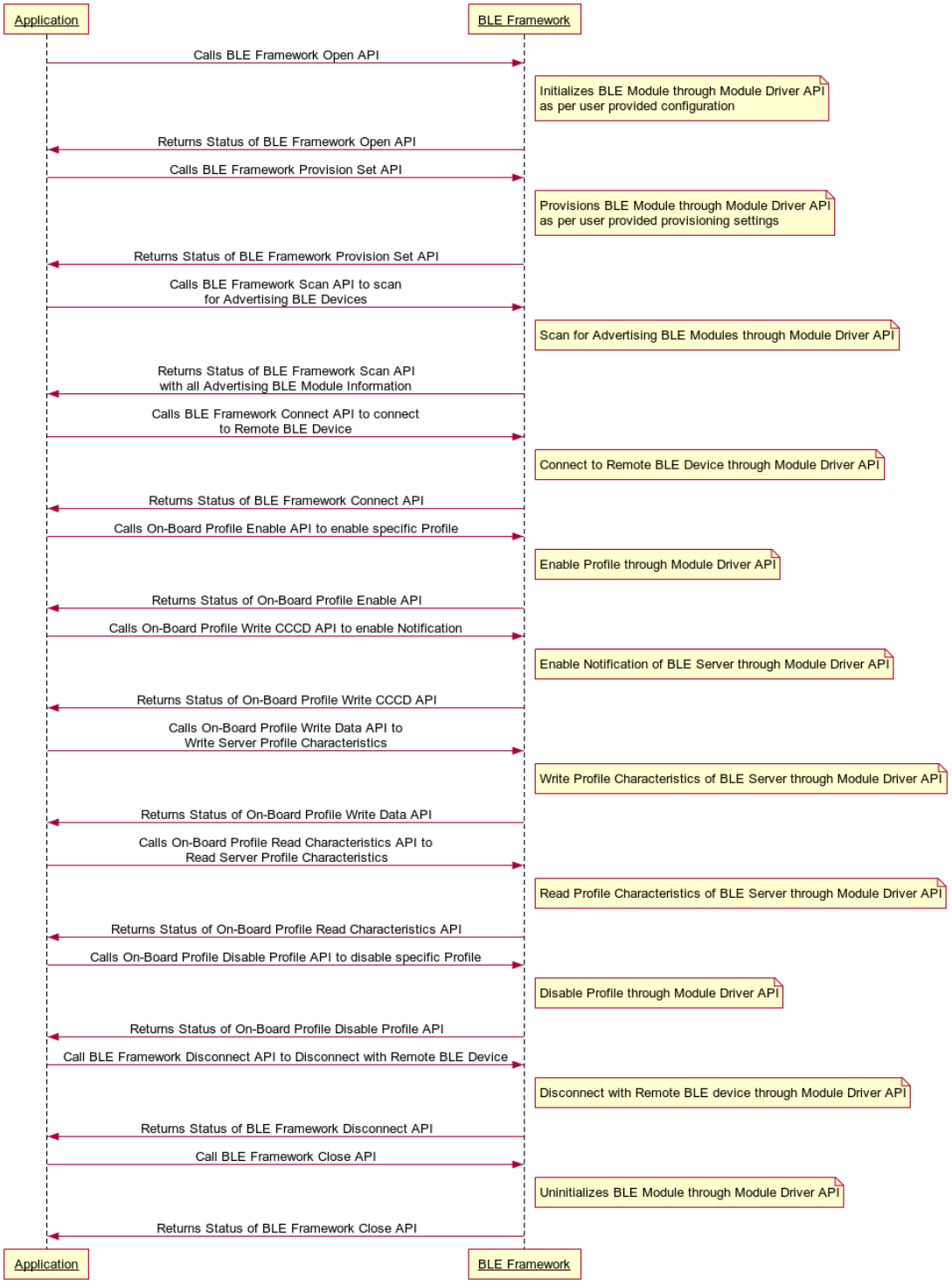


図 137:　オンボードプロファイルのクライアントアプリケーションフロー

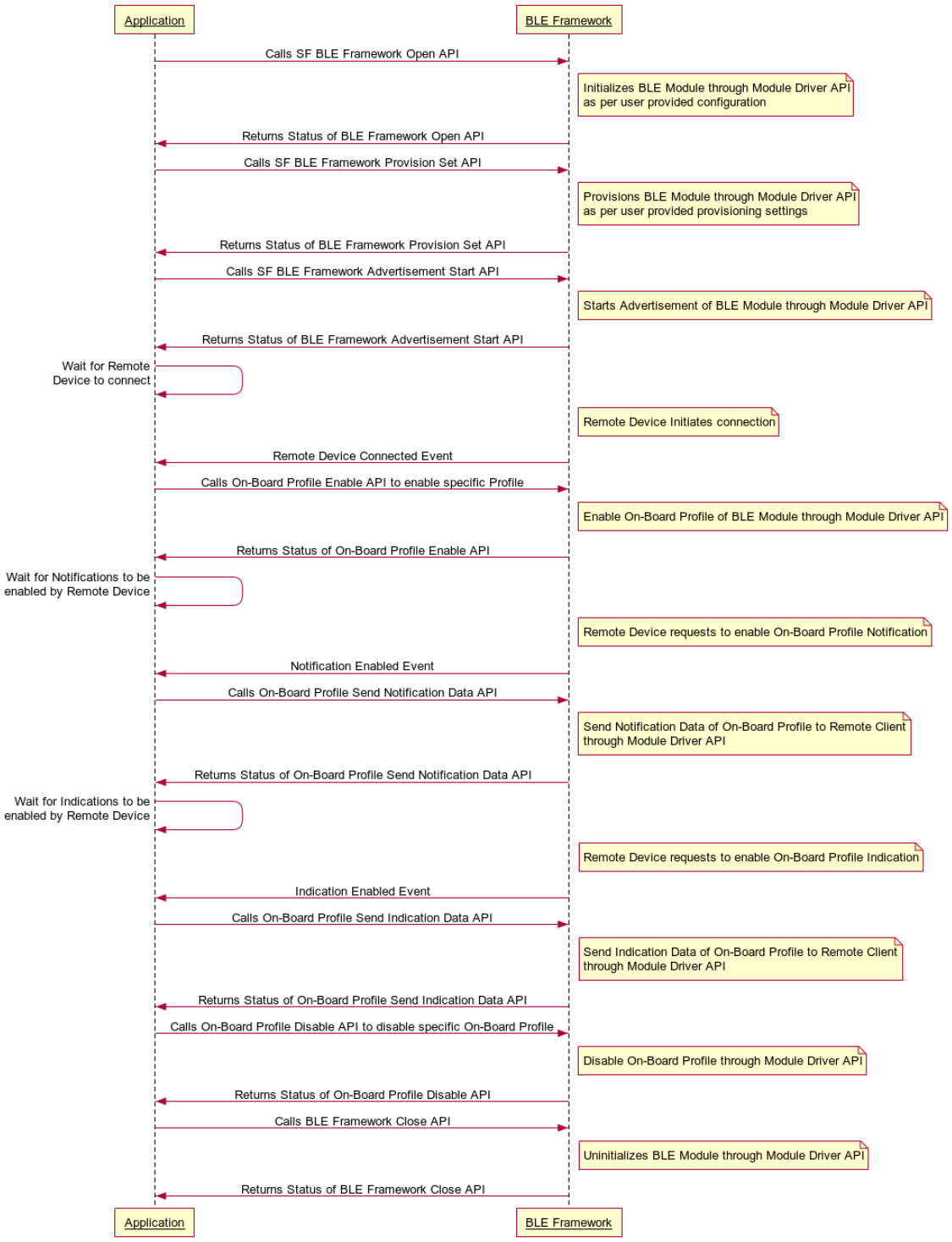


図 138: オンボードプロファイルのサーバーアプリケーションフロー

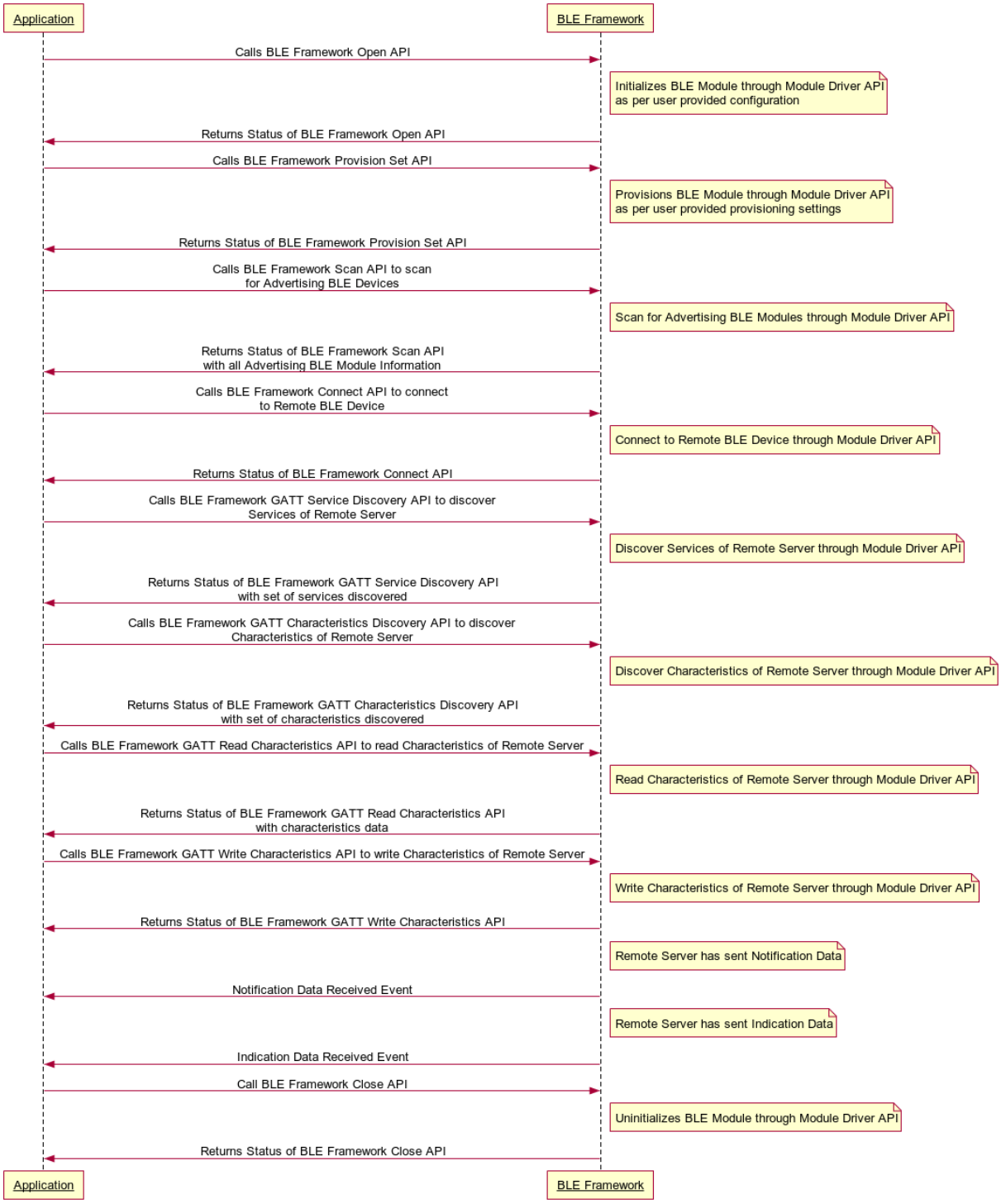


図 139: GAP および GATT のクライアントアプリケーションフロー

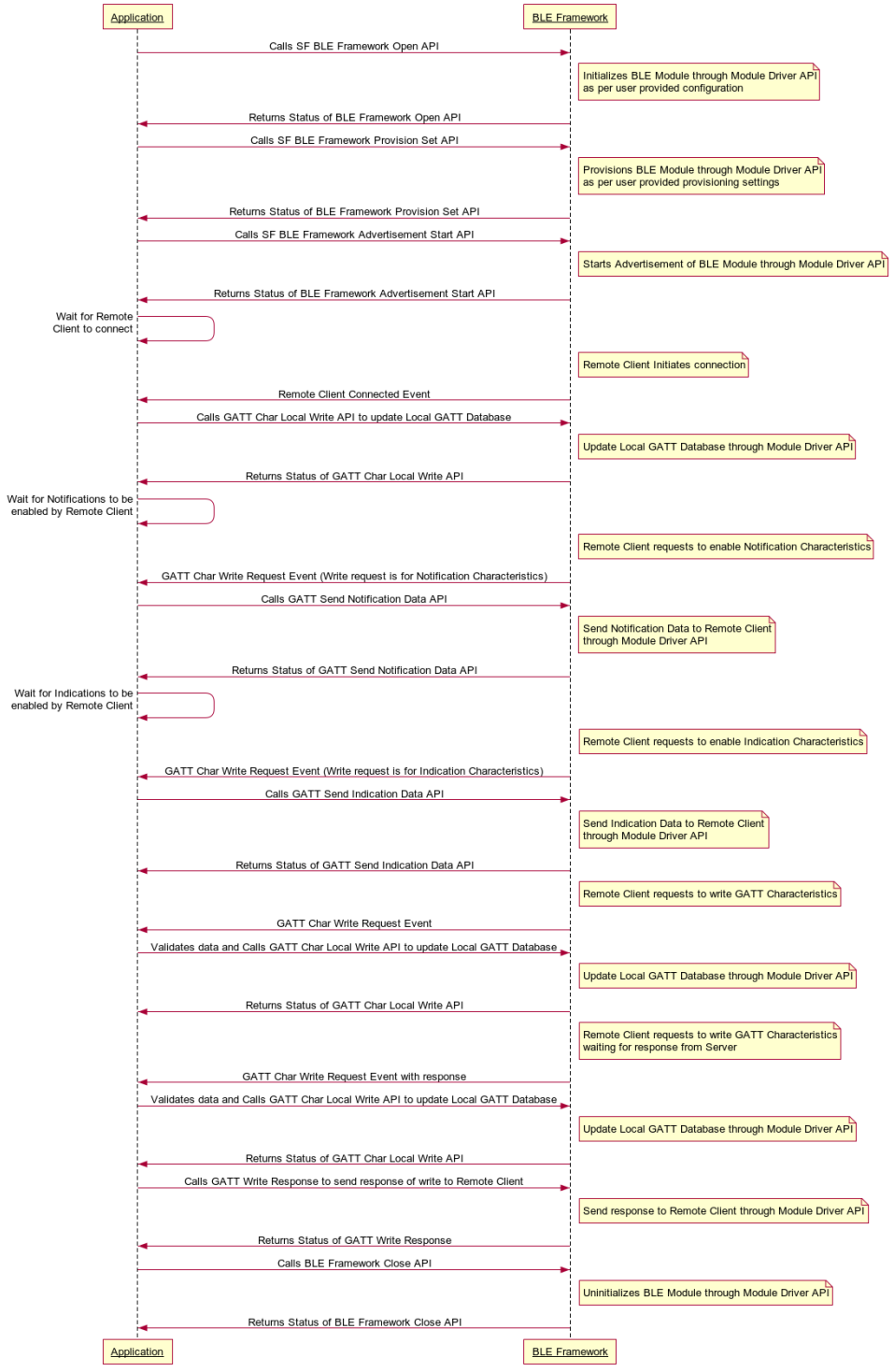


図 140: GAP および GATT のサーバーアプリケーションフロー

BLE フレームワークセキュリティ

セキュリティマネージャは、通信リンクの暗号化に使用されるセキュリティキーの生成と交換の機能を BLE プロトコルスタックに提供します。セキュリティマネージャには次の 2 つの機能があります。

- **イニシエータ**
  - これは GAP マスタ（セントラル）デバイスです。
- **レスポンダ**
  - これは GAP スレーブまたは周辺デバイスです。

イニシエータはセキュリティ手順を開始するマスタデバイスです。ただし、スレーブデバイスは、イニシエータがセキュリティ手順を開始するように非同期で要求できます。

BLE セキュリティで提供されるモードには、各モードに関連付けられたレベルがあります。セキュリティのモードおよびレベルは、認証または未認証のペアリング、暗号化、データ署名のサポートの組み合わせです。ペアリングはさまざまなセキュリティ要件を満たすために必要です。 2 種類のペアリングがあります。

認証されたペアリング : デバイスが MITM（中間者）攻撃から保護されます。

認証されていないペアリング : デバイスが MITM から保護されません。

- **セキュリティモード 1**
  - セキュリティレベル 1: セキュリティなし
  - セキュリティレベル 2: 未認証ペアリングと暗号化
  - セキュリティレベル 3: 認証ペアリングと暗号化
  - セキュリティレベル 4: 認証 LE セキュア接続ペアリングと暗号化
- **セキュリティモード 2**
  - セキュリティレベル 1: 未認証ペアリングとデータ署名
  - セキュリティレベル 2: 認証ペアリングとデータ署名

***注:** RL78G1D BLE モジュールでは、セキュリティレベル 4 のセキュリティモード 1 はサポートされません。*

BLE セキュリティには以下の手順があります。

ペアリング

この手順は、通信リンクを暗号化する一時暗号化キーを生成するために使用されます。

永続暗号化キーはこの暗号化済みの通信リンクで共有して、追加の通信で使用できます。

ボンディング

これは、ペアリングと永続キーの格納を組み合わせたものです。ペアリングの後で永続キーが不揮発性メモリに格納されると、それによって 2 つのデバイス間の永続的なボンディングが形成されます。その後の通信では、デバイスがボンディング手順を実行する必要はありません。

暗号化の確立

通信は永続キーを使用して暗号化されます。

ペアリングによって作成されたセキュアなリンクは接続の間のみ継続しますが、ボンディングではボンドと呼ばれる永続的な関係が形成されます。

BLE セキュリティには 3 つのフェーズがあります。2 つのデバイスが GAP 接続手順を使用して接続を確立した後で、3 つのフェーズによりセキュアな通信リンクを確立します。

フェーズ 1（ペアリングフェーズ、情報共有）

最初のフェーズ 1 では、一時キーの生成に必要なすべての情報が 2 つのデバイス間で共有されます。

フェーズ 2（ペアリングフェーズ、一時キー共有）

このフェーズでは、一時暗号化キー（Short Term Key すなわち STK）が両方のデバイス上で生成されます。これは接続の暗号化に使用されます。こうして暗号化されたリンクをさらに他の通信に使用できます。ピアデバイスが接続されている間、この通信リンクは暗号化されたままになります。

フェーズ 3（ボンディング、永続キーの共有と格納）

ボンディングが必要な場合、デバイスはこのフェーズに進みます。このフェーズでは、暗号化されたリンク（フェーズ 2 で一時キーを使用して確立）を使用して、永続キー（Long Term Key すなわち LTK）が 2 つのデバイス間で交換されます。これらの永続キーは不揮発性メモリに格納され、各接続のためにデバイスが使用できるようになります。

5.1.6.5 BLE ボタンフレームワークの制限事項

1) BLE フレームワークのテストは RL78G1D BLE ハードウェアモジュールでしか行われていません。さまざまな BLE モジュールのサポートは今後のバージョンで追加される予定です。
2) RL78G1D を使用する BLE フレームワークではコンパイル時に警告が表示されます。すべての警告はサードパーティの RL78G1D ドライバーコードに対するものです。BLE フレームワークファイルには警告はありません。これらの警告はユーザーアプリケーションには影響しません。
3) BLE フレームワークでのカスタムプロファイルサポートは、RL78G1D タイプの BLE ハードウェアモジュールのみに制限されています。
4) HID プロファイルクライアントモードは RL78G1D BLE ハードウェアモジュールによってサポートされていません。この結果、HID プロファイルの BLE フレームワーク実装でも HID プロファイルクライアントモードはサポートされません。RL78G1D 対応の BLE フレームワークを使用するアプリケーションは、クライアントモードで HID プロファイルを使用できません。
5) 複数のスレーブ BLE デバイスを RL78G1D BLE モジュールに接続することはできません。
6) モジュールの制限事項については、SSP の最新のリリースノートを参照してください。

5.1.6.6 アプリケーションへの BLE フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して BLE フレームワークモジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

BLE フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。(BLE フレームワークのデフォルト名は g_sf_ble0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

BLE フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_ble0 RL78G1D BLE GAP and GATT on sf_ble_rl78g1d	Threads	New Stack> Framework> Networking> BLE > RL78G1D BLE GAP and GATT on sf_ble_rl78g1d
g_sf_ble_onboard_profile0 On-Board Profile on RL78G1D BLE Framework	Threads	New Stack> Framework> Networking> BLE > On-Board Profile on RL78G1D BLE Framework

次の図に示すように、BLE フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルドライバーの追加が必要な場合は、モジュールの説明に 「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

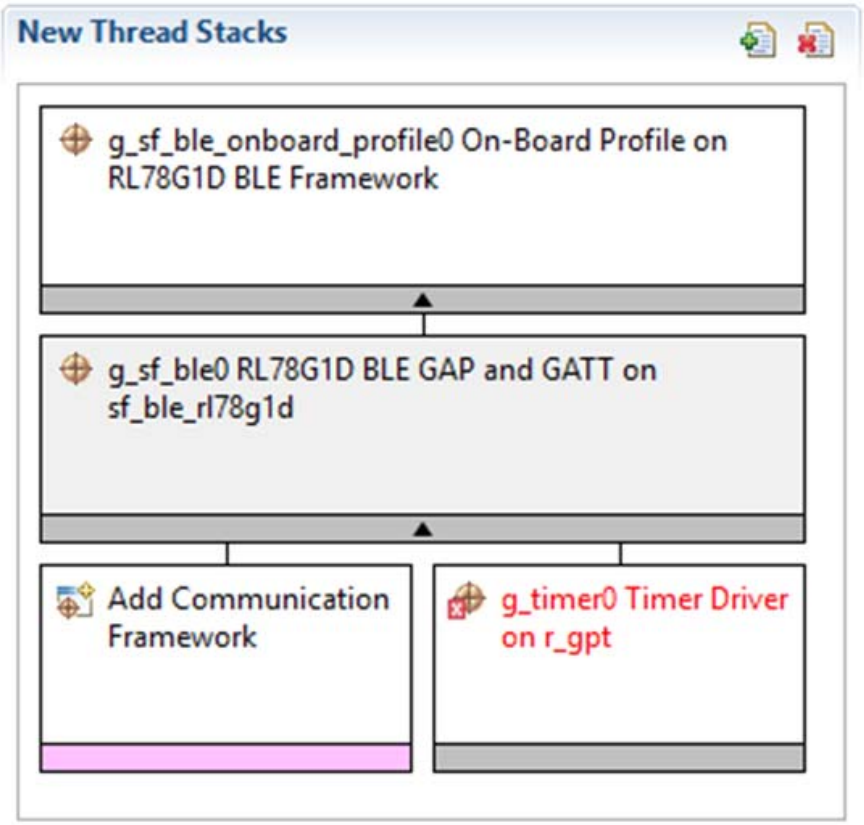


図 141: BLE フレームワークモジュールのスタック

5.1.6.7 BLE フレームワークモジュールの設定

必要な動作に合わせて BLE フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

RL78G1D BLE フレームワーク上のオンボードプロファイルの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Heart Rate Profile	Enabled, Disabled Default: Enabled	心拍数プロファイルの選択
Alert Notification Profile	Enabled, Disabled Default: Disabled	アラート通知プロファイルの選択
Blood Pressure Profile	Enabled, Disabled Default: Disabled	血圧プロファイルの選択
Find Me Profile	Enabled, Disabled Default: Disabled	ファインドミープロファイルの選択
HID Over GATT Profile	Enabled, Disabled Default: Disabled	HID gatt プロファイルの選択
Health Thermometer Profile	Enabled, Disabled Default: Disabled	体温計プロファイルの選択
Phone Status Alert Profile	Enabled, Disabled Default: Disabled	電話アラートプロファイルの選択

ISDE Property	Value	説明
Proximity Profile	Enabled, Disabled Default: Disabled	近接プロファイルの選択
Scan Parameter Profile	Enabled, Disabled Default: Disabled	スキャンパラメータプロファイルの選択
Time Profile	Enabled, Disabled Default: Disabled	時間プロファイルの選択
Name	g_sf_ble_onboard_profile0	モジュール名

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

BLE GAP および GATT の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_sf_ble0	モジュール名
Bluetooth Device Address (Restart Board after first run to see changed Address)	{0x0, 0x0, 0x0, 0x0, 0x0}	Bluetooth デバイスアドレスの選択
Address Type	Public Address, Random Address Default: Public Address	アドレスタイプの選択
Scan Interval	48	スキャンインターバルの選択
Scan Window	48	スキャンウィンドウの選択
Maximum Connection Interval	40	最大接続インターバルの選択

ISDE Property	Value	説明
Connection Slave Latency	0	接続スレーブレイテンシの選択
Supervision Timeout	80	スーパービジョンタイムアウトの選択
BLE Driver Thread Priority	1	BLE ドライバースレッドプライオリティの選択
BLE Serial Thread Priority	1	BLE シリアルスレッドプライオリティの選択

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	チャネルの選択。
Mode	Periodic	警告:ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。
Period Value	10	「タイマ期間の計算」を参照
Period Unit	Milliseconds	「タイマ期間の計算」を参照
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts	デューティサイクル単位の選択

ISDE Property	Value	説明
Auto Start	True	設定後にタイマを起動するには、 true に設定します。 start を呼び出すまで測定を無効状態にするには false に設定します。
GTIOCA Output Enabled	False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、 true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCA Stop Level	Pin Level Low	タイマが停止したときの出力ピンレベルを制御します。
GTIOCB Output Enabled	False	GPT 用に設定されたポートピン上でタイマ信号を出力するには、 true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCB Stop Level	Pin Level Low	タイマが停止したときの出力ピンレベルを制御します。
Callback	RBLE_Timer_cb	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

BLE フレームワークモジュールのクロック構成

BLE フレームワークモジュールは、低レベルモジュールによって指定されるクロックを使用します。

BLE フレームワークモジュールのピン構成

BLE フレームワークモジュールは、低レベルモジュールによって指定されるピンを使用します。

5.1.6.8 アプリケーションでの BLE フレームワークモジュールの使用

***注:** BLE フレームワークモジュールの詳しい使用例は、『BLE Framework- Application Project』にあります。プロジェクトと関連するアプリケーションに関する完全な説明は、 http://www.renesas.com ホームページの上部の検索バーで 「r30an0309eu」を検索すると見つかります。*

オンボードプロファイルクライアントフローを使用して、単純なアプリケーションで BLE フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) 初期化 （open API を使用した、コールバック関数の登録、プロビジョニング、アドバタイズメント）
2) provisionSet API を使用した BLE モジュールのプロビジョニング
3) advertisementStart API を使用したモジュールのアドバタイズ、リモートデバイスの接続を待機
4) onbpEnable API を使用したオンボードプロファイルの有効化、リモートデバイスによる通知の有効化を待機
5) onbpServerSendNotification API を使用した通知の送信、リモートデバイスによる指示の有効化を待機
6) onbpServerSendIndication API を使用した指示の送信
7) onbpDisable API を使用したプロファイルの無効化
8) close API を使用した BLE モジュールのクローズ

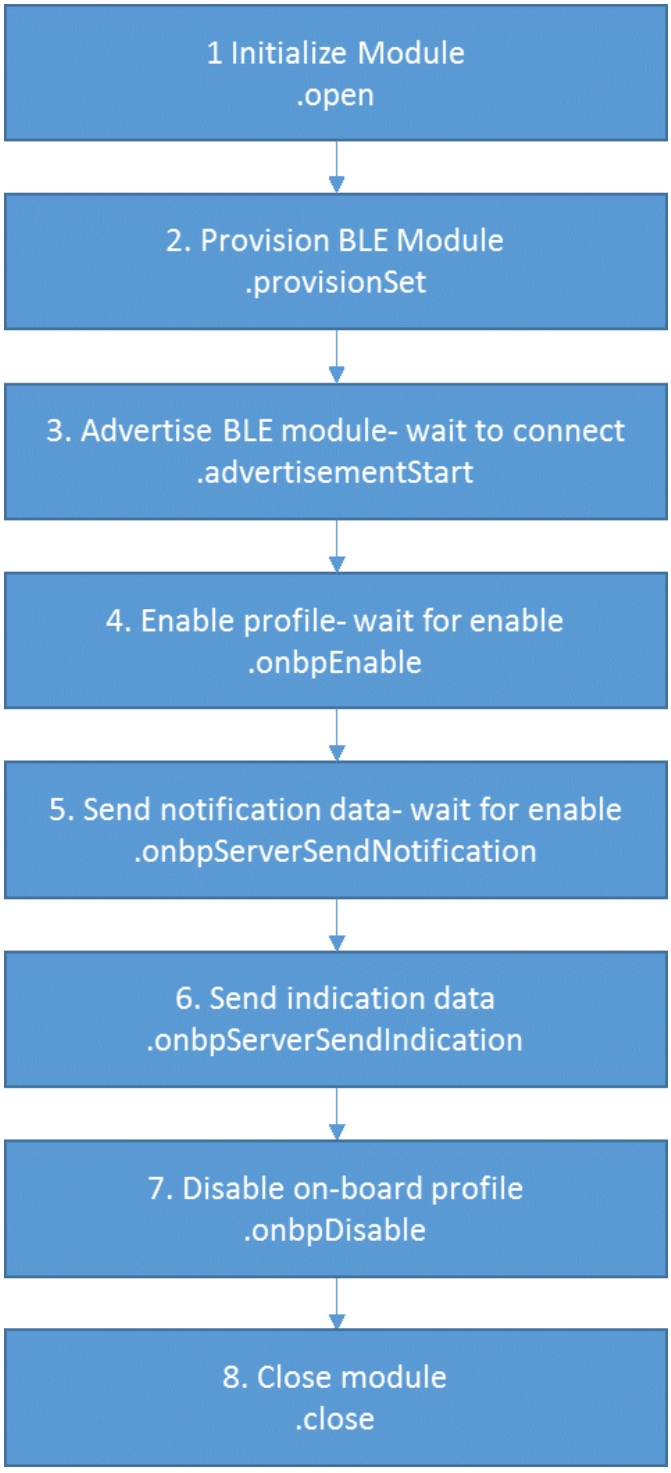


図 142: オンボードプロファイルクライアントでの通常の BLE フレームワークモジュールアプリケーションのフロー

5.1.7 セルラーフレームワーク

セルラーフレームワークモジュールによって、SSP でのセルラーモデム統合のために高レベルアプリケーションレイヤーインタフェースが提供されます。セルラーフレームワークによって、さまざまなベンダのセルラーモデムをアプリケーションで使用するための汎用インタフェースが提供されます。

セルラーフレームワークで提供される一連の API を使用し、アプリケーションはデータ通信のためにセルラーネットワークのプロビジョニング、構成、通信を行います。セルラーフレームワークはコンソールフレームワークを使用し、AT コマンドを使用してシリアルインタフェースを介してセルラーモデムと通信します。セルラーフレームワークは、NetX™ によって提供される PPP WAN プロトコルを利用して、データ通信用のシリアルインタフェース上でシリアルデータパイプを作成します。TCP/IP を使用するデータ通信は、NetX アプリケーションプロトコル、ソケット、あるいは IoT プロトコル（MQTT など）を使用して、この Wide Area Network（WAN）リンク上で確立できます。

また、セルラーフレームワークではフレームワークレベルのソケット API も提供されます。これは、特定のセルラーハードウェアモジュール内のオンチップ（セルラーハードウェアモジュール内）の TCP/IP スタックや、さらにはソケット API を使用するネットワークの TCP/IP リンクと通信するためのものです。

- 以下を使用する接続がサポートされます。
  - CAT1 および CAT3 モデム
  - セルラーモジュールに存在するオンチップスタック対応の BSD ソケットインタフェース
  - NSAL インタフェースを使用する Synergy MCU（ホスト）上の oNetX スタック
- このフレームワークは、主に、ネットワーキングスタックに対応する API の共通セットと、さまざまなセルラーハードウェアモジュール対応の汎用インタフェースで構成されます。
  - NimbeLink CAT3 (NL-SW-LTE-TSVG) Verizon-US
  - NimbeLink CAT3 (NL-SW-LTE-TEUG) India and Europe
  - NimbeLink CAT1 (NL-SW-LTE-GELS3-B) Verizon-US

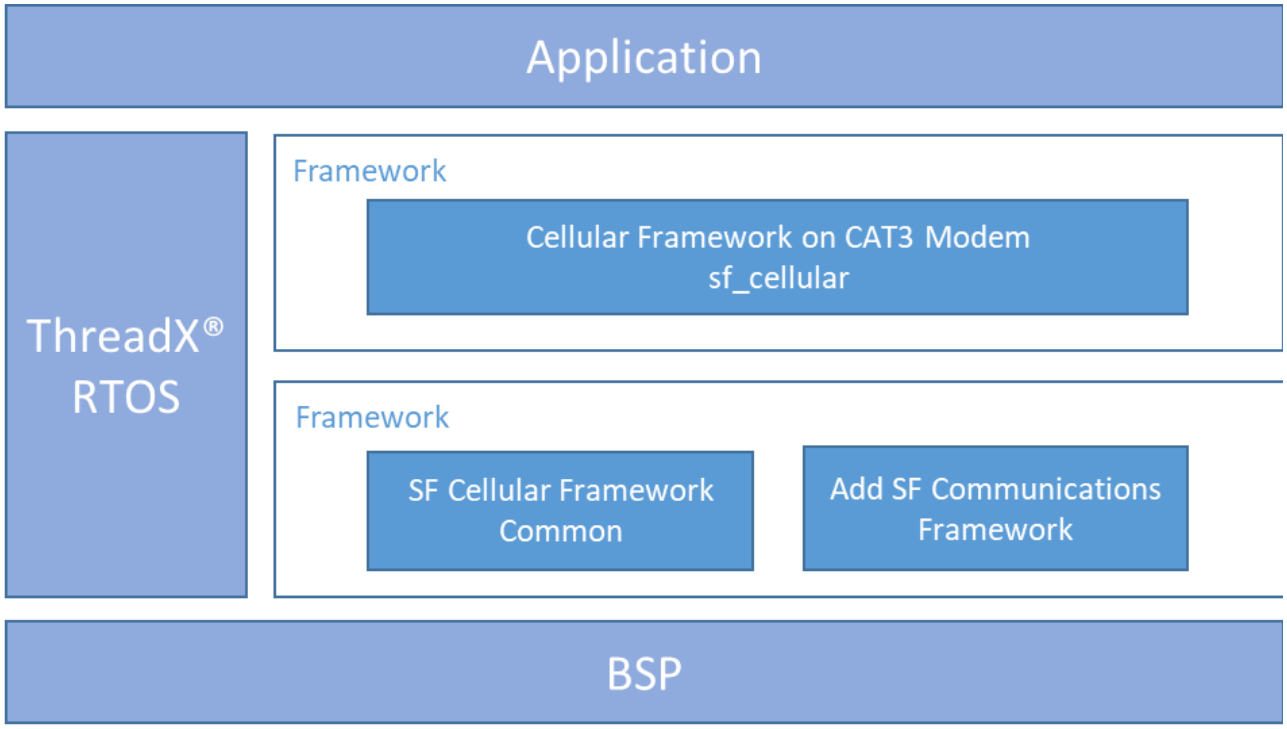


図 143: CAT3 モデム上のセルラーフレームワークモジュールの編成、オプション、スタックの実装

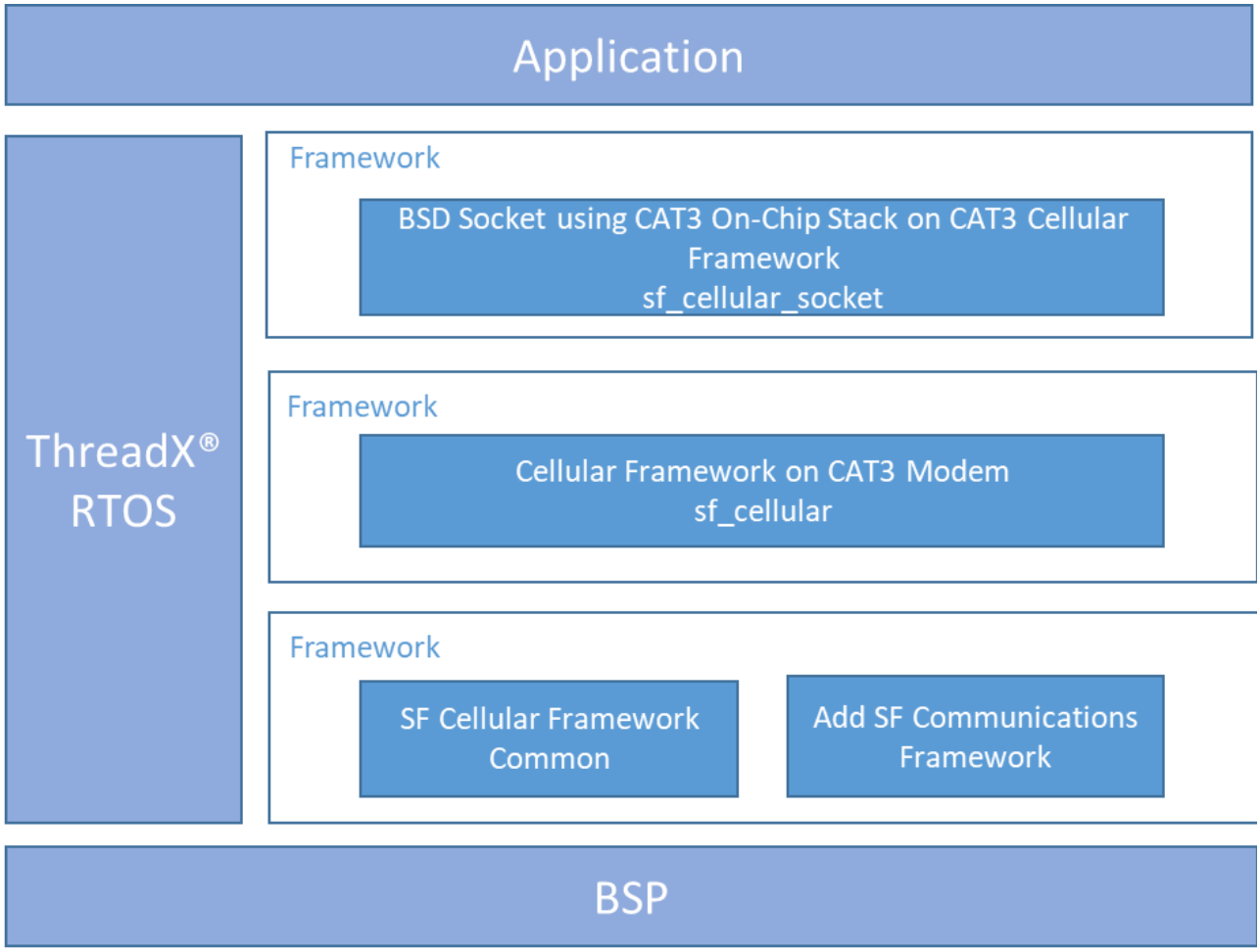


図 144: **CAT3 セルラーフレームワークモジュールでの CAT3 オンチップスタックを使用した BSD ソケットの編成、オプション、スタックの実装**

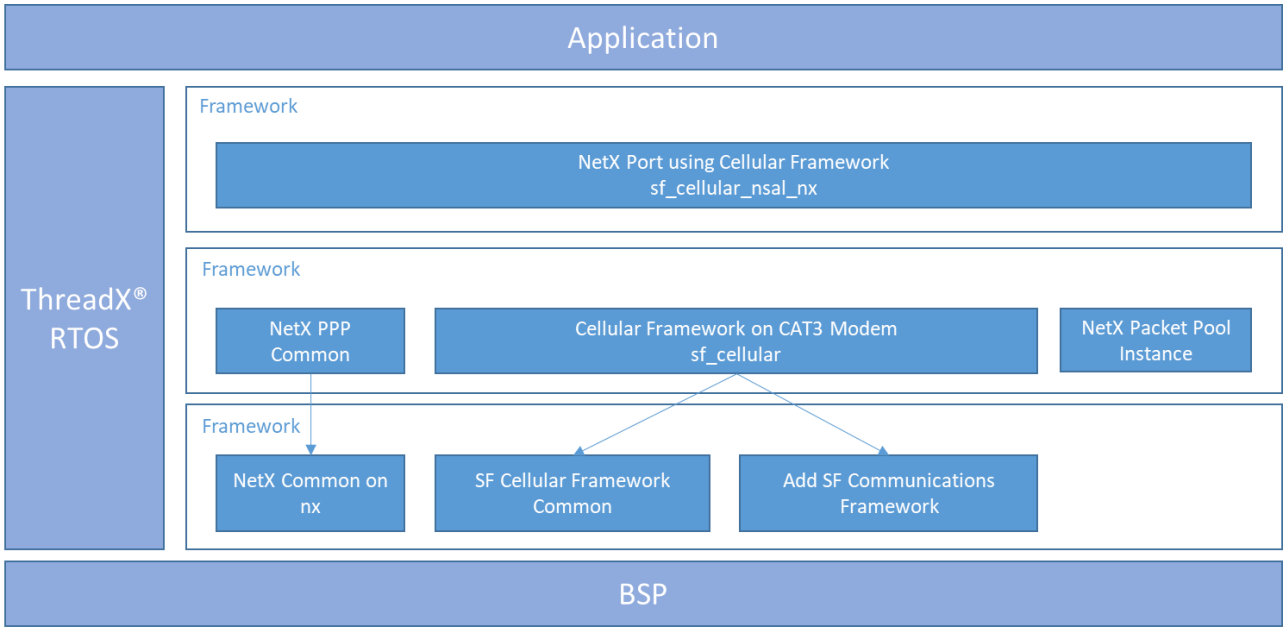


図 145: セルラーフレームワークモジュールを使用した NetX ポートの編成、オプション、スタックの実装

5.1.7.1 セルラーフレームワークモジュールの API の概要

セルラーフレームワークモジュールは、関連する各モジュールの API を定義します。この後で各 API の動作について説明します。その他の詳細については、対応するモジュールの API リファレンスセクションを参照してください。

API の詳しい説明と API の使用方法を示すアプリケーションプロジェクトが、Renesas Web サイトの「Cellular Application Note」にあります。上部の検索バーで 「R11AN0082」を検索すると、アプリケーションの注意事項とアプリケーションプロジェクトが検索結果に表示されます。

セルラーフレームワークモジュールの API 要約

セルラーフレームワークは、ネットワークスタックとアプリケーションのために汎用インタフェースを提供します。このフレームワークで提供される API を次に示します。

セルラーフレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
.open	g_sf_cellular0.p_api->open (g_sf_cellular0.p_ctrl, g_sf_cellular0.p_cfg); この API 関数は、セルラーモジュールを初期化して有効にします。 open 関数は、セルラーフレームワークのインスタンスを一意に特定するセルラー制御構造体を返します。セルラーフレームワークの open 関数は、以下のパラメータを使用してセルラーモジュール構成を引数として受け取ります。 - オペレータ選択モード （列挙） - オペレータ名フォーマット （列挙） - オペレータ名 （文字列） - 優先オペレータリスト （構造体の配列） - タイムゾーン更新ポリシー （列挙）
.close	g_sf_cellular0.p_api->close (g_sf_cellular0.p_ctrl); この API は、セルラーモジュールの初期化を解除して無効にします。引数としてセルラー制御構造体を取ります。
.infoGet	g_sf_cellular0.p_api->infoGet (g_sf_cellular0.p_ctrl, p_cellular_info); この API は、セルラー制御構造体を引数として取り、セルラーモジュールから取得した以下の情報を返します。 - チップセットまたはドライバーの情報 （文字列） - メーカー名 （文字列） - ファームウェアバージョン （文字列） - IMEI 番号 （文字列） - RSSI 値 （無符号整数 16 ビット） - ビットエラー率 （無符号整数 16 ビット）
.statisticsGet	g_sf_cellular0.p_api->statisticsGet (g_sf_cellular0.p_ctrl, p_stats); この API はセルラーモジュールのデータ統計を取得します。これはセルラー制御構造体を引数として取り、以下の統計を返します。 - 受信パケット数 （無符号整数 32 ビット） - 送信パケット数 （無符号整数 32 ビット） - 送信パケットエラー数 （無符号整数 32 ビット）
.transmit	g_sf_cellular0.p_api->transmit (g_sf_cellular0.p_ctrl, p_buffer, length); この API はデータすなわちパケットを送信します。これは、セルラー制御構造体、データバッファ、データバッファ長を引数として取ります。セルラーフレームワークの transmit 関数は、データバッファを PPP ドライバーに渡して送信します。

Function Name	API の呼び出し例と説明
.provisioningGet	g_sf_cellular0.p_api->provisioningGet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); この API は、セルラー制御構造体を引数として取り、以下のパラメータを返します。 - 認証 type(enumeration) - ユーザー名 （文字列） - パスワード （文字列） - APN 名 （文字列） - PDP コンテキスト ID （整数） - PDP コンテキストタイプ （列挙） - 機内モード （列挙）
.provisioningSet	g_sf_cellular0.p_api->provisioningSet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); この API は認証の資格情報を設定します。セルラー制御構造体と以下のパラメータを引数として取り、セルラーモジュールをプロビジョニングします。 - 認証 type(enumeration) - ユーザー名 （文字列） - パスワード （文字列） - APN 名 （文字列） - PDP コンテキスト ID （整数） - PDP コンテキストタイプ （列挙） - 機内モード （列挙）
.networkConnect	g_sf_cellular0.p_api-> networkConnect (g_sf_cellular0.p_ctrl); この API は、ネットワークスタックの支援によりリモートホストと通信できるアプリケーションを使用して、セルラーネットワーク上でネットワーク接続を確立します。引数としてセルラー制御構造体を取ります。
.networkDisconnect	g_sf_cellular0.p_api->networkDisconnect (g_sf_cellular0.p_ctrl); この API は、 networkConnect API を使用して確立されたネットワーク接続を終了します。引数としてセルラー制御構造体を取ります。
.simPinSet	g_sf_cellular0.p_api->simPinSet (g_sf_cellular0.p_ctrl, p_pin); この API を使用すると、アプリケーションやユーザーがセルラーネットワークに登録する必要があるピンを変更できます。これは、セルラー制御構造体、古いピン、新しいピンを引数として取ります。

Function Name	API の呼び出し例と説明
.simLock	g_sf_cellular0.p_api->simLock (g_sf_cellular0.p_ctrl, p_pin); この API は SIM をロックします。引数としてセルラー制御構造体と SIM ピンを取ります。
.simUnlock	g_sf_cellular0.p_api->simUnlock (g_sf_cellular0.p_ctrl, p_pin); この API は SIM のロックを解除します。引数としてセルラー制御構造体と SIM ピンを取ります。
.simIDGet	g_sf_cellular0.p_api->simIDGet (g_sf_cellular0.p_ctrl, p_sim_id); この API はセルラーモジュールの SIM ID を読み取ります。セルラー制御構造体を引数として取り、セルラーモジュールから読み取った SIM ID を返します。
.networkStatusGet	g_sf_cellular0.p_api->networkStatusGet (g_sf_cellular0.p_ctrl, p_status); このAPIはセルラーモジュールのネットワークステータス情報を取得します。これは、セルラー制御構造体を引数として取り、以下のパラメータを返します。 - 国コード （整数） - オペレータコード （整数） - RSSI （整数） - セル ID （文字列） - IMSI （文字列） - オペレータ名 （文字列） - サービスドメイン （整数） - アクティブバンド （整数）
.fotaCheck	g_sf_cellular0.p_api->fotaCheck (g_sf_cellular0.p_ctrl, p_fota_check); このAPIは使用可能なファームウェアアップグレードをチェックします。これは、セルラー制御構造体と fota チェック用構造体を引数として取ります。
.fotaStart	g_sf_cellular0.p_api->fotaStart (g_sf_cellular0.p_ctrl, p_fota_start); このAPIはファームウェアアップグレードを開始します。これは、セルラー制御構造体と fota 開始用構造体を引数として取ります。
.fotaStop	g_sf_cellular0.p_api->fotaStop (g_sf_cellular0.p_ctrl, p_fota_stop); このAPIはファームウェアアップグレードを停止します。これは、セルラー制御構造体と fota 停止用構造体を引数として取ります。

Function Name	API の呼び出し例と説明
.versionGet	g_sf_cellular0.p_api->versionGet (p_version); この API は、バージョンポインタを使用して API バージョンを取得します。
.reset	g_sf_cellular0.p_api->reset (g_sf_cellular0.p_ctrl, reset_type); セルラーハードウェアモジュールをリセットします。

セルラーソケットフレームワークモジュールの API 要約

セルラーソケットフレームワークでは、以下の表に示す追加の API が提供されます。

セルラーソケットフレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
.open	g_sf_cellular_socket0.p_api->open (g_sf_cellular_socket0.p_ctrl, g_sf_cellular_socket0.p_cfg); このAPIは、セルラーフレームワークの低レベルOpen() APIを呼び出して、セルラーデバイスドライバーを初期化します。
.close	g_sf_cellular_socket0.p_api->close (g_sf_cellular_socket0.p_ctrl); このAPIは、セルラーフレームワークの低レベルClose() APIを呼び出して、セルラーデバイスドライバーを閉じます。
.versionGet	g_sf_cellular_socket0.p_api->versionGet (p_version); この API は、バージョンポインタを使用して API バージョンを取得します。

セルラーソケットフレームワークモジュールの API 要約

これらの API は、アプリケーションがソケットを使用したデータ転送を実行するために使用できます。これには BSD API に準拠するソケット API が含まれます。

- socket
- close
- connect

- send
- recv
- sendto
- recvfrom
- select

セルラーフレームワークモジュールのエラーコード

以下の表に、セルラーフレームワーク固有のエラーコードを示します。これらのエラーコードは ssp_err_t. に含まれます。

セルラーフレームワークのエラーコード

Error Codes	説明
SSP_SUCCESS	呼び出し成功
SSP_ERR_CELLULAR_CONFIG_FAILED	構成失敗
SSP_ERR_CELLULAR_INIT_FAILED	初期化失敗
SSP_ERR_CELLULAR_TRANSMIT_FAILED	送信失敗
SSP_ERR_CELLULAR_FW_UPTODATE	最新
SSP_ERR_CELLULAR_FW_UPGRADE_FAILED	アップグレード失敗
SSP_ERR_CELLULAR_FAILED	一般的な障害
SSP_ERR_CELLULAR_INVALID_STATE	無効な状態

5.1.7.2 セルラーフレームワークモジュールの動作の概要

セルラーフレームワークモジュールの動作について

セルラーフレームワークによって、アプリケーションを変更せずに、さまざまなベンダのセルラーハードウェアモジュールとシームレスに通信するための汎用インタフェースが提供されます。このフレームワークは、主に、ネットワーキングスタックとさまざまなセルラーハードウェアモジュールのインタフェースになる共通セットの API で構成されます。このセクションでは、セルラーフレームワークの基本ブロックと重要な機能について説明します。これに基づいて、セルラーフレームワークを使用して目的のセルラーアプリケーションを開発できるかどうかを判断できます。

汎用 API と抽象化により、セルラーハードウェアモジュール用に開発されたアプリケーションを、別のセルラーハードウェアモジュールに簡単に移植できます。ネットワーキングスタック NetX も、ネットワークソフトウェア抽象化レイヤー（NSAL）を使用してこのフレームワークに統合されます。

Synergy セルラーフレームワークは以下の論理ブロックで構成されています。

• Synergy セルラーフレームワークアプリケーションインタフェース。

- NetX TCP/IP スタック用のネットワークスタック抽象化レイヤー （NSAL）
- セルラーデバイスドライバー（セルラーチップセットとやりとりするための AT コマンドインタフェース）
- オンチップネットワーキングスタックをサポートするセルラーハードウェアモジュールのインタフェースになる BSD ソケット互換 API
- Synergy ソフトウェアパッケージ （SSP） HAL インタフェース

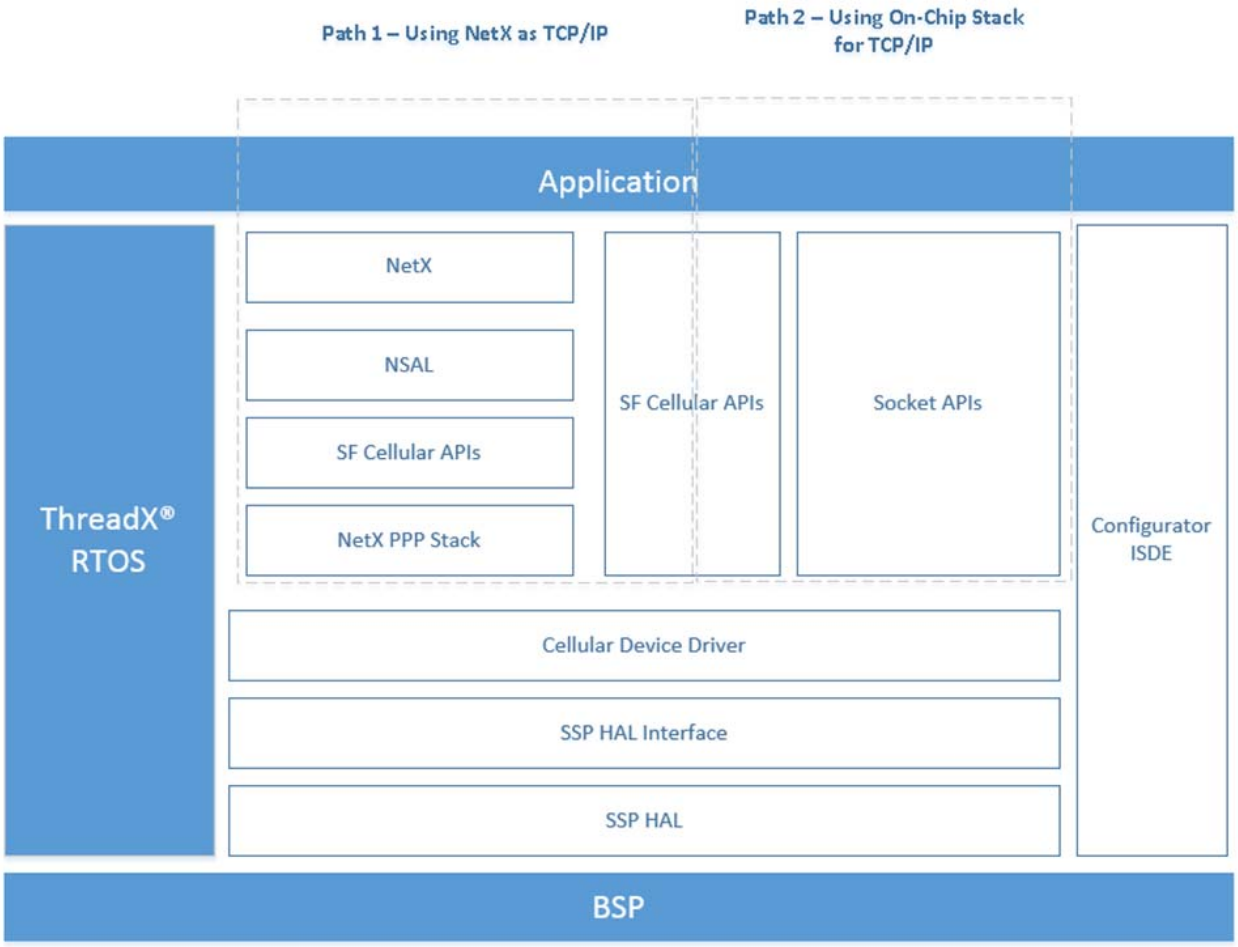


図 146: セルラーフレームワークモジュールのアプリケーションの観点

セルラーフレームワークによってアプリケーションにインタフェースの共通セットが提供され、アプリケーションはセルラーハードウェアモジュールの構成、プロビジョニング、通信を行います。ユーザーは、このような汎用インタフェースを使用し、Synergy MCU を使用してセルラーベースのアプリケーションを開発できます。セルラーハードウェアモジュールには、一連の 3GPP 規格で指定されているさまざまな構成パラメータがあります。個別のデバイスドライバーまたはセルラーチップセットやモジュール （あるいは両方）ですべての構成パラメータがサポートされるとは限りません。ネットワークオペレータ、アクセスポイント名（APN）、セキュリティ資格証明は、このモジュールが機能するために最低限必要です。

ネットワークスタック抽象化レイヤー

セルラーフレームワークは、ネットワークスタック抽象化レイヤー（NSAL）を提供します。NSAL は、WAN リンク上のデータ通信に使用される PPP スタックを使用して NetX とセルラードライバーに接続するレイヤーです。

ソケットインタフェースレイヤー

セルラーフレームワークによってアプリケーションにソケットレベル API が提供され、アプリケーションはセルラーハードウェアモジュールに存在するオンチップネットワーキングスタックとやりとりします。これには、セルラーハードウェアモジュールまたはドライバーが、オンチップネットワーキングスタックとソケットインタフェースをサポートする必要があります。アプリケーションがこれらの API を使用するとき、セルラーハードウェアモジュールに存在するオンチップネットワーキングスタックを使用し、NSAL または NetX とソケット API は使用しません。また、Synergy MCU グループで実行しているネットワーキングスタックも使用しません。

PPP スタック

ポイントツーポイントプロトコル （PPP）はデータ通信で一般的に使用される WAN プロトコルです。 NetX は、SSP の一部として PPP スタックサポートを提供します。NSAL は、PPP スタックを利用して、シリアルインタフェースを介してセルラーサービスプロバイダのネットワークと通信します。PPP は、PAP/CHAP などの認証方法を扱うオプションを提供します。このような認証メカニズムはオプションですが、NSAL はフレームワーク API を活用してセルラーハードウェアモジュールとの間でデータを送受信します。NSAL を使用すると、ネットワークスタックに関して変更せずにセルラーデバイスドライバーを再利用できます。

セルラーデバイスドライバー（セルラーチップセットとやりとりするための AT コマンドインタフェース）

セルラーフレームワークは AT コマンドセットを使用し、シリアルドライバーを使用してセルラーモデムとやりとりします。モデムとのやりとりに使用されるシリアルインタフェースは UART です。フレームワークのデフォルトで使用される UART 速度は、最大 115200 ビット / 秒です。

セルラーフレームワークモジュールの動作の説明

上の図に示すようにユーザーアプリケーションの観点では、セルラーモデムで使用可能なサポートに応じて通信のために 2 つのパスでアプリケーションをこのフレームワークで使用できます。TCP/IP スタックをホストエンドで使用するオプションが提供されるモジュールや、セルラーモデムそのものに存在する TCP/IP スタックを使用するオプションが提供されるモジュールがあります。場合によっては、セルラーハードウェアモジュールによって両方が提供されます。ホスト TCP/IP スタック（NetX）が使用されるとき、NetX、NSAL、PPP の論理レイヤーはアーキテクチャ図のように使用されます。オンチップスタックが使用されると、ソケット API が、セルラーモデムに存在する TCP/IP スタックと通信するために使用されます。ただし、ユーザーは同時に両方を使用できません。

次の制御フロー図に示すように、セルラーモデムの要件に応じてユーザーが提供する構成を使用した初期化の際には、NetX nx_ip_create が呼び出されます。これは、内部で NSAL ドライバーのエントリ関数（リンクレベルの初期化を行いセルラーハードウェアモジュールを初期化する）を呼び出します。また、これは、モジュールをプロビジョニングし、PPP インタフェースを使用してネットワーク接続を確立します。

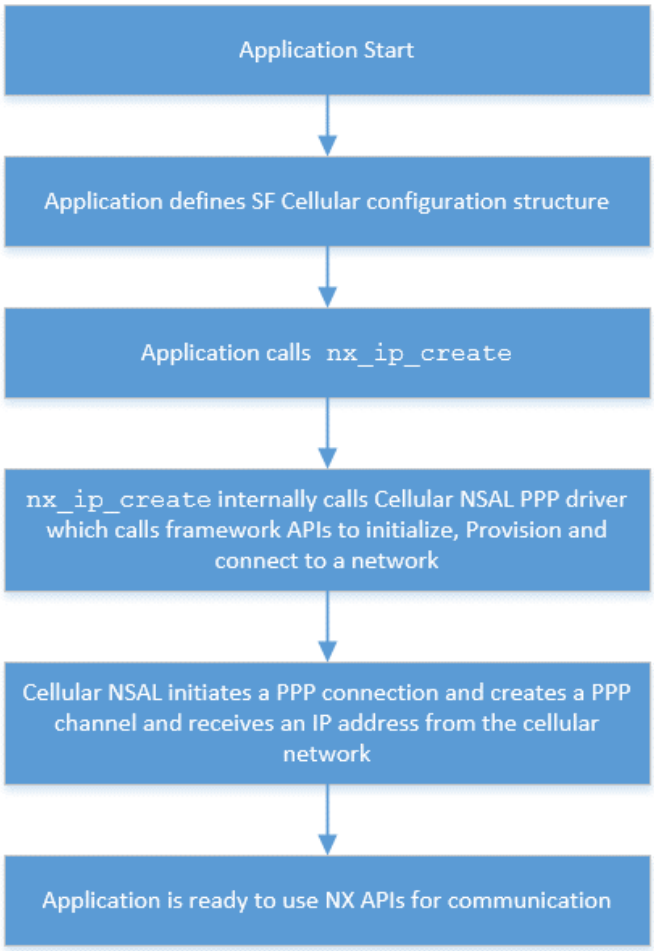


図 147:　セルラーフレームワークモジュールの初期化シーケンス

セルラーハードウェアモジュールのプロビジョニング

セルラーモデムのプロビジョニング中には、制御構造体とユーザー構成構造体が引数として渡されます。プロビジョニングで使用される各ユーザー引数は、認証、APN、ユーザー名、パスワードです。

ソケットインタフェースを使用するアプリケーションフロー制御

次のフロー図には、セルラーソケットインタフェースでのオンチップスタックパスの使用に関するフローが示されます。

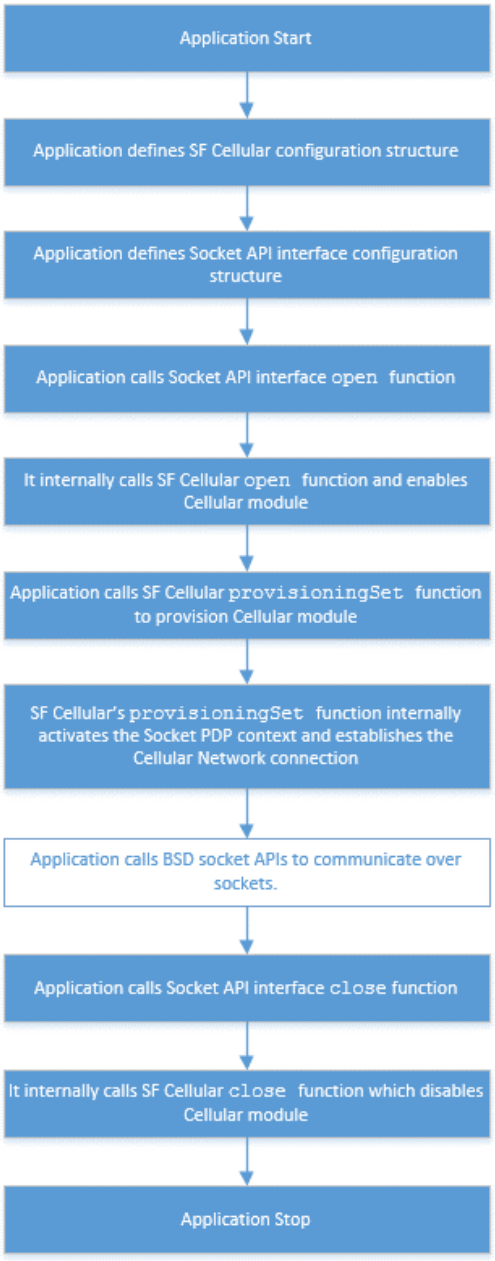


図 148:　セルラーフレームワークモジュールのソケットインタフェース

次のフロー図には、パケット送信が NetX アプリケーションで使用する手順のシーケンスが示されます。

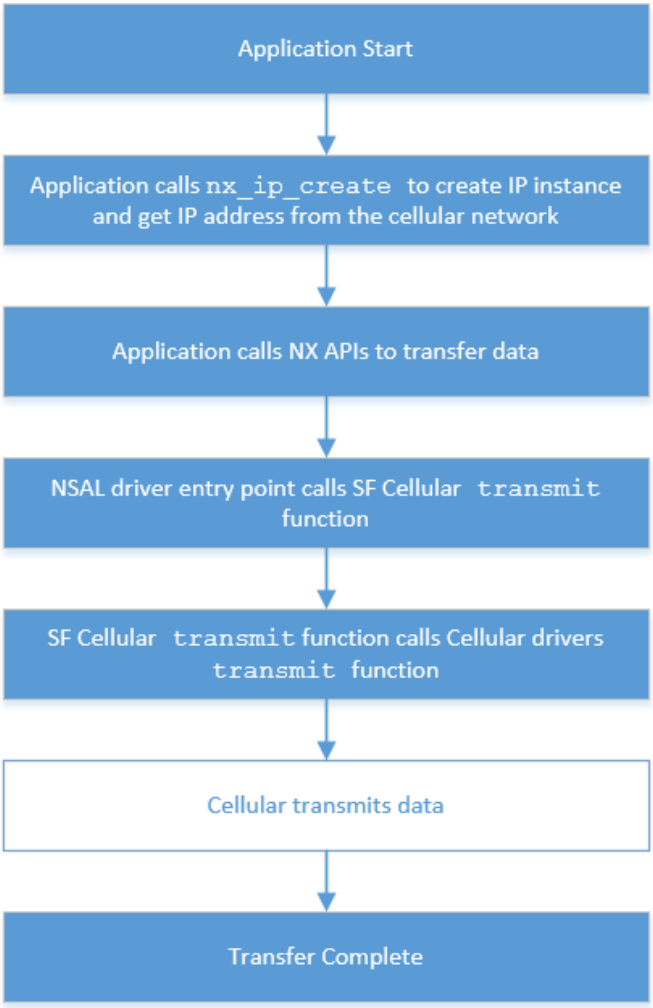


図 149:　セルラーフレームワークのパケット送信シーケンス

次のフロー図には、NetX を使用したセルラーフレームワークのパケット受信が示されます。受信のケースでは、データがシリアルインタフェースで受信されるとき、処理スレッドがコールバック関数をトリガし、コールバック関数がデータを処理し、さらに処理するために NetX レイヤーに送信します。

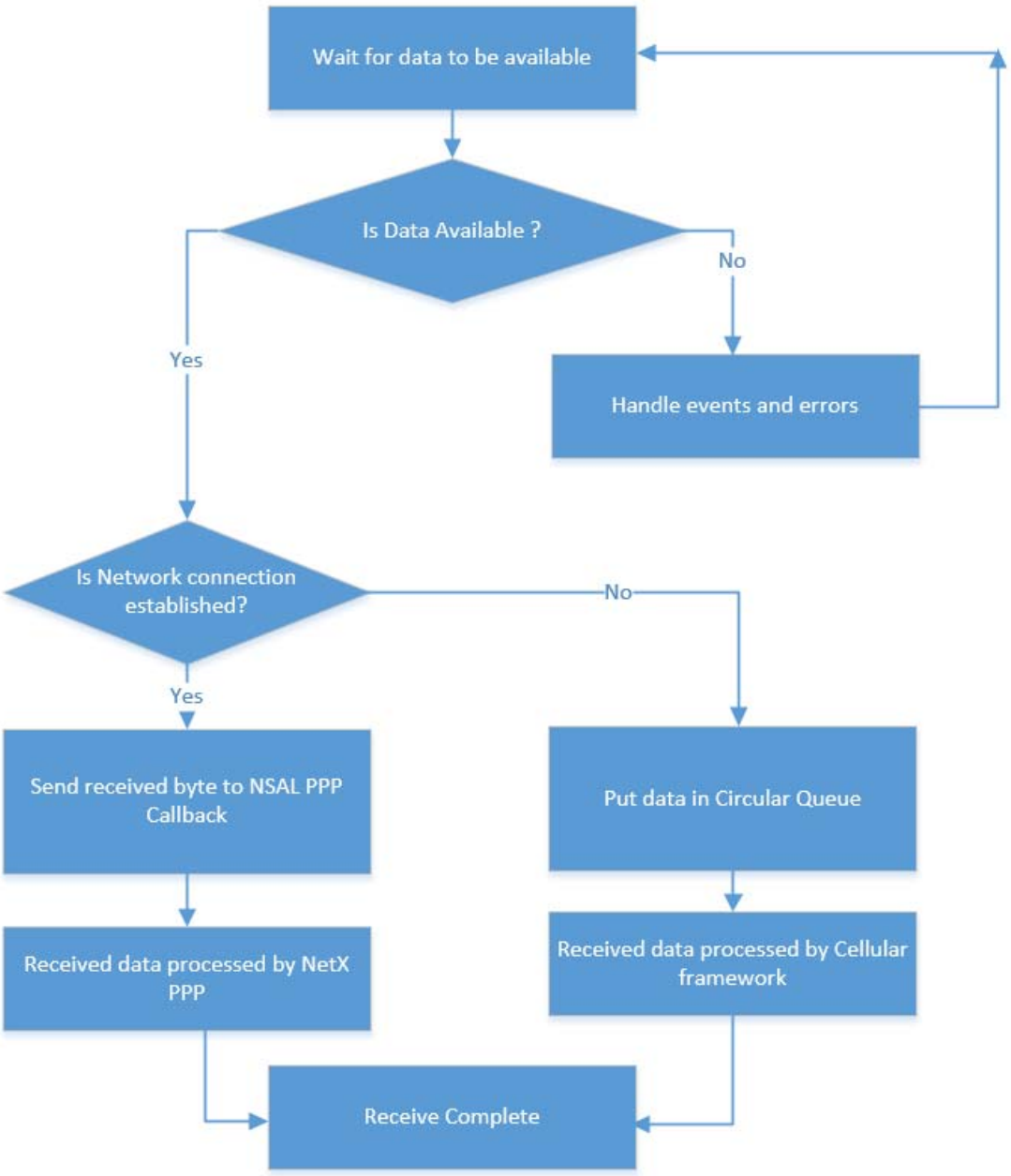


図 150:　セルラーフレームワークのパケット受信シーケンス

セルラーフレームワークモジュールの動作に関する重要な注意事項と制限事項

- 現在のフレームワークでサポートされるのは、NimbeLink CAT3 および CAT1 セルラーハードウェアモジュールのみです。

- 無線によるファームウェアアップグレード（FOTA）は、NimbeLink CAT3 および CAT1 セルラーハードウェアモジュールではサポートされません。
- このモジュールに関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

5.1.7.3 アプリケーションへのセルラーフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにセルラーフレームワークモジュールを組み込む方法について説明します。

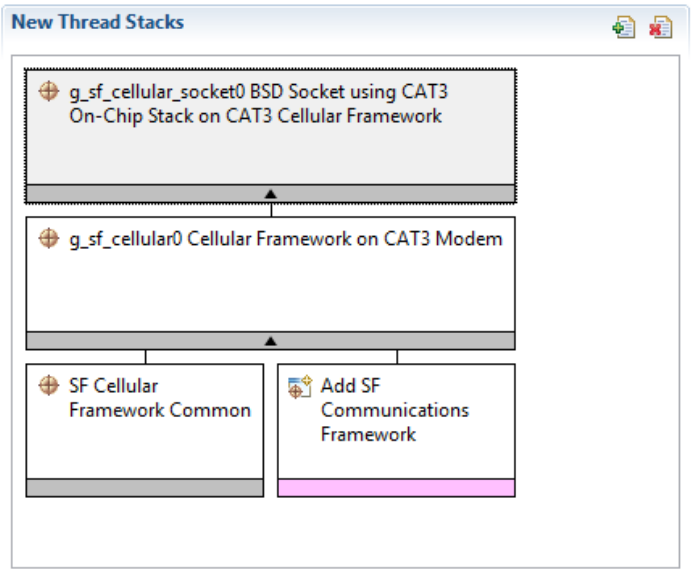
***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの手順に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照してください。*

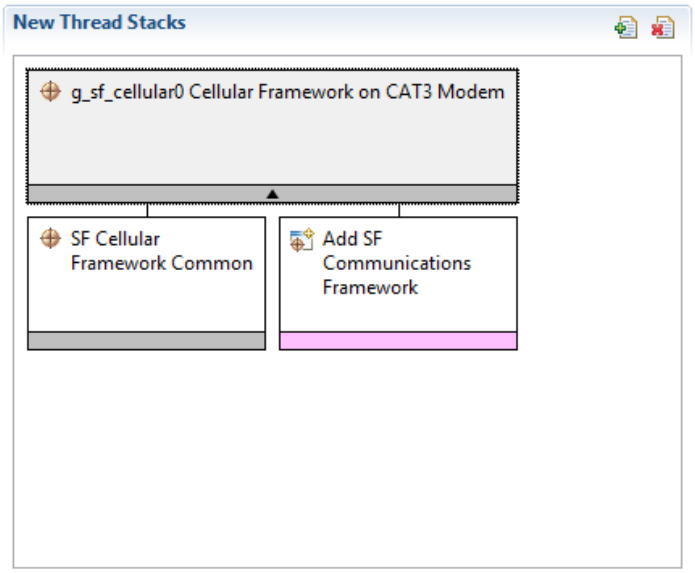
セルラーフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。

セルラーフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_cellular_socket0 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework	Threads	New Stack> Framework> Networking> Cellular > BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework
g_sf_cellular_0 Cellular Framework on CAT3 Modem	Threads	New Stack> Framework> Networking> Cellular > Cellular Framework on CAT3 Modem
g_sf_el_nx0 NetX Port using Cellular Framework on sf_cellular_nsal_nx	Threads	New Stack> Framework> Networking> Cellular > NetX Port using Cellular Framework on sf_cellular_nsal_nx

次の図に示すようにセルラーフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。ピンクの帯で示されるモジュールでは実装オプションのの選択が必要です。


New Thread Stacks
g_sf_cellular_socket0 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework
g_sf_cellular0 Cellular Framework on CAT3 Modem
SF Cellular Framework Common
Add SF Communications Framework


New Thread Stacks
g_sf_cellular0 Cellular Framework on CAT3 Modem
SF Cellular Framework Common
Add SF Communications Framework

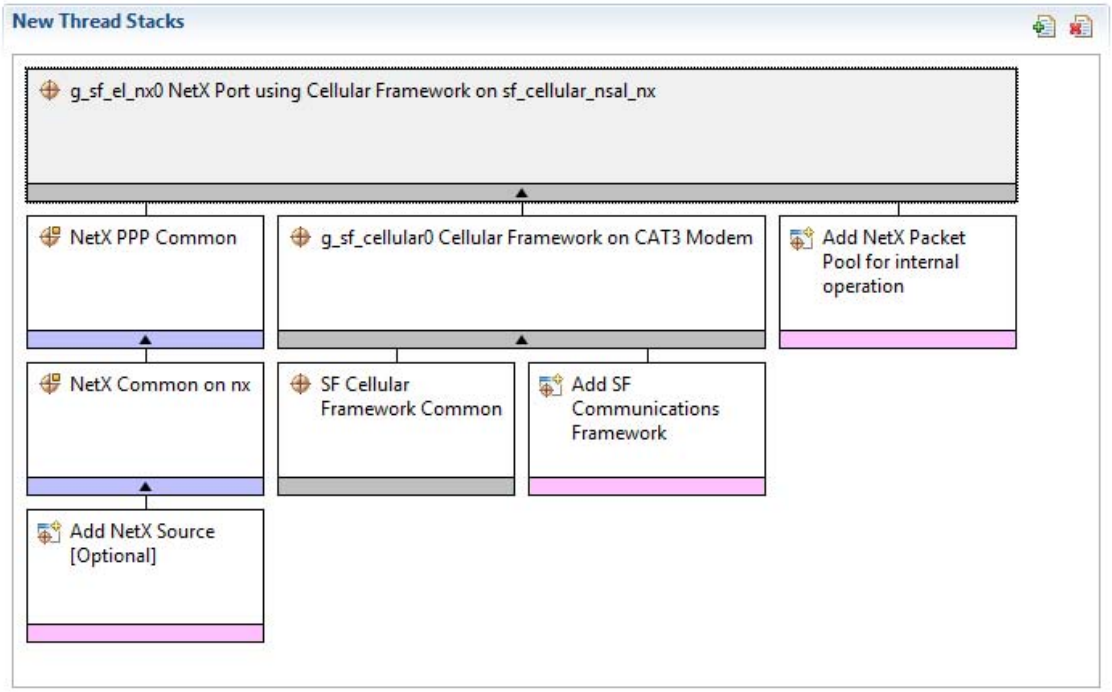


図 151:　セルラーフレームワークモジュールのスタックオプション

SF_CELLULAR_NSAL_NX の選択

sf_cellular_nsal_nx の実装は、NetX IP のインスタンスと組み合わせて使用できます。このケースでは、これは IP インスタンススタックの下にオプションとして追加されます。次の表の選択手順を使用して、 NetX IP のインスタンスを追加します。

Resource	ISDE Tab	Stacks Selection Sequence
g_ip0 NetX IP Instance	Threads	New Stack> X-Ware> NetX> NetX IP Instance

使用可能なオプションを使用して、sf_cellular_nsal_nx を NetX IP インスタンスに NetX ネットワークドライバーとして追加します。結果のスタックを次の図に示します。

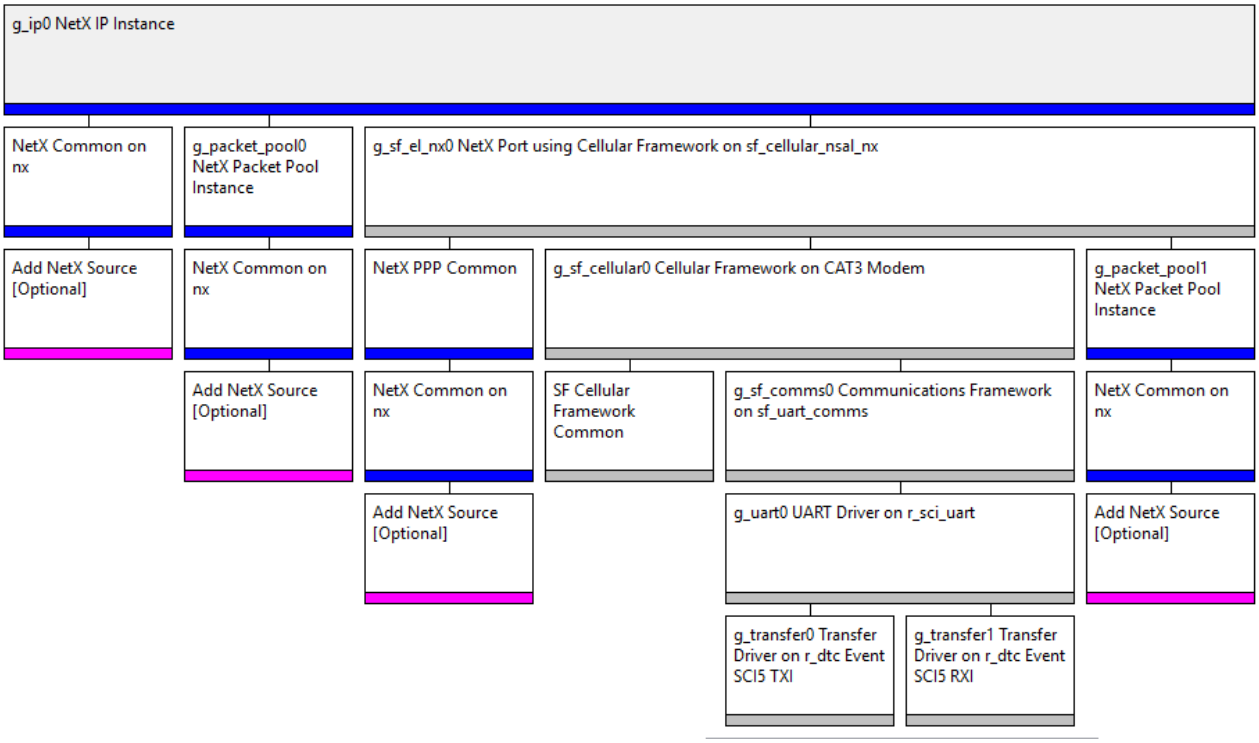


図 152: sf_cellular_nsal_nx の NetX IP インスタンスの使用

5.1.7.4 セルラーフレームワークモジュールの構成

ユーザーは必要な動作に合わせてセルラーフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: ISDE を開き、セルラーフレームワークを作成し、次に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

セルラーフレームワークモジュールの各実装については後で別のセクションで説明します。

CAT3 モデム上のセルラーフレームワークモジュールの構成

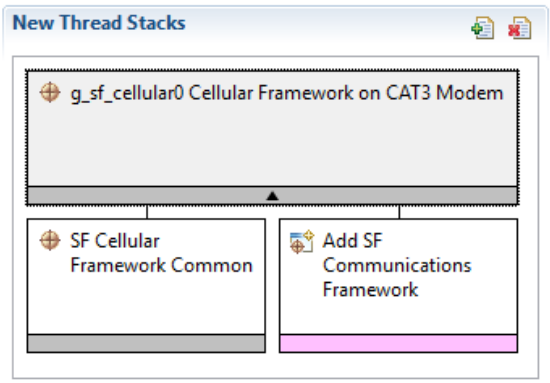


図 153: CAT3 モデム上のセルラーフレームワークのスレッドスタック

sf_cellular 上の CAT3 モデム上のセルラーフレームワークの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled Default: Disabled	オンチップスタックサポートの選択
Modem	TEUG, TSVG Default: TEUG	モデムの選択
Name	g_sf_cellular0	モジュール名
SIM Pin (Used to Unlock SIM)	1111	SIM ピンの選択
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK ピンの選択

ISDE Property	Setting	説明
Number of Preferred Operator	0	優先オペレータ番号の選択
Preferred Operator 1 Name	40422	優先オペレータ 1 の名前の選択
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 1 の名前フォーマットの選択
Preferred Operator 2 Name	40424	優先オペレータ 2 の名前の選択
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 2 の名前フォーマットの選択
Preferred Operator 3 Name	40422	優先オペレータ 3 の名前の選択
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 3 の名前フォーマットの選択
Preferred Operator 4 Name	40424	優先オペレータ 4 の名前の選択
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 4 の名前フォーマットの選択
Preferred Operator 5 Name	40422	優先オペレータ 5 の名前の選択
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 5 の名前フォーマットの選択
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	オペレータ選択モードの選択
Operator Name (Manual Mode Selection)	40422	オペレータの名前の選択

ISDE Property	Setting	説明
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	オペレータ名フォーマットの選択
Time Zone Update Policy	Enabled, Disabled Default: Enabled	タイムゾーン更新ポリシーの選択
Receive Data Callback	sf_cellular_nsal_recv_callback	受信データコールバックの選択
Provisioning Callback	celr_prov_callback	プロビジョニングコールバックの選択
Circular Queue Size in Bytes	256	循環キューサイズの選択
SF Communications Framework Thread Stack Size	512	SF通信フレームワークスレッドスタックサイズの選択
Numerical priority of SF Communication Framework Thread.Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	5	SF通信フレームワークスレッドの数値プライオリティの選択
Cellular Module Reset IO Pin	IOPORT_PORT_10_PIN_05	セルラーモジュールリセットIOピンの選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SF セルラーフレームワーク共通の構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

CAT3 セルラーフレームワークでの CAT3 オンチップスタックを使用した BSD ソケットの構成

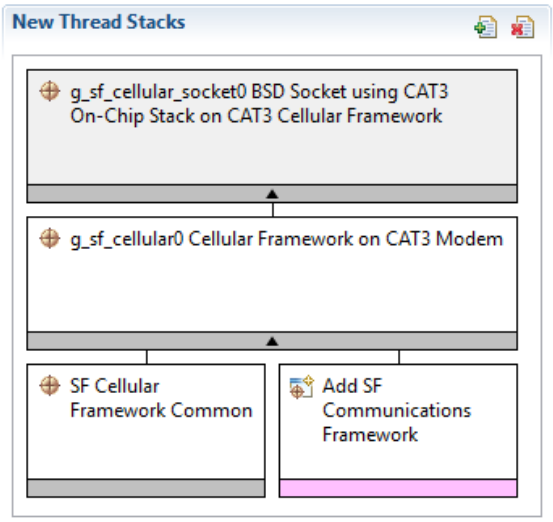


図 154: CAT3 セルラーフレームワークでの CAT3 オンチップスタックを使用した BSD ソケットのスレッドスタック

CAT3 セルラーフレームワークでの CAT3 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
Name	g_sf_cellular	モジュール名

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

CAT3 モデム上のセルラーフレームワークの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。

ISDE Property	Setting	説明
On-Chip Stack Support	Enabled, Disabled Default: Disabled	オンチップスタックサポートの選択
Modem	TEUG, TSVG Default: TEUG	モデムの選択
Name	g_sf_cellular0	モジュール名
SIM Pin (Used to Unlock SIM)	1111	SIM ピンの選択
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK ピンの選択
Number of Preferred Operator	0	優先オペレータ番号の選択
Preferred Operator 1 Name	40422	優先オペレータ 1 の名前の選択
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 1 の名前フォーマットの選択
Preferred Operator 2 Name	40424	優先オペレータ 2 の名前の選択
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 2 の名前フォーマットの選択
Preferred Operator 3 Name	40422	優先オペレータ 3 の名前の選択
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 3 の名前フォーマットの選択
Preferred Operator 4 Name	40424	優先オペレータ 4 の名前の選択
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 4 の名前フォーマットの選択
Preferred Operator 5 Name	40422	優先オペレータ 5 の名前の選択

ISDE Property	Setting	説明
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 5 の名前フォーマットの選択
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	オペレータ選択モードの選択
Operator Name (Manual Mode Selection)	40422	オペレータの名前の選択
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	オペレータ名フォーマットの選択
Time Zone Update Policy	Enabled, Disabled Default: Enabled	タイムゾーン更新ポリシーの選択
Receive Data Callback	sf_cellular_nsal_recv_callback	受信データコールバックの選択
Provisioning Callback	celr_prov_callback	プロビジョニングコールバックの選択
Circular Queue Size in Bytes	256	循環キューサイズの選択
SF Communications Framework Thread Stack Size	512	SF通信フレームワークスレッドスタックサイズの選択
Numerical priority of SF Communication Framework Thread.Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the hightest priority.	5	SF通信フレームワークスレッドの数値プライオリティの選択
Cellular Module Reset IO Pin	IOPORT_PORT_10_PIN_05	セルラーモジュールリセット IO ピンの選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SF セルラーフレームワーク共通の構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

セルラーフレームワークを使用した NetX ポートの構成

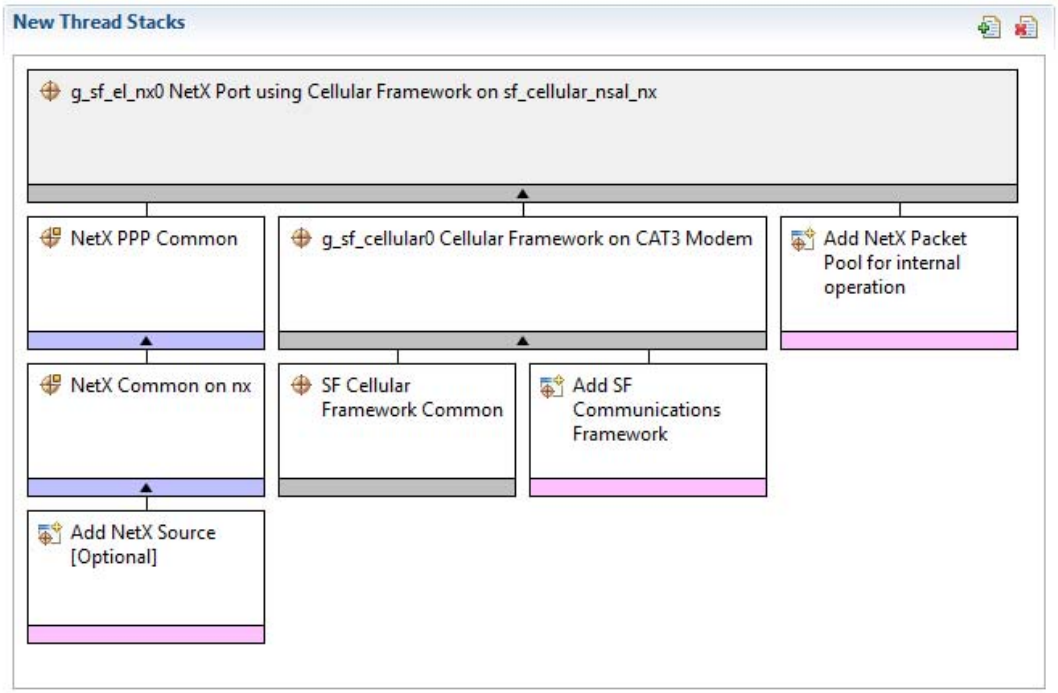


図 155: セルラーフレームワークを使用した NetX ポートのスレッドスタック

sf_cellular_nsal_nx 上のセルラーフレームワークの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。

ISDE Property	Setting	説明
Name	g_sf_el_nx0	モジュール名
PPP Stack Size in Bytes	2048	PPP スタックサイズの選択
Name	g_nx_ppp0	モジュール名
Numerical priority of PPP Thread (Priority must be lower than IP Helper thread).Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	3	PPP スレッドの数値プライオリティの選択
Authentication Method	None, PAP, CHAP Default: None	認証方法の選択
Invalid Packet Handler Callback	NULL	無効パケットハンドラコールバックの選択
Link Down Callback	ppp_link_down_callback	リンクダウンコールバックの選択
Link Up Callback	ppp_link_up_callback	リンクアップコールバックの選択
PAP Login Callback	NULL	PAP ログインコールバックの選択
PAP Verify Login Callback	NULL	PAP 確認ログインコールバックの選択
Get Challenges Values Callback	NULL	チャレンジ値取得コールバックの選択
Get Responder Values Callback	NULL	レスポンダ値取得コールバックの選択
Get Verification Callback	NULL	確認取得コールバックの選択
Local IPv4 Address (use commas for separation)	0,0,0,0	ローカル IPv4 アドレスの選択
Peer IPv4 Address (use commas for separation)	0,0,0,0	ピア IPv4 アドレスの選択

***注：** 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

NetX PPP 共通の構成設定

ISDE Property	Setting	説明
Name	g_nx_ppp_common0	モジュール名

CAT3 モデム上のセルラーフレームワークの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled Default: Disabled	オンチップスタックサポートの選択
Modem	TEUG, TSVG Default: TEUG	モデムの選択
Name	g_sf_cellular0	モジュール名
SIM Pin (Used to Unlock SIM)	1111	SIM ピンの選択
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK ピンの選択
Number of Preferred Operator	0	優先オペレータ番号の選択
Preferred Operator 1 Name	40422	優先オペレータ 1 の名前の選択
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 1 の名前フォーマットの選択
Preferred Operator 2 Name	40424	優先オペレータ 2 の名前の選択
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 2 の名前フォーマットの選択

ISDE Property	Setting	説明
Preferred Operator 3 Name	40422	優先オペレータ 3 の名前の選択
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 3 の名前フォーマットの選択
Preferred Operator 4 Name	40424	優先オペレータ 4 の名前の選択
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 4 の名前フォーマットの選択
Preferred Operator 5 Name	40422	優先オペレータ 5 の名前の選択
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	優先オペレータ 5 の名前フォーマットの選択
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	オペレータ選択モードの選択
Operator Name (Manual Mode Selection)	40422	オペレータの名前の選択
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	オペレータ名フォーマットの選択
Time Zone Update Policy	Enabled, Disabled Default: Enabled	タイムゾーン更新ポリシーの選択
Receive Data Callback	sf_cellular_nsal_recv_callback	受信データコールバックの選択
Provisioning Callback	celr_prov_callback	プロビジョニングコールバックの選択
Circular Queue Size in Bytes	256	循環キューサイズの選択

ISDE Property	Setting	説明
SF Communications Framework Thread Stack Size	512	SF通信フレームワークスレッドスタックサイズの選択
Numerical priority of SF Communication Framework Thread.Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	5	SF通信フレームワークスレッドの数値プライオリティの選択
Cellular Module Reset IO Pin	IOPORT_PORT_10_PIN_05	セルラーモジュールリセットIOピンの選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SF セルラーフレームワーク共通の構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

NetX パケットプールインスタンスの構成設定

ISDE Property	Setting	説明
Name	g_packet_pool0	モジュール名
Packet Size (bytes)	128	パケットサイズの選択
Number of Packets in Pool	16	プール内のパケット数の選択
Name of generated initialization function	packet_pool_init0	生成される初期化関数の名前の選択

ISDE Property	Setting	説明
Auto initialization	Enable, Disable Default: Enable	自動初期化の選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

NetX 共通の構成設定

ISDE Property	Setting	説明
Name of generated initialization function	nx_common_init0	生成される初期化関数の名前の選択
Auto initialization	Enable, Disable Default: Enable	自動初期化の選択

セルラーフレームワークモジュールのクロック構成

セルラーフレームワークモジュールは、選択された特定の低レベルモジュール（UART または USB など）に必要なクロックを使用します。

セルラーフレームワークモジュールのピン構成

セルラーフレームワークモジュールは、選択された低レベルモジュール（UART または USB など）に依存する入力ピンと出力ピンを使用します。

5.1.7.5 アプリケーションでのセルラーフレームワークモジュールの使用

セルラーフレームワークは、実装ごとにターゲットアプリケーションによる扱いが異なります。初期化、パケット送信、パケット受信の使用に関する一般的な制御フローは、多くのアプリケーションにおいて重要です。これらの機能の一般的な実装のフロー例を次の図に示します。

詳しい説明とアプリケーションプロジェクトが、Renesas Web サイトの「Cellular Application Note」にあります。上部の検索バーで「R11AN0082」を検索すると、アプリケーションの注意事項とアプリケーションプロジェクトが検索結果に表示されます。

セルラーフレームワークのパス 1

1) セルラーモジュールの初期化

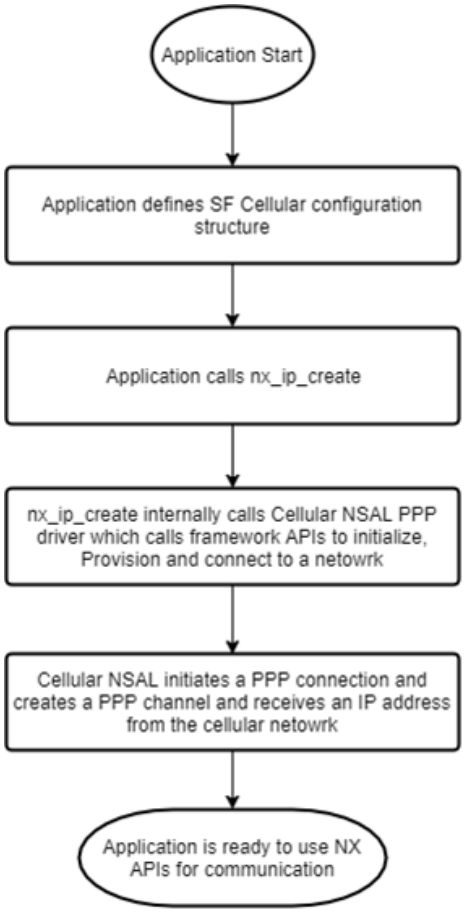


図 156: セルラーモジュールの初期化を使用したアプリケーションの制御フロー

2) パケット送信

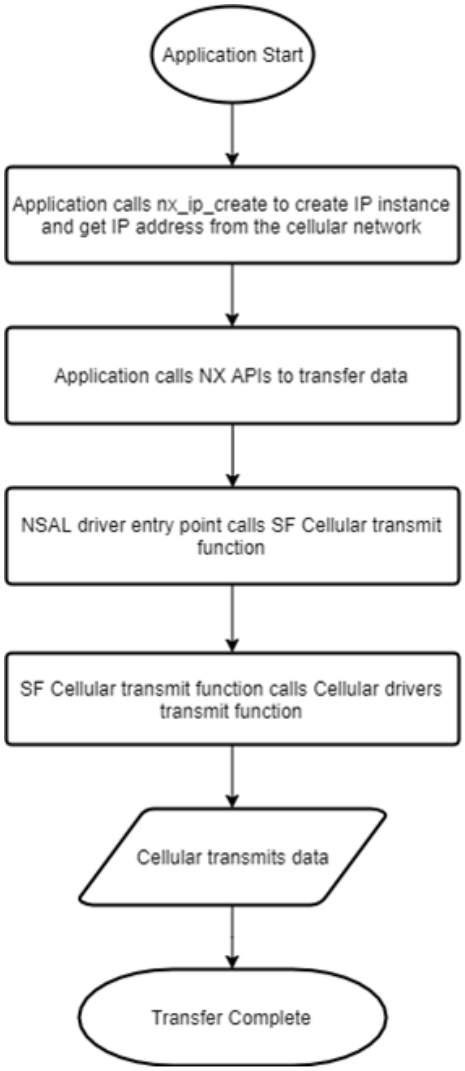


図 157: NetX を使用してパケット送信を行うアプリケーションの制御フロー

3) パケット受信

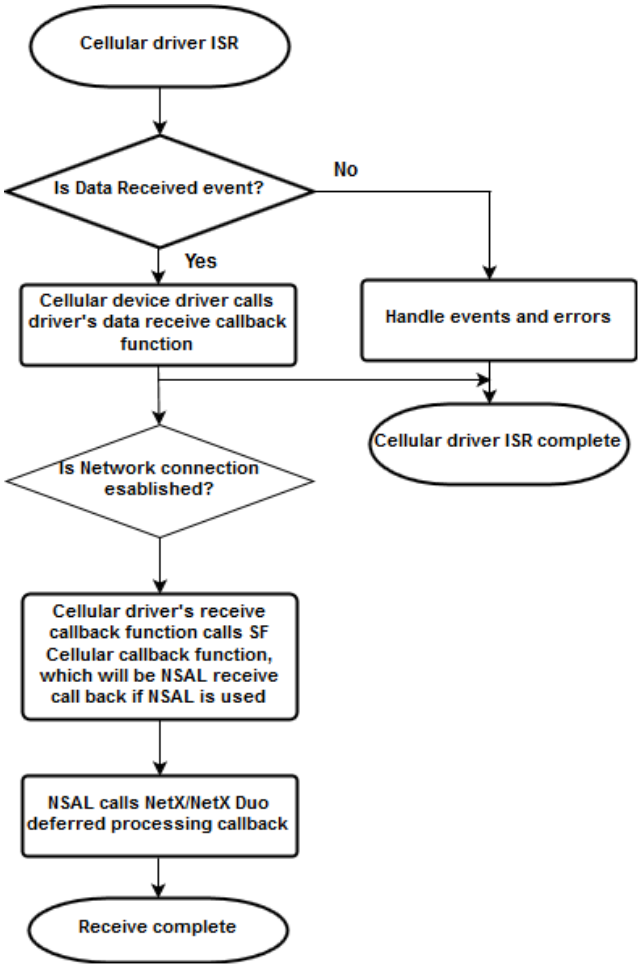


図 158: NetX を使用してパケットを受信するアプリケーションの制御フロー

セルラーフレームワークモジュールのパス 2

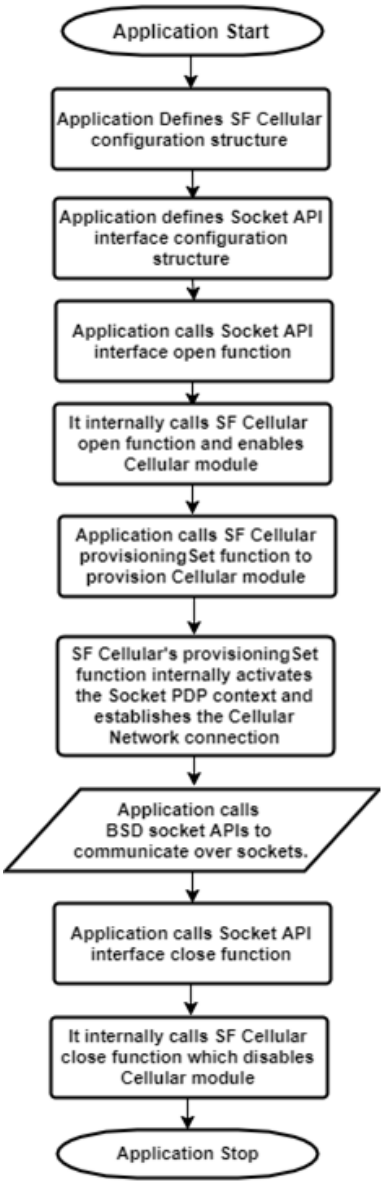


図 159: BSD ソケットインタフェースを使用するアプリケーションのフロー制御

5.1.8 UART 通信フレームワーク

このモジュールは、ThreadX 対応の通信フレームワークです。ThreadX オブジェクトを使用して動作がスレッドセーフになるようにします。主な特長は次のとおりです。

- UART 通信プロトコルのサポート
- 専用アクセスを確保するためのチャネルロックのサポート

- ThreadX 対応実装

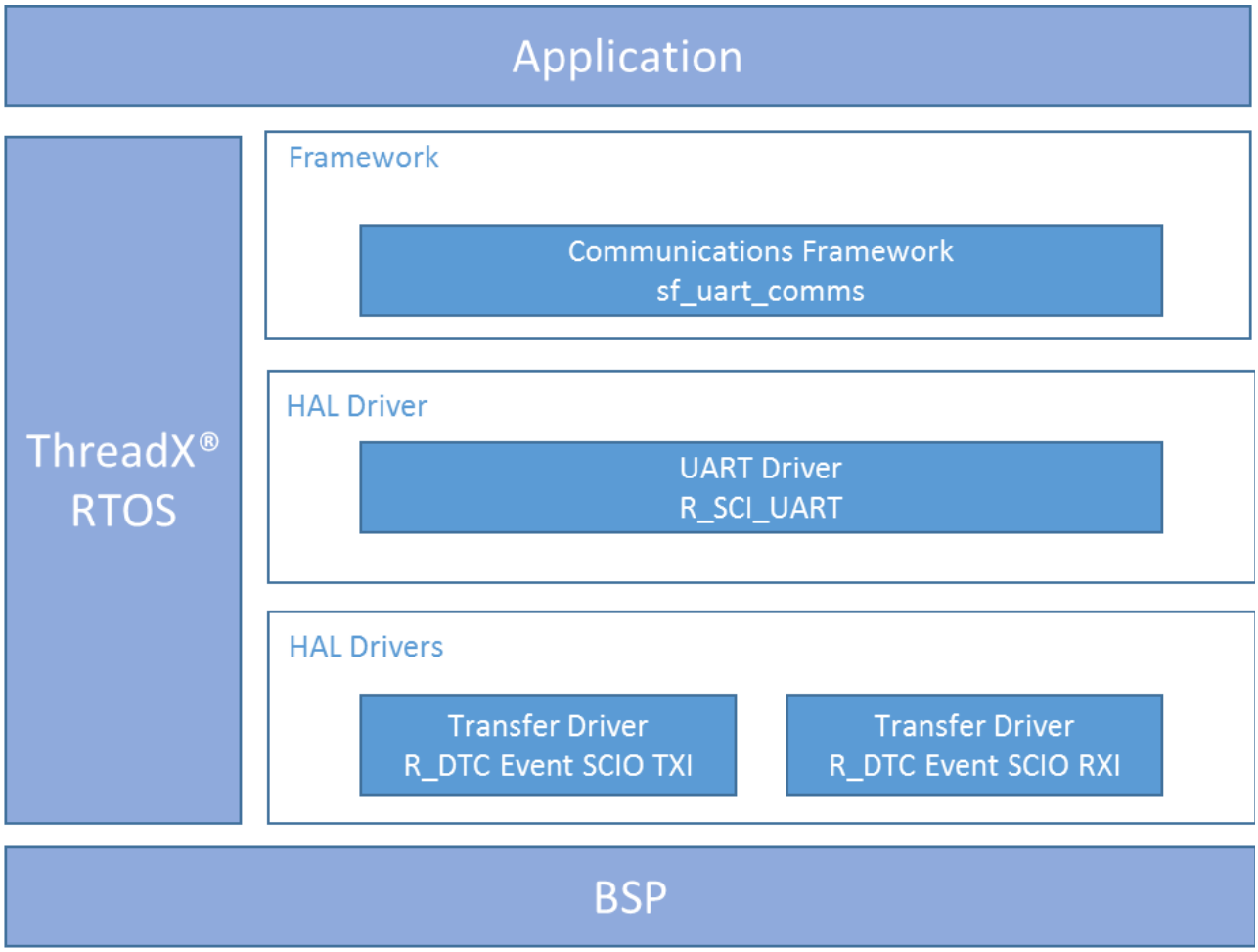


図 160: UART 通信フレームワークの編成、オプション、スタックの実装

5.1.8.1 UART 通信フレームワークモジュールの API の概要

UART 通信フレームワークモジュールは、通信チャネルのオープン、クローズ、リード、ライトの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

UART 通信フレームワークの API 要約

Function Name	API の呼び出し例と説明
open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); 通信ドライバーを初期化します。
close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); 通信ドライバーをクリーンアップします。
read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); 通信ドライバーからデータを読み取ります。この呼び出しは、 要求された数のバイトを読み取った後、またはドライバへのアクセスを 待っている間にタイムアウトが発生した場合に復帰します。
write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); 通信ドライバーにデータを書き込みます。この呼び出しは、 要求された数のバイトを読み取った後、またはドライバへのアクセスを 待っている間にタイムアウトが発生した場合に復帰します。

Function Name	API の呼び出し例と説明
lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); 通信ドライバーをロックします。排他的アクセスの確保 （通信ドライバーへの）を行います。
unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); 通信ドライバーをロック解除します。排他的アクセスの解放 （通信ドライバーへの）を行います。
versionGet	g_sf_comms0.p_api->version(&version); ドライバーバージョンを指定された version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、SSP ユーザーズマニュアルで関連モジュールの API リファレンスを参照してください。*

エラーステータス戻り値

Name	説明
SSP_SUCCESS	チャネルが正常に開かれました。
SSP_ERR_IN_USE	チャネルは既に使用中です。
SSP_ERR_ASSERTION	UART 制御ブロックまたは設定構造体へのポインタが NULL です。
SSP_ERR_HW_LOCKED	チャネルがロックされています。

Name	説明
SSP_ERR_INVALID_MODE	チャネルが非 UART モード用に使用されているか、設定されているモードが誤っています。
SSP_ERR_INVALID_ARGUMENT	設定構造体に無効なパラメータ設定値が見つかりました。
SSP_ERR_QUEUE_UNAVAILABLE	送信キューと受信キューのいずれかまたは両方が開かれていません。
SSP_ERR_INTERNAL	この関数はあらゆるチャネルで再入可能です。
SSP_ERR_TIMEOUT	タイムアウトエラー。
SSP_ERR_INSUFFICIENT_DATA	受信循環バッファに十分なデータがありません。
SSP_ERR_RXBUF_OVERFLOW	キューオーバーフローを受け取ります。
SSP_ERR_OVERFLOW	ハードウェアオーバフロー。
SSP_ERR_FRAMING	フレーミングエラー。
SSP_ERR_PARITY	パリティエラー。
SSP_ERR_INSUFFICIENT_SPACE	伝送循環バッファに十分なスペースがありません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.8.2 UART 通信フレームワークモジュールの動作の概要

UART フレームワークによって、標準 UART プロトコルを使用した、簡単に使用できる通信フレームワークが提供されます。 UART デバイスのデータをスレッドセーフ方式で読み取りおよび書き込みするための高レベル API に加え、このフレームワークでは、アプリケーションが UART チャネルをスレッドに対してロック（およびロック解除）するための API も提供されます。これが特に役立つのは、複数のアプリケーションスレッドが同じ UART デバイスとの通信を試行するときに、コンテキストスイッチのために、UART に実装されている高レベルのアプリケーションプロトコルまたは状態マシン （あるいは両方）（たとえば Kermit など）で異常が発生する可能性がある場合です。

UART 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

UART フレームワークモジュールはすべてのチャネルで再入可能です。

UART フレームワークは、UART デバイスと通信するために r_sci_uart モジュール上の UART ドライバーを使用します。 UART ドライバーに DTC ドライバーを追加して （ISDE の Synergy コンフィギュレータを使用）拡張すると、CPU の割り込みを行わずに、UART デバイスに対するリードおよびライトトランザクションを実行できます。UART フレームワークを使用して UART デバイスからデータを読み取るとき、UART ドライバーのコールバック機能が利用され、**DTC は使用されません** （DTC モジュールのサポートがコンフィギュレータで UART ドライバーに追加されている場合でも）。これは、ドライバーが DTC を使用してデバイスの

データを読み取る際に発生する可能性があるタイミングや同期の問題を回避するためです。UART フレームワークを使用して UART デバイスにデータを書き込むときは、トランザクションを実行するために DTC が使用されます（ドライバーが DTC を使用するように構成されている場合）。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.8.3 アプリケーションへの UART 通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに UART 通信フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

UART 通信フレームワークモジュールは sf_uart_comms. に実装されます。UART 通信フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（通信フレームワークのデフォルト名は g_sf_comms0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

通信フレームワークの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_uart_comms	Threads	[Framework] > [Connectivity] > [Communications Framework on sf_uart_comms]

次の図に示すように、通信フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

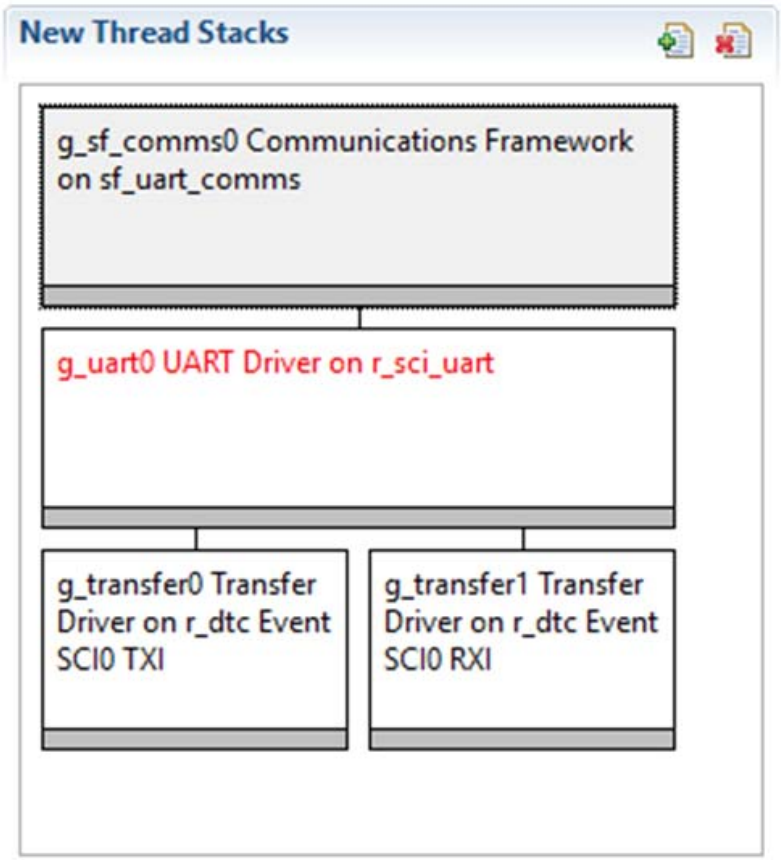


図 161: UART 通信フレームワークのスタック実装オプション

5.1.8.4 UART 通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、UART 通信フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。こ

のレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

UART 通信フレームワークモジュールのスタックの構成設定

実装に対して UART 通信フレームワークのスタックオプションが選択されている場合は、次のモジュールがスタックに自動的に追加されます。次の表に使用可能な構成設定の詳細を示します。これらの多くは、ほとんどのアプリケーションで使用できるデフォルト設定が移入されています。

UART 通信フレームワーク (sf_uart_comms) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを含めるかどうかを選択します。
Read Input Queue Size (4-Byte Words)	15	データ受信キューのバッファサイズ。sf_uart_comms はキュー管理に ThreadX キューを利用します。
Name	g_sf_comms0	UART 通信フレームワークモジュールの名前。

(r_sci_uart) 上の UART ドライバーの構成設定

ISDE Property	Value	説明
External RTS Operation	Enable, Disable Default: Disable	RTS 信号として使用する IOPORT ピンを有効にします。RTS 機能では、この設定パラメータを「有効」にし、設定「RTS 外部ピン制御のための UART コールバック関数名」を指定します。

ISDE Property	Value	説明
Reception	Enable, Disable Default: Enable	SCI 上のすべての UART チャネルについて UART の受信を有効または無効にします。この設定パラメータに「無効」を設定すると、コードサイズが小さくなります。UART 受信のためのコード部分がコンパイルされないためです。このパラメータは、個々の UARTT チャネルに対しては設定できません。
Transmission	Enable, Disable Default: Enable	SCI 上のすべてのチャネルについて UART 送信を有効または無効にします。この設定に「無効」を設定すると、UART 送信のためのコード部分がコンパイルから除外されるため、コードサイズが小さくなりますが、この設定に「無効」を設定できるのは、UART ポートとして機能する他の SCI チャネルが送信を行わない場合だけです。
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_uart0	SCI 上の UART モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Channel	0-9	SCI チャネル番号。
Baud Rate	9600	ボーレートの選択。
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART データビット。
Parity	None, Odd, Even Default: None	UART パリティビット。

ISDE Property	Value	説明
Stop Bits	1 bit, 2 bits Default: 1 bit	UART ストップビット。
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	SCI チャネル n の CTSn/RTSn ピンに対して CTS または RTS を選択します。SCI ハードウェアは、このピン上で CTS と RTS のいずれかの制御信号をサポートしますが、両方はサポートしません。CTS と RTS の両方を使用するアプリケーションでは、この設定パラメータに「CTS」を選択し、設定「外部 RTS 操作」を有効にし、設定「RTS 外部ピン制御のための UART コールバック関数名」を指定します。
Name of UART callback function to be defined by user	user_uart_callback	名前は有効な C シンボルである必要があります。
Name of UART callback function for the RTS external pin control to be defined by user	NULL	名前は有効な C シンボルである必要があります。
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	ボーレートクロック発信器ブロックで使用するクロックソースを選択します。
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	選択したチャネル n に対し SCKn ピン上でボーレートクロックを出力するためのオプション設定。
Start bit detection	Falling Edge, Low Level Default: Falling Edge	受信におけるスタートビット検出モード。この設定には、通常は「立ち下がりエッジ」を設定します。

ISDE Property	Value	説明
Noise Cancel	Enable, Disable Default: Disable	RXDn ピン上でデジタルノイズキャンセルを有効にします。SCI のデジタルノイズフィルターブロックは、2ステージのフリップフロップ回路で構成されます。詳細については、Renesas Synergy ハードウェアマニュアルのノイズキャンセルのセクションを参照してください。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調有効化の選択。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信割り込み優先順位の選択。

ISDE Property	Value	説明
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信終了割り込み優先順位の選択。
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	エラー割り込み優先順位の選択。

r_dtc 上の TXI 転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択

ISDE Property	Value	説明
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

r_dtc 上の RXI 転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Setting	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

UART 通信フレームワークモジュールのクロック構成

UART 通信フレームワークには、固有のクロック構成の要件はありません。

UART 通信フレームワークモジュールのピン構成

UART 通信フレームワークは、MCU のピンを使用し、選択した低レベル実装に基づいて外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は低レベル実装のピンの選択例を示しています。

注: *動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

UART、 USB、または Telnet （受信部）の通信フレームワークのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select **Peripherals > Connectivity: SCI > SCI8**

注: *上記の選択シーケンスは選択した実装に対応する例です。ターゲットハードウェアによっては他の選択シーケンスも可能です。*

UART のピン構成設定

Pin Configuration Property	設定値	説明
Pin Group Selection	Mixed, _A Only, _B Only	ピングループの選択

Pin Configuration Property	設定値	説明
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard (Default: Disabled)	UART 受信部実装では動作モードとして非同期 UART を選択します。
TXD_MOSI	None, PB05, P105 (Default: None)	TXD ピン P105
RXD_MISO	None, PB05, P104 (Default: None)	RXD ピン P104

UART、 USB、または Telnet（トランスミッタ）の通信フレームワークのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select **Peripherals > Connectivity: SCI > SCI3**

注：*上記の選択シーケンスは選択した実装に対応する例です。ターゲットハードウェアによっては他の選択シーケンスも可能です。*

UART のピン構成設定

Pin Configuration Property	設定値	説明
Pin Group Selection	Mixed, _A Only, _B Only	ピングループの選択
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard (Default: Disabled)	UART トランスミッタ実装では動作モードとして非同期 UART を選択します。

Pin Configuration Property	設定値	説明
TXD_MOSI	None, P707, P409 (Default: None)	TXD ピン P707
RXD_MISO	None, P706, P408 (Default: None)	RXD ピン P706

***注：** 前の設定例は、 Synergy S7G2 および DK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.8.5 アプリケーションでの UART 通信フレームワークモジュールの使用

アプリケーションで UART 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して UART 通信フレームワークを初期化します。
2) lock API を使用して連続通信のためにチャネルをロックします（必要な場合）。
3) read API を使用してデータを受信します。
4) write API を使用してデータを送信します。
5) unlock API を使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) close API を使用して、チャネルを閉じます。

上記の手順を、次の図の通常の動作フロー図に示します。

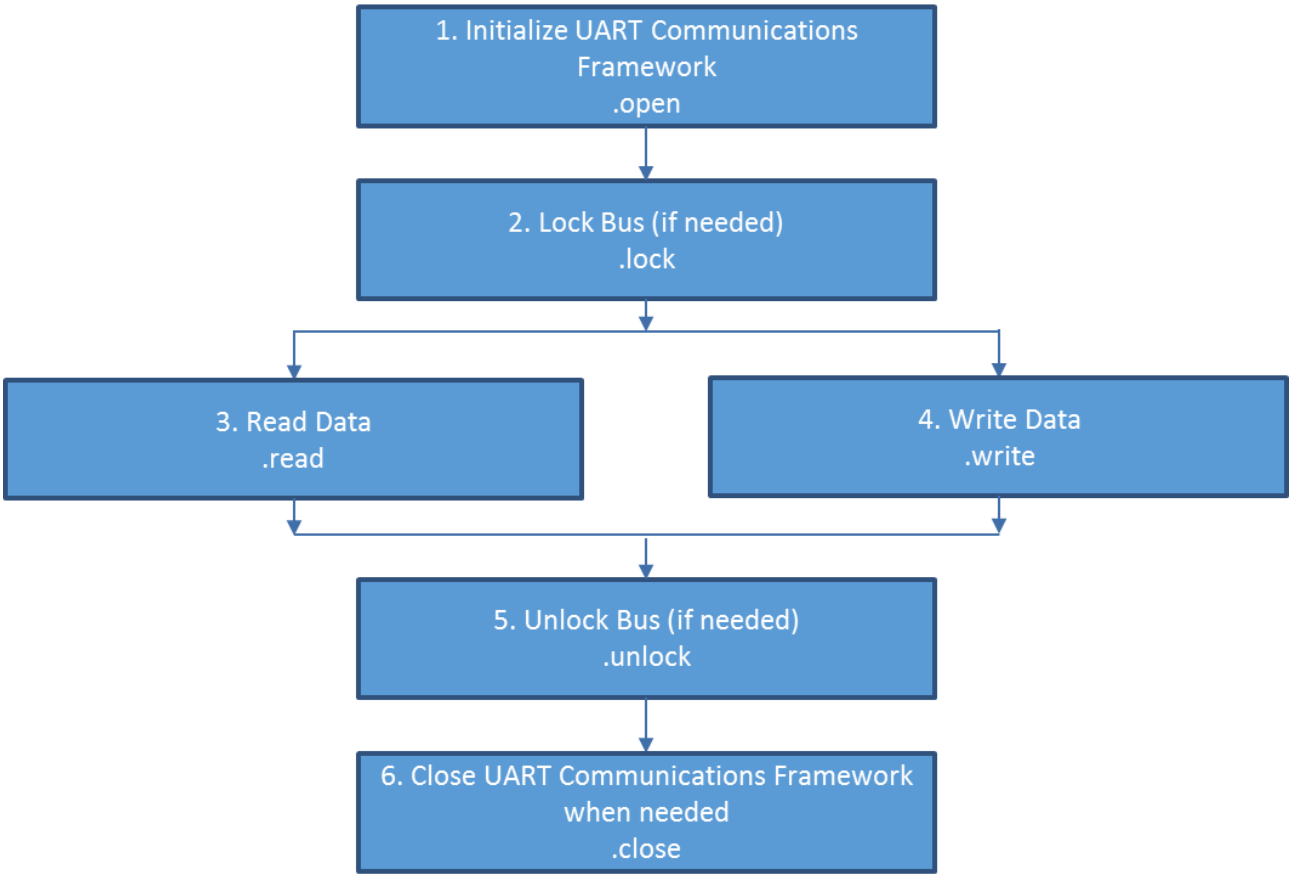


図 162: 通常の UART 通信フレームワークアプリケーションのフロー図

5.1.9 USBX™ 上の通信フレームワークモジュール

- USB 上で高レベルの接続性がサポートされますが、API を変更せずに UART やイーサネット接続に簡単に変更できます。
- 専用アクセスのためのチャネルロックをサポートします。
- USB のハイスピード（HS）またはフルスピード（FS）動作をサポートします。
- データ転送（DMA または DTC）の動作をサポートします。
- ThreadX® 対応実装はミューテックスとイベントフラグを内部で使用します。

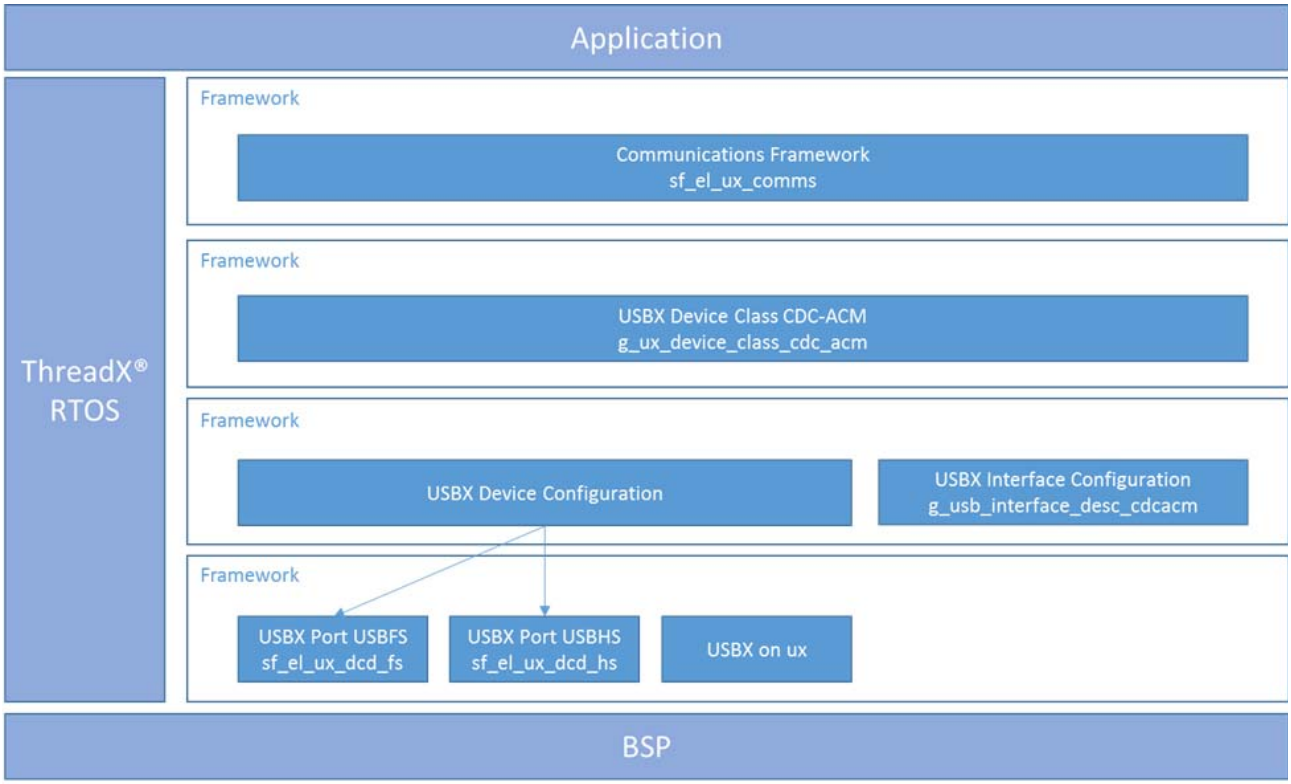


図 163: USBX 上の通信フレームワークモジュールのブロック図

5.1.9.1 USBX 上の通信フレームワークモジュールの API の概要

USBX 上の通信フレームワークは、USB 接続を介したオープン、クローズ、リード、ライトの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX 上の通信フレームワークモジュールの API 要約

Function Name	API の呼び出し例と説明
.open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); 通信ドライバーを初期化します。

Function Name	API の呼び出し例と説明
.close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); 通信ドライバーをクリーンアップします。
.read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); 通信ドライバーからデータを読み取ります。この呼び出しは、要求された数のバイトを読み取った後、またはドライバーへのアクセスを待っている間にタイムアウトが発生した場合に復帰します。
.write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); 通信ドライバーにデータを書き込みます。この呼び出しは、すべてのバイトが書き込まれた後、またはドライバーへのアクセスを待っている間にタイムアウトが発生した場合に復帰します。
.lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); 通信ドライバーをロックします。通信ドライバーへの排他的アクセスを予約します。
.unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); 通信ドライバーをロック解除します。通信ドライバーへの排他的アクセスを解放します。
.versionGet	g_sf_comms0.p_api->versionGet(&version); バージョンポインタに API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	チャネルが正常に開かれました。
SSP_ERR_IN_USE	チャネルは既に使用中です。
SSP_ERR_ASSERTION	UART 制御ブロックまたは設定構造体へのポインタが NULL です。
SSP_ERR_INTERNAL	この関数はあらゆるチャネルで再入可能です。
SSP_ERR_TIMEOUT	タイムアウトエラー。
SSP_ERR_NOT_OPEN	モジュールが開かれていません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.9.2 USBX 上の通信フレームワークモジュールの動作の概要

USBX 上の通信フレームワークによって、USB ポートを介した使用しやすい接続が提供されます。このフレームワークの高レベル API は、他の接続実装（UART やイーサネットなど）と互換性があるため、API を変更せずに簡単に実装を切り替えることができます。このモジュールは、トランザクションの完了に同期するためにミューテックスなどの ThreadX オブジェクトを使用します。USBX 通信フレームワークモジュールはロック機能をサポートします。つまり、ユーザーが通信フレームワークをスレッドにロックすることができ、複数のスレッドが同一の USBX ポートを安全に使用できます。ロックすると、アプリケーションが一定時間（API をロックする呼び出しから API をロック解除する呼び出しまで）USB ポートを確保できます。高レベル API（read API、write API）を使用して、USBX CDC-ACM 通信インタフェースを介してホストとのデータの送受信をサポートします。

USBX 上の通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

USBX ドライバーは、ターゲット MCU でサポートされるバージョンに応じて、HS または FS の USB に実装できます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.9.3 アプリケーションへの USBX 上の通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX 上の通信フレームワークを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

USBX 上の通信フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。（USBX 上の通信フレームワークのデフォルト名は g_sf_ux. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

USBX 上の通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_el_ux_comms	**Threads**	**New Stack> Framework> Connectivity> Communications Framework on sf_el_ux_comms**

次の図に示すように、sf_el_ux_commsのUSBX上の通信フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、低レベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）低レベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

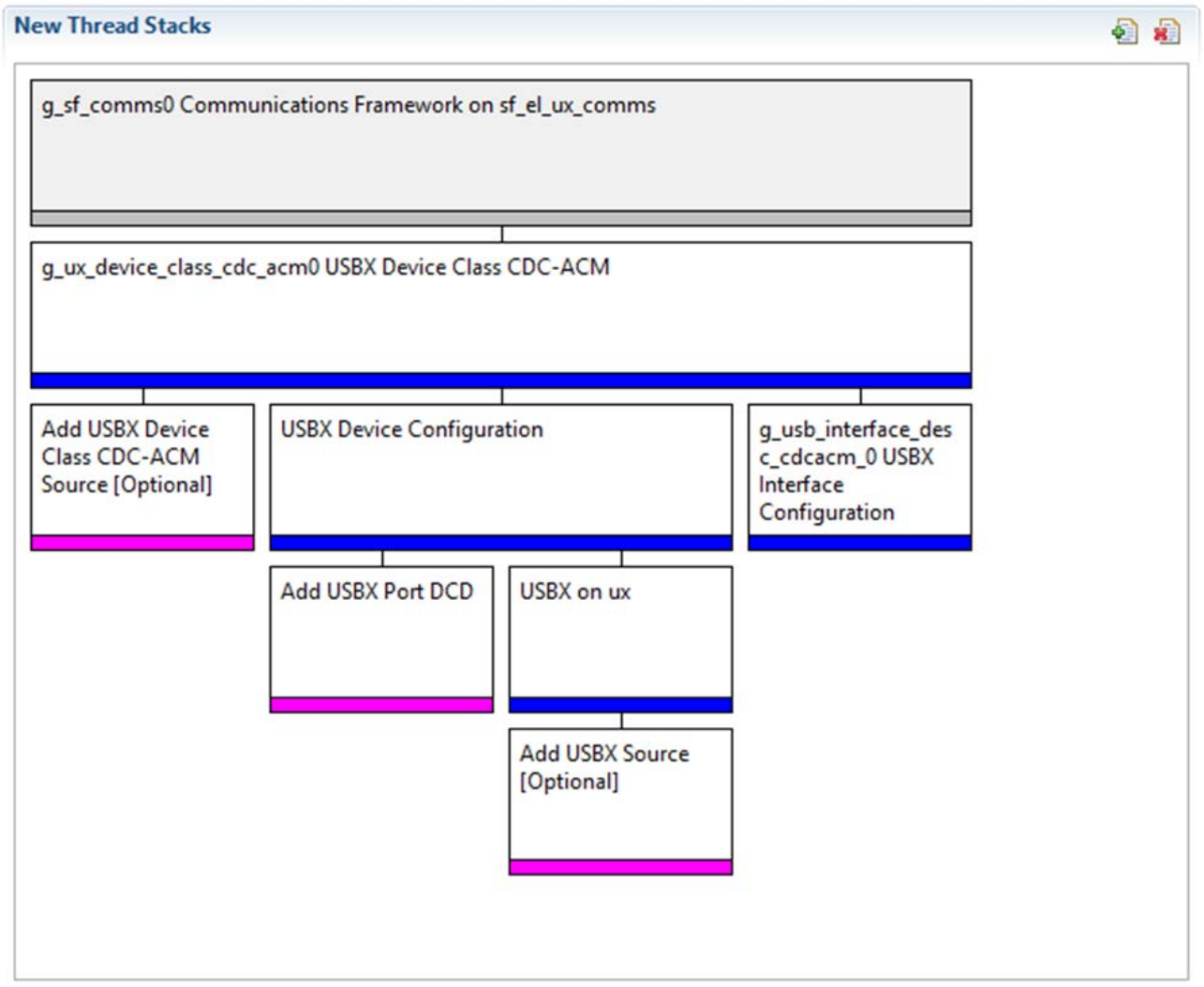


図 164: USBX 上の通信フレームワークモジュールのスタック

5.1.9.4 USBX 上の通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、 USBX 上の通信フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の **[Properties]** ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

sf_el_ux_comms の USBX 上の通信フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Read Input Buffer Size (Bytes)	128	read API で一度に受信できる最大バイト数。
Timeout in ticks	1000	open API での USBX CDC インスタンス作成をサスペンドするタイムアウト値。
Name	g_sf_comms0	モジュール名
Name of the sf_comms initialization function	sf_comms_init0	通信フレームワークを初期化するヘルパー関数の名前。この関数は、<xxx_thread>.c 内に自動生成されるコードで示されます。この <xxx_thread> は、スレッドプロパティで設定したスレッド記号の名前です。この関数は、自動 sf_comms 初期化プロパティが有効の場合に、自動生成コード内でコールされます。無効の場合は、この関数をユーザーアプリケーションでコールすることができます。
Auto sf-comms Initialization	Enable, Disable Default: Enable	通信フレームワークの自動初期化サポート。この構成を有効にすると、自動生成コードで上記のヘルパー関数がコールされます。無効の場合は、関数が自動でコールされず、ユーザーが後でコールすることができます。

注: 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、アプリケーションに応じた異なるバッファサイズの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: モジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX 上の通信フレームワークのローレベルドライバーの構成設定

USBX デバイスクラス CDC-ACM の構成設定

ISDE Property	Value	説明
Name	g_ux_device_class_cdc_acm0	モジュール名
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	USBX CDC-ACM instance_activate 関数コールバックの選択
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	USBX CDC-ACM instance_deactivate 関数コールバックの選択

注: 設定例とデフォルトは、 Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成の構成設定

ISDE Property	Value	説明
Vendor ID	0x045B	ベンダ ID の選択
Product ID	0x0000	製品 ID の選択
Device Release Number	0x0000	デバイスリリース番号の選択
Index of Manufacturing String Descriptor	0x00	メーカー文字列記述子のインデックスの選択
Index of Product String Descriptor	0x00	製品文字列記述子のインデックスの選択
Index of Serial Number String Descriptor	0x00	シリアル番号文字列記述子のインデックスの選択

ISDE Property	Value	説明
Class Code	Device, Communications (CDC), HID, Mass Storage, Miscellaneous, Vendor Specific Default: Communications (CDC)	クラスコードの選択
Index of String Descriptor describing this configuration	0x00	この構成を記述する文字列記述子のインデックスの選択
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	この構成の USB 記述子のサイズ（バイト）の選択（この値はベンダ固有クラスのみ変更、それ以外は 0 に設定）
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	インタフェース数の選択（この値はベンダ固有クラスのみ変更、それ以外は 0 に設定）
Self-Powered	Enable, Disable Default: Enable	電源内蔵の選択
Remote Wakeup	Enable, Disable Default: Disable	リモート起動の選択
Maximum Power Consumption (in 2mA units)	50	最大消費電力（2mA 単位）の選択
Supported Language Code	0x0409	サポートされる言語コードの選択
Name of USBX String Framework	NULL	USBX 文字列フレームワークの名前の選択
Total index number of USB String Descriptors in USB String Framework	0	USB 文字列フレームワークの USB 文字列記述子の合計インデックス数の選択
Name of USBX Language Framework	NULL	USBX 言語フレームワークの名前の選択
Number of Languages to support (US English is applied if zero is set)	0	サポートされる言語の数（0 に設定すると米国英語が適用）の選択

注: 設定例とデフォルトは、Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成の構成設定

ISDE Property	Value	説明
Name	g_usb_interface_desc_cdcacm_0	モジュール名
Interface Number of Communications Class interface	0x00	通信クラスインタフェースのインタフェース番号の選択
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 Default: Endpoint 3	通信クラスで使用するインタラプト転送エンドポイントの選択
Polling period for Interrupt Endpoint (in mS/125us units for FS/HS)	0x0F	割り込みエンドポイントに対するポーリング間隔（FS/HS の場合は、ms/125µs 単位）の選択
Interface Number of Data Class interface	0x01	データクラスインタフェースのインタフェース番号の選択
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 1	データクラスで使用するバルクイン転送エンドポイントの選択
Bulk Out Transfer endpoint to use for Data Class	End Default: Endpoint 2	データクラスで使用するバルクアウト転送エンドポイントの選択
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	通信クラスインタフェースを記述する文字列記述子のインデックス（インタフェース記述子: インタフェース）の選択
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	データクラスインタフェースを記述する文字列記述子のインデックス（インタフェース記述子: インタフェース）の選択

注: 設定例とデフォルトは、Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS 用の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	説明
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フルスピードの割り込み優先順位の選択
Name	g_sf_el_ux_dcd_fs_0	モジュール名
USB Controller Selection	USBFS	USB コントローラの選択

注: *設定例とデフォルトは、Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBHS 用の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	説明
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ハイスピードの割り込み優先順位の選択。
Name	g_sf_el_ux_dcd_hs_0	モジュール名。
USB Controller Selection	USBHS	USB コントローラの選択。

注: *設定例とデフォルトは、Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ux 上の USBX の構成設定

ISDE Property	Value	説明
USBX Pool Memory Name	g_ux_pool_memory	モジュール名

ISDE Property	Value	説明
USBX Pool Memory Size	18,432	USBX プールメモリサイズの選択
User Callback for Host Event Notification (Only valid for USB Host)	Default: NULL	ホストイベント通知を処理するコールバック関数。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBX 上の通信フレームワークモジュールのクロック構成

USB 周辺モジュールは UCLK をドライバーとして使用するため、SSP コンフィギュレータの [Clocks] タブで 48MHz に設定する必要があります。

USBX 上の通信フレームワークモジュールのピン構成

USB 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は USBFS0 のピンの選択例を示しています。

注: *USBHS ピンは次の表に示すピンと似ているため含めていません。*

注: *動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。*

USBFS0 でのピンの選択

Resource	ISDE Tab	Pin selection Sequence
USBFS0	Pins	Select Peripherals > Connectivity:USBFS > USBFS0

注: *選択シーケンスでは、USBFS0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

USBFS0 のピン構成設定

Property	設定値	説明
Operation Mode	Disabled, Custom, Device, Host, OTG (Default: Disabled)	通信フレームワークのデバイスを選択
USBDP	USBDP	USBDP ピン
USBDM	USBDM	USBDM ピン

Property	設定値	説明
OVRCURB	None, P204, P205 (Default: None)	OVRCURB ピン
OVRCURA	None, P205, P501 (Default: None)	OVRCURA ピン
VBUSEN	None, P206, P500 (Default: None)	VBUSEN ピン
VBUS	None, P406 (Default: None)	VBUS ピン
EXICEN	None, P409, P503 (Default: None)	EXICEN ピン
ID	None, P408, P504 (Default: None)	ID ピン
VCCUSB	VCCUSB	VCCUSB ピン
VSSUSB	VSSUSB	VSSUSB ピン

5.1.9.5 アプリケーションでの USBX 上の通信フレームワークモジュールの使用

アプリケーションで USBX 上の通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して USBX 上の通信フレームワークを初期化します。
2) lock API を使用して連続通信のためにチャネルをロックします（必要な場合）。
3) read API を使用してデータを受信します。
4) write API を使用してデータを送信します。
5) unlock API を使用して連続通信用のチャネルのロックを解除します（必要な場合）。

6) close API を使用して、チャネルを閉じます。

これらの手順を、次の図の通常の動作フロー図に示します。

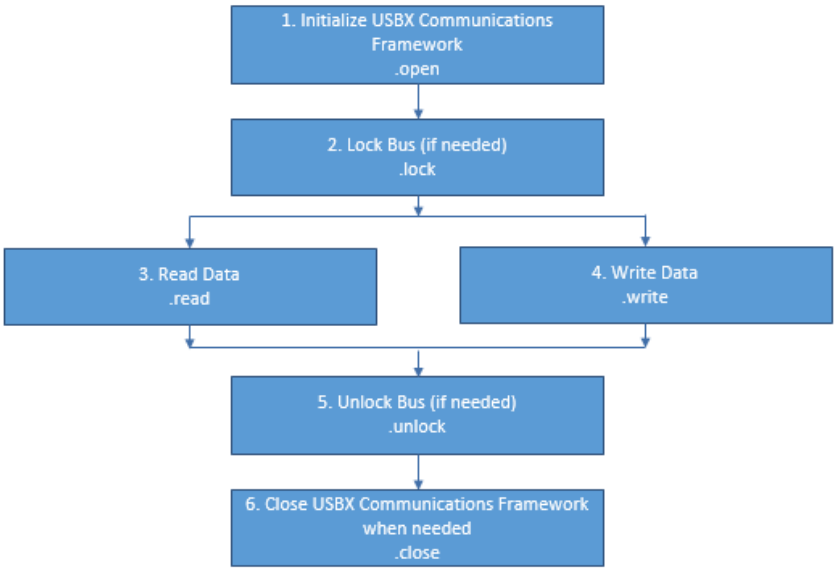


図 165: 通常の USBX 上の通信フレームワークアプリケーションのフロー図

5.1.10 コンソールフレームワーク

コンソールフレームワークは以下の機能に対応します。

- メニューに基づくコマンドラインインタフェースの作成
- サブメニューと、単一呼び出しによる複数メニュー内の移動
- 親メニューへの移動、またはメニューのルートに戻る
- メニューごとのヘルプメニュー
- NULL 終端文字列の書き込みと、改行文字までの読み取り
- 引数をコマンドラインに解析する API
- 大文字と小文字を区別しない入力

コンソールフレームワークモジュールの編成（SSP コンフィギュレータの [Thread Stack] ウィンドウに表示される）を次の図に示します。各実装の選択肢（イーサネット、UART、USB）にはそれぞれ低レベルモジュールがあり、開発者が選択した実装に基づいて自動的に追加されます。ほとんどのケースで、必要なすべての構成情報がモジュールに自動的に追加されるため、開発者が選択する必要があるのははごくわずかの重要な構成設定のみです。

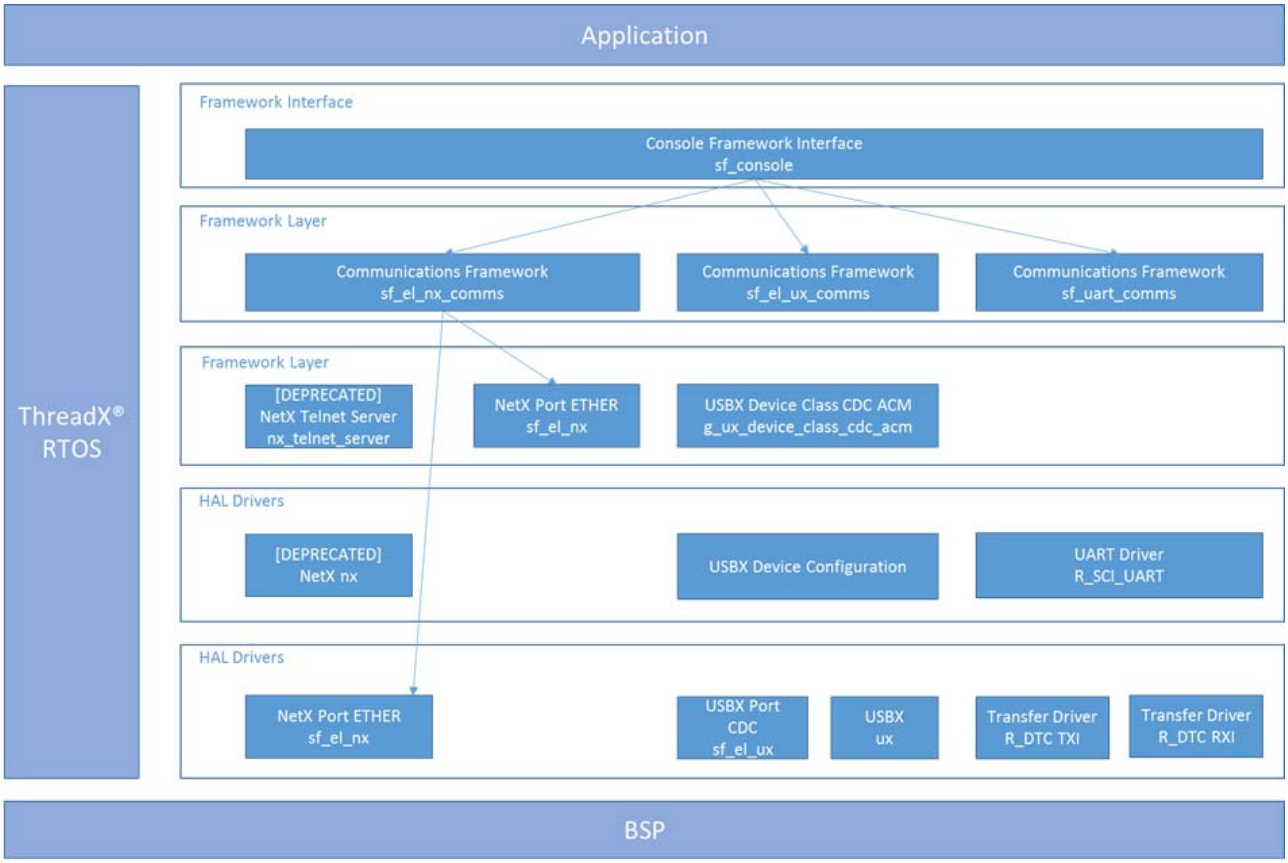


図 166:　コンソールフレームワークモジュールのブロック図

5.1.10.1 コンソールフレームワークモジュール API の概要

コンソールフレームワークモジュールは、入力プロンプトのオープン、クローズ、リード、ライト、発行の API を定義します。また、解析や augumentFind など、複雑なコマンドの処理に役立つその他の機能も提供されます。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

コンソールフレームワークモジュール API 要約

Function Name	API の呼び出し例と定義
open	g_sf_console0.p_api->open(g_sf_console0.p_ctrl, g_sf_console0.p_cfg) open APIはコンソールを構成します。この関数は、他のコンソール関数の前に呼び出す必要があります。***注:*** *この呼び出しは、ユーザースレッドに入る前、システムの初期化の間に自動的に行われます。ユーザーがコンソールを閉じる場合を除き、open を呼び出す必要はありません。*
close	g_sf_console0.p_api->close(g_sf_console0.p_ctrl); close API は、内部ドライバーデータのクリーンアップを処理します。
.prompt	g_sf_console0.p_api->prompt(g_sf_console0.p_ctrl, NULL, TX_WAIT_FOREVER); prompt API は、メニューのプロンプト文字列を出力して入力を待ちます。メニューに基づいて入力を解析し、コマンドが識別された場合は、コールバック関数を呼び出します。
parse	g_sf_console0.p_api->parse(g_sf_console0.p_ctrl, commands, input, s_length); parse API は、コマンドメニューで入力文字列を探し、見つかった場合には該当するコールバック関数を呼び出します。
read	g_sf_console0.p_api->read(g_sf_console0.p_ctrl, ch, 1, TX_WAIT_FOREVER); read API は、宛先にデータをバイト単位で挿入し、入力をコンソールにエコーします。バックスペースキー、デリートキー、左右の矢印キーがサポートされています。改行を示す CR または CR + LF、CR + NULL が受信されるか、入力したデータが許可されているバイト数を超えると、読み取りが完了します。バッファが SF_CONSOLE_MAX_INPUT_LENGTH, をオーバーフローした場合は、read によってエラーコードが返されます。
write	g_sf_console0.p_api->write(g_sf_console0.p_ctrl, (uint8_t*)data_string, TX_WAIT_FOREVER); write API はバッファミューテックスオブジェクトを取得し、HAL レイヤーでのデータ送信を処理します。これには、データ転送の完了と同期するためのイベントフラグが含まれます。
argumentFind	g_sf_console0.p_api->argumentFind("LED", p_args->p_remaining_string, NULL, &led_num); argumentFind API は入力文字列内でコマンドライン引数を検索し、引数の直後の文字のインデックスを返します。文字列の数はすべて整数に変換されます。

Function Name	API の呼び出し例と定義
versionGet	g_sf_console0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	p_ctrl が NULL です。
SSP_ERR_UNSUPPORTED	コマンドが現在のメニューで見つかりませんでした。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.10.2 コンソールフレームワークモジュールの動作の概要

コンソールフレームワークモジュールは、ThreadX 対応のコマンドラインインタフェース（CLI）です。このモジュールは、ミューテックスなどの ThreadX オブジェクトを使用してブロックを行い、イベントフラグなどの同期手法を使用してトランザクションを完了します。コンソールフレームワークの動作の重要な要素は初期化および入力処理です。それぞれについて次のセクションで説明します。

コンソールフレームワークモジュールの初期化

open 呼び出しは、モジュールが追加された src/synergy_gen/toggle_thread.c ファイル内に ISDE によって自動的に生成されます。この open 呼び出しを実行するには、デフォルトでコンフィギュレータにおける名前 (g_sf_console_root_menu) と一致する変数名を持つルートメニューをアプリケーションで定義する必要があります。必要なハードウェア接続が確立されていれば、実行が src/toggle_thread_entry.c, に達するまでにモジュールが使用可能になります。

コンソールフレームワークモジュールの入力処理

コンソールフレームワークモジュールでは、一連のメニュー、コマンド構造体、およびコールバックが必要です。コンソールフレームワークモジュールは通常はプロンプトで作動し、多くの場合、entry スレッド内の while ループに配置されます。このフレームワークの promptAPI は、現在のメニューをプロンプトとして出力してから、入力を読み取り、それをコンソールにエコーバックします（プロパティでエコーが無効にされている場合は除く）。

次の動作がユーザー入力に対して実行されます。

コンソールで入力が行われる際の動作は次のとおりです。

- BackSpace キーを押すとカーソルの前の文字が削除されます。
- Delete キーを押すとカーソルの後の文字が削除されます。
- 左右の矢印キーを押すとカーソルが移動します。
- 上方向キーを押すと、他に何も入力していない場合にのみ、最後のコマンドが入力されます。

注 : *最後のコマンドより前の履歴は保持されません。つまり、上方向キーを 2 回押しても、最後のコマンドの前に入力されたコマンドの情報は残っていないため、最後のコマンドが引き続き表示されます。*

読み取り入力時に改行文字が現れると、入力文字列が解析され、関連するコールバックが呼び出されるか、次のメニューに切り替わります（コマンドのコールバックの代わりに SF_CONSOLE_CALLBACK_NEXT_FUNCTION が使用されている場合）。解析はコールバック関数が呼び出されるまで続行されます。prompt API が再び呼び出された場合は、コールバック関数を含むメニューを表示して入力待ちになります。親メニューに移動するには、'^' を押します。サブメニューからルートメニューに移動するには、'~' を押します。

コンソールフレームワークモジュールの必須構造体の作成 – メニュー

コンソールフレームワークではメニューが必要です。メニューの実装にコンソールフレームワークが使用する構造体の作成は開発者が行います。コンソールメニューの構造体（次の図を参照）には、前のメニューへのポインタ、メニューの名前（複数レベルのメニューを作成するため）、メニュー内のコマンド数、コマンド構造体の配列へのポインタが含まれます。次の図に示すように、コマンド配列の各エントリには、コマンド名文字列へのポインタ、help コマンドの説明文字列、関連するコマンドコールバック関数、コールバックに提供されるコンテキストパラメータが含まれます。

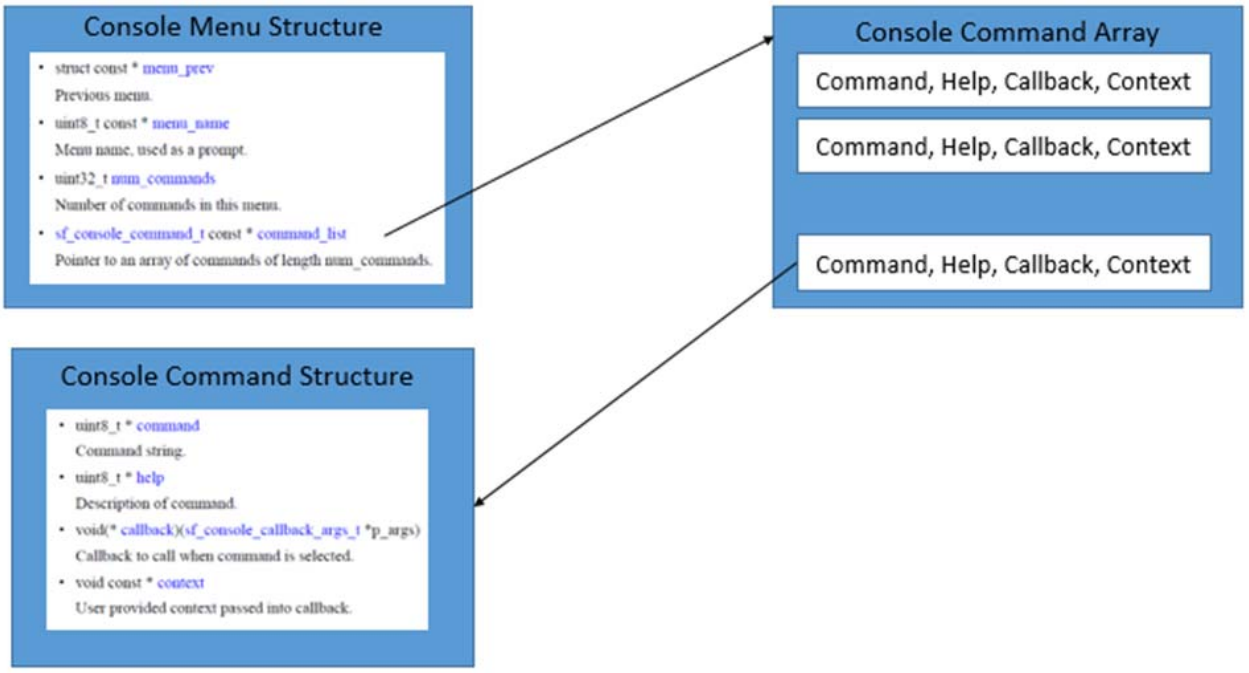


図 167: コンソールフレームワークモジュールのメニュー構造体

アプリケーションプロジェクトの例として、1 つのメニューがあるコンソールフレームワークを示します。これは、 CLI で LED の切り替えを制御します。コンソールコマンド配列 （g_sf_console_commands, 下図右側）にはコマンドの配列が格納されます （このケースでは 1 つのコマンドのみ）。コマンド構造体によって、コマンドが 「LED TOGGLE」、ヘルプの説明が 「Toggle an LED」、コールバックが led_toggle_callback, コンテキストが NULL （この例では使用されないため）であることが定義されます。

ルートメニュー（下図左側）は g_sf_console_root_menu 構造体です。この構造体では、menu_prev エントリが NULL （1 つのメニューしかないため）、 menu_name が 「Root」、 num_commands は 「1」（エントリの合計数を求めるために配列サイズをエントリサイズで割った値）、command_list の開始アドレスは「address」（コマンド配列の最初のエントリの場所）が定義されます。

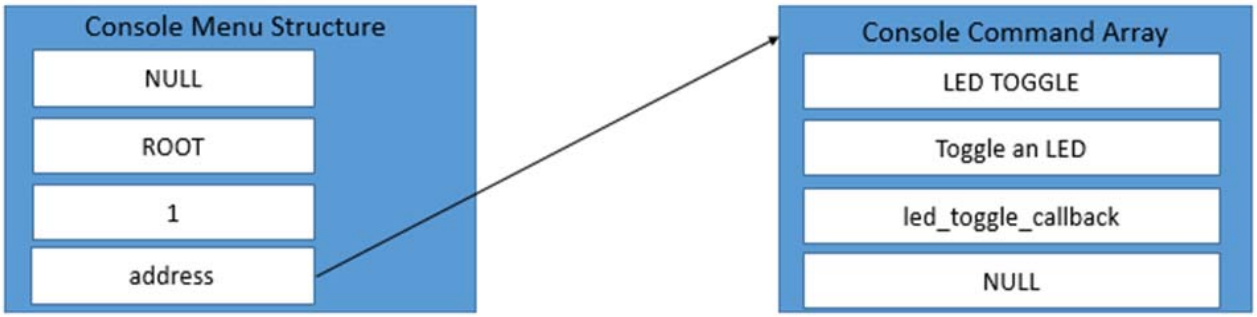


図 168: メニュー構造体例の図

コンソールフレームワークモジュールの動作に関する重要な注意事項と制限事項

コンソールフレームワークモジュールの prompt API を使用するには、まずメニュー、コマンド構造体、およびコールバックを設定します。

このモジュールを使用するうえで既知の制限事項はありません。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.10.3 アプリケーションへのコンソールフレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションにコンソールフレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

コンソールフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。（コンソールフレームワークモジュールのデフォルト名は g_sf_console0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

コンソールフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_console0 Console Framework on sf_console	Threads	New Stack > Framework > Services > Console Framework on sf_console

次の図に示すように、sf_console 上のコンソールフレームワークがスレッドスタックに追加されると、コンフィギュレータは、コンソールフレームワークを完成させるために少なくとも 1 つの通信フレームワークが必要であることを（[Add Communications Framework] ブロックを介して）レポートします。通信フレームワークは、コンソールフレームワークが使用する通信インタフェースのタイプ（および基礎となるハードウェア実装）を判別します。追加の構成情報を必要とする低レベルモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、低レベルモジュールの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。低レベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

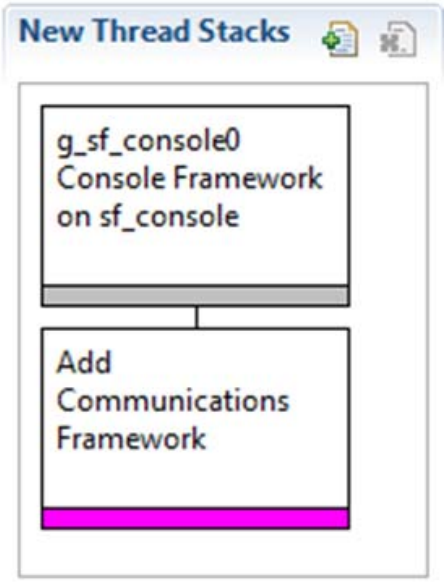


図 169: コンソールフレームワークモジュールのスタック

現在、コンソールフレームワークで選択できる通信フレームワークは UART、USB、Telnet の 3 種類です。それぞれの構成を次のスレッドスタック図に示します。これらは、[Add Communications Framework] ブロックをクリックして、必要な通信フレームワークを選択すると簡単にインポートできます。（Telnet オプションでは非推奨の実装が使用されることに注意してください。これは正常に機能しますが、将来のリリースでは差し替えになるため非推奨とされます。非推奨モジュールを使用する設計は、そのリリース後に新しい実装に更新する必要があります。）他の通信フレームワークが追加され、使用できるようになると表示されます。

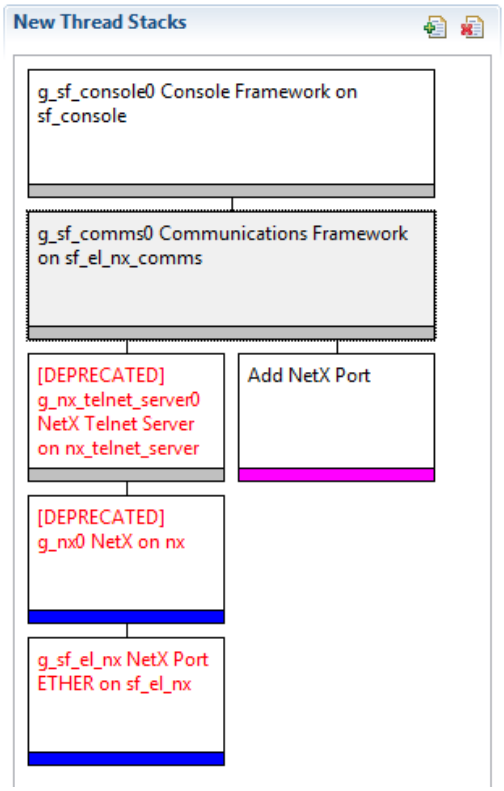

New Thread Stacks
g_sf_console0 Console Framework on sf_console
g_sf_comms0 Communications Framework on sf_el_nx_comms
[DEPRECATED] g_nx_telnet_server0 NetX Telnet Server on nx_telnet_server
Add NetX Port
[DEPRECATED] g_nx0 NetX on nx
g_sf_el_nx NetX Port ETHER on sf_el_nx

図 170: イーサネット

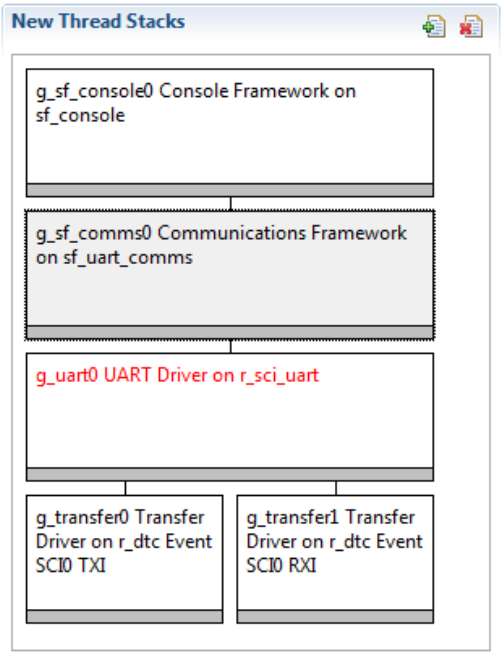

New Thread Stacks
g_sf_console0 Console Framework on sf_console
g_sf_comms0 Communications Framework on sf_uart_comms
g_uart0 UART Driver on r_sci_uart
g_transfer0 Transfer Driver on r_dtc Event SCI0 TXI
g_transfer1 Transfer Driver on r_dtc Event SCI0 RXI

図 171: UART

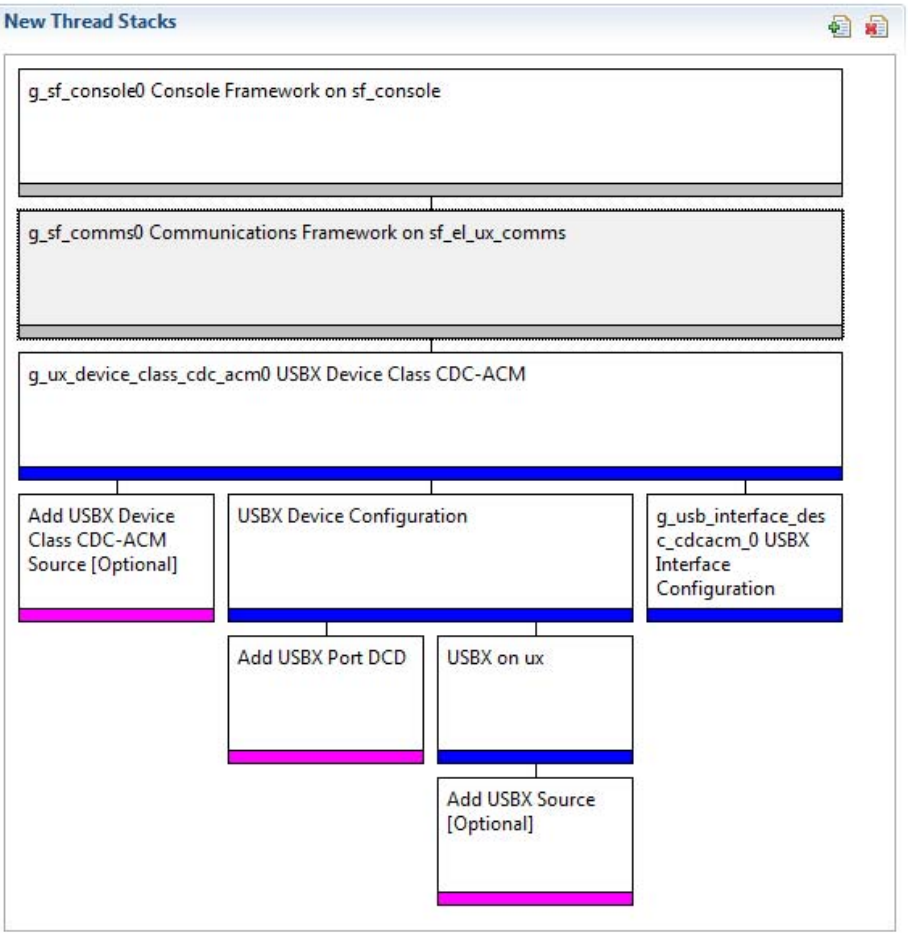


図 172: USB

通信フレームワークモジュールのオプション

5.1.10.4 コンソールフレームワークモジュールの構成

ユーザーは必要な動作に合わせてコンソールフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties]

ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_console 上のコンソールフレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enable, Disable Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択します。
Maximum Input String Length	128	入力文字列の長さ。
Maximum Write String Length	128	出力文字列の長さ。
Name	g_sf_console0	コンソールフレームワークモジュール名。
Name of Initial Menu (Application Defined)	g_sf_console_root_menu	開始メニューの名前。
Echo	True, False Default: True	プロンプトから端末へのエコーを有効または無効にします。
Autostart	True, False Default: False	True の場合は、初期化の後に最上位メニューを使用したプロンプトが発生します。False の場合は、プロンプトをアプリケーションで呼び出す必要があります。
Name of the sf_console Initialization Function	sf_console_init0	sf_console 初期化関数の名前の選択。
Auto sf_console Initialization	Enable, Disable Default: Enable	自動 sf_console 初期化の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、Telnet 接続の IP アドレスまたは UART のボーレートを変更する必要があります。低レベルスタックモジュールで構成可能なプロパティは参照のためにすべて記載しています。

注： *モジュールのプロパティ設定の大半は直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

コンソールフレームワークスタックモジュールの構成設定

通常、低レベルモジュールでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

低レベルの実装に対して Telnet オプションが選択されている場合は、次のモジュールがスタックに自動的に追加されます。次の表に使用可能な構成設定の詳細を示します。これらの多くは、ほとんどのアプリケーションで使用できるデフォルト設定が移入されています。

Telnet オプション (sf_el_nx_comms) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled	パラメータチェックを有効または無効にします。
	Default: BSP	
Name	g_sf_comms0	モジュール名。
Channel	0	イーサネットドライバーによって使用される基底チャネル。
IP Address Byte 1	192	IP アドレスバイト 1 の選択
IP Address Byte 2	168	IP アドレスバイト 2 の選択
IP Address Byte 3	0	IP アドレスバイト 3 の選択
IP Address Byte 4	0	IP アドレスバイト 4 の選択
Subnet Mask Byte 1	255	サブネットマスクバイト 1 の選択
Subnet Mask Byte 2	255	サブネットマスクバイト 2 の選択
Subnet Mask Byte 3	255	サブネットマスクバイト 3 の選択
Subnet Mask Byte 4	0	サブネットマスクバイト 4 の選択

***注：** IP アドレスや関連するマスクの有効な設定や共通設定の情報と説明は、このドキュメントの最後に記載した、Synergy プラットフォームのナレッジベースの IP アドレスの制限事項に関する記事を参照してください。設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

スタックモジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なる IP アドレスの使用が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: 次に示すNetX Telnet サーバーモジュールとNetX モジュールの非推奨の構成は、sf_el_nx_comms. 上の通信フレームワークモジュールを使用する際に必要です。非推奨の構成の使用を避けるため、 sf_el_nx_comms 上の通信フレームワークモジュールは将来のリリースで更新されます。

NetX Telnet サーバーモジュール (nx_telnet_server) の構成設定

ISDE Property	Value	説明
Name	g_nx_telnet_server0	モジュール名
Show deprecation warning	Enabled, Disabled Default: Enabled	非推奨警告の表示の選択

注: 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX モジュール （nx）の構成設定

ISDE Property	Value	説明
Name	Default: g_nx0	NetX モジュール名。
Show deprecation warning	Enabled, Disabled Default: Enabled	非推奨警告の表示の選択

注: 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX ポート ETHER(sf_el_nx) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
Channel 0 Phy Reset Pin	IOPORT_PORT_09_PIN_03	チャネル 0 Phy リセットピンの選択

ISDE Property	Value	説明
Channel 0 MAC Address High Bits	0x00002E09	チャネル 0 MAC アドレス高ビットの選択
Channel 0 MAC Address Low Bits	0x0A0076C7	チャネル 0 MAC アドレス低ビットの選択
Channel 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	チャネル 1 Phy リセットピンの選択
Channel 1 MAC Address High Bits	0x00002E09	チャネル 1 MAC アドレス高ビットの選択
Channel 1 MAC Address Low Bits	0x0A0076C8	チャネル 1 MAC アドレス低ビットの選択
Number of Receive Buffer Descriptors	8	受信バッファ記述子の数の選択
Number of Transmit Buffer Descriptors	32	送信バッファ記述子の数の選択
Ethernet Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	イーサネット割り込み優先順位の選択
Name	g_sf_el_nx	モジュール名
Channel	0	チャネルの選択
Callback	NULL	コールバックの選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ローレベルの実装に対して USB オプションが選択されている場合は、次のモジュールがスタックに自動的に追加されます。次の表に使用可能な構成設定の詳細を示します。これらの多くは、ほとんどのアプリケーションで使用できるデフォルト設定が移入されています。

USB 通信フレームワークモジュール (sf_el_ux_comms) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Read Input Buffer Size (Bytes)	128	read() API で一度に受信できる最大バイト数。
Timeout in ticks	1000	open() API での USBX CDC インスタンス作成をサスペンドするタイムアウト値。
Name	g_sf_comms0	モジュール名。
Name of the sf_comms initilization function	sf_comms_init0	通信フレームワークを初期化するヘルパー関数の名前。この関数は、<xxx_thread>.c 内に自動生成されるコードで示されます。この <xxx_thread> は、スレッドプロパティで設定したスレッド記号の名前です。この関数は、自動 sf_comms 初期化プロパティが有効の場合に、自動生成コード内でコールされます。無効の場合は、この関数をユーザーアプリケーションでコールすることができます。
Auto sf-comms Intialization	Enable, Disable Default: Enable	通信フレームワークの自動初期化サポート。この構成を有効にすると、自動生成コードで上記のヘルパー関数がコールされます。無効の場合は、関数が自動でコールされず、ユーザーが後でコールすることができます。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBX デバイスクラス CDC-ACM モジュール (g_ux_device_class_cdc_acm0) の構成

ISDE Property	Value	説明
Name	g_ux_device_class_cdc_acm0	モジュール名

ISDE Property	Value	説明
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	USBX CDC-ACM instance_activate 関数コールバックの選択
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	USBX CDC-ACM instance_deactivate 関数コールバックの選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBX デバイス構成の構成設定

ISDE Property	Value	説明
Vendor ID	0x045B	ベンダ ID の選択
Product ID	0x0000	製品 ID の選択
Device Release Number	0x0000	デバイスリリース番号の選択
Index of Manufacturing String Descriptor	0x00	メーカー文字列記述子のインデックスの選択
Index of Product String Descriptor	0x00	製品文字列記述子のインデックスの選択
Index of Serial Number String Descriptor	0x00	シリアル番号文字列記述子のインデックスの選択
Class Code	Device, Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor Specific Default: Communications	クラスコードの選択
Index of String Descriptor describing this configuration	0x00	この構成を記述する文字列記述子のインデックスの選択
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	この構成の USB 記述子のサイズ（バイト）の選択（この値はベンダ固有クラスのみ変更、それ以外は 0 に設定）

ISDE Property	Value	説明
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	インタフェース数の選択（この値はベンダ固有クラスのみ変更、それ以外は 0 に設定）
Self-Powered	Enable, Disable Default: Enable	電源内蔵の選択
Remote Wakeup	Enable, Disable Default: Disable	リモート起動の選択
Maximum Power Consumption (in 2mA units)	50	最大消費電力（2mA 単位）の選択
Supported Language Code	0x0409	サポートされる言語コードの選択
Name of USBX String Framework	NULL	USBX 文字列フレームワークの名前の選択
Total index number of USB String Descriptors in USB String Framework	0	USB 文字列フレームワークの USB 文字列記述子の合計インデックス数の選択
Name of USBX Language Framework	NULL	USBX 言語フレームワークの名前の選択
Number of Languages to support (US English is applied if zero is set)	0	サポートされる言語の数（0 に設定すると米国英語が適用）の選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBX インタフェース構成の構成設定

ISDE Property	Value	説明
Name	g_usb_interface_desc_cdcacm_0	モジュール名
Interface Number of Communications Class interface	0x00	通信クラスインタフェースのインタフェース番号の選択

ISDE Property	Value	説明
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 Default: Endpoint 3	通信クラスで使用するインタラプト転送エンドポイントの選択
Polling period for Interrupt Endpoint (in mS/125us units for FS/HS)	0x0F	割り込みエンドポイントに対するポーリング間隔（FS/HS の場合は、mS/125us 単位）の選択
Interface Number of Data Class interface	0x01	データクラスインタフェースのインタフェース番号の選択
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 1	データクラスで使用するバルクイン転送エンドポイントの選択
Bulk Out Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 2	データクラスで使用するバルクアウト転送エンドポイントの選択
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	通信クラスインタフェースを記述する文字列記述子のインデックス（インタフェース記述子: インタフェース）の選択
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	データクラスインタフェースを記述する文字列記述子のインデックス（インタフェース記述子: インタフェース）の選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBFS 用の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	説明
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フルスピードの割り込み優先順位の選択。
Name	g_sf_el_ux_dcd_fs_0	モジュール名。
USB Controller Selection	USBFS	USB コントローラの選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBHS 用の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	説明
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ハイスピードの割り込み優先順位の選択。
Name	g_sf_el_ux_dcd_hs_0	モジュール名。
USB Controller Selection	USBHS	USB コントローラの選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ux 上の USBX の構成設定

ISDE Property	Value	説明
USBX Pool Memory Name	g_ux_pool_memory	USBX プールメモリ名の選択。
USBX Pool Memory Size	18432	USBX プールメモリサイズの選択。
User Callback for Host Event Notification (Only valid for USB Host)	NULL	ホストイベント通知のユーザーコールバック（USB ホストでのみ有効）

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ローレベルの実装に対して UART オプションが選択されている場合は、次のモジュールがスタックに自動的に追加されます。次の表に使用可能な構成設定の詳細を示します。これらの多くは、ほとんどのアプリケーションで使用できるデフォルト設定が移入されています。

UART 通信フレームワークモジュール (sf_uart_comms) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled	パラメータチェックを含めるかどうかを選択します。
	Default: BSP	
Read Input Queue Size (4-Byte Words)	15	データ受信キューのバッファサイズ。sf_uart_comms はキュー管理に ThreadX キューを利用します。
Name	g_sf_comms0	UART 通信フレームワークモジュールの名前。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

UART ドライバーモジュール (r_sci_uart) の構成設定

ISDE Property	Value	説明
External RTS Operation	Enable, Disable Default: Disable	RTS 信号として使用する IOPORT ピンを有効にします。RTS 機能では、この設定パラメータを「有効」にし、設定「RTS 外部ピン制御のための UART コールバック関数名」を指定します。

ISDE Property	Value	説明
Reception	Enable, Disable Default: Enable	SCI 上のすべての UART チャネルについて UART の受信を有効または無効にします。この設定パラメータに「無効」を設定すると、コードサイズが小さくなります。UART 受信のためのコード部分がコンパイルされないためです。このパラメータは、個々の UARTT チャネルに対しては設定できません。
Transmission	Enable, Disable Default: Enable	SCI 上のすべてのチャネルについて UART 送信を有効または無効にします。この設定に「無効」を設定すると、UART 送信のためのコード部分がコンパイルから除外されるため、コードサイズが小さくなりますが、この設定に「無効」を設定できるのは、UART ポートとして機能する他の SCI チャネルが送信を行わない場合だけです。
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_uart0	SCI 上の UART モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Channel	0-9	SCI チャネル番号。
Baud Rate	9600	ボーレートの選択。
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART データビット。
Parity	None, Odd, Even Default: None	UART パリティビット。

ISDE Property	Value	説明
Stop Bits	1 bit, 2 bits Default: 1 bit	UART ストップビット。
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	SCI チャネル n の CTSn/RTSn ピンに対して CTS または RTS を選択します。SCI ハードウェアは、このピン上で CTS と RTS のいずれかの制御信号をサポートしますが、両方はサポートしません。CTS と RTS の両方を使用するアプリケーションでは、この設定パラメータに「CTS」を選択し、設定「外部 RTS 操作」を有効にし、設定「RTS 外部ピン制御のための UART コールバック関数名」を指定します。
Name of UART callback function to be defined by user	user_uart_callback	名前は有効な C シンボルである必要があります。
Name of UART callback function for the RTS external pin control to be defined by user	NULL	名前は有効な C シンボルである必要があります。
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	ボーレートクロック発信器ブロックで使用するクロックソースを選択します。
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	選択したチャネル n に対し SCKn ピン上でボーレートクロックを出力するためのオプション設定。
Start bit detection	Falling Edge, Low Level Default: Falling Edge	受信におけるスタートビット検出モード。この設定には、通常は「立ち下がりエッジ」を設定します。

ISDE Property	Value	説明
Noise Cancel	Enable, Disable Default: Disable	RXDn ピン上でデジタルノイズキャンセルを有効にします。SCI のデジタルノイズフィルターブロックは、2ステージのフリップフロップ回路で構成されます。詳細については、Renesas Synergy ハードウェアマニュアルのノイズキャンセルのセクションを参照してください。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調有効化の選択。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信割り込み優先順位の選択。

ISDE Property	Value	説明
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信終了割り込み優先順位の選択。
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	エラー割り込み優先順位の選択。

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 TXI の転送ドライバーモジュール

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択

ISDE Property	Value	説明
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI の転送ドライバーモジュール

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Value	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ローレベルの実装に対して USB オプションが選択されている場合は、次のモジュールがスタックに自動的に追加されます。次の表に使用可能な構成設定の詳細を示します。これらの多くは、ほとんどのアプリケーションで使用できるデフォルト設定が移入されています。

USB 通信フレームワークモジュール (sf_el_ux_comms) の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enable, Disable Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択します。
Memory Size (Bytes)	65536	メモリサイズの選択。
Timeout in ticks	1000	open() API での USBX CDC インスタンス作成をサスペンドするタイムアウト値。
Read Input Buffer Size (Bytes)	128	これは、 read() API で一度に受信できる最大バイト数です。
Name	g_sf_comms0	USB 通信フレームワークモジュールの名前。
Name of the sf_comms Initialization Function	sf_comms_init0	sf_comms 初期化関数の名前の選択。
Auto sf_comms Initialization	Enable, Disable Default: Enable	自動 sf_comms 初期化の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

USBX デバイスクラス CDC-ACM モジュール (g_ux_device_class_cdc_acm0) の構成

ISDE Property	Value	説明
Name	Default: g_ux_device_class_cdc_acm0	USBX デバイスクラス cdc-acm モジュールの名前。
Show Deprecation Warning	Enabled, Disabled Default: Enabled	非推奨警告の表示の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ux 上の USXB の構成設定

ISDE Property	Value	説明
Name	Default: g_ux0	ux 上の USBX モジュールの名前。
Show Deprecation Warning	Enabled, Disabled Default: Enabled	非推奨警告の表示の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

sf_el_ux 上の USBX ポート HS および FS の構成設定

ISDE Property	Value	説明
VBUSEN Pin Signal Logic	Active Low, Active High Default: Active High	VBUSEN ピンの信号ロジックの選択。
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ハイスピードの割り込み優先順位の選択。

ISDE Property	Value	説明
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フルスピードの割り込み優先順位の選択。
Name	Default: g_sf_el_ux	USBX ポート HS および FS の名前。
Show deprecation warning	Enabled, Disabled Default: Enabled	非推奨警告の表示の選択。

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

5.1.10.5 アプリケーションでのコンソールフレームワークモジュールの使用

シンプルなコンソールフレームワークを構築する際の重要な要素は、スタックの選択と構成、メニュー構造体の定義、メニューコマンドコールバックのコードの記述、ターゲットアプリケーション内で必要な CLI 関数を実装するための API コールの使用です。アプリケーションでコンソールフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) メニューおよびコマンド構造体を作成します。
2) 必要なコールバックを実装します。
3) open API を使用して SF_CONSOLE を初期化します。
4) プロンプト API を使用してプロンプトを生成し、コマンドを処理します。
5) 必要に応じて他の API（read、 write、 parse または argumentFind）を使用してコマンドを処理します。
6) 必要に応じて close API を使用してモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

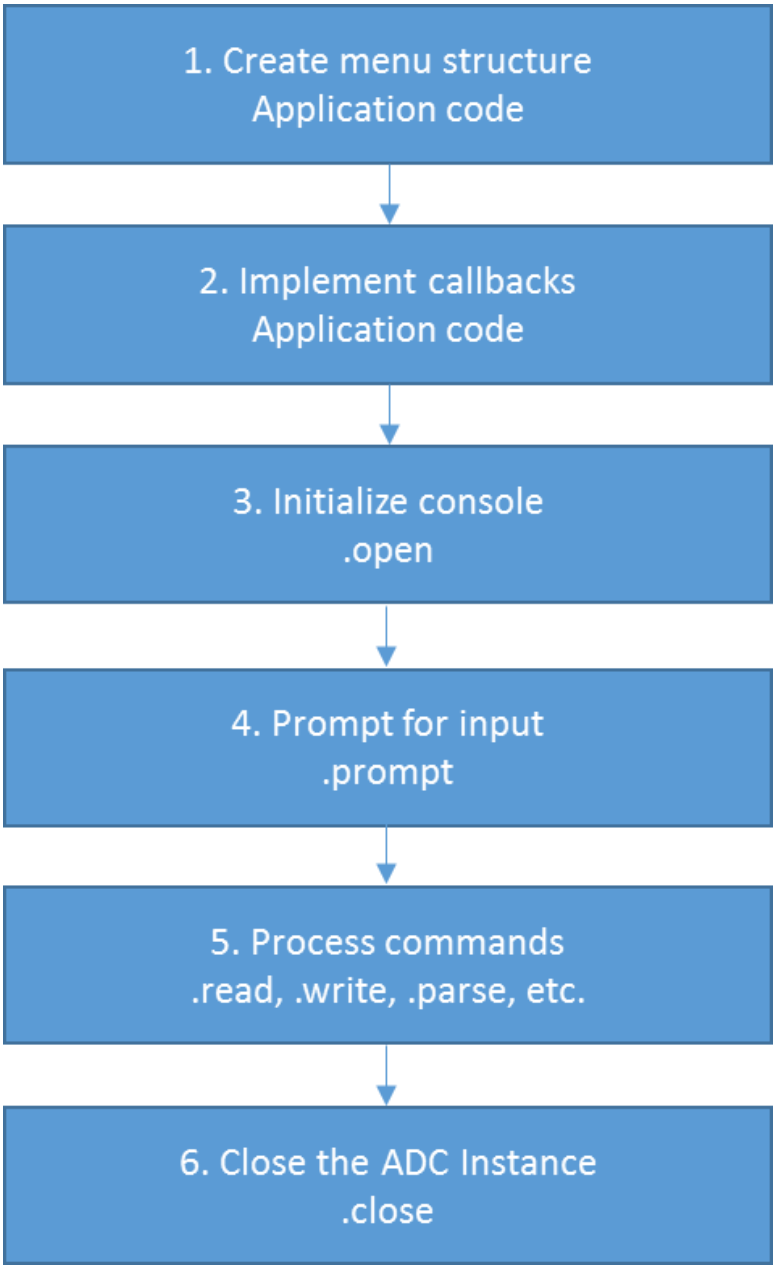


図 173: 一般的なコンソールフレームワークモジュールアプリケーション

5.1.11 暗号化フレームワーク

- SF CRYPTO フレームワーク
  - SF CRYPTO フレームワークは、基礎となるセキュア暗号化エンジンを開きます。

  - 他の暗号化フレームワークモジュール SF_CRYPTO_TRNG, SF_CRYPTO_HASH、SF_CRYPTO_KEY. の共有リソースにアクセスするためのサービスを提供します。
- SF CRYPTO TRNG フレームワーク
  - SF CRYPTO TRNG フレームワークモジュールは、基礎となるセキュア暗号化エンジンへの TRNG HAL インタフェースを使用します。
  - SF CRYPTO フレームワークモジュールを通じて共有リソースにアクセスします。
- SF CRYPTO HASH フレームワーク
  - SF CRYPTO HASH フレームワークは、基礎となるセキュア暗号化エンジンを利用して HASH サービスを提供します。
- SF CRYPTO KEY フレームワークモジュールの特長
  - RSA 2048 ビット、 1024 ビットプレーンテキストまたは未加工キー。
  - RSA 2048 ビット、 1024 ビットラップキー。
  - ECB、CBC、CTR および GCM チェーンモード用の AES 128 ビット、192 ビットおよび 256 ビットラップキー。
  - XTS チェーンモード用の AES 128 ビットおよび 256 ビットラップキー。
  - ラップキーは、暗号化キー、キーハンドル、ラップキーなどの異なる名前で参照されることがあります。 Synergy プラットフォームでのキーのラップは安全とみなされます。これらのキーはプラットフォームの外部で使用できないためです。

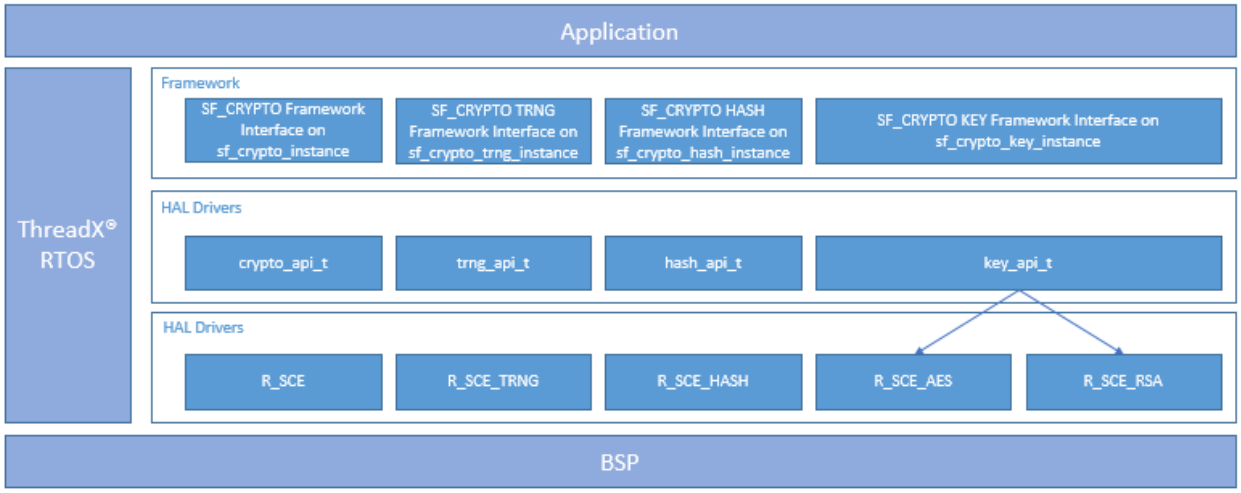


図 174: 暗号化フレームワークモジュールの編成、オプション、スタックの実装

5.1.11.1 暗号化フレームワークモジュール API の概要

暗号化フレームワークモジュールは、コールバックの登録、周期レポートの開始、周期レポートの停止、レポートの取得、レポートの設定に関する API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

暗号化フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_crypto0.p_api->open(g_sf_crypto0.p_ctrl, g_sf_crypto0.p_cfg); この関数を使用して、暗号化フレームワークモジュールと r_sce HAL モジュールを開きます。
.close	g_sf_crypto0.p_api->close(g_sf_crypto0.p_ctrl); この関数を使用して、暗号化フレームワークモジュールと r_sce HAL モジュールを閉じます。
.lock	g_sf_crypto0.p_api->lock(g_sf_crypto0.p_ctrl); この関数は、暗号操作用の共有リソースをロックします。
.unlock	g_sf_crypto0.p_api->unlock(g_sf_crypto0.p_ctrl, &p_status); この関数は、暗号操作のステータスを取得し、それを p_status ポインタに格納します。
.getStatus	g_sf_crypto0.p_api->unlock(g_sf_crypto0.p_ctrl); この関数は、暗号操作用の共有リソースをロック解除します。
.versionGet	g_sf_crypto0.p_api->versionGet(&version); この関数は、バージョンポインタを使用して API バージョンを取得します。

暗号化フレームワーク HASH モジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_crypto_hash0.p_api->open(g_sf_crypto_hash0.p_ctrl, g_sf_crypto_hash0.p_cfg); この関数を使用して、暗号化ハッシュフレームワークモジュールを開きます。
.close	g_sf_crypto_hash0.p_api->close(g_sf_crypto_hash0.p_ctrl); この関数を使用して、暗号化ハッシュフレームワークモジュールを閉じます。
.hashInit	g_sf_crypto_hash0.p_api->hashInit(g_sf_crypto_hash0.p_ctrl); この関数は、バージョンポインタを使用して API バージョンを取得します。
.hashUpdate	g_sf_crypto_hash0.p_api->hashUpdate(g_sf_crypto_hash0.p_ctrl, &p_data_in); この関数は、p_data_in ポインタが指すデータに対してハッシュステップを実行します。
.hashFinal	g_sf_crypto_hash0.p_api->hashUpdate(g_sf_crypto_hash0.p_ctrl, &p_data_in, &p_size); この関数は、p_data_in ポインタが指すデータに対して最後のハッシュステップを実行し、出力サイズを p_size ポインタで定義された 32 ビットワードに格納します。
.versionGet	g_sf_crypto_hash0.p_api->versionGet(&version); この関数は、バージョンポインタを使用して API バージョンを取得します。

暗号化フレームワーク TRNG モジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_crypto_trng0.p_api->open(g_sf_crypto_trng0.p_ctrl, g_sf_crypto_trng0.p_cfg); この関数を使用して、暗号化 TRNG フレームワークモジュールを開きます。
.close	g_sf_crypto_trng0.p_api->close(g_sf_crypto_trng0.p_ctrl); この関数を使用して、暗号化 TRNG フレームワークモジュールを閉じます。
.randomNumberGenerate	g_sf_crypto_trng0.p_api-> randomNumberGenerate (&p_buffer); この関数は、乱数を生成し、それを p_buffer ポインタで定義されたアドレスから開始するメモリに格納します。
.versionGet	g_sf_crypto_trng0.p_api->versionGet(&version); この関数は、バージョンポインタを使用して API バージョンを取得します。

暗号化フレームワーク KEY モジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_crypto_key0.p_api->open(g_sf_crypto_key0.p_ctrl, g_sf_crypto_key0.p_cfg); この関数を使用して、暗号化キーフレームワークモジュールを開きます。
.keyGenerate	g_sf_crypto_key0.p_api->keyGenerate(g_sf_crypto_key0.p_ctrl, &p_secret_key, &p_public_key); この関数を使用して、p_secret_key および p_public_key ポインタを使用してキーを生成します。生成されたキーは、p_sectret_key の場所に格納されます。

Function Name	API の呼び出し例と説明
.close	g_sf_crypto_key0.p_api->close(g_sf_crypto_key0.p_ctrl); この関数を使用して、暗号化キーフレームワークモジュールと r_sce HAL モジュールを閉じます。
.versionGet	g_sf_crypto_key0.p_api->versionGet(&version); この関数は、バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	操作が正常に終了しました。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.11.2 暗号化フレームワークモジュールの動作の概要

SF CRYPTO フレームワークは高レベルの API で、 sf_crypto. に実装されます。 SF CRYPTO フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じてフレームワーク暗号化サービスの基盤を提供します。

次の注意事項は、SF_CRYPTO, SF_CRYPTO_TRNG, SF_CRYPTO_HASH、SF_CRYPTO_KEY モジュールに適用されます。

SF CRYPTO フレームワークサービス

SF CRYPTO フレームワークモジュールには、真性乱数生成用の SF_CRYPTO_TRNG、メッセージダイジェスト生成用の SF_CRYPTO_HASH、 RSA および AES キー生成サービス用の SF_CRYPTO_KEY が含まれます。

暗号化、復号、署名生成、署名検証用のフレームワークサービスは、今後の SSP バージョンで提供される予定です。現時点では、 HAL API を直接使用する必要があります。

エンディアン構成パラメータ

- Synergy コンフィギュレータでは、暗号化フレームワークモジュールなしで暗号化 HAL ドライバーのみが選択されている場合、エンディアンは構成可能なパラメータであり、デフォルトで *ビッグエンディアン*に設定されます。これは、以前のリリースでサポートされていたモードです。

- Synergy コンフィギュレータでは、暗号化フレームワークモジュールが選択されている場合（対応する HAL コンポーネントが自動的に含まれます）、エンディアンフラグは、バイト配列フォーマットのみがサポートされるリトルエンディアンモードにロックされます。
- 暗号化フレームワークレイヤ API では、入力および出力データに対してバイト配列のみがサポートされます。
- 暗号化 HAL API では、入力および出力データに対してワード /32 ビット (uint32_t) 配列がサポートされます。
- バイト配列フォーマットのデータを持つユーザーは、暗号化フレームワークモジュールを使用する必要があります。
- 対応するフレームワーク API のない暗号化 HAL API を使用している既存のプロジェクトでは、ビッグエンディアンデータが使用されている可能性があり、その場合は、リトルエンディアンに変換してフレームワーク API を使用するか、引き続き既存の HAL API を直接使用する必要があります。
- HAL API を直接使用する場合、ユーザーは (uint32_t) としてキャストされたデータが、使用されている HAL モジュールのエンディアンと一致することを確認する必要があります。

データバッファのアラインメント

- 入力バッファに渡すために割り当てられたすべてのデータバッファは、ワード境界でアラインする必要があります。

VersionGet API

- この API は、モジュールが開かれる前であっても、いつでも呼び出すことができます。

SF CRYPTO TRNG フレームワークモジュールの概要説明

SF CRYPTO TRNG フレームワークは高レベルの API で、sf_crypto_trng. に実装されます。SF CRYPTO TRNG フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じて真性乱数生成サービスを提供します。

SF CRYPTO HASH フレームワークモジュールの概要説明

SF CRYPTO HASH フレームワークは高レベルの API で、sf_crypto_hash. に実装されます。SF CRYPTO HASH フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じてハッシュおよびメッセージダイジェストサービスを提供します。

サポートされるハッシュ関数は、 SHA-1、 SHA-224、 SHA-256 です。

FIPS セキュアハッシュ標準から：

すべてのアルゴリズムは反復的な一方向ハッシュ関数で、メッセージを処理して*メッセージダイジェスト*と呼ばれる凝縮表現を生成できます。これらのアルゴリズムより、メッセージの整合性を判定できます。メッセージに変更を加えると、非常に高い確率で、異なるメッセージダイジェストが生成されます。この特性はデジタル署名とメッセージ認証コードの生成および検証と、乱数またはビットの生成に役立ちます。

サポートされるハッシュ関数については、メッセージサイズを $<2^{64}$ ビットにする必要があります。

メッセージダイジェストサイズは次のとおりです。

SHA-1: 20 バイト

SHA-224: 28 バイト

SHA-256: 32 バイト

SF CRYPTO KEY フレームワークモジュールの概要説明

SF CRYPTO KEY フレームワークは高レベルの API で、sf_crypto_key. に実装されます。SF CRYPTO KEY フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じて暗号キー生成サービスを提供します。

キーのフォーマット

keyGenerate API によって生成される RSA キーのフォーマットは次のとおりです。

- RSA 公開キー
  - バイト 0 ～バイト 3: 公開キー指数
  - バイト 4: RSA modulus(128 bytes for RSA 1024-bit and 256 bytes for RSA 2048-bit keys) の開始
- 標準フォーマットの RSA 秘密キー
  - バイト 0: 秘密キー指数 （RSA 1024 ビットキーの場合は 128 バイト、 RSA 2048 ビットキーの場合は 256 バイト）
  - RSA 係数が続きます。（RSA 1024 ビットキーの場合は 128 バイト、 RSA 2048 ビットキーの場合は 256 バイト）
- CRT formatTheのRSA秘密キー。コンポーネントはバイト0をexponent2として次の順序で並べられます。
  - exponent2 // 第 2 係数の CRT 指数、正の整数
  - prime2 // 第 2 係数、正の整数
  - exponent1 // 第 1 係数の CRT 指数、正の整数
  - prime1 // 第 1 係数、正の整数
  - coefficient // （第 1） CRT 係数、正の整数

暗号化フレームワークモジュールの動作に関する重要な注意事項と制限事項

- ロック API とロック解除 API は、SF_CRYPTO_XXX モジュールで使用するためのものです。ただし、アプリケーションがフレームワークレイヤでまだサポートされていない HAL API を直接呼び出す場合は、HAL モジュールを呼び出す直前にロック API を使用します。ロック解除 API は、HAL モジュールの呼び出しから戻った直後に使用します。
- statusGet API は、入力パラメータとして制御ブロック /p_ctrl を必要とします。このため、SF_CRYPTO モジュールの open API が呼び出された後にのみ呼び出す必要があります。
- すべての暗号化フレームワーク API はブロッキング呼び出しです。

このモジュールの制限事項については、 SSP の最新のリリースノートを参照してください。

5.1.11.3 アプリケーションへの暗号化フレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに暗号化フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない*

場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

暗号化フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（次の表に示すように、暗号化フレームワークモジュールのデフォルトの名前は g_ux_host_class_hid0 です。この名前は、対応する [Properties] ウィンドウで変更できます。）

暗号化フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_crypt0 Crypto Framework on sf_crypto	Threads	New Stack> Framework> Crypto > Crypto Framework on sf_crypto
g_sf_crypt_key0 Crypto Key Framework on sf_crypto_key	Threads	New Stack> Framework> Crypto > Crypto Key Framework on sf_crypto_key
g_sf_crypt0 Crypto TRNG Framework on sf_crypto_trng	Threads	New Stack> Framework> Crypto > Crypto TRNG Framework
g_sf_crypt0 Crypto HASH Framework on sf_crypto_hash	Threads	New Stack> Framework> Crypto > Crypto HASH Framework

***注:** 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

次の図に示すように暗号化フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、複数のスタックで使用できるため 1 回のみ追加する必要があります。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、このテキストを含むブロックで示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に 「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

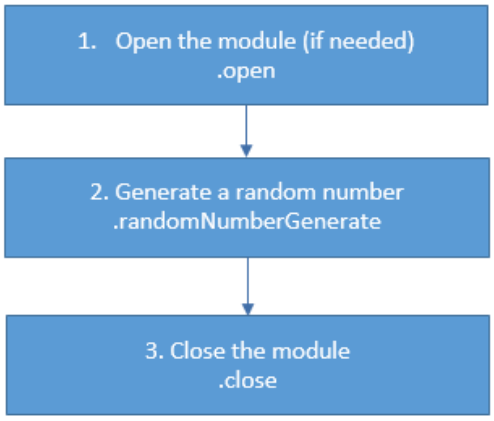


図 175: 暗号化フレームワークモジュールスタック - 暗号化、ハッシュ、キー、 TRNG

5.1.11.4 暗号化フレームワークモジュールの構成

暗号化フレームワークモジュールには、操作パラメータの定義に使用される構成可能なプロパティがあります。

注: *ISDE を開き、暗号化フレームワークモジュールを作成し、次に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、 SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

SF CRYPTO フレームワークモジュールの構成設定

Configuration parameter	説明
uint32_t wait_option	RTOS サービス呼び出しの待機オプション 使用可能なメモリが十分にない場合のサービスの動作を定義します。待機オプションは次のように定義されます。 TX_NO_WAIT (0x00000000) T X_WAIT_FOREVER (0xFFFFFFFF) タイムアウト値（0x00000001 ～ 0xFFFFFFFE） TX_NO_WAIT を選択すると、成功したかどうかにかかわらず、このサービスからすぐに戻ります。これは、サービスが初期化から呼び出された場合に有効な唯一のオプションです。TX_WAIT_FOREVER を選択すると、十分なメモリが使用可能になるまで呼び出し側のスレッドが無限にサスペンドされます。 数値（1 ～ 0xFFFFFFFE）を選択すると、メモリの待機中にサスペンド状態になるタイマティックの最大数が指定されます。 ISDE を通じて設定されるデフォルトは TX_WAIT_FOREVER. です。
crypto_instance_t * p_lower_lvl_crypto	ローレベル暗号化エンジン HAL ドライバーインスタンスへのポインタ。 Synergy コンフィギュレータによって構成されます。
void const * p_extend	ハードウェア固有の設定値に対応するための拡張パラメータです。 Synergy コンフィギュレータによって構成されます。

Configuration parameter	説明
void const * p_context	ユーザーデータのプレースホルダー。 今後の拡張用です。
void * p_memory_pool	バイトプールアドレス。 作成されたSF_CRYPTO_XXXモジュールインスタンスの数に基づいてプールを割り当てる呼び出し側です。
uint32_t memory_pool_size	バイトプールサイズ。 ISDE によって設定されるデフォルトは 128 バイトです。 推奨サイズ： RSA キーが必要な場合は SF_CRYPTO_KEY モジュールのインスタンスあたり 20 バイト。 AES キーが必要な場合は SF_CRYPTO_KEY モジュールのインスタンスあたり 270 バイト。 AES キーが必要な場合は SF_CRYPTO_KEY モジュール。 SF_CRYPTO_HASH モジュールのインスタンスあたり 112 バイト。

Configuration parameter	説明
sf_crypto_close_option_t close_option	クローズオプション SCE モジュールを閉じる方法を選択します。 SF_CRYPTO_CLOSE_OPTION_DEFAULT - 他に開かれているSF_CRYPTO_XXXモジュールがない場合にのみモジュールを閉じます。 SF_CRYPTO_CLOSE_OPTION_FORCE_CLOSE - SF_CRYPTO_XXX モジュールのステータスに関係なくモジュールを閉じます。 SF_CRYPTO_CLOSE_OPTION_DEFAULT はデフォルト設定です。

sf_crypto の構成パラメータ

SF CRYPTO TRNG フレームワークモジュールの構成設定

Configuration parameter	説明
sf_crypto_instance_t * p_lower_lvl_common;	ローレベル暗号化エンジン HAL ドライバーインスタンスへのポインタ。 Synergy コンフィギュレータによって構成されます。
trng_instance_t * p_lower_lvl_instance;	TRNG HAL ドライバーインスタンスへのポインタ。 Synergy コンフィギュレータによって構成されます。
void * p_extend;	ハードウェア固有の設定値に対応するための拡張パラメータです。 今後使用される予定です。

sf_crypto_trng モジュールの構成パラメータ

構成は、 TRNG HAL ドライバーの ISDE を通じて設定されます。

Configuration parameter	説明
uint32_t num_attempts	真性乱数の生成を試行する最大回数。 デフォルトでは 2 に設定されます。

R_SCE_TRNG HAL モジュールの構成パラメータ

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SF CRYPTO HASH フレームワークモジュールの構成設定

sf_crypto_hash モジュールの構成パラメータ

Configuration parameter	説明
sf_crypto_hash_type_t hash_type;	HASH アルゴリズムタイプ。
crypto_instance_t * p_lower_lvl_crypto	ローレベル暗号化エンジン HAL ドライバーインスタンスへのポインタ。 Synergy コンフィギュレータによって構成されます。
hash_instance_t * p_lower_lvl_instance;	HASH 下位レベルモジュールインスタンスへのポインタ。 ISDE によって構成されます。
void * p_extend	ハードウェア固有の設定値に対応するための拡張パラメータ。オプションです。 Synergy コンフィギュレータによって構成されます。

SF CRYPTO KEY フレームワークモジュールの構成設定

sf_crypto_key モジュールの構成パラメータ

Configuration parameter	説明
sf_crypto_key_type_t key_type	生成するキータイプ

Configuration parameter	説明
sf_crypto_key_size_t key_size	生成するキーサイズ
sf_crypto_data_handle_t domain_params;	要求されたキータイプのドメインパラメータへのポインタ。 構造体には、データ（ドメインデータを含む）へのポインタとデータ長が含まれます。 RSA および AES キータイプの場合は NULL に設定します。
sf_crypto_instance_t * p_lower_lvl_crypto_common	暗号化フレームワーク共通インスタンスへのポインタ。 Synergy コンフィギュレータによって構成されます。
void const * p_extend	ハードウェア固有の設定値に対応するための拡張パラメータです（今後の使用のために予約済み）。 Synergy コンフィギュレータによって構成されます。

暗号化フレームワークモジュールのクロック構成

暗号化フレームワークモジュールは、SCE と同じクロックを使用します。

暗号化フレームワークモジュールのピン構成

CRYPTO フレームワークモジュールは外部ピンを使用しません。

5.1.11.5 アプリケーションでの暗号化フレームワークモジュールの使用

コンフィギュレータは、暗号化フレームワークモジュールの作成と登録に必要な処理を生成します。ただし、CRYPTO モジュールは他のモジュールを使用する前に開く必要があります。

重要な注意 :

- オートスタート設定が有効になっている場合は、open API を自動的に呼び出すことができます。
- リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- 暗号化フレームワークサービスと SCE を終了するには close API を使用します。

アプリケーションでの SF CRYPTO TRNG フレームワークモジュールの使用

SF CRYPTO TRNG フレームワークに関する注意 :

- 稀に RandomNumberGenerate API がエラーを返した場合は、すべての CRYPTO フレームワークモジュールを閉じ、 SF CRYPTO モジュールを再度開く必要があります。
- TRNG の 1 つのインスタンスのみが必要です。
- CRYPTO フレームワークの open API は、SF_CRYPTO_TRNG フレームワークを開く前に呼び出す必要があります。

アプリケーションで SF CRYPTO TRNG フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) オートスタートオプションが有効になっていない限り、TRNG open API を使用して SF_CRYPTO_TRNG および SCE TRNG HAL モジュール (R_SCE_TRNG) を初期化します。
2) randomNumberGenerate API を使用して乱数を生成します。要求できる乱数の最小の長さは 1 バイトです。
3) TRNG サービスが不要になった場合は、close API を使用して暗号化フレームワークサービスと SCE を終了します。

上記の手順を次のフローチャートに示します。

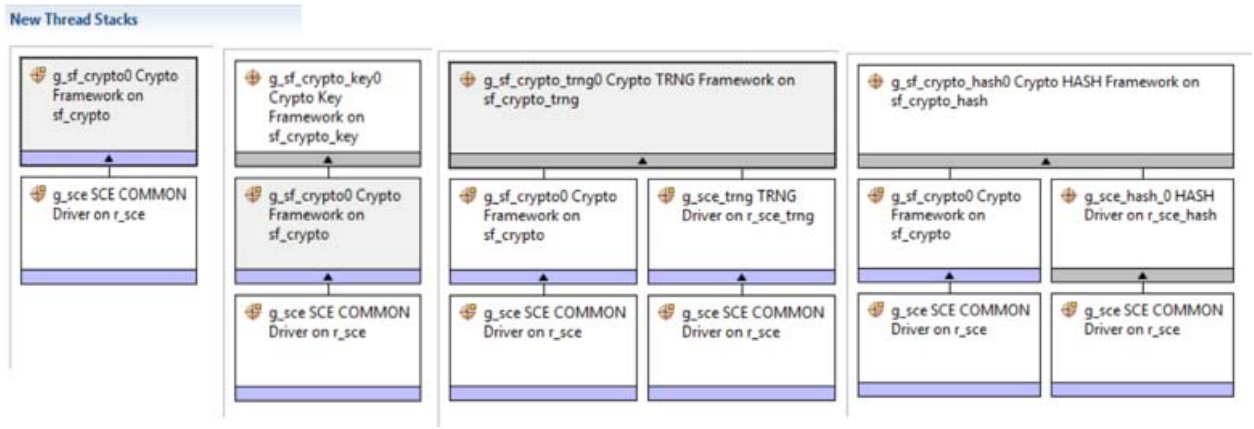


図 176: 暗号化 TRNG フレームワークの一般的なアプリケーションフロー

アプリケーションでの SF CRYPTO HASH フレームワークモジュールの使用

SF CRYPTO HASH フレームワークに関する注意 :

- リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。（モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。）
- オートスタートオプションが有効になっていない限り、最初に SF CRYPTO モジュールを開く必要があります。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。

アプリケーションで SF CRYPTO HASH フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) 自動初期化が有効になっていない限り、 open API を使用して、 SCE HASH ドライバーを通じて SF CRYPTO HASH フレームワークモジュールおよび SCE HASH HAL モジュール (R_SCE_HASH) を初期化します。
2) hashInit API を使用して、選択した HASH アルゴリズムの HASH 操作を初期化します。
   a) hashInit を open API、hashUpdate API、hashFinal API の後に呼び出して、構成した HASH アルゴリズムで新しい操作を初期化することができます。
3) hashUpdate API を使用して入力データをハッシュします。すべてのデータが完全に使い尽くされるまで、複数回呼び出すことができます。
4) hashFinal API を使用して、hashUpdate API でハッシュされたすべてのデータのメッセージダイジェストを取得します。
   a) hashFinal API が呼び出されたら、 hashInit を呼び出して、同じ HASH アルゴリズムを使用して次のデータセットの動作を初期化するか、 close API を呼び出してモジュールを閉じることができます。その後、別の HASH アルゴリズムについて再度開くことができます。
5) 暗号化 HASH フレームワークサービスを終了するには close API を使用します。

上記の手順を次のフローチャートに示します。

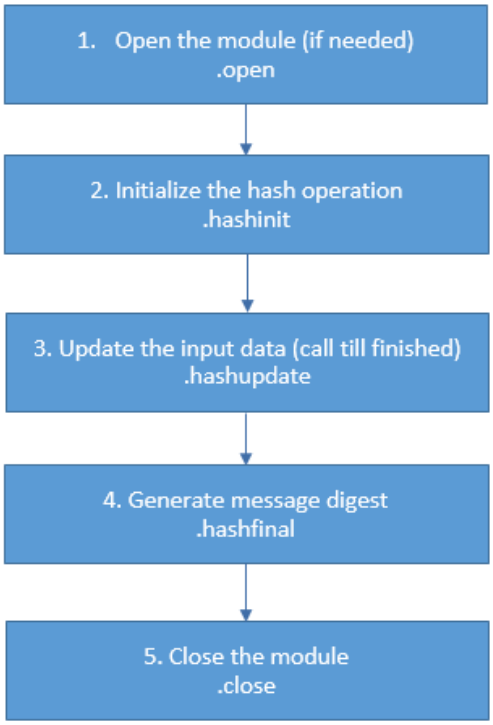


図 177: 暗号化 HASH フレームワークの一般的なアプリケーションフロー

アプリケーションでの SF CRYPTO KEY フレームワークモジュールの使用

SF CRYPTO KEY フレームワークに関する注意 :

- リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。（モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。）
- SF CRYPTO モジュールを最初に開く必要があります。これは、オートスタートオプションが有効になっている場合は自動的に行われます。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。

アプリケーションで SF CRYPTO KEY フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

- オートスタートオプションが有効になっていない限り、 open API を使用して SF CRYPTO KEY フレームワークモジュールを初期化します。
- keyGenerate API を使用して、 open 呼び出し時に構成パラメータによって設定されたタイプとサイズの暗号キーを生成します。
- 暗号化キーフレームワークモジュールサービスを終了するには close API を使用します。

上記の手順を次のフローチャートに示します。

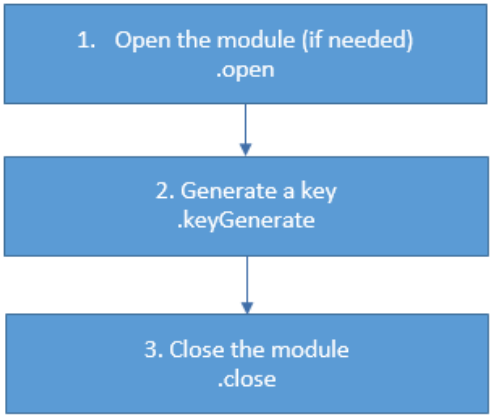


図 178: 暗号化キーフレームワークの一般的なアプリケーションフロー

5.1.12 静電容量式タッチフレームワーク

CTSU フレームワークインタフェースは、静電容量式タッチパネルのハードウェアスキャンを実行し、構成で指定された周期でパネルを更新するプライベートスレッドを作成します。以下の主要機能を提供します。

- 複数のアプリケーションレイヤがコールバックを登録できます。これにより、スライダやボタンなどの複数のウィジェットがこのレイヤを使用できます。
- 周期的な CTSU スキャンを自律的に実行し、ユーザーが指定したレートでプロセスを更新します。
- S124、 S128、 S3A6、 S3A7、 S5D5、 S5D9、 S7G2 をサポートします

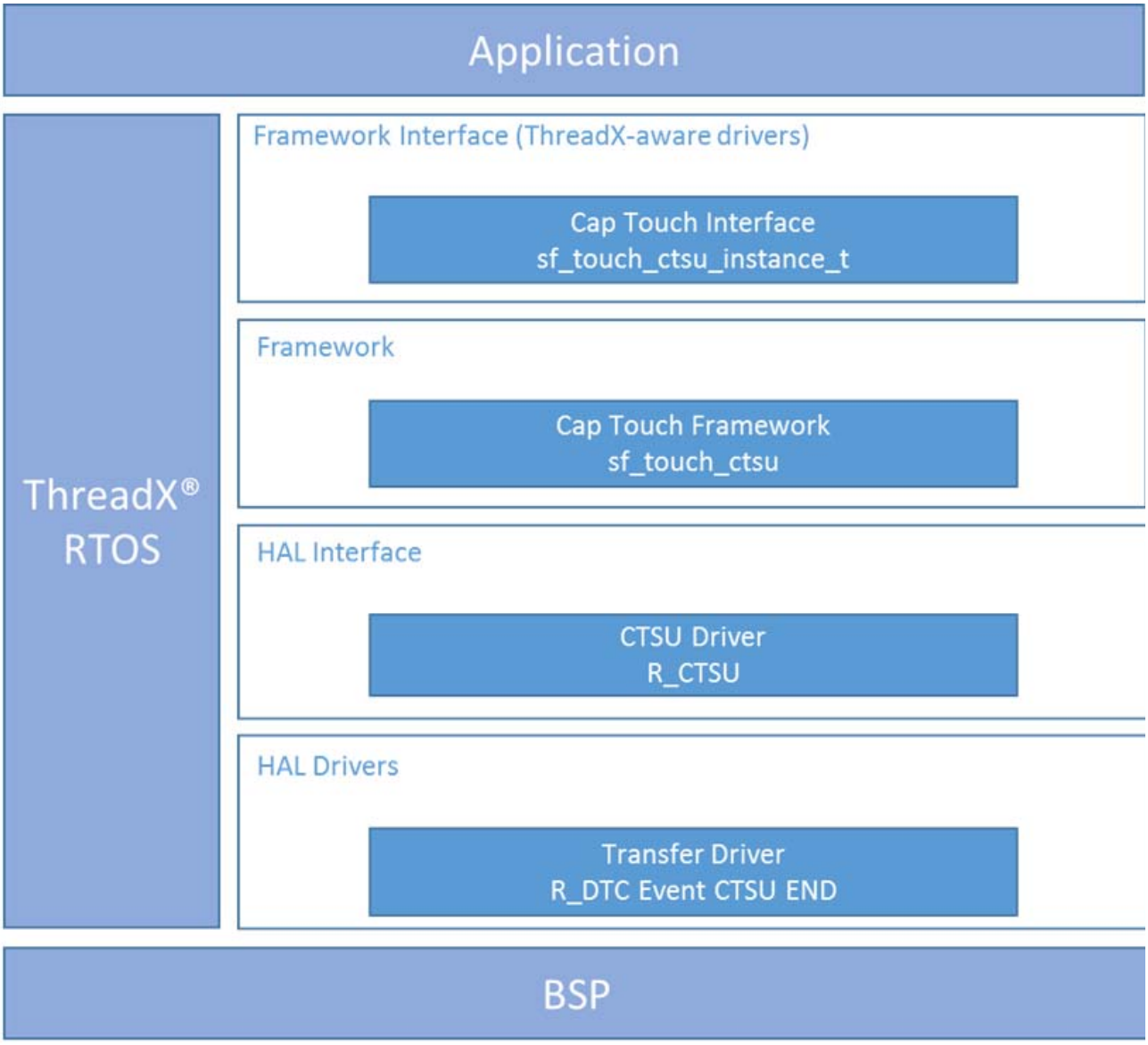


図 179: 静電容量式タッチの編成、オプション、スタックの実装

5.1.12.1 静電容量式タッチフレームワークモジュール API の概要

静電容量式タッチは、オープン、有効化、無効化、クローズの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

静電容量式タッチフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_touch_ctsu0.p_api->open(g_sf_touch_ctsu0.p_ctrl, g_sf_touch_ctsu0.p_cfg); タッチ CTSU ボタンフレームワークを初期化します。また低レベルハードウェアを構成し、すべてのボタンに対してコールバック関数を登録します。
read	g_sf_touch_ctsu0.p_api->read(g_sf_touch_ctsu0.p_ctrl, p_dest, read_options, channels, count); タッチ CTSU フレームワークからデータを読み取ります。
close	g_sf_touch_ctsu0.p_api->close(g_sf_touch_ctsu0.p_ctrl); タッチ CTSU フレームワークを閉じます。他のモジュールによって使用されていない場合は、CTSU フレームワークと下位レイヤを閉じます。
versionGet	g_sf_touch_ctsu0.p_api->versionGet(&p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	フレームワークが既に初期化されています。
SSP_ERR_NOT_OPEN	デバイスは開かれていません。
SSP_ERR_INTERNAL	内部エラー。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.12.2 静電容量式タッチフレームワークモジュールの動作の概要

CTSU フレームワークインタフェースは、構成された CTSU チャネルのハードウェアスキャンと更新を周期的に実行する内部スレッドを作成します。このフレームワークモジュールは、HAL レイヤ CTSU ドライバーを使用してスキャン結果を読み取ります。スキャンが完了すると、アプリケーションレイヤーによって登録されたコールバックが呼び出されます。複数の上位レイヤがこのフレームワークを使用している場合（ボタンやスライダ、ホイールなど）、このレイヤは初期化を行った順にこれら各レイヤのコールバックを呼び出します。サポートされるコールバック数は現在 3 に制限されています。

内部スレッド (update_hz) の周波数は、このフレームワークに実行される後続の各 open() 呼び出しで設定または変更されます。スキャンレートを設定する場合は、ハードウェアスキャンの完了と処理に必要な時間によってスキャンレートが制限されることに注意してください。スキャンの完了に必要な時間は、構成されているチャネルの数と、基礎となる r_ctsu レイヤが構成されているモードによって異なります。詳細については、Renesas Synergy ウェブサイトにある静電容量式タッチのアプリケーションノートを参照してください。

静電容量式タッチフレームワークモジュールの動作に関する注意事項

静電容量式タッチフレームワークの構成情報は、[SSP Utilities] タブで Renesas Synergy Gallery からダウンロードできる Capacitive Touch Workbench for Renesas Synergy ツールによって生成されます。

静電容量式タッチフレームワークモジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.12.3 アプリケーションへの静電容量式タッチフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチをアプリケーションに組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

静電容量式タッチフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（静電容量式タッチのデフォルト名は g_sf_touch_button0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

静電容量式タッチフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_ctsu0 Capacitive Touch Framework on sf_touch_ctsu	**Threads**	**New Stack > Framework > Input > Capacitive Touch Framework on sf_touch_ctsu**

次の図に示すように sf_touch_ctsu の静電容量式タッチフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

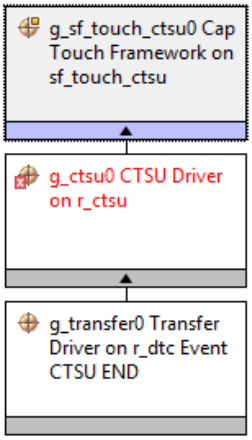


図 180: 静電容量式タッチフレームワークモジュールスタック

5.1.12.4 静電容量式タッチフレームワークモジュールの構成

必要な動作に合わせて静電容量式タッチモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

sf_touch_ctsu 上の静電容量式タッチフレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効化または無効化します。
Name	g_sf_touch_ctsu0	モジュール名。
Thread Priority	3	システムの他のスレッドのプライオリティに基づいてユーザーが決定します。高いプライオリティを使用することを推奨します。
Update Hz	50	CTSU ハードウェアスキャンを実行するレート。最大スキャンレートは、ThreadX のティックレート設定値未満にする必要があります。ハードウェアスキャンを実行する合計時間は、使用するチャネル数によって決定されます。各チャネルは、スキャンを完了するのに約 600µsec かかります。そのため、10 個のチャネルがセルフキャパシタンスモードで 10 個のボタンに使用される場合は、合計ハードウェアスキャン時間は最大 6 ミリ秒または 166 Hz となります。同様に、7 個のチャネルが相互容量モード（5x2 レイアウト）で 5 個のボタンに使用される場合は、合計ハードウェアスキャン時間は最大 4.2 ミリ秒または 250Hz となります。これにソフトウェア処理時間が加わり、その長さはコアクロックと CTSU によって使用されるソフトウェアのフィルタ深さに応じて変わります。したがって最大スキャンレートは、構成内で使用されるすべてのチャネルのスキャンと処理に必要な時間より小さくし、RTOS ティックレート未満とする必要があります。

ISDE Property	Value	説明
Callback	NULL	各スキャンが完了したときに呼び出されるユーザーコールバックの名前。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する* **[PROPERTIES]** *ウィンドウを調べることで決定されます。*

静電容量式タッチフレームワークの下位レベルモジュールの構成設定

通常は、ローレベルドライバーの少数の設定のみをデフォルトから変更する必要があり、これらは [Thread Stack] ブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Offset Adjustment	Enabled, Disabled Default: Enabled	オフセット調整を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。(レートは、*センサー値調整のランタイムレート*により決定されます。)

ISDE Property	Value	説明
Drift Compensation	Enabled, Disabled Default: Enabled	ドリフト補正を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。ドリフト補正により、ボードやサーフェスの老化あるいは温度や環境の変化に応じて、タッチの存在の有無を決定するベースライン値が経時的に変化するよう設定できます。ドリフト補正レートは、*定常状態のドリフト補正レート*および*スタートアップドリフト補正レート*、*チャネル解放補正レート*により決定されます。
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	ドリフト補正のアルゴリズムを提供します。現在サポートされているのは Alternate 1 のみです。その他のオプションは、今後のリリースでサポートされる可能性があります。ドリフト補正方法 Alternate 1 を使うと、定常状態、スタートアップ、チャネル解放の各イベントに対し、異なるドリフト補正レートを用いることができます。
Steady state drift compensation rate, drift compensation will be applied per n scans	500	チャネルが定常状態にある場合のドリフト補正レート（N スキャンごとに適用）を決定します。（スキャンレートに依存します。）
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	ドライバーの初期化実行時におけるドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*よりも低く設定する必要があります。）
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	チャネルが押された状態から開放された状態に移行する際のドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*以下に設定する必要があります。）

ISDE Property	Value	説明
Default filter depth (used in sensor count filter provided by driver)	1	ドライバーが提供するソフトウェアセンサーカウントフィルターで用いられます。このデフォルトフィルターは、比較的一般的な「漏れ積分回路」ソフトウェアフィルターを改良したもので、深さ 4 は約 16 のフィルター定数と同等となります。入力が 16 以上の定数値である場合、フィルター出力はフィルターの 16 サイクル / 繰り返し後に定数入力値の 68 ～ 69% に達します。
Runtime rate of tuning of sensor values (if drift compensation is used this value is overridden to be twice the rate of the steady state drift compensation)	800	オフセット調整のレート。（ドリフト補正を使用する場合、この値は定常状態ドリフト補正レートの 2 倍に設定されます。）
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	すべてのチャネルがタッチされていない場合にのみ、自動調整とドリフト補正を実行します。
Max. active channels	1	アプリケーションによって使用されるチャネルの最大数を指定します。 Self Capacitance Mode では、使用されるチャネル数を指します。 Mutual Capacitance Mode では、Rx チャネルと Tx チャネルの乗算の結果です。たとえば、Rx チャネル 4 つと Tx チャネル 3 つの場合、最大アクティブチャネル数は 12 です。
Name	g_ctsu0	モジュール名。
CTSU configuration used	g_ctsu0_config	CTWによって生成された構成構造体の名前。
Callback	NULL	スキャンが完了するたびに呼び出されるユーザーコールバック関数。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。
Data Processing Option	Default Processing (recommended), No Processing (tuning only) Default: Default Processing	完了した前回のデータ後にスキャンが開始されるかどうかや、自動較正がスキャンの間に実行されるかどうかを指定するために使用できます。調整を行う場合以外はデフォルト処理を使用してください。

ISDE Property	Value	説明
Write Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ライト割り込み優先順位の選択。
Read Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	リード割り込み優先順位の選択。
End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	終了割り込み優先順位の選択。

r_dtc EVENT CTSU END での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table

ISDE Property	Value	説明
Name	g_transfer0	モジュール名
Mode	Block	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode)	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	1	転送回数の選択
Number of Blocks (Valid only in Block Mode)	1	ブロック数の選択
Activation Source (Must enable IRQ)	Event CTSU END	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

***注:** 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

静電容量式タッチフレームワークモジュールのクロック構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。 CTSU クロック速度を、静電容量式タッチ調整ツール Capacitive Touch Workbench for Renesas Synergy（CTW）で選択したクロック速度と同じ値に設定します。

静電容量式タッチフレームワークモジュールのピン構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

***注:** 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

CTSU のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
CTSU	**Pins**	Select **Peripherals > Input: CTSU > CTSU0**

CTSU のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Enabled (Default: Disabled)	CTSU の動作モードとして Enabled を選択します
TS00 to TS11	None, Pn, Pm (Default: None)	TS ピン
TSCAP	None, Pn, Pm (Default: None)	TSCAP ピン

***注:** 前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.12.5 アプリケーションでの静電容量式タッチフレームワークモジュールの使用

アプリケーションで静電容量式タッチフレームワークモジュールを使用する際の一般的な手順では、最初に次の操作を行います。

1) ISDE を通じてアプリケーションにフレームワークを追加します。
2) 内部スレッドがチャネルスキャンを開始するレートを構成します。
3) スキャンが完了するたびに呼び出す必要のあるアプリケーションコールバックを指定します。

これらの手順が完了したら、必要に応じてモジュールの API にアクセスできます。

1) open API を使用して CTSU タッチフレームワークを初期化します。
2) コールバックによって新しいスキャンの可用性が通知されたら、read API を使用してスキャンしたデータを読み取ります。
3) close API を使用して CTSU タッチフレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

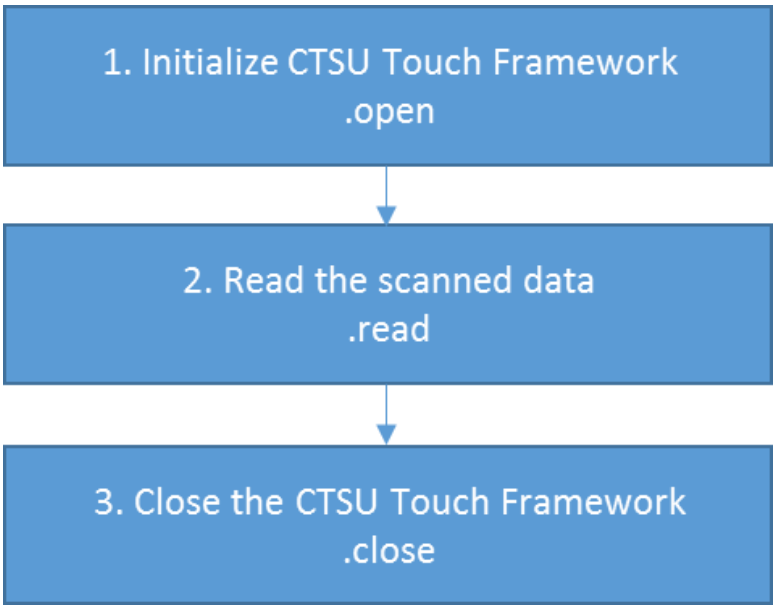


図 181: 通常の静電容量式タッチフレームワークモジュールアプリケーションのフロー図

5.1.13 静電容量式タッチボタンフレームワーク

静電容量式タッチボタンフレームワークモジュールは、システムに存在するすべてのボタンの CTSU データを解釈するために使用されます。主な特長をいくつか以下に示します。

- 構成データを生成する Capacitive Touch Workbench for Renesas Synergy （CTW）ツールと連携します。
  - S124、 S128、 S3A6、 S3A7、 S5D5、 S5D9、 S7G2 をサポートします
- コールバック関数をイベントに提供します。
  - デバウンシングを実行します
  - Press、 Release、 LongTouch などの複数のタイプのイベントをサポートします
  - 各ボタンのコールバックをボタン構成テーブルに列挙された順に呼び出します。

静電容量式タッチボタンフレームワークには静電容量式タッチフレームワークが必要です。ほとんどの場合、必要な構成情報はモジュールに自動的に追加されます。ISDE 構成オプションについては、このドキュメントで詳しく説明します。

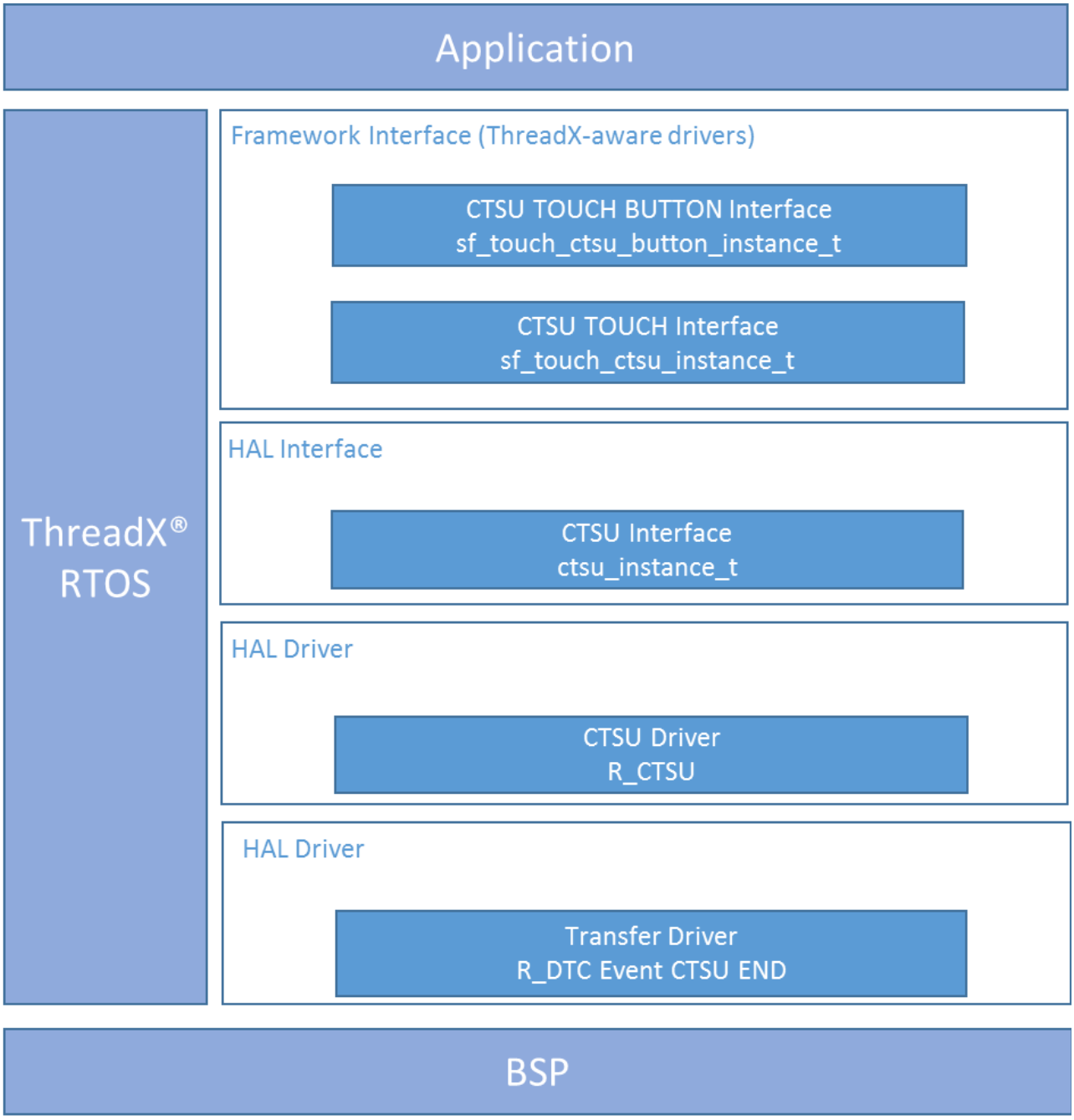


図 182:　静電容量式タッチボタンの編成、オプション、スタックの実装

5.1.13.1 静電容量式タッチボタンフレームワークモジュール API の概要

静電容量式タッチボタンは、オープン、有効化、無効化、クローズの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

静電容量式タッチボタンフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_touch_button0.p_api->open(g_sf_touch_button0.p_ctrl, g_sf_touch_button0.p_cfg); タッチ CTSU ボタンフレームワークを初期化します。また低レベルハードウェアを構成し、すべてのボタンに対してコールバック関数を登録します。
enable	g_sf_touch_button0.p_api->enable(g_sf_touch_button0.p_ctrl, button_id); 構成されたボタンのコールバック通知を有効化します。
disable	g_sf_touch_button0.p_api->disable(g_sf_touch_button0.p_ctrl, button_id); 構成されたボタンのコールバック通知を無効化します。
close	g_sf_touch_button0.p_api->open(g_sf_touch_button0.p_ctrl); タッチ CTSU ボタンフレームワークを閉じます。他のモジュールによって使用されていない場合は、ボタンフレームワークと下位レイヤーを閉じます。
versionGet	g_sf_touch_button0.p_api->versionGet(&p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

***注:** 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	フレームワークはこの特定のウィジェットによって既に初期化されています。
SSP_ERR_NOT_OPEN	デバイスは開かれていません。
SSP_ERR_INTERNAL	内部エラー。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.13.2 静電容量式タッチボタンフレームワークモジュールの動作の概要

静電容量式タッチボタンフレームワークは、システムに存在するすべてのボタンの CTSU データを解釈するために使用されます。CTSU フレームワークレイヤも初期化し、さらにハードウェアレイヤを初期化します。静電容量式タッチボタンフレームワークには、処理されたデータが使用可能になるたびに CTSU フレームワークレイヤを伴って呼び出されるコールバックがあります。静電容量式タッチボタンフレームワークはこの処理済みのデータ（バイナリ）を使用してデバウンシングを実行し、構成されたイベント（Press、Release、LongTouch など）のいずれが各ボタンに対して有効かを判断します。

フレームワークは、各ボタンのコールバックをボタン構成テーブルに列挙された順に呼び出します。ユーザーがスキャンプロセスを停止しない限り、引き続きタイマによってスキャンがトリガされ、データがユーザーバッファに書き込まれます。ユーザーバッファはフレームワークによって循環バッファとして扱われます。バッファの名前と長さは ISDE コンフィギュレータによって指定します。

静電容量式タッチボタンフレームワークモジュールの動作に関する注意事項

このフレームワークは、[SSP Utilities] タブで Renesas Synergy Gallery からダウンロードできる Capacitive Touch Workbench for Renesas Synergy ツールによって生成される構成データとともに使用するように設計されています。

静電容量式タッチボタンフレームワークモジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.13.3 アプリケーションへの静電容量式タッチボタンフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチボタンをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない*

場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

静電容量式タッチボタンをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。(静電容量式タッチボタンのデフォルト名は g_sf_touch_button0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。)

静電容量式タッチボタンフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_button0 Capacitive Touch Button Framework on sf_touch_ctsu_button	**Threads**	**New Stack > Framework > Input > Capacitive Touch Button Framework on sf_touch_ctsu_button**

次の図に示すように sf_touch_ctsu_button の静電容量式タッチボタンフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションまたは推奨であり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

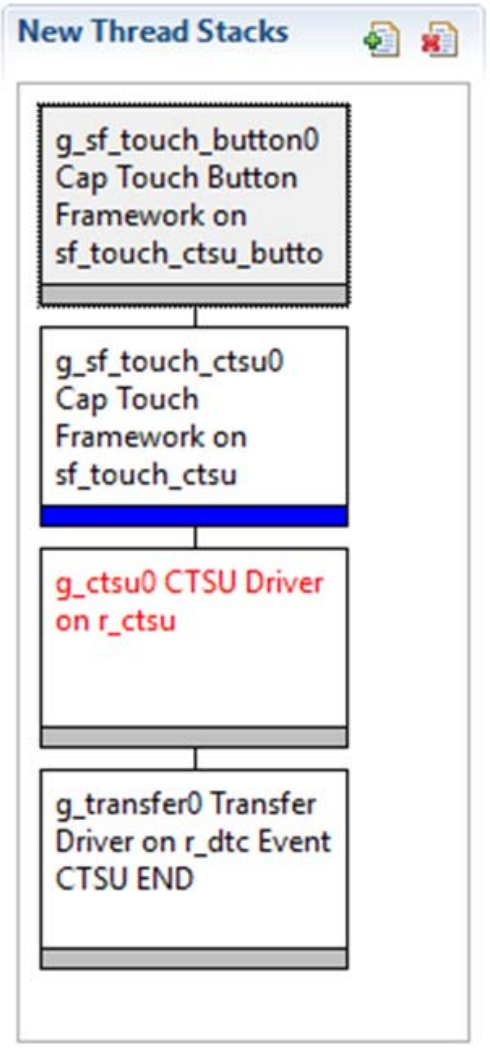


図 183: 静電容量式タッチボタンフレームワークモジュールのスタック

5.1.13.4 静電容量式タッチボタンフレームワークモジュールの構成

必要な動作に合わせて静電容量式タッチボタンモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_touch_ctsu_button 上の静電容量式タッチボタンフレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
Number of Buttons	1	CTW で構成されたボタンの数
Short hold debounce multiplier	1	ボタンの短押しイベントの乗数を指定します
Long hold debouce multiplier	5	ボタンの長押しイベントの乗数を指定します
Stuck in debounce multiplier	10	debounce 関数で使用される乗数
Multi touch enable	Enabled, Disabled Default: Disabled	マルチタッチ検出を有効化/無効化します
Enable stuck at condition detection	Enabled, Disabled Default: Disabled	ボタンのスタックイベントの乗数を指定します
Name	g_sf_touch_button0	モジュール名
Button Configuration Structure Name	touch_buttons	CTWによって生成されたボタン構成構造体の名前
Callback	g_button_framework_user_callback	ユーザーアプリケーションによって作成されたコールバック関数の名前。

注：上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

静電容量式タッチボタンフレームワークの下位レベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

sf_touch_ctsu 上の静電容量式タッチフレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効化または無効化します。
Name	g_sf_touch_ctsu0	モジュール名。
Thread Priority	3	システムの他のスレッドのプライオリティに基づいてユーザーが決定します。高いプライオリティを使用することを推奨します。

ISDE Property	Value	説明
Update Hz	50	CTSU ハードウェアスキャンを実行するレート。最大スキャンレートは、ThreadX のティックレート設定値未満にする必要があります。ハードウェアスキャンを実行する合計時間は、使用するチャネル数によって決定されます。各チャネルは、スキャンを完了するのに約 600 usec かかります。そのため、10 個のチャネルがセルフキャパシタンスモードで 10 個のボタンに使用される場合は、合計ハードウェアスキャン時間は最大 6 ミリ秒または 166 Hz となります。同様に、7 個のチャネルが相互容量モードで 5 個のボタンに使用される場合は （5x2 レイアウト）、合計ハードウェアスキャン時間は、最大 4.2 ミリ秒または 250Hz になります。これにソフトウェア処理時間が加わり、その長さはコアクロックと CTSU によって使用されるソフトウェアのフィルタ深さに応じて変わります。したがって最大スキャンレートは、構成内で使用されるすべてのチャネルのスキャンと処理に必要な時間より小さくし、RTOS ティックレート未満とする必要があります。
Callback	NULL	各スキャンが完了したときに呼び出されるユーザーコールバックの名前。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。

ISDE Property	Value	説明
Offset Adjustment	Enabled, Disabled Default: Enabled	オフセット調整を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。（レートは、*センサー値調整のランタイムレート*により決定されます。）
Drift Compensation	Enabled, Disabled Default: Enabled	ドリフト補正を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。ドリフト補正により、ボードやサーフェスの老化あるいは温度や環境の変化に応じて、タッチの存在の有無を決定するベースライン値が経時的に変化するよう設定できます。ドリフト補正レートは、*定常状態のドリフト補正レート*および*スタートアップドリフト補正レート*、*チャネル解放補正レート*により決定されます。
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	ドリフト補正のアルゴリズムを提供します。現在サポートされているのは Alternate 1 のみです。その他のオプションは、今後のリリースでサポートされる可能性があります。ドリフト補正方法 Alternate 1 を使うと、定常状態、スタートアップ、チャネル解放の各イベントに対し、異なるドリフト補正レートを用いることができます。
Steady state drift compensation rate, drift compensation will be applied per n scans	500	チャネルが定常状態にある場合のドリフト補正レート（N スキャンごとに適用）を決定します。（スキャンレートに依存します。）
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	ドライバーの初期化実行時におけるドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*よりも低く設定する必要があります。）
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	チャネルが押された状態から開放された状態に移行する際のドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*以下に設定する必要があります。）

ISDE Property	Value	説明
Default filter depth (used in sensor count filter provided by driver)	1	ドライバーが提供するソフトウェアセンサーカウントフィルターで用いられます。このデフォルトフィルターは、比較的一般的な「漏れ積分回路」ソフトウェアフィルターを改良したもので、深さ 4 は約 16 のフィルター定数と同等となります。入力が 16 以上の定数値である場合、フィルター出力はフィルターの 16 サイクル / 繰り返し後に定数入力値の 68 ～ 69% に達します。
Runtime rate of tuning of sensor values (if drift compensation is used this value is overridden to be twice the rate of the steady state drift compensation)	800	オフセット調整のレート。（ドリフト補正を使用する場合、この値は定常状態ドリフト補正レートの 2 倍に設定されます。）
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	すべてのチャネルがタッチされていない場合にのみ、自動調整とドリフト補正を実行します。
Max. active channels	1	アプリケーションによって使用されるチャネルの最大数を指定します。 Self Capacitance Mode では、使用されるチャネル数を指します。 Mutual Capacitance Mode では、Rx チャネルと Tx チャネルの乗算の結果です。たとえば、Rx チャネル 4 つと Tx チャネル 3 つの場合、最大アクティブチャネル数は 12 です。
Name	g_ctsu0	モジュール名。
CTSU configuration used	g_ctsu0_config	CTWによって生成された構成構造体の名前。
Callback	NULL	スキャンが完了するたびに呼び出されるユーザーコールバック関数。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。
Data Processing Option	Default Processing (recommended), No Processing (tuning only) Default: Default Processing	完了した前回のデータ後にスキャンが開始されるかどうかや、自動較正がスキャンの間に実行されるかどうかを指定するために使用できます。調整を行う場合以外はデフォルト処理を使用してください。

ISDE Property	Value	説明
Write Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ライト割り込み優先順位の選択。
Read Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	リード割り込み優先順位の選択。
End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	終了割り込み優先順位の選択。

r_dtc EVENT CTSU END での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table

ISDE Property	Value	説明
Name	g_transfer0	モジュール名
Mode	Block	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode)	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	1	転送回数の選択
Number of Blocks (Valid only in Block Mode)	1	ブロック数の選択
Activation Source (Must enable IRQ)	Event CTSU END	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

***注:** 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

静電容量式タッチボタンフレームワークモジュールのクロック構成

CTSU ボタンフレームワークは CTSU HAL ドライバーと同じ構成を使用します。 CTSU クロック速度を、静電容量式タッチ調整ツール Capacitive Touch Workbench for Renesas Synergy で選択したクロック速度と同じ値に設定します。

静電容量式タッチボタンフレームワークモジュールのピン構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

***注:** 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

静電容量式タッチセンシングユニット （CTSU）のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
CTSU	**Pins**	Select **Peripherals > Input: CTSU > CTSU0**

CTSU のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Enabled (Default: Disabled)	CTSU の動作モードとして Enabled を選択します
TS00 to TS11	None, Pn, Pm (Default: P204, P206, P207, None, P408, P409, None, None, None, P414, P415)	TS ピン
TSCAP	None, Pn, Pm (Default: None)	TSCAP ピン

***注 :** 前の設定例は、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.1.13.5 アプリケーションでの静電容量式タッチボタンフレームワークモジュールの使用

静電容量式タッチボタンフレームワークを使用する前に、ISDE のプロジェクトにフレームワークを追加し、CTW の構成データがプロジェクトに追加されていることを確認して、ボタンの数、CTW によって生成された構成構造体の名前、使用するコールバックなどのフレームワークのプロパティを更新します。これを実行

した後、アプリケーションで静電容量式タッチボタンフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して静電容量式タッチボタンフレームワークモジュールを初期化します。
2) コールバック関数で、タッチボタンイベントを検出し、処理します。
3) disable API を使用してボタンのコールバックを無効化します（オプション）。
4) enable API を使用してボタンのコールバックを有効化します（オプション）。
5) close API を使用して静電容量式タッチボタンフレームワークモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

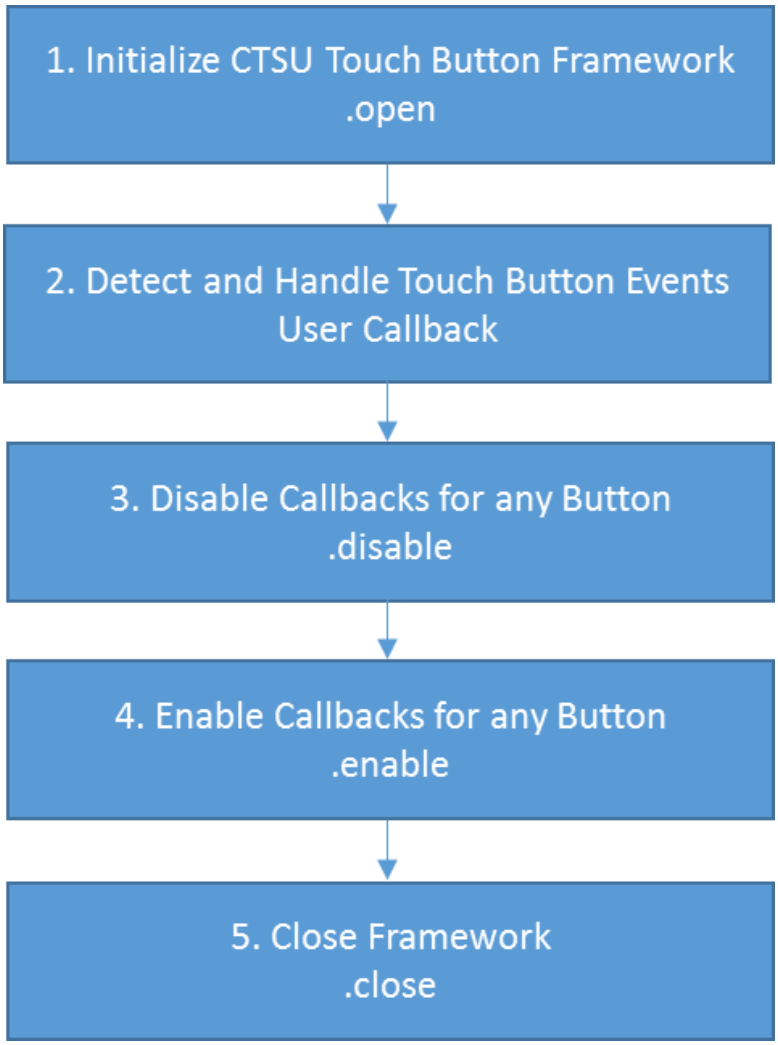


図 184: 通常の静電容量式タッチボタンフレームワークモジュールアプリケーションのフロー図

5.1.14 静電容量式タッチスライダフレームワーク

静電容量式タッチスライダフレームワークモジュールは、システムによって初期化されるすべてのスライダ構成の CTSU データを解釈するために使用されます。主な特長を以下に示します。

- スライダとホイールをサポート
- スライダとホイールの複数インスタンスをサポート
- コールバックを使用してタッチ処理を簡略化
  - CTW から生成された構成データを使用
  - 状態が変化したときにコールバックを生成
  - コールバックは各スライダに関連付けられ、イベントと位置を含む
  - コールバックは構成テーブルに列挙された順に呼び出される
- マルチタッチ検出をサポート（ビルド時にオプションで無効化できる）
- S124、 S128、 S3A6、 S3A7、 S5D5、 S5D9、 S7G2 をサポートします

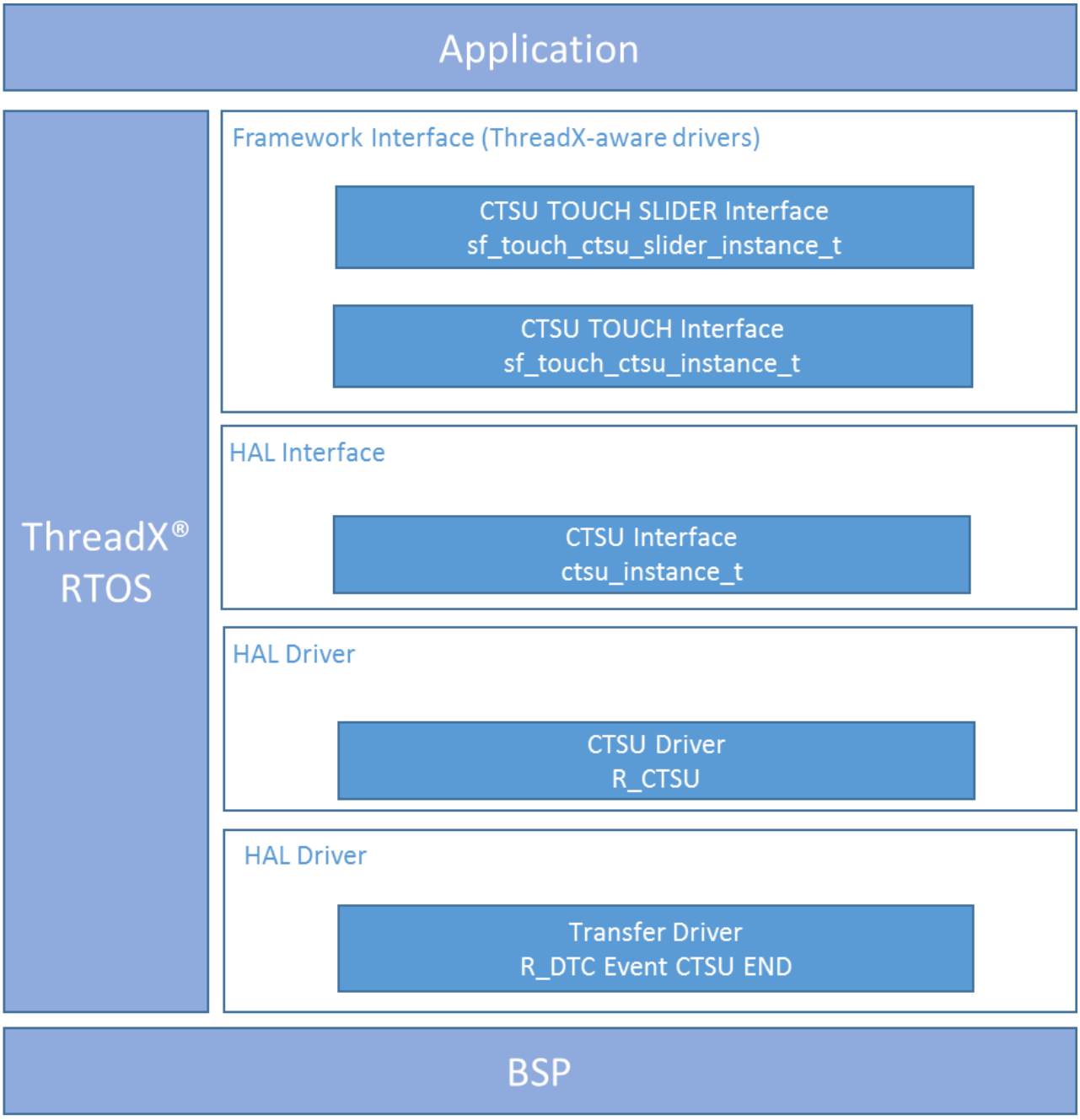


図 185:　静電容量式タッチスライダフレームワークの編成、オプション、スタックの実装

5.1.14.1 静電容量式タッチスライダフレームワークモジュール API の概要

静電容量式タッチスライダフレームワークは、オープン、クローズ、有効化、無効化の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

静電容量式タッチスライダフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_touch_slider0.p_api->open(g_sf_touch_slider0.p_ctrl, g_sf_touch_slider0.p_cfg); タッチ CTSU ボタンフレームワークを初期化します。また低レベルハードウェアを構成し、すべてのボタンに対してコールバック関数を登録します。
enable	g_sf_touch_slider0.p_api->enable(g_sf_touch_slider0.p_ctrl, slider_id); 構成されたボタンのコールバック通知を有効化します。
disable	g_sf_touch_slider0.p_api->disable(g_sf_touch_slider0.p_ctrl, slider_id); 構成されたボタンのコールバック通知を無効化します。
close	g_sf_touch_slider0.p_api->close(g_sf_touch_slider0.p_ctrl); タッチ CTSU ボタンフレームワークを閉じます。他のモジュールによって使用されていない場合は、ボタンフレームワークと下位レイヤーを閉じます。
versionGet	g_sf_touch_slider0.p_api->versionGet(&p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	フレームワークは既に使用中です。
SSP_ERR_NOT_OPEN	デバイスは開かれていません。
SSP_ERR_INTERNAL	内部エラー。
SSP_ERR_UNSUPPORTED	サポートされないデバイス。
SSP_ERR_INVALID_ARGUMENT	無効な引数。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.14.2 静電容量式タッチスライダフレームワークモジュールの動作の概要

静電容量式タッチスライダフレームワークモジュールは、システムによって初期化されるすべてのスライダ構成の CTSU データを解釈するために使用されます。また、CTSU フレームワークレイヤーの初期化も行います。静電容量式タッチスライダフレームワークには、処理されたデータが使用可能になるたびに CTSU フレームワークレイヤを伴って呼び出されるコールバックがあります。スライダーフレームワークは、このデータ（未加工値）を使用して、タッチまたは解放が発生したかどうか、また発生した場合はどこで発生したかを判断します。状態に変化があった場合、フレームワークはイベントと位置とともにスライダー構成テーブルに列挙された順に各スライダーのコールバックを呼び出します。スライダーフレームワークは、タッチイベントと解放イベント間の更新レート (sf_touch_ctsu configuration update_hz) でコールバックを実行します。アプリケーションコードは、これらのコールバックを使用してスライダー上の位置を追跡する必要があります。スライダフレームワークは、タッチイベントと解放イベント間の更新レート（update_hz）でコールバックを実行します。この機能は、ビルド時にオプションで無効にできます。

静電容量式タッチスライダフレームワークモジュールの動作に関する注意事項

このフレームワークは、[SSP Utilities] タブで Renesas Synergy Gallery からダウンロードできる Capacitive Touch Workbench for Renesas Synergy ツールによって生成される構成データとともに使用するように設計されています。

静電容量式タッチスライダフレームワークモジュールの制限事項

このフレームワークは、セルフキャパシタンス動作モードで配置された静電容量式タッチスライダおよびホイールのみをサポートします。それ以外には、このモジュールを使用するうえで既知の制限事項はありません。その他の制限事項については、SSP の最新のリリースノートを参照してください。

5.1.14.3 アプリケーションへの静電容量式タッチスライダフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチスライダフレームワークモジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

静電容量式タッチスライダフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（静電容量式タッチスライダフレームワークモジュールのデフォルト名は g_sf_touch_slider0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

静電容量式タッチスライダフレームワークの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_slider0 Capacitive Touch Slider/Wheel Framework on sf_touch_ctsu_slider	**Threads**	**New Stack> Framework> Input> Capacitive Touch Slider Framework on sf_touch_ctsu_slider**

次の図に示すように sf_touch_ctsu_slider の静電容量式タッチスライダフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

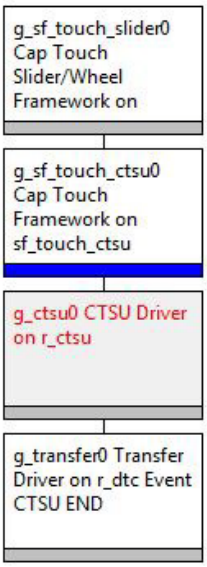


図 186: 静電容量式タッチスライダフレームワークモジュールのスタック

5.1.14.4 静電容量式タッチスライダフレームワークモジュールの構成

必要な動作に合わせて静電容量式タッチスライダフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_touch_ctsu_slider 上の静電容量式タッチスライダフレームワークモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
Number of Sliders/Wheels	1	CTW で構成されたスライダの数
Multi touch enable	Enabled, Disabled Default: Enabled	マルチタッチ検出を有効化 / 無効化します
Name	g_sf_touch_slider0	フレームワーク名
Slider/Wheel Configuration Structure Name (generated by Workbench)	all_sliders	CTWによって生成されたスライダ構成構造体の名前
Callback	g_slider_framework_user_callback	ユーザーアプリケーションによって作成されたコールバック関数の名前。

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルプロパティは Capacitive Touch Workbench for Renesas Synergy（CTW）によって構成されるため、CTSU の動作に関する詳細な知識がない場合は構成変更を行わないことが最善です。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

静電容量式タッチスライダフレームワークモジュールの構成設定

通常は、下位レベルモジュールの少数の設定のみをデフォルトから変更する必要があります。（これらは [thread stack] ブロックに赤いテキストで示されます。）一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

sf_touch_ctsu 上の静電容量式タッチフレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効化または無効化します。

ISDE Property	Value	説明
Name	g_sf_touch_ctsu0	モジュール名。
Thread Priority	3	システムの他のスレッドのプライオリティに基づいてユーザーが決定します。高いプライオリティを使用することを推奨します。
Update Hz	50	CTSU ハードウェアスキャンを実行するレート。最大スキャンレートは、ThreadX のティックレート設定値未満にする必要があります。ハードウェアスキャンを実行する合計時間は、使用するチャネル数によって決定されます。各チャネルは、スキャンを完了するのに約 600 usec かかります。そのため、10 個のチャネルがセルフキャパシタンスモードで 10 個のボタンに使用される場合は、合計ハードウェアスキャン時間は最大 6 ミリ秒または 166 Hz となります。同様に、7 個のチャネルが相互容量モード（5x2 レイアウト）で 5 個のボタンに使用される場合は、合計ハードウェアスキャン時間は最大 4.2 ミリ秒または 250Hz となります。これにソフトウェア処理時間が加わり、その長さはコアクロックと CTSU によって使用されるソフトウェアのフィルタ深さに応じて変わります。したがって最大スキャンレートは、構成内で使用されるすべてのチャネルのスキャンと処理に必要な時間より小さくし、RTOS ティックレート未満とする必要があります。
Callback	NULL	各スキャンが完了したときに呼び出されるユーザーコールバックの名前。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Offset Adjustment	Enabled, Disabled Default: Enabled	オフセット調整を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。(レートは、*センサー値調整のランタイムレート*により決定されます。)
Drift Compensation	Enabled, Disabled Default: Enabled	ドリフト補正を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。ドリフト補正により、ボードやサーフェスの老化あるいは温度や環境の変化に応じて、タッチの存在の有無を決定するベースライン値が経時的に変化するよう設定できます。ドリフト補正レートは、*定常状態のドリフト補正レート*および*スタートアップドリフト補正レート*、*チャネル解放補正レート*により決定されます。
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	ドリフト補正のアルゴリズムを提供します。現在サポートされているのは Alternate 1 のみです。その他のオプションは、今後のリリースでサポートされる可能性があります。ドリフト補正方法 Alternate 1 を使うと、定常状態、スタートアップ、チャネル解放の各イベントに対し、異なるドリフト補正レートを用いることができます。
Steady state drift compensation rate, drift compensation will be applied per n scans	500	チャネルが定常状態にある場合のドリフト補正レート (N スキャンごとに適用) を決定します。(スキャンレートに依存します。)
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	ドライバーの初期化実行時におけるドリフト補正レート (N スキャンごとに適用) を決定します。(*定常状態ドリフト補正レート*よりも低く設定する必要があります。)

ISDE Property	Value	説明
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	チャネルが押された状態から開放された状態に移行する際のドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*以下に設定する必要があります。）
Default filter depth (used in sensor count filter provided by driver)	1	ドライバーが提供するソフトウェアセンサーカウントフィルターで用いられます。このデフォルトフィルターは、比較的一般的な「漏れ積分回路」ソフトウェアフィルターを改良したもので、深さ 4 は約 16 のフィルター定数と同等となります。入力が 16 以上の定数値である場合、フィルター出力はフィルターの 16 サイクル / 繰り返し後に定数入力値の 68 ～ 69% に達します。
Runtime rate of tuning of sensor values (if drift compensation is used this value is overrriden to be twice the rate of the steady state drift compensation)	800	オフセット調整のレート。（ドリフト補正を使用する場合、この値は定常状態ドリフト補正レートの 2 倍に設定されます。）
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	すべてのチャネルがタッチされていない場合にのみ、自動調整とドリフト補正を実行します。
Max. active channels	1	アプリケーションによって使用されるチャネルの最大数を指定します。 Self Capacitance Mode では、使用されるチャネル数を指します。 Mutual Capacitance Mode では、Rx チャネルと Tx チャネルの乗算の結果です。たとえば、Rx チャネル 4 つと Tx チャネル 3 つの場合、最大アクティブチャネル数は 12 です。
Name	g_ctsu0	モジュール名。
CTSU configuration used	g_ctsu0_config	CTWによって生成された構成構造体の名前。
Callback	NULL	スキャンが完了するたびに呼び出されるユーザーコールバック関数。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。

ISDE Property	Value	説明
Data Processing Option	Default Processing (recommended), No Processing (tuning only) Default: Default Processing	完了した前回のデータ後にスキャンが開始されるかどうかや、自動較正がスキャンの間に実行されるかどうかなどの指定に使用することはできません。調整を行う場合以外はデフォルト処理を使用してください。
Write Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ライト割り込み優先順位の選択。
Read Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	リード割り込み優先順位の選択。
End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	終了割り込み優先順位の選択。

r_dtc EVENT CTSU END での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択

ISDE Property	Value	説明
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table
Name	g_transfer0	モジュール名
Mode	Block	モードの選択
Transfer Size	2 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	1	転送回数の選択
Number of Blocks (Valid only in Block Mode)	1	ブロック数の選択
Activation Source (Must enable IRQ)	Event CTSU END	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

注: 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

静電容量式タッチスライダフレームワークモジュールのクロック構成

CTSU スライダフレームワークは CTSU HAL ドライバーと同じ構成を使用します。 CTSU クロック速度を、静電容量式タッチ調整ツールで選択したクロック速度と同じ値に設定します。

静電容量式タッチスライダフレームワークモジュールのピン構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の最初の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は CTSU ピンの選択例を示しています。

注: 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

静電容量式タッチセンシングユニット （CTSU）のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
CTSU	Pins	Select **Peripherals > CTSU> Cap_Touch_Sensing_Unit**

CTSU のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Enabled (Default: Disabled)	CTSU の動作モードとして Enabled を選択します
TS00 to TS11	None, Pn, Pm (Default: None)	TS ピン
TSCAP	None, P205, P203 (Default: P205)	TSCAP ピン

注: 前の設定例は、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.1.14.5 アプリケーションでの静電容量式タッチスライダフレームワークモジュールの使用

静電容量式タッチスライダフレームワークを使用する前に、ISDE のプロジェクトにフレームワークを追加し、CTW の構成データがプロジェクトに追加されていることを確認して、スライダの数、CTW によって生成された構成構造体の名前、使用するコールバックなどのフレームワークのプロパティを更新します。これを実行した後、アプリケーションで静電容量式タッチスライダフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して静電容量式タッチスライダフレームワークモジュールを初期化します。
2) コールバック関数で、タッチスライダイベントを検出し、処理します。
3) disable API を使用してスライダのコールバックを無効化します（オプション）。
4) enable API を使用してスライダのコールバックを有効化します（オプション）。
5) close API を使用して静電容量式タッチボタンフレームワークモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

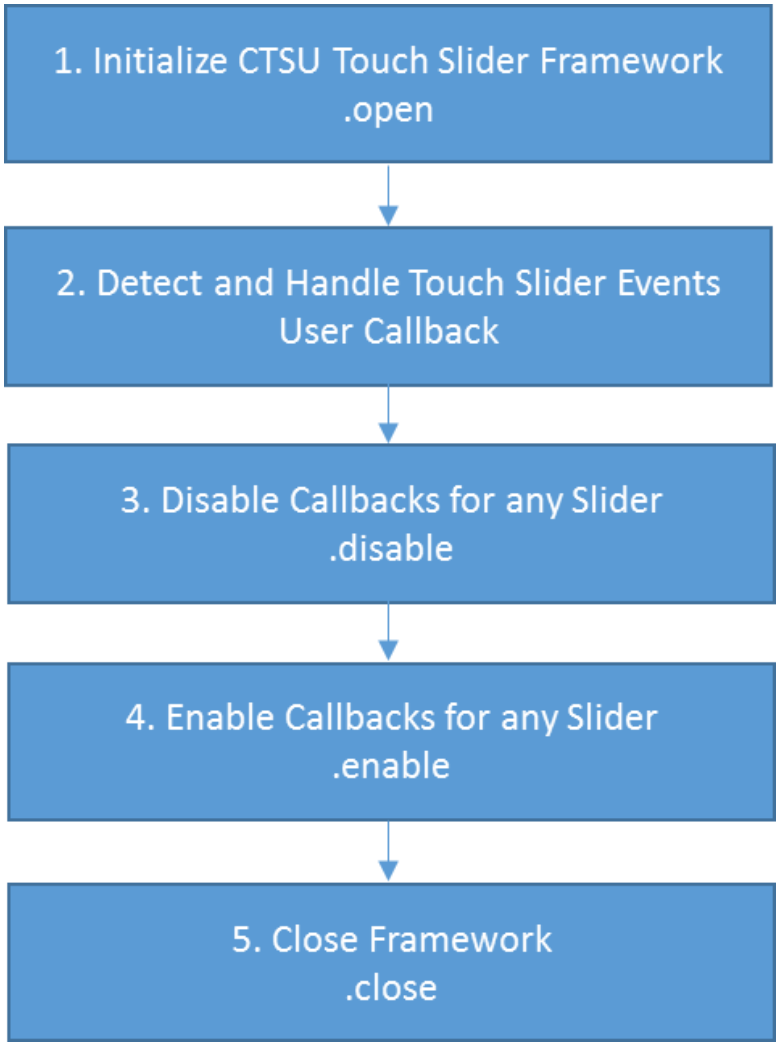


図 187: 通常の静電容量式タッチスライダフレームワークアプリケーションのフロー図

5.1.15 外部 IRQ フレームワーク

- 外部割り込み入力に応答します
- RTOS はスレッド同期に内部セマフォを使用して実装を認識します
  - 内部スレッドに信号を送信できます
  - イベントリンクコントローラ（ELC）を介して転送をトリガできます
- Synergy MCU で使用可能なポートピンを使用します
  - ピンは MCU 間で異なる可能性があるため、詳細については MCU のユーザーズマニュアルを参照してください

- 以下のような複数のハードウェア機能をサポートします
  - チャネルの選択
  - トリガ条件
  - デジタルフィルタリング
  - オートスタート

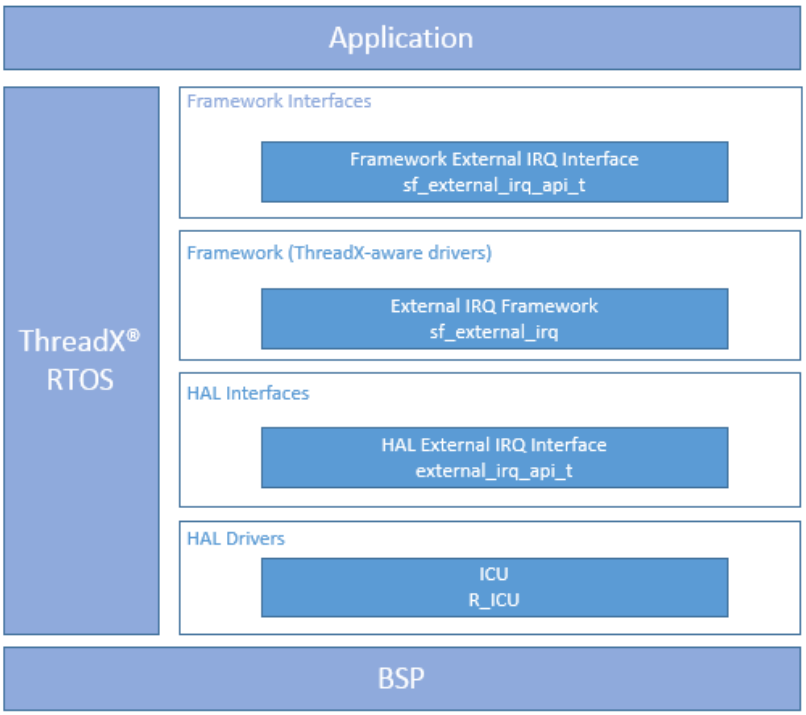


図 188: 外部 IRQ フレームワークの編成、オプション、スタックの実装

5.1.15.1 外部 IRQ フレームワークモジュール API の概要

外部 IRQ フレームワークモジュールは、モジュールのオープン、待機、クローズの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

外部 IRQ フレームワーク API の要約

Function Name	API の呼び出し例と説明
open	g_sf_external_irq.p_api->open(g_sf_external_irq.p_ctrl, g_sf_external_irq.p_cfg); ミューテックスを取得した後、 HAL レイヤーでドライバーの初期化を処理します。
wait	g_sf_external_irq.p_api->wait(g_sf_external_irq.p_ctrl, TX_WAIT_FOREVER); 外部割り込みの終了後、リターンします。
versionGet	g_sf_external_irq.p_api->versionGet(&version); API バージョンを取得し、バージョンポインタに格納します。
close	g_sf_external_irq.p_api->close(g_sf_external_irq.p_ctrl); チャネルミューテックスを解放し、 HAL レイヤーでチャネルを閉じます。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、 SSP ユーザーズマニュアルで関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	デバイスが使用中です。
SSP_ERR_NOT_OPEN	デバイスが開かれていません。
SSP_ERR_TIMEOUT	タイムアウトエラー。
SSP_ERR_WAIT_ABORTED	サスペンションが中断されました。

Name	説明
SSP_ERR_UNSUPPORTED	HAL ドライバーでサポートされない機能です。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、関連モジュールの 『Renesas Synergy Software Package Reference』を参照してください。*

5.1.15.2 外部 IRQ フレームワークモジュールの動作の概要

外部 IRQ フレームワークは、ThreadX 対応の一連のフレームワーク API です。外部 IRQ フレームワークを使用すると、スイッチなどの外部入力は、内部セマフォを通じてスレッドに信号を送信したり、イベントリンクコントローラ (ELC) を通じて転送をトリガすることができます。適切な動作のために、外部 IRQ フレームワークモジュールと外部 IRQ HAL モジュールの両方を構成する必要があります。 HAL 構成設定では、トリガレベルやデジタルフィルタリング設定などのハードウェアオプションを制御できます。

外部 IRQ フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 外部割り込み機能をサポートするポートピンを調べるときや、特定のポートピンの外部 IRQ 番号が知りたいときには、プログラムする Synergy デバイスのデータシートを参照してください。
- 外部 IRQ 番号は、 ISDE で外部 IRQ ドライバーの [Properties] ウィンドウに示されるチャネルの設定に対応します。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.15.3 アプリケーションへの外部 IRQ フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して外部 IRQ フレームワークをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

外部 IRQ フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。（外部 IRQ フレームワークのデフォルトの名前は g_sf_cexternal_irq0 で、これは次の表に示されています。この名前は、対応する **[Properties]** ウィンドウで変更できます）。

外部 IRQ フレームワークの構成設定

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_external_irq0 External IRQ Framework on sf_external_irq	**Threads**	**New Stack> Framework> Input> External IRQ Framework on sf_external_irq**

次の図に示すように sf_external_irq の外部 IRQ フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

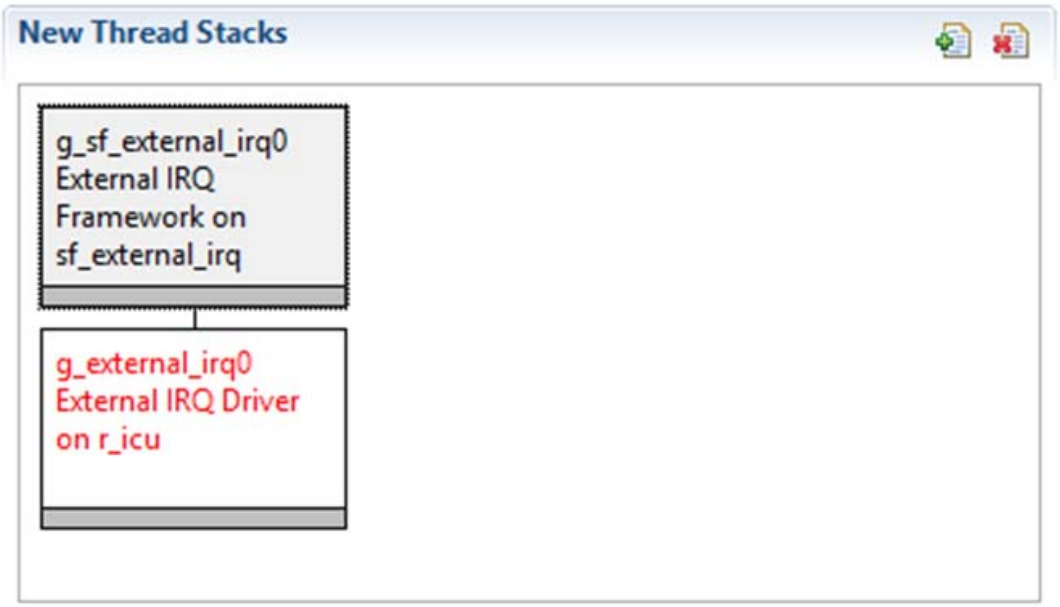


図 189: 外部 IRQ フレームワークのスレッドスタック

5.1.15.4 外部 IRQ フレームワークモジュールの設定

必要な動作に合わせて外部 IRQ フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_external_irq 上の外部 IRQ フレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択。
Name	g_sf_external_irq0	フレームワーク名。
Event	Semaphore Put, None (Default: Semaphore Put)	イベントの選択。

注: *上記の値の例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

スタックモジュールのデフォルト以外の値が望ましい場合もあります。たとえば、トリガ設定やデジタルフィルタリング値の選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

5.1.15.5 外部 IRQ フレームワークモジュールのローレベルドライバーの構成設定

通常は、下位レベルモジュールの少数の設定のみをデフォルトから変更する必要があります。（これらは [thread stack] ブロックに赤いテキストで示されます。）一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_icu 上の外部 IRQ ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択。
ICU IRQ0	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	ICU の選択。
Name	g_external_irq0	ドライバー名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling, Rising, Both Edges, Low Level (Default: Rising)	トリガの選択。
Digital Filtering	Enabled, Disabled (Default: Disabled)	デジタルフィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 (Default: PCKL/64)	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	True, False (Default: True)	割り込み要求イネーブルの選択。
Callback	NULL	コールバックの選択。

注: *前述の値の例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

5.1.15.6 アプリケーションでの外部 IRQ フレームワークモジュールの使用

アプリケーションで外部 IRQ フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API で外部 IRQ フレームワークモジュールを開きます
2) wait API を使用して割り込みを待ちます
3) 外部 IRQ イベントを処理します
4) close API を使用してモジュールを閉じます。

上記の一般的な手順を、次の図の通常の動作フロー図に示します。

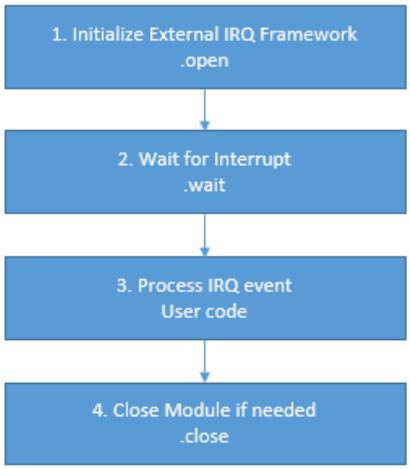


図 190: 通常の外部 IRQ フレームワークモジュールアプリケーションのフロー図

5.1.16 FileX ポートブロックメディアフレームワーク

- FileX ポートブロックメディアフレームワークモジュールの特長は次のとおりです。
  - FAT32、 FAT16、 FAT12 をサポート
  - 特定のメディアでの最初の FAT パーティションのマウントをサポート
  - 無制限の FileX オブジェクト（メディア、ディレクトリ、ファイル）
  - 動的 FileX オブジェクト作成と削除
  - 柔軟なメモリ使用
  - サイズを自動的に拡張
  - 小さなフットプリント
  - ThreadX との完全な統合
- SD カードインタフェース
  - SD、 SDHC、 SDXC フォーマットと互換

  - 1 ビットと 4 ビットのバス幅をサポート
  - ハードウェアでサポートされる場合のカード検出機能
  - ライトプロテクトのサポート
- eMMC インタフェース
  - 1 ビット、 4、ビット、 8 ビットのバス幅をサポート
- SD バスインタフェース
  - SD メモリカードおよび SDIO カードと互換
  - 転送バスモードを 4 ビットのワイドバスモードまたは 1 ビットのデフォルトバスモードから選択可能
  - SD、 SDHC、 SDXC フォーマットと互換
- SD および MMC 共有
  - SBFAI 割り込み SD バッファによってトリガできる DMAC および DTC は、 DMAC を使用してリードおよびライトアクセス可能

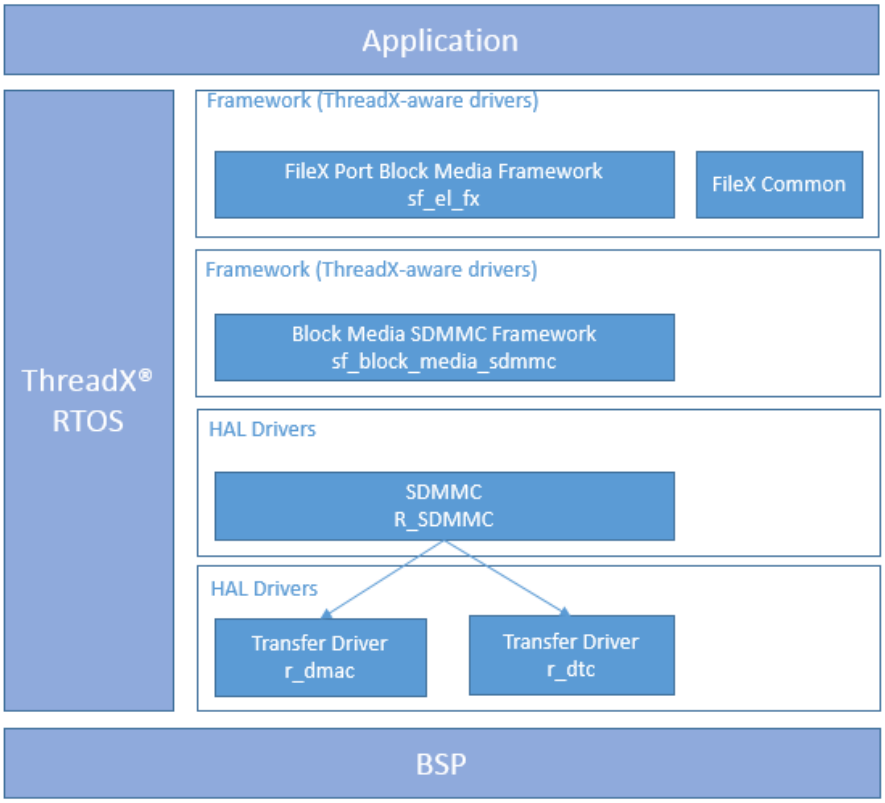


図 191: sf_block_media_sdmmc での FileX ポートブロックメディアフレームワークスタック

5.1.16.1 FileX ポートブロックメディアフレームワーク API の概要

FileXポートブロックメディアフレームワークは、ファイルの整理とアクセスのためのExpress Logic社のFileXの操作を実装します。FileX 関連の API の完全なリストはここで示すには長すぎるため、次の表では最も重要な API の一部のみを示します。このドキュメントの「参考文献」セクションの説明に従って入手可能な『FileX User's Manual』を参照してください。

FileX ポートブロックメディアフレームワーク API の要約 （選択した例のみ）

Function Name	API の呼び出し例と説明
fx_directory_attributes_read	fx_directory_attributes_read(&my_media, "mydir", &attributes); 指定したメディアからディレクトリ属性を読み取ります
fx_file_open	fx_file_open(&my_media, &my_file, "myfile.txt", FX_OPEN_FOR_READ); 読み取る 「myfile.txt」ファイルを開きます
fx_file_create	fx_file_create(&my_media, "myfile.txt"); 「myfile.txt」という名前のファイルを作成します
fx_media_read	fx_media_read(&my_media, 22, my_buffer); &my_media, で指定されたメディアから論理セクター（この例ではセクター 22）を読み取り、バッファ my_buffer に配置します

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作、ステータス戻り値、定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している『SSP Reference Manual』を参照してください。FileX API ドキュメントについては、「参考文献」セクションで説明しているようにRenesas SynergyのX-Ware コンポーネントとNetX コンポーネントのドキュメントを参照してください。*

5.1.16.2 FileX ポートブロックメディアと R_SDMMC の動作の概要

FileX ポートブロックメディアの動作の概要

FileX は、ディープエンベデッドアプリケーション用の完全なファイルアロケーションテーブル (FAT) フォーマットメディアであり、ファイル管理システムでもあります。FileX は以下のメディアデバイスをサポートします。同時にサポートできるデバイスの数に制限はありません。

- RAM ディスク
- FLASH マネージャー
- その他複数の物理デバイス

FileX は、12 ビット、16 ビット、および 32 ビットの FAT フォーマットと連続的なファイル割り当てをサポートします。FileX はサイズと性能の両面で高度に最適化されています。FileX モジュールの API リファレンスは、このドキュメントの最後の「参考文献」セクションの説明に従って入手できる『FileX User's manual』で参照できます。

FileX ポートブロックメディアフレームワークの動作に関する注意事項

メディアは、開く前に FAT12、FAT16、FAT32 のいずれかのファイルシステムにフォーマットする必要があります。この操作は、メディアの挿入前または実行時に行うことができます。

FileX ブロックメディアフレームワークの制限事項

- SF_EL_FX では現在 exFAT がサポートされていません。
- SF_EL_FX を SD または eMMC とともに使用する場合は、FileX ブロックサイズを 512 バイトにする必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

SDMMC HAL モジュールの動作の概要

SD/MMC ドライバーと SDIO ドライバーは r_sdmmc 上に実装され、SD/MMC メディアデバイスならびに SDIO カードの読み取り / 書き込みと制御に使用されます。SD/MMC モジュールはスタンドアロンドライバーとして使用できます。また、FileX やその他の互換性のあるファイルシステムとともに使用することもできます。

SDMMC HAL モジュールは次のメモリデバイスをサポートしています。

- SDSC（SD 標準容量）、SDHC（SD 大容量）、SDXC（SD 拡張容量）、eMMC（組み込みマルチメディアカード）
  - SD および eMMC メモリデバイスのリード、ライト、消去をサポート
  - 1 ビット、4 ビット、8 ビットデータバスをサポート（8 ビットバスは eMMC でのみサポート）
  - ハードウェアライトプロテクトの検出をサポート（SD カードのみ）
  - 下位互換モードと高速 SRD モード（eMMC）間を自動的に選択
- SDIO のサポート
  - SDIO シングルレジスタアクセスをサポート（CMD52）
  - SDIO マルチレジスタアクセスをサポート（CMD53）
  - SDIO 割り込みをサポート
- ホスト（MCU）とデバイスの両方でサポートされる最大クロックレートにクロックを自動的に構成

割り込み構成

FileX、sf_el_fx、sf_block_media_sdmmc, を使用する場合、コールバックは sf_block_media_sdmmc レイヤで実装されます。SDMMC モジュールレベルでの設定は不要です。ただし、割り込みおよび転送ドライバーについては設定が必要です。必要な割り込みは次のとおりです。

DTC で SD/MMC を使用 :

- アクセス割り込み
- DTC 割り込み

DMAC で SD/MMC を使用 :

- アクセス割り込み
- DMAC 割り込み （DMAC インスタンス内）

DTC で SDIO を使用 :

- アクセス割り込み
- SDIO 割り込み
- DTC 割り込み

DMAC で SDIO を使用 :

- アクセス割り込み
- SDIO 割り込み
- DMAC 割り込み （DMAC インスタンス内）

カード割り込みはオプションで、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン（n = チャネル番号）が使用可能になっている MCU パッケージでのみ使用可能です。

ブロックサイズ構成

ブロックサイズ構成は、SDIO カードで使用する場合にのみ提供されます。SDIO カードの場合、ブロックサイズは 1 ～ 512 バイトに構成できます。ブロックサイズは、SD カードと e/MMC メモリデバイスではデフォルトの 512 バイトのままにする必要があります。

カード検出構成

r_sdmmc ドライバーでカード検出を使用する前に、「カード検出の制限事項」を参照してください。カード検出が使用不能な場合や、アプリケーションに不要な場合は、 Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [Not Used] に設定する必要があります。カード検出を有効化するには、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [CD Pin] に、[Media Type] を [Card] に設定する必要があります。

アクセス割り込み優先順位

データ転送が 4 バイトでアラインされていないか、 4 バイトの倍数でない場合、ブロックサイズ （最大 512 バイト）のソフトウェアコピーはアクセス割り込みで実行されます。これにより、ソフトウェアコピーが完了するまで、プライオリティがアクセス割り込み以下の他の割り込みがブロックされます。

一般的なタイミングに関する注意事項

このドライバーの一部の関数はブロックします。open と erase は、動作全体が完了するまでブロックします。read、write、readIo、writeIo、readIoExt、writeIoExt は、単一コマンドレスポンスサイクルにわたってブロックします。マルチスレッドシステムでは、これらのいずれかの関数呼び出し中にこのドライバーがブロックする可能性のある時間よりも短いレスポンスタイムを他のスレッドが必要とする場合、このドライバーをプライオリティの低いスレッドで使用するには注意が必要です。

Open のタイミングに関する注意事項

open API は、デバイスの識別および構成プロセス全体を実行します。これには、1 ビットのバス幅と 400kHz 以下のバス速度でのいくつかのコマンドレスポンスサイクルが関係します。

Read、 Write、 ReadIoExt、 WriteIoExt のタイミングに関する注意事項

メディアのリードおよびライト（read および write）と拡張リードおよびライト SDIO API（readIoExt および writeIoExt）はデバイスからレスポンスを受信するまでブロックします。データ転送操作は非ブロックであり、割り込みおよび転送インスタンス（DMAC または DTC）を必要とします。これらの API は、初期動作が正常に開始したことを示す SSP_SUCCESS を返します。アプリケーションは、イベント SDMMC_EVENT_TRANSFER_COMPLETE または SDMMC_EVENT_TRANSFER_ERROR でコールバックを待って、リードまたはライトの完了を示す必要があります。

ReadIo と WriteIo のタイミングに関する注意事項

SDIO readIo および writeIo API は、デバイスからレスポンスを受信するまでブロックします。リードまたはライト動作はこれらの API が戻ると完了します。

消去のタイミングに関する注意事項

erase API は、消去動作が完了するまでブロックします。これは、消去されるセクター数に応じて数秒以上の場合があります。

SDIO 割り込み

多くの SDIO カードはレベル割り込みを使用するため、割り込みは割り込みがデバイスでクリアされるまでアサート解除されません。SDIO 割り込みが適切にクリアされるようにするには、次のいずれかの方法をお勧めします。

- コールバック関数がイベント SDMMC_EVENT_SDIO. で終了する前に SDIO 割り込みがデバイスでクリアされることを確認します。この方法を選択し、 SDIO API を割り込みで使用する必要がある場合は、アクセス割り込みのプライオリティを SDIO 割り込みよりも高くする必要があります。
- IoIntEnable を使用してイベント SDMMC_EVENT_SDIO の発生時のコールバック関数の SDIO 割り込みを無効化します。アプリケーションの他の場所の SDIO 割り込みをクリアしてから、必要に応じて IoIntEnable を使用して SDIO 割り込みを再有効化します。

SD HALA への準拠

SD の仕様に準拠したホストデバイスを開発する場合、ホストは SD ホスト機器およびペリフェラルの使用許諾契約（SD Host/Ancillary Product License Agreement）（SD HALA）に従う必要があります。

データのアラインメントとサイズ

データ転送は、4 バイトでアラインされる必要があり、可能な場合にはサイズが 4 バイトの倍数である必要があります。この推奨事項は、read()、write()、readIoExt()、writeIoExt() API に適用されます。データ転送が 4 バイトでアラインされ、4 バイトの倍数である場合、r_sdmmc ドライバーはゼロコピーで、DMAC または DTC によるハードウェア加速を十分に活用します。データ転送が 4 バイトでアラインされていないか、4 バイトの倍数でない場合、転送されるブロックごとに特別な CPU 割り込みが必要です（「アクセス割り込み優先順位」を参照）。

カード検出の制限事項

r_sdmmc ドライバーでのカード検出のサポートは制限されています。カード検出は open が完了した後でのみ使用可能で、 open() はカードが挿入されていない限り完了できません。したがって、r_sdmmc ドライバーでのカード検出は、カードの取り外しと再挿入の検出にのみ役立ちます。再挿入が検出された後、ドライバーを終了して再開する必要があり、カード検出は再開が完了するまで使用可能になりません。カード検出は、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン （n = チャネル番号）が使用可能になっている MCU パッ

ケージでのみ使用可能です。MCU に SDHIn CD ピンがないか、カード検出がアプリケーションに不要な場合は、 Synergy 構成ツールの r_sdmmc インスタンスの [Properties] でカード検出を無効化する必要があります。r_sdmmc ドライバーのカード検出機能の使用の代替方法は、外部 IRQ インスタンスを使用し、アプリケーションレイヤでカード検出を処理することです。カード検出がアプリケーションレイヤーで処理される場合は、カード挿入が検出された後に open を呼び出す必要があり、カード取り外しが検出された後に close を呼び出す必要があります。

その他の制限事項

このモジュールの最新の制限事項については、 SSP の最新のリリースノートを参照してください。

5.1.16.3 アプリケーションへの FileX ポートブロックメディアフレームワークの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに FileX ポートブロックメディアフレームワークを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（モジュールのデフォルト名を次の表に示します。この名前は、対応する [Properties] ウィンドウで変更できます）。次の図に示すように、ドライバーが追加されるとスレッドに表示されます。

モジュール選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_fx0 FileX Port Block Media Framework on sf_el_fx	**Threads**	**New Stack> Framework > File System> FileX Port Block Media Framework on sf_el_fx**

次の図に示すように sf_el_fx 上の FileX ポートブロックメディアフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、複数のスタックで使用できるため 1 回のみ追加する必要があります。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、このテキストを含むブロックで示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、 [New] アイコンが表示され、可能な選択肢が表示されます。

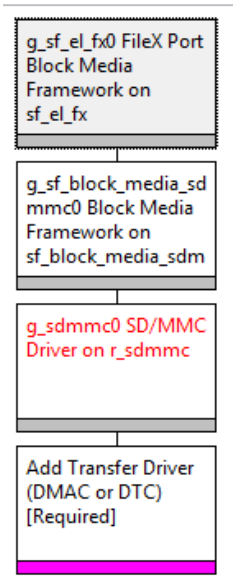


図 192: FileX ポートブロックメディアフレームワークスタック

5.1.16.4 FileX ポートブロックメディアフレームワークモジュールの構成

ユーザーは必要な動作に合わせてモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注：** ISDE を開き、FileX ポートブロックメディアを作成し、次に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

FileX ポートブロックメディアモジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_sf_el_fx0	FileXポートブロックメディアフレームワークモジュール名

sf_block_media_sdmmc モジュール上のブロックメディアフレームワークの構成設定

ISDE Property	Setting	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_sf_block_media_sdmmc0	ブロックメディアフレームワークモジュール名
Block size of media in bytes	512	メディアブロックサイズ

注: *上記の設定例とデフォルトは、Synergy S7 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、別のメディアタイプではバス幅が異なる可能性があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

5.1.16.5 SD/MMC ドライバーと SDIO ドライバーの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

SD/MMC および SDIO ドライバーの構成

ISDE Property	Setting	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_sdmmc0	モジュール名
Channel	0,1 Default 1	SDMMC のチャネル（チャネル 0 または 1）
Media type	Embedded, Card Default: Embedded	メディアは、カードと組み込みデバイスのどちらかです。この設定により、カードの挿入/取り出しがあるのか、あるいは書き込み保護ピンが存在するのかをファームウェアに知らせます。
Bus Width	1 bit, 4 bits, 8 bits	ハードウェアインタフェースによって定義されたバス幅。（8 ビットは eMMC のみ）
Block Size	512	ブロックのサイズ

ISDE Property	Setting	説明
Callback	NULL (default) Name of user callback function.	（FileX を使用する場合は不要）ユーザーコールバック関数の名前に設定します。割り込みを発生させたイベントを提供します。 SDMMC_EVENT_CARD_REMOVED, SDMMC_EVENT_CARD_INSERTED, SDMMC_EVENT_ACCESS, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANSFER_COMPLETE, SDMMC_EVENT_TRANSFER_ERROR
Access Interrupt Priority	Priority level 0-15 Default: Disabled	SD および eMMC、SDIO への読み取り、書き込み、エラーに必要となる割り込み優先順位。有効にすることで、割り込みコールバックを呼び出します。
Card Interrupt Priority	Priority level 0-15 Default: Disabled	（オプション）カード取り外しの割り込み。カードが取り外された際、自動でデバイスを閉じます。有効にすることで、割り込みコールバックを呼び出します。
DMA Request Interrupt Priority	Priority level 0-15 Default: Disabled	SDIO の読み取り、書き込み、エラーに必要となる割り込みプライオリティレベル。有効にすることで、割り込みコールバックを呼び出します。メモリカードに使用されません。

注: *上記の設定例とデフォルトは、Synergy S7 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SDHI のクロック設定

SDHI は PCLKA をそのクロックソースとして使用します。データレートを最適化する必要がない限り、SDMMC 周辺デバイスに対して固有のクロックを設定する必要はありません。SDMMC ドライバーは、PCLKA 周波数と、SD、SDIO または eMMC デバイスで許可される最大クロックレート（これはメディアデバイスの初期化時に取得されます）に基づいて、適切な内蔵分周器を選択します。

SDMMC ピン構成

e^2 studio ピンコンフィギュレーターを使用して、SDMMC の I/O ピンを設定します。ほとんどのボード、高速メモリ、SDIO デバイスにおいて、各品のドライブ能力は「中」または「高」に設定する必要があります。次の各表は [SSP configuration] ウィンドウ内でのピンの選択方法と、モジュールピンの選択例を示しています。

***注：** 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

SDHI のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SDHI	**Pins**	Select **Peripherals > Storage:SHDI > SDHI0**

***注：** 上記の選択シーケンスでは、SDHI0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

SDHI のピン構成設定

Pin Configuration Property	設定値	説明
Operation Mode	Disabled, Custom, SD_MMC 1 bit SD_MMC 4 bit MMC 8 bit (Default: Custom)	アプリケーションに従ってモードを選択します

Pin Configuration Property	設定値	説明
CLK	None, P413 (Default: P413)	クロックピン
CMD	None, P412 (Default: P412)	コマンドピン
DAT0-7	None, PXXX (Default: PXXX)	データピン
CD	None, P903 (Default: P903)	カード検出ピン
WP	None, P414 (Default: P414)	カードライトプロテクションピン

***注：** 前の設定例は、 Synergy S7G2 および DK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

その他の設定

リリース 1.1.0 ではメディアのリード / ライト関数 (R_SDMMC_Read および R_SDMMC_Write) および SDIO の拡張リード / ライト関数 (R_SDMMC_ReadIoExt および R_SDMMC_WriteIoExt) が非ブロッキング関数になったため、DMAC または DTC で割り込みと転送関数が必要です。リード / ライト関数は、初期動作が正常に開始したことを示す SSP_SUCCESS を返します。ただし、ユーザーアプリケーションはユーザーコールバックを待ち、読み取り / 書き込みの完了を示す SDMMC_EVENT_TRANSFER_COMPLETE または SDMMC_EVENT_TRANSFER_ERROR イベントを確認する必要があります。

5.1.16.6 アプリケーションでの FileX ポートブロックメディアフレームワークモジュールまたは SDMMC HAL モジュールの使用

アプリケーションの sf_el_fx で FileX ポートブロックメディアフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) FileX API fx_system_initialize を使用してメディアを初期化します （sf_el_fx は [FileX Common on fx] で [Auto Initialization] プロパティが [Enabled] に設定されている場合にこれを自動的に呼び出します）。
2) オプションで、 FileX API fx_media_format を使用してメディアをフォーマットします

- [Format media during initialization] プロパティが [Enabled] に設定されている場合、 sf_el_fx は生成された初期化関数中にメディアを自動的にフォーマットします）。
- [Filesystem on SDMMC] が有効な場合は、隠しセクターの後のメディア全体がフォーマットされ、[FileX on Block Media] の [Total Sectors] 構成は無視されます。

3) FileX API fx_media_open を使用してメディアを開きます（sf_el_fx は [FileX on Block Media] インスタンスで [Auto Initialization] が [Enabled] に設定されている場合にメディアを自動的に開きます）。
4) FileX API の 1 つ（たとえば、fx_file_create, fx_file_delete, fx_directory_create, fx_directory_delete). を使用して、必要に応じてファイルとディレクトリを作成および削除します。
5) FileX API の 1 つ （たとえば、 fx_file_read や fx_file_write). を使用して、必要に応じてメディア上のファイルに対してリードやライトを行います。
6) FileX API の 1 つ （たとえば、 fx_media_read や fx_media_write). を使用して、必要に応じてメディアディレクトリに対してリードやライトを行います。
7) FileX API fx_media_close. を使用して物理メディアを閉じます。

注: *fx_media_open 呼び出しに成功した後は、すべての FileX ファイルおよびディレクトリ関連 API を使用できます。すべての使用可能な関数については、『FileX ユーザーガイド』を参照してください。*

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

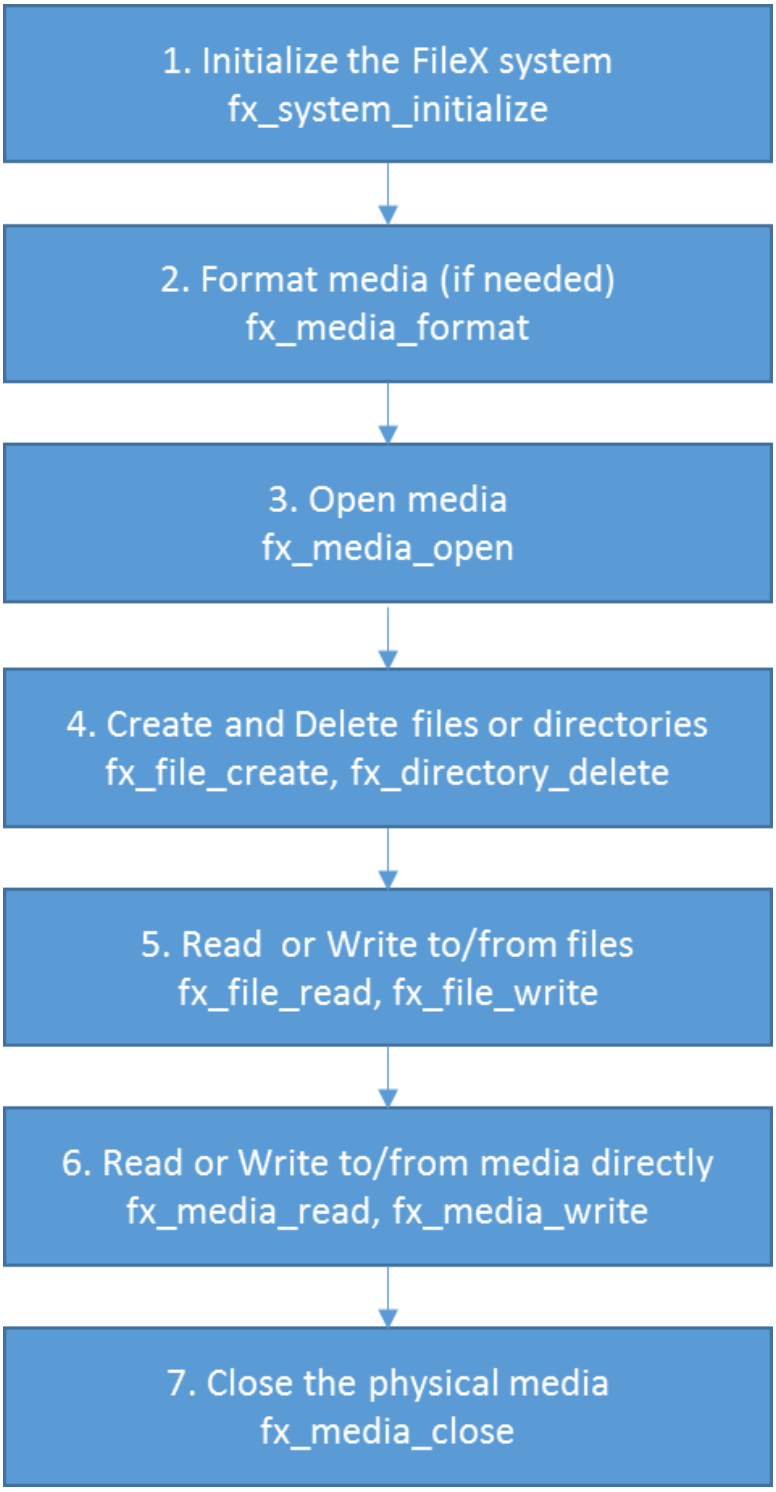


図 193: 通常の **FileX** ポートブロックメディアアプリケーションのフロー図

SD または eMMC メモリデバイスで r_sdmmc ドライバーを使用する一般的な手順は次のとおりです。

1) open を使用してドライバーを開きます。
2) read を使用してカードからデータを読み取るか、 write を使用してカードにデータを書き込みます。
3) read または write が呼び出された後、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR （操作が正常に完了しなかったことを意味します）のコールバックを待ちます。

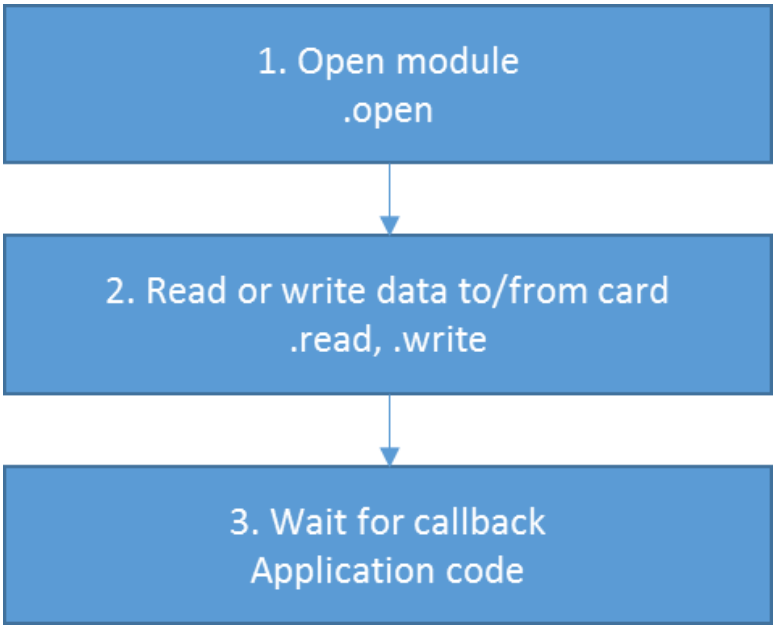


図 194: SDMMC SD または eMMC アプリケーションのフロー図

SDIO カードで r_sdmmc ドライバーを使用する一般的な手順は次のとおりです。

1) open を使用してドライバーを開きます。
2) readIo または writeIo を使用して単一レジスタを設定するか読み戻します。動作はこれらの呼び出しの直後に完了し、コールバックを待つ必要はありません。
3) readIoExt または writeIoExt を使用して複数レジスタを設定するか読み戻します。必要に応じて呼び出しの間に control を使用してブロックサイズを変更できます。
4) readIoExt または writeIoExt が呼び出された後、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR （操作が正常に完了しなかったことを意味します）のコールバックを待ちます。
5) カードが割り込みを要求する場合は、イベント SDMMC_EVENT_SDIO. でコールバックが呼び出されます。カードのドキュメントの説明に従って SDIO 割り込みを処理します（このドキュメントの「SDIO 割り込み」セクションを参照してください）。カードからの SDIO 割り込みは、IoIntEnable を使用していつでも有効化または無効化できます。

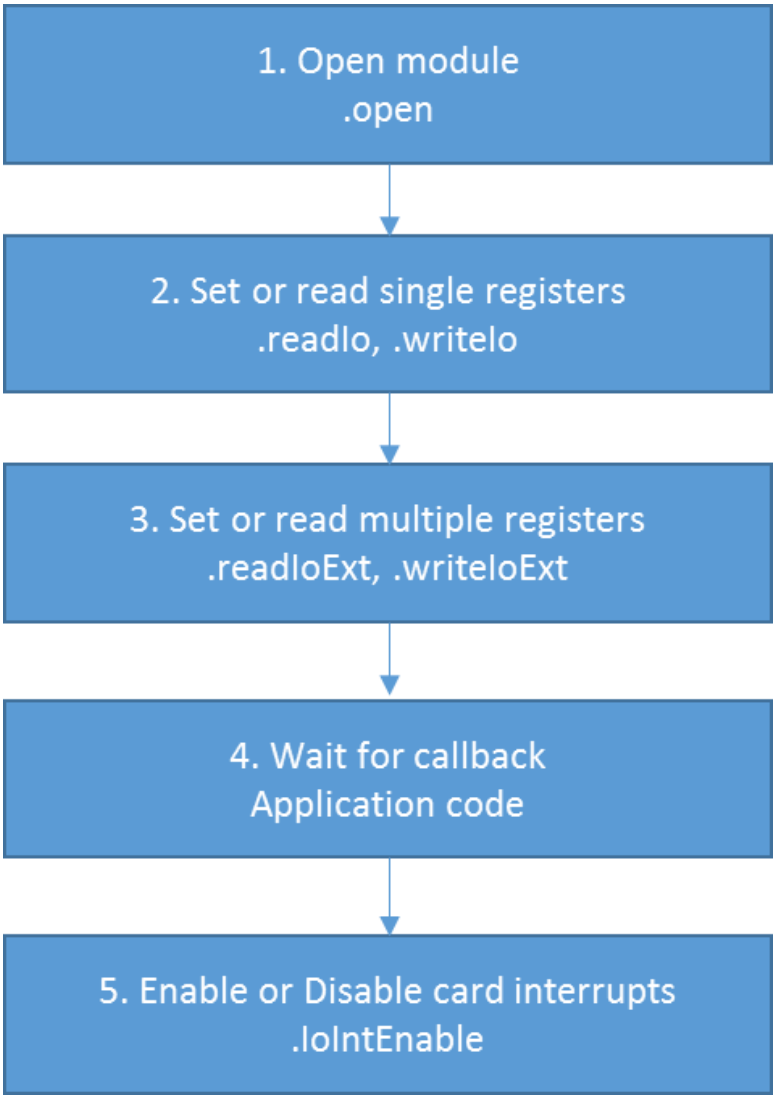


図 195:　一般的な SDMMC SDIO アプリケーションのフロー図

5.1.17 I2C フレームワーク

- ThreadX 対応フレームワーク
- I2C バス上の複数の I2C の統合と同期を処理します
- SCI I2C ドライバーと RIIC ドライバーの両方にアクセスするための単一のインタフェースを提供します
- I2C フレームワークモジュールはマスタモードで I2C 通信を構成します
- I2C フレームワークモジュールは 100kbps、 400kbps、 1Mbps の 3 つのデータレートをサポートします

- I2C フレームワークモジュールは 7 ビットアドレッシングと 10 ビットアドレッシングの両方をサポートします
- I2C フレームワークモジュールは、内部的なコールバックのサポートも提供します。ユーザー定義のコールバックは使用されません。コールバック関数は、イベント i2c_event_t: で呼び出されます。
  - 転送中断
  - 転送完了
  - 受信完了
- コールバック構造体 i2c_callback_args_t には、送信または受信したバイト数も設定されます。
- 以下によって実装されます。
  - SCI 上の簡易 I2C
  - RIIC

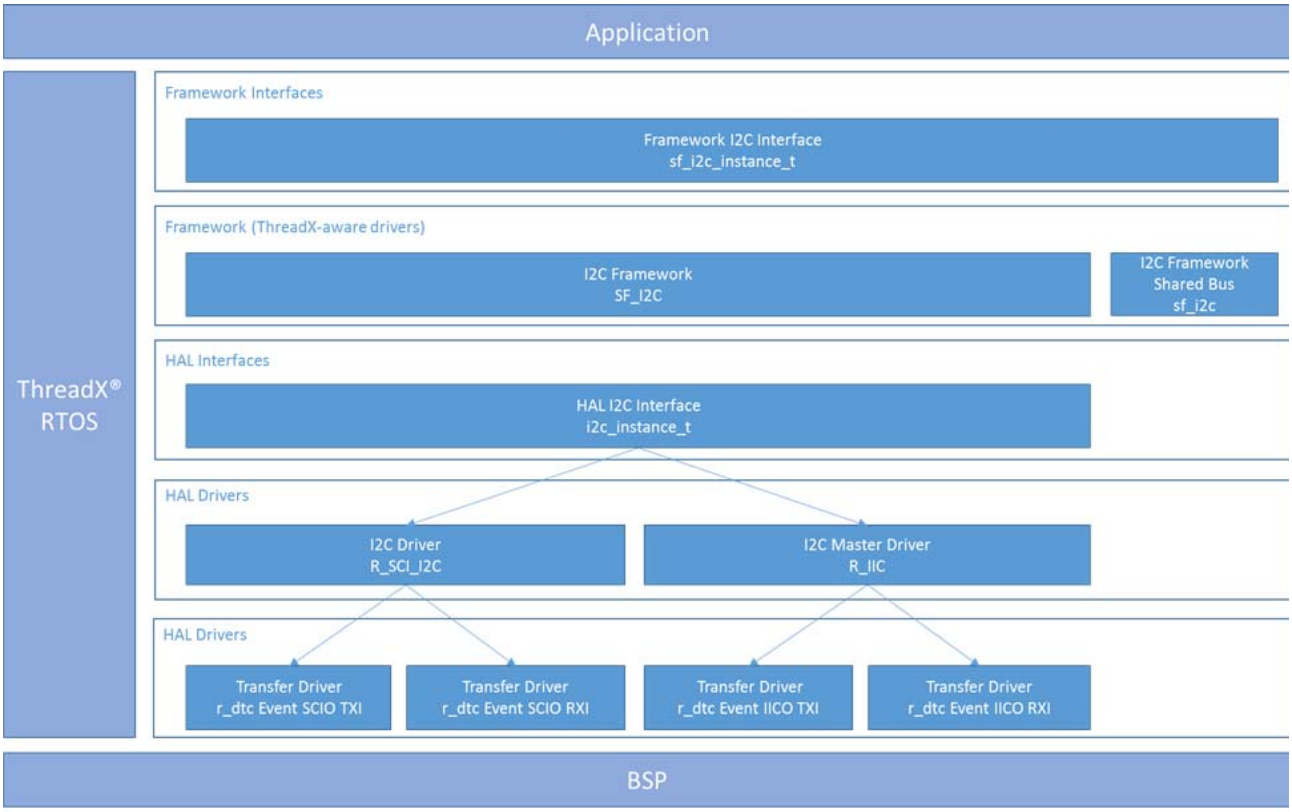


図 196: I2C フレームワークモジュールのブロック図

5.1.17.1 I2C フレームワークモジュール API の概要

I2C フレームワークインタフェースは、I2C フレームワークを使用したバスのオープン、クローズ、リード、ライト、ロック、ロック解除、リセットの API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

I2C フレームワークモジュール API の要約

Function Name	API の呼び出し例と定義
	g_sf_i2c_device.p_api->open(g_sf_i2c_device.p_ctrl, g_sf_i2c_device.p_cfg) 指定された I2C デバイスをバス上で開きます。
	g_sf_i2c_device.p_api->close (g_sf_i2c_device.p_ctrl); 制御ハンドルで指定された I2C デバイスを無効化します。バスにデバイスが接続されていない場合は、バスが使用する RTOS サービスを終了します。
	g_sf_i2c_device.p_api->read (g_sf_i2c_device.p_ctrl, &reg, no_of_bytes_to_read, 0, TX_WAIT_FOREVER); I2C デバイスからデータを受信します。
	g_sf_i2c_device.p_api->write (g_sf_i2c_device.p_ctrl, command, no_of_bytes_to_write, false, TX_WAIT_FOREVER); I2C デバイスにデータを送信します。
	g_sf_i2c_device.p_api->lock (g_sf_i2c_device.p_ctrl); バスをデバイスにロックします。ロックするとロックを解除するまでバスが予約され、割り込みなしで複数のリードとライトを実行できます。
	g_sf_i2c_device.p_api->unlock (g_sf_i2c_device.p_ctrl); バスを特定のデバイスからロック解除し、他のデバイスで使用できるようにします。
	g_sf_i2c_device.p_api->reset (g_sf_i2c_device.p_ctrl, TX_NO_WAIT); 進行中の転送を中止し、I2C ペリフェラルを強制的にレディ状態にします。
	g_sf_i2c_device.p_api->version(version); バージョンポインタを使用してバージョン情報を取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	I2C 関数が正常に実行されました
SSP_ERR_INVALID_MODE	指定された I2C モードが不正です
SSP_ERR_INVALID_CHANNEL	指定された I2C チャネルは除外されています
SSP_ERR_IN_USE	I2C チャネルが既に開かれています
SSP_ERR_INVALID_ARGUMENT	引数が、事前に定義された値に含まれていません
SSP_ERR_INVALID_POINTER	NULL pointer(s) が指定されています
SSP_ERR_INTERNAL	内部エラーが発生しました
SSP_ERR_ASSERTION	この関数を呼び出す前に、呼び出し側が制御ハンドルをクリアする必要があります。
SSP_ERR_NOT_OPEN	デバイスインスタンスが開かれていません
SSP_ERR_TRANSFER_ABORTED	データ転送が中止されました
SSP_ERR_INVALID_RATE	要求されたレートを設定できません
SSP_ERR_TIMEOUT	タイムアウトエラーが発生しました。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.17.2 I2C フレームワークモジュールの動作の概要

I2C フレームワークモジュールは、SSP の階層化されたドライバーアーキテクチャに準拠しています。低レベルの I2C HAL モジュールを使用して I2C と通信し、ユーザーの構成に従って Synergy マイクロコントローラ上の I2C 対応ペリフェラルを制御します。I2C フレームワークモジュールを使用すると、1 つ以上の I2C バスを作成し、複数の I2C をそれぞれの I2C バスに接続できます。I2C フレームワークモジュール API は、ThreadX-Mutex を使用して、I2C スレーブデバイスの共有バスを取得および解放します。取得とリリースは、それぞれ I2C フレームワークモジュールの lock および unlock API によって実装されます。

I2C フレームワークモジュールはマスタモードで I2C 通信を構成するため、ユーザーは次の操作を行うことができます。

- スレーブデバイスからの読み取り
- スレーブデバイスへの書き込み
- I2C のリセット

- このインタフェースでは、コールバックもサポートしています。
- I2C バスのロック
- I2C バスのロック解除

I2C フレームワークモジュールは、Synergy MCU I2C ハードウェアモジュール（RIIC および SCI HAL モジュール）と連携します。両方の I2C モジュールが、最大ビットレート 400kbps の I2C 高速モードをサポートしています。 IIC と RIIC HAL モジュールは、ビットレートが 1Mbps の高速モードプラスをサポートしています。このモジュールは、両方の実装について、マスタモードのみをサポートしています。

I2C フレームワークモジュールは、バスとバス上のデバイスによるアーキテクチャを使用します。どのデバイスも、接続先のバスに関連付けられています。ユーザーは、フレームワーク共有バスと、バスに接続されている各フレームワークデバイスの低レベル I2C HAL レイヤを構成する必要があります。ユーザーは、複数のデバイスを同じバス上で構成するときに既存のフレームワーク共有バスを追加することができます。共通の開始および停止手順が、すべての I2C データ転送操作に使用されます。低レベルに対してはデバイスを 1 つだけ構成します。残りのデバイスは、フレームワーク内のデバイスアドレスを切り替えることによって、リードまたはライト動作を実行します。

同じバス上のすべての I2C フレームワークデバイスは、同じ低レベル構成設定（I2C HAL モジュールなど）を使用しますが、スレーブアドレスとアドレシングモードは例外です。フレームワークは、アプリケーションで最初に開かれるデバイスの構成を使用して、バスを構成します。つまり、同じバス上のすべての I2C フレームワークデバイスは、低レベル構成設定が同じでなければなりません（スレーブアドレスとアドレシングモードを除く）。異なる構成が使用されると、適切な動作は保証されません。

I2C フレームワークはバスロック機能をサポートするため、特定のペリフェラルに対してバスをロックできます。ロックすると、一定時間（ロックとロック解除の間）、デバイスがバスを予約できるようになります。これにより、（一部のニーズに対応して）デバイスが複数のリードおよびライトを中断することなく完了できるようになります。

I2C フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 現在の PCLKB 設定で達成可能な最も近いボーレート（要求されたレートより小さいか等しいもの）が計算されて使用されます。有効なクロックレートを計算できなかった場合は、エラーが返されます。
- I2C は、ELC から使用可能な他のペリフェラルの起動をトリガできます。詳細については、『ELC Module Guide』を参照してください。
- すべてのデバイスに対しクロックレートが同じに保たれる場合、I2C フレームワークは、同一バス上の複数の I2C デバイスをサポートできます。つまり、クロックレートが同じであれば、同一バス内で複数のデバイスを開くことができます。デバイスのクロックレートが異なる場合、一度に開くことのできるデバイスは 1 つのみです。
- SCI で I2C を使用する場合、SDA および SCL 出力ピンタイプは、n チャネルオープンドレインである必要があります。
- I2C フレームワーク構成では、このバスに指定されたチャネル番号により HAL モジュールで指定されたチャネル番号が上書きされます。
- 共有バスは、それぞれの構成を持つ複数のスレーブデバイスで使用できます。複数のデバイスが同じ I2C チャネルを使用している場合、フレームワークは lock および unlock API で相互排除も処理します。
- 複数の I2C デバイスを同じバス上で構成するには、バスに接続する各デバイスについて、以下のモジュールを追加して構成します。
  - I2C フレームワークデバイスモジュール

  - 最初のデバイスの構成で I2C 共有バスモジュールを構成してから、残りのデバイスに対して同じバスを使用します。
  - I2C HAL モジュール
  - DTC module(Optional)
- ロック機能は、異なるスレッドのデバイスに対して有効になります。バスに接続されている複数のデバイスが同じスレッドからのものである場合、I2C バスはそのスレッドのすべてのデバイスに対してロックされます。このような場合、バスがロックされると、同じスレッドのすべてのデバイスがバスにアクセスできます。

注: *各 I2C フレームワークデバイスは、ISDE コンフィギュレータで一意の名前を使用して構成する必要があります。*

注: *同じバス上で接続されているすべてのデバイスに同じ構成設定を指定する必要があります（スレーブアドレスとアドレシングモードは例外です。）*

I2C フレームワークモジュールでは現在、以下の機能はサポートされていません。

- GPT 以外のタイマの使用
- DMA の使用
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.17.3 アプリケーションへの I2C フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して I2C フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注: *このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

I2C フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（I2C フレームワークモジュールのデフォルト名は g_sf_i2c_device0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

I2C フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_i2c_device0 I2C Framework on sf_i2c	Threads	New Stack> Framework> Connectivity> I2C Device Framework on sf_i2c

次の図に示すように、sf_i2c の I2C フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタン

ドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、 1 回のみ追加すれば複数のスタックで使用できます。

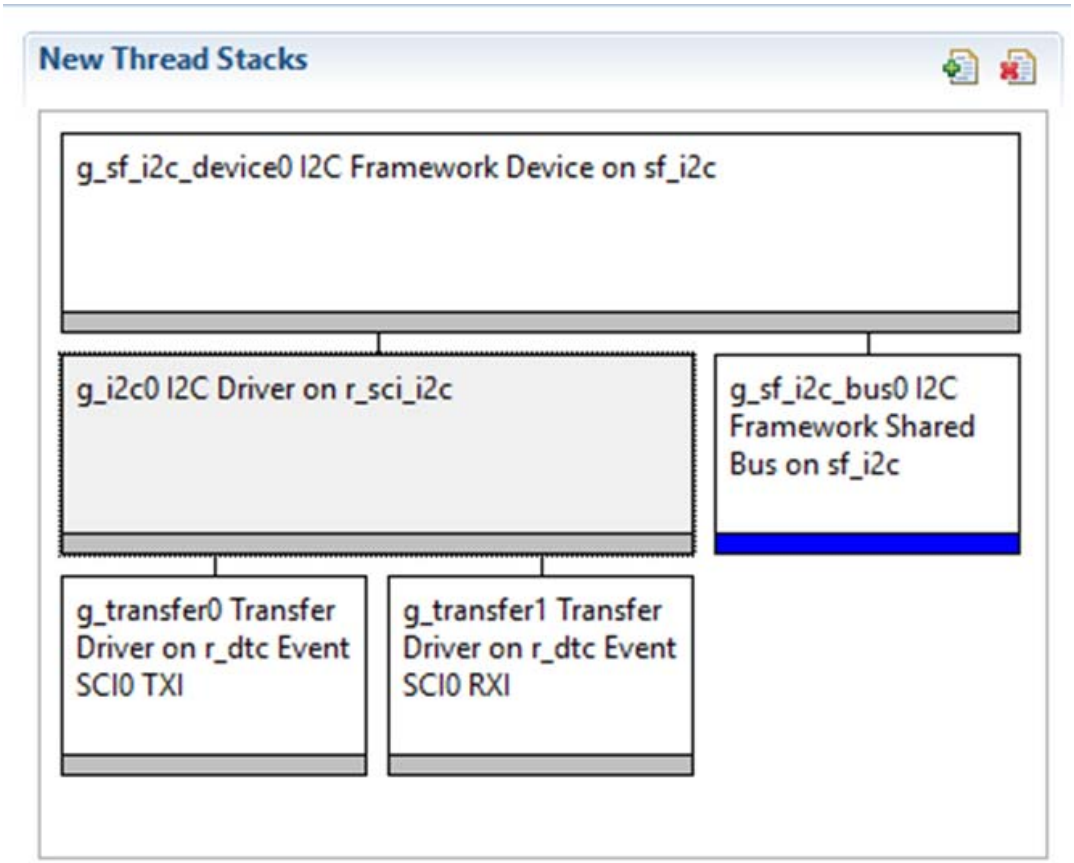


図 197: I2C フレームワークモジュールのスタック

5.1.17.4 I2C フレームワークモジュールの設定

必要な動作に合わせて I2C フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ター

ゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_i2c 上の I2C フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	パラメータエラーチェックを有効または無効にします。
Name	g_sf_i2c_device0	I2C フレームワークデバイスを識別するための名前を設定します。この名前に基づいて、API、Config および Control インスタンスが作成されます。

注: *設定例とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、異なるバイト順序やピクセルカラーフォーマットの選択が役立つ場合です。低レベルモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *低レベルモジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。*

I2C フレームワークのローレベルドライバーの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_sci_i2c 上の I2C マスタ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_i2c0	モジュール名。
Channel	0 to 9	この設定で使用する SCI チャネルを指定します。 SCI には以下のチャネルがあります:Series S7: 0 1 2 3 4 5 6 7 8 9; Series S3: 0 1 2 3 4 - - - - 9; Series S1: 0 1 - - - - - - - 9。
Rate	Standard, Fast-mode Default: Standard	標準および高速。
Slave Address	0x00	スレーブデバイスのアドレス。
Address Mode	7-Bit, 10-Bit Default: 7-Bit	現在、7ビットアドレスのみがサポートされています。
SDA Output Delay (nano seconds)	300	SDA 出力遅延 （ナノ秒単位）。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数を有効化します。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 i2c_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択。

ISDE Property	Value	説明
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信終了割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_riic 上の I2C マスタ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_i2c0	モジュール名。
Channel	0 to 9	この設定で使用する SCI チャネルを指定します。 SCI には以下のチャネルがあります : Series S7: 0 1 2 3 4 5 6 7 8 9; Series S3: 0 1 2 3 4 - - - - 9; Series S1: 0 1 - - - - - - - 9。
Rate	Standard, Fast-mode Default: Standard	標準および高速。
Slave Address	0x00	スレーブデバイスのアドレス。

ISDE Property	Value	説明
Address Mode	7-Bit, 10-Bit Default: 7-Bit	現在、7ビットアドレスのみがサポートされています。
SDA Output Delay (nano seconds)	300	SDA 出力遅延 （ナノ秒単位）。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数を有効化します。
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 i2c_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択。

ISDE Property	Value	説明
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信終了割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

sf_i2c 上の I2C フレームワーク共有バスの構成設定

ISDE Property	Value	説明
Name	g_sf_i2c_bus0	バスを識別するための名前を設定します。このバス名は、フレームワーク設定で I2C をバスに関連付けるために使用されます。
I2C Implementation	SCI I2C, RIIC Default: RIIC	フレームワークで使用する低レベルインタフェースを選択します。

ISDE Property	Value	説明
Channel	0-9	デバイスが接続されている SCI または RIIC チャネル番号。フレームワークでは、このバス設定で指定されたチャネル番号により HAL ドライバーで指定されたチャネル番号が上書きされます。

注: 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc イベント SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択

ISDE Property	Value	説明
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI 時の転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。

ISDE Property	Value	説明
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

***注:** 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント IIC0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IICI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Value	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid)	ELC ソフトウェアイベント割り込み優先順位の選択。

r_dtc イベント IIC0 RXI 時の転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択

ISDE Property	Value	説明
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IIC0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

r_dtc イベント IIC0 RXI 時の転送ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。

ISDE Property	Value	説明
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IIC0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

***注:** 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

I2C フレームワークモジュールのクロック構成

SCI 周辺モジュールは PCLKB をそのクロックソースとして使用します。 PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、ランタイムで CGC インタフェースを使用

します。構成時に、I2C 転送レートは、ユーザー選択 PCLB レートとユーザー選択転送レートに基づいて、ドライバーにより内部で計算および設定されます。ユーザー選択レートを実現できないような構成が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

I2C フレームワークモジュールのピン構成

SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

SCI1 のピン選択シーケンス

Resource	ISDE Tab	Pin Selection Sequence
SCI	Pins	Select Peripherals > SCI1_3_5_7_9 > SCI1

***注:** 選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。*

SCI 上の I2C フレームワークモジュールのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled	SCI 上の I2C の動作モードとして簡易 I2C を選択します
RXD1_SCL1_MISO1	None, P212, P708 (Default: None)	SCL ピン
TXD1_SDA1_MOSI1	None, P213, P709 (Default: None)	SDA ピン

***注:** 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

I2C フレームワークモジュールのその他の設定

SCL ピンと SDA ピンに加えて、I2C スレーブデバイスをリセットするために I2C RESET 信号が必要な場合があります。この場合、RESET 信号は GPIO を使用して追加でき、アプリケーションプログラムで直接制御する必要があります。外部デバイスのリセット機能は、r_sci_i2c モジュール内でサポートされていません。

5.1.17.5 アプリケーションでの I2C フレームワークモジュールの使用

モジュールの構成が完了し、ファイルが生成され、I2C ピンが構成されると、I2C フレームワークモジュールはアプリケーションで使用できる状態になります。アプリケーションで I2C モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して I2C モジュールを初期化します。
2) reset API を使用してモジュールをリセットします（必要な場合）
3) lock API を使用してバスをロックします（必要な場合）
4) write API を使用してチャネルを構成します
5) unlock API を使用してバスをロック解除します（必要な場合）
6) リードが必要な場合は、 lock API を使用してバスをロックします（必要な場合）
7) プログラムは read API を使用してバッファからのリードを開始します
8) unlock API を使用してバスをロック解除します（必要な場合）
9) 必要に応じて close API を使用してモジュールを閉じることができます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

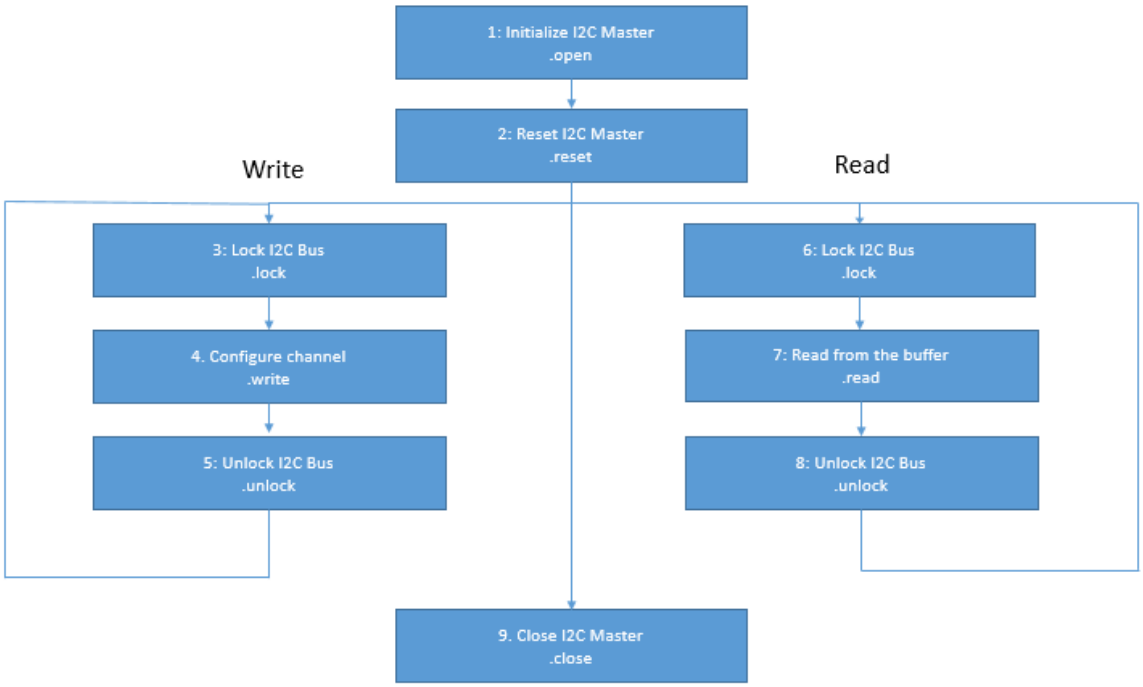


図 198: 通常の I2C フレームワークモジュールアプリケーションのフロー図

5.1.18 JPEG デコードフレームワーク

- Synergy JPEG ハードウェアへのスレッドセーフなアクセスを提供します。
- JPEG デコード HAL モジュールを使用した JPEG 解凍をサポート。
- JPEG　デコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直サブサンプル値、水平ストライド、デコードされたピクセルフォーマット、入力および出力データフォーマット、色空間などのパラメータを設定。
- イメージをデコードする前にそのサイズを取得。
- デコードされたイメージフレームの保管のため、コード化されたデータを入力バッファーおよび出力バッファーに格納する操作をサポート。
- JPEG デコーダモジュールへのコード化されたデータのストリーム転送をサポート。この機能により、アプリケーションは、ファイルやネットワークからのコード化された JPEG イメージを、イメージ全体をバッファリングすることなく読み取ることができます。
- デコードするイメージ行数を設定。この機能により、アプリケーションは、デコードされたイメージの処理を、フレーム全体をバッファーリングすることなくオンザフライで実行可能。
- 入力値のデコードされたフォーマットとして YCbCr444、YCbCr422、YCbCr420、YCbCr411 をサポート。
- 出力値のデコードされたフォーマットとして ARGB8888 と RGB565 をサポート。
- JPEG イメージのサイズ、高さ、幅が要件を満たさない場合にエラーを返すことが可能。
- JPEG　ハードウェアサポートイベントと同期するためのスレッドをサスペンドまたは再開するための wait API 関数をサポート。

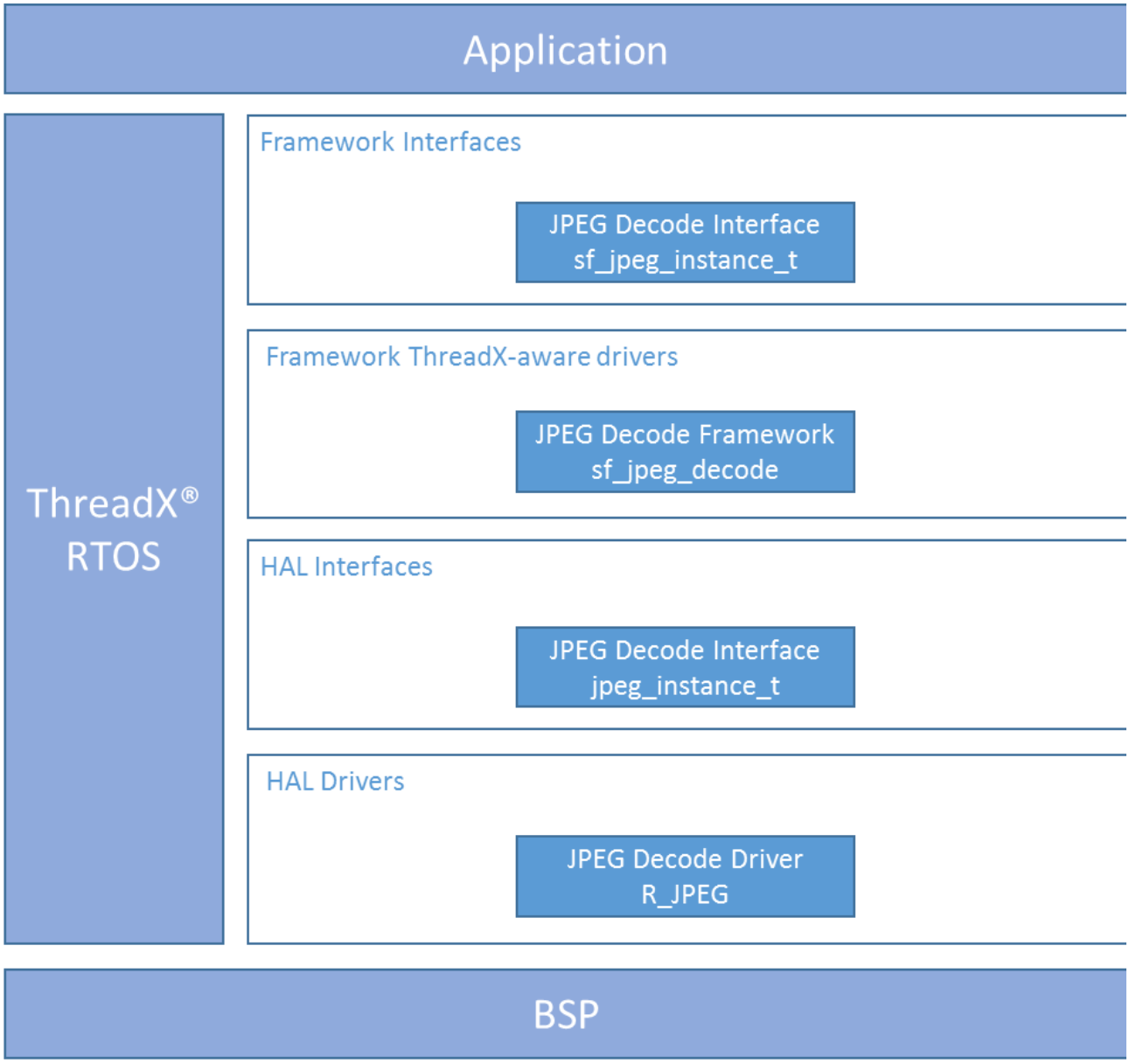


図 199: JPEG デコードフレームワークモジュールのブロック図

5.1.18.1 JPEG デコードフレームワークモジュール API の概要

JPEG デコードフレームワークモジュールは、オープン、クローズ、アラーム設定、RTC 操作の開始と停止の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

JPEG デコードフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_jpeg_decode.p_api->open(g_sf_jpeg_decode.p_ctrl, g_sf_jpeg_decode.p_cfg); JPEG デコードフレームワークを開きます。
close	g_sf_jpeg_decode.p_api->close(g_sf_jpeg_decode.p_ctrl); JPEG デコードフレームワークを閉じます。
inputBufferSet	g_sf_jpeg_decode.p_api->inputBufferSet(g_sf_jpeg_decode .p_ctrl, &buffer, size); JPEG コーデックにデータを供給します。
outputBufferSet	g_sf_jpeg_decode.p_api->outputBufferSet(g_sf_jpe\g_decode.p_ctrl, &buffer, size); JPEG コーデックから処理済みのデータを読み取ります。
linesDecodedGet	g_sf_jpeg_decode.p_api->linesDecodedGet(g_sf_jpeg_decode.p_ctrl, &lines); デコードされたライン数を取得します。
horizontalStrideSet	g_sf_jpeg_decode.p_api->horizontalStrideSet(g_sf_jpeg_decode.p_ctrl, stride); ストライド値を構成します。
imageSubsampleSet	g_sf_jpeg_decode.p_api->imageSubsampleSet(g_sf_jpeg_decode.p_ctrl, h_sub, v_sub); カレンダカウンタを開始します。
wait	g_sf_jpeg_decode.p_api->wait(g_sf_jpeg_decode.p_ctrl, timeout); 現在の操作を待機します。
statusGet	g_sf_jpeg_decode.p_api->statusGet(g_sf_jpeg_decode.p_ctrl, &status); JPEG コーデックのステータスを取得します。
versionGet	g_sf_jpeg_decode.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	JPEG デコードドライバーが正常に開かれました。
SSP_ERR_ASSERTION	アサートエラーです。

Name	説明
SSP_ERR_IN_USE	モジュールは既に使用中です。
SSP_ERR_TIMEOUT	待機操作がタイムアウトし、基礎となるドライバーが時間内に応答しませんでした。
SSP_ERR_WAIT_ABORTED	システム内部エラーが発生しました。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの **API リファレンス** *を参照してください。*

5.1.18.2 JPEG デコードフレームワークモジュールの動作の概要

JPEG デコードフレームワークモジュールは、標準 JPEG デコード操作を実装します。入力バッファのデータを取得し、定義済みの JPEG デコードアルゴリズムをバッファに適用します。出力は、定義済みの出力バッファの場所に配信されます。JPEG ハードウェアサポートイベントと同期するためのスレッドのサスペンドや再開には wait API 関数を使用できます。

JPEG デコードフレームワークモジュールの動作に関する重要な注意事項と制限事項

- JPEG エンコードされたデータのデコードは、open API を呼び出すことで開始できます。モジュールを開くには、JPEG デコードフレームワークモジュールインスタンスを使用します。このインスタンスには、API 関数のポインタ、制御ブロックへのポインタ、ISDE の e^2 studio の Synergy プロジェクトコンフィギュレータを通じて生成される静的構成が含まれています。
- JPEG デコードフレームワークモジュールを停止するには、 close API を呼び出します。
- 入力バッファストリーミングモードは、入力中心の機能が必要な場合に使用できます。
- 出力バッファストリーミングモードは、出力中心の機能が必要な場合に使用できます。
- 出力データのカラーフォーマットとして RGB565 と ARGBB888 をサポートします。
- JPEG　デコードフレームワークモジュールの制御ブロックにはステータスフラグがあり、ユーザーは statusGet API を通じてモジュールの現在のステータスを取得することができます。また、モジュールで特定のイベントが発生すると、ユーザーコールバック関数を通じてステータスがレポートされます。
- JPEG デコードフレームワークモジュールは、入力バッファがソースイメージファイルよりも小さい場合に入力バッファのバッファストリーミングモードをサポートします。ハードウェアで生成された INPUT_PAUSE 割り込みが発生するたびに、次の入力フレームが入力バッファとして設定されます。
- JPEG デコードフレームワークモジュールは、結果のイメージが出力バッファサイズよりも大きい場合に出力バッファのバッファストリーミングモードをサポートします。ハードウェアで生成された OUTPUT_PAUSE割り込みが発生するたびに、次のデータ用のスペースを空けるために出力バッファからデータが読み取られ、格納されます。
- JPEG デコードフレームワークモジュールが正常に動作するためには、入力および出力バッファが 8 バイトでアラインされている必要があります。そうでないと、API は実行に失敗したことを示すエラーコードを返します。
- JPEG フレームワークモジュールは、 JPEG エンコード処理をサポートしていません。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.18.3 アプリケーションへの JPEG デコードフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに JPEG デコードフレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

JPEG デコードフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（JPEG デコードフレームワークモジュールのデフォルト名は g_sf_jpeg_decode0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

RTC 選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_jpeg_decode0 JPEG Framework	Threads	New Stack> Framework> Graphics> JPEG Decode Framework on sf_jpeg_decode

次の図に示すように、sf_jpeg_decode の JPEG デコードフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。必要な特定の設定は、強調表示されたテキストにカーソルを移動することで表示できます。または、強調表示されたスタックフレームで推奨されます。このスタックでは、レポートされる JPEG HAL モジュールの要件は、解凍割り込み優先順位とデータ転送割り込み優先順位の割り込みを有効にすることです。

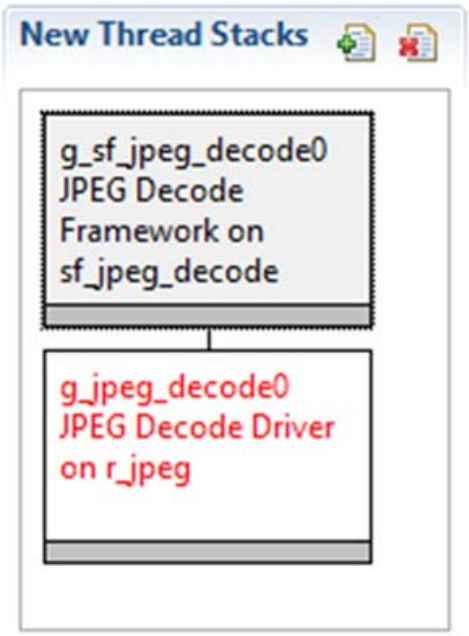


図 200: JPEG デコードフレームワークモジュールのスタック

解凍プロセス割り込み (JEDI)

JPEG 解凍プロセス割り込みは、次のイベントが検出されたときに発生します。

- 現在の解凍プロセスが正常に完了した。
- 解凍プロセス中にエラーが発生した。
- イメージのサイズとピクセルフォーマットが正常に読み出された。

データ転送インタフェース (JDTI)

JPEG データ転送割り込みは、次のイベントが検出されたときに発生します。

- JPEG コードされたデータがすべて正常に完了した。
- linesDecodedGet で指定された出力イメージデータ行数が転送された。
- inputBufferSet で指定された入力イメージデータ行数が転送された。

5.1.18.4 JPEG デコードフレームワークモジュールの構成

ユーザーは必要な動作に合わせてJPEGデコードフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべての

プロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

***注:** ISDE を開き、JPEG デコードフレームワークを作成し、次の表に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_jpeg 上の JPEG HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_jpeg_decode0	JPEG デコードモジュールインスタンスに使用する名前。
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) (Default: Normal Byte order)	入力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) (Default: Normal Byte order)	出力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format (Default: Pixel Data RGB565 format)	出力データフォーマットを指定します。

ISDE Property	Value	説明
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	出力データフォーマットのアルファ値を指定します（ARGB8888 フォーマットに対してのみ有効）。
Name of user callback function	NULL	ユーザーコールバック関数の名前を指定します。
Decompression Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	JPEG JEDI 割り込みの選択
Data Transfer Interrupt priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	JPEG JDTI 割り込みの選択

JPEG デコードフレームワークモジュールのクロック構成

JPEG フレームワークモジュールは、周辺モジュールクロック A（PCLKA）を使用して内部論理を実行します。

JPEG デコードフレームワークモジュールの割り込み構成

割り込みを有効化するには、ISDE の JPEG デコードフレームワークモジュールの [Properties] ウィンドウで解凍割り込みとデータ転送割り込みのプライオリティを設定します。

JPEG デコードフレームワークモジュールのピン構成

JPEG デコードフレームワークモジュールはピンを使用しません。

5.1.18.5 アプリケーションでの JPEG デコードフレームワークモジュールの使用

アプリケーションで JPEG デコードフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して JPEG デコードを初期化します。
2) imageSubSampleSet API を使用してイメージサブサンプルを設定します。
3) horizontalStrideSet API を使用して水平ストライドを設定します。
4) outputBufferSet API を使用して出力バッファを設定します。
5) inputBufferSet API を使用して入力バッファを設定します。
6) wait API を使用してデコードの完了を待ちます。
7) statusGet API を使用してデコードステータスを確認します。
8) close API を使用してインスタンスを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

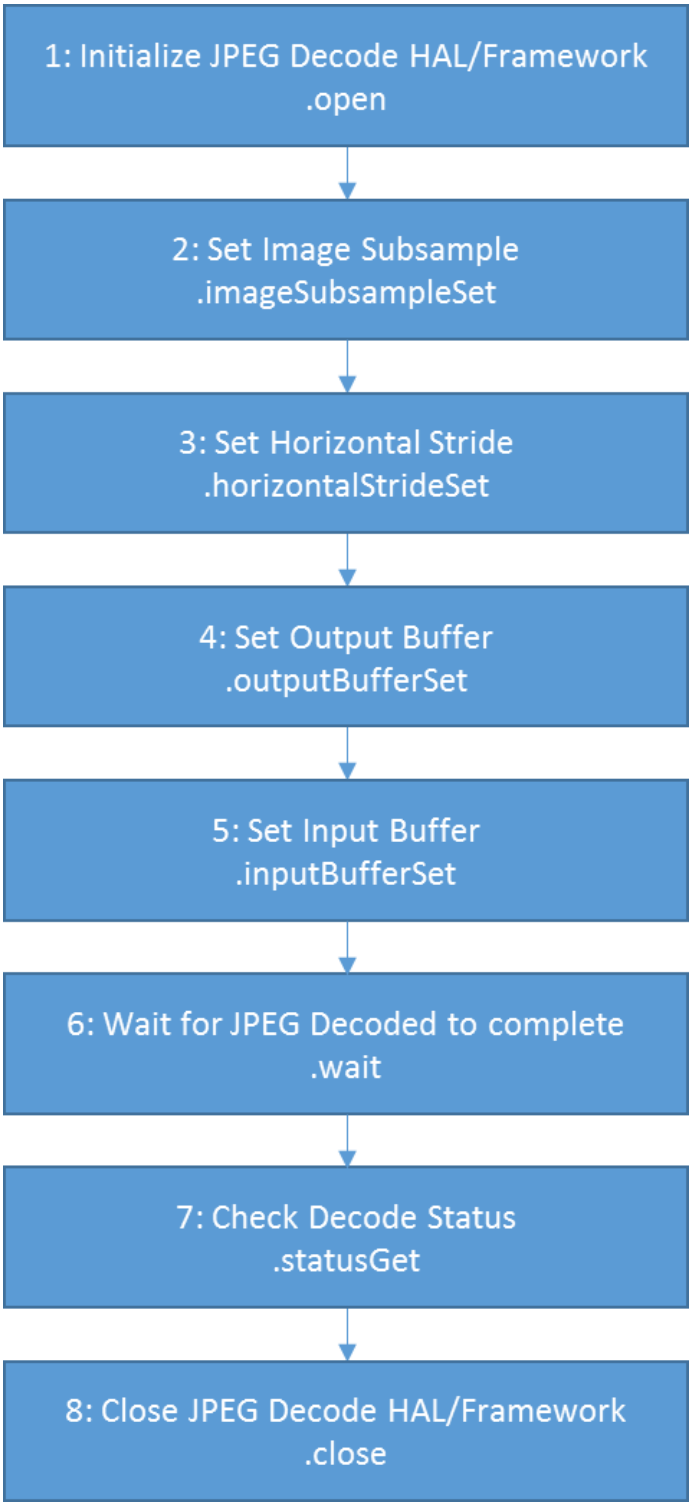


図 201:　通常の JPEG HAL およびフレームワークモジュールアプリケーションのフロー図

5.1.19 メッセージングフレームワーク

メッセージフレームワークモジュールは以下の機能をサポートしています。

- スレッド間通信-異なるデバイスを制御したりサブシステムを管理したりするアプリケーションスレッドが互いに通信できます。
- パブリッシュ / サブスクライブ方式 - フレームワークの設計は、疎結合型のメッセージング方式に基づいています。複数のスレッドが 1 つのイベントクラスを受信待ちできる設計になっています。メッセージ作成元スレッドは、イベントクラスのメッセージにサブスクライブしているスレッドを知る必要がありません。サブスクライバーは、メッセージの作成元を知る必要がありません。
- メッセージ管理 - フレームワークでは、バッファを制御するフラグやハンドシェイクのためのコールバック関数ポインタなど、各メッセージを管理するためのバッファ制御ブロックがサポートされています。
- メッセージバッファリング - フレームワークは、メッセージング用のバッファの割り当てと解放を管理します。アプリケーションは、割り当てられたバッファを利用してメッセージを書き込んだり、不要になったメッセージを破棄することができます。
- 同期通信 - フレームワークでは、ThreadX メッセージキューを使用した非同期メッセージングがサポートされていますが、メッセージ作成元とサブスクライバスレッドの間のハンドシェイクを確立するためのオプションも提供されています。ハンドシェイクは、サブスクライバスレッドから作成元スレッドのユーザーコールバック関数を呼び出すことで実装されています。
- メッセージ構造の指定 - 定義済みの共通メッセージヘッダが提供されています。また、いくつかの典型的なペイロード構造体テンプレートも例として提供されています。
- メッセージのプライオリティ - 高プライオリティのメッセージを送信して、サブスクライバスレッドが、メッセージキューにある他のメッセージの前にそのメッセージを受け取れるようにすることができます。

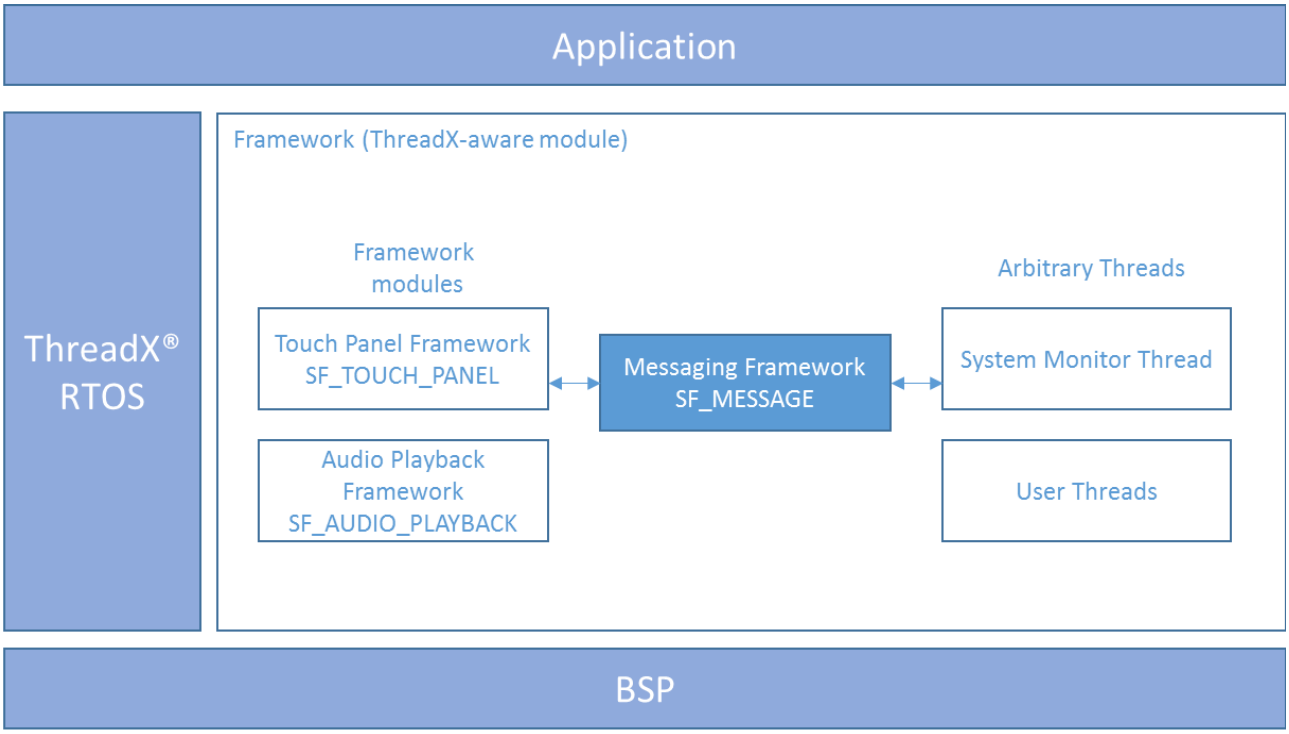


図 202: メッセージフレームワークモジュールの編成および使用例

5.1.19.1 メッセージフレームワークモジュール API の概要

メッセージフレームワークモジュールは、フレームワークのオープンとクローズ、バッファの取得と解放、サブスクライバへのメッセージのポスト用の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

メッセージフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_message.p_api->open (g_sf_message.p_ctrl, g_sf_message.p_cfg); メッセージフレームワークを初期化します。メッセージングフレームワーク制御ブロックを開始し、構成パラメータに対応する作業メモリを構成します。

Function Name	API の呼び出し例と説明
close	g_sf_message.p_api->close (g_sf_message.p_ctrl); メッセージフレームワークを終了処理します。
bufferAcquire	g_sf_message.p_api->bufferAcquire (g_sf_message.p_ctrl, &p_buffer, &acquire_cfg, wait_option); ブロックから渡されるメッセージ用に、バッファを取得します。
bufferRelease	g_sf_message.p_api->bufferRelease (g_sf_message.p_ctrl, &p_buffer, option); bufferAcquire から取得されたバッファを解放します。
post	g_sf_message.p_api->post (g_sf_message.p_ctrl, (sf_message_header_t *) p_payload, &post_cfg, &err_post, wait_option); サブスクライバーに対してメッセージをポストします。
pend	g_sf_message.p_api->pend (g_sf_message.p_ctrl, &my_queue, &p_buffer, wait_option); メッセージを保留します。
versionGet	g_sf_message.p_api->versionGet (&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。

Name	説明
SSP_ERR_ASSERTION	必要なポインタが NULL です。
SSP_ERR_BUFFER_RELEASED	バッファが解放されていません。
SSP_ERR_ILLEGAL_SUBSCRIBER_LISTS	不正なメッセージサブスクライバリストです。
SSP_ERR_IN_USE	メッセージフレームワークが使用中です。
SSP_ERR_INTERNAL	OS サービス呼び出しが失敗しました。
SSP_ERR_INVALID_MSG_BUFFER_SIZE	メッセージバッファサイズが無効です。
SSP_ERR_INVALID_WORKBUFFER_SIZE	作業バッファサイズが無効です。
SSP_ERR_MESSAGE_QUEUE_EMPTY	キューが空です。（タイムアウトオプションが指定されている場合は、メッセージの受信前にタイムアウトが発生します。）
SSP_ERR_MESSAGE_QUEUE_FULL	キューが一杯です。（タイムアウトオプションが指定されている場合は、メッセージの送信前にタイムアウトが発生します。）
SSP_ERR_NO_MORE_BUFFER	メモリブロックプールにバッファがこれ以上見つかりません。
SSP_ERR_NO_SUBSCRIBER_FOUND	サブスクライバーが見つかりません。
SSP_ERR_NOT_OPEN	メッセージフレームワークモジュールが開かれていません。
SSP_ERR_TIMEOUT	OS サービス呼び出しがタイムアウトを返しました。
SSP_ERR_TOO_MANY_BUFFERS	メッセージバッファが多すぎます。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

5.1.19.2 メッセージフレームワークモジュールの動作の概要

下図に、メッセージフレームワークモジュールを利用するシステム内の、メッセージ作成元スレッドとサブスクライバ thread(s) の間のメッセージングデータフローの概要を示します。

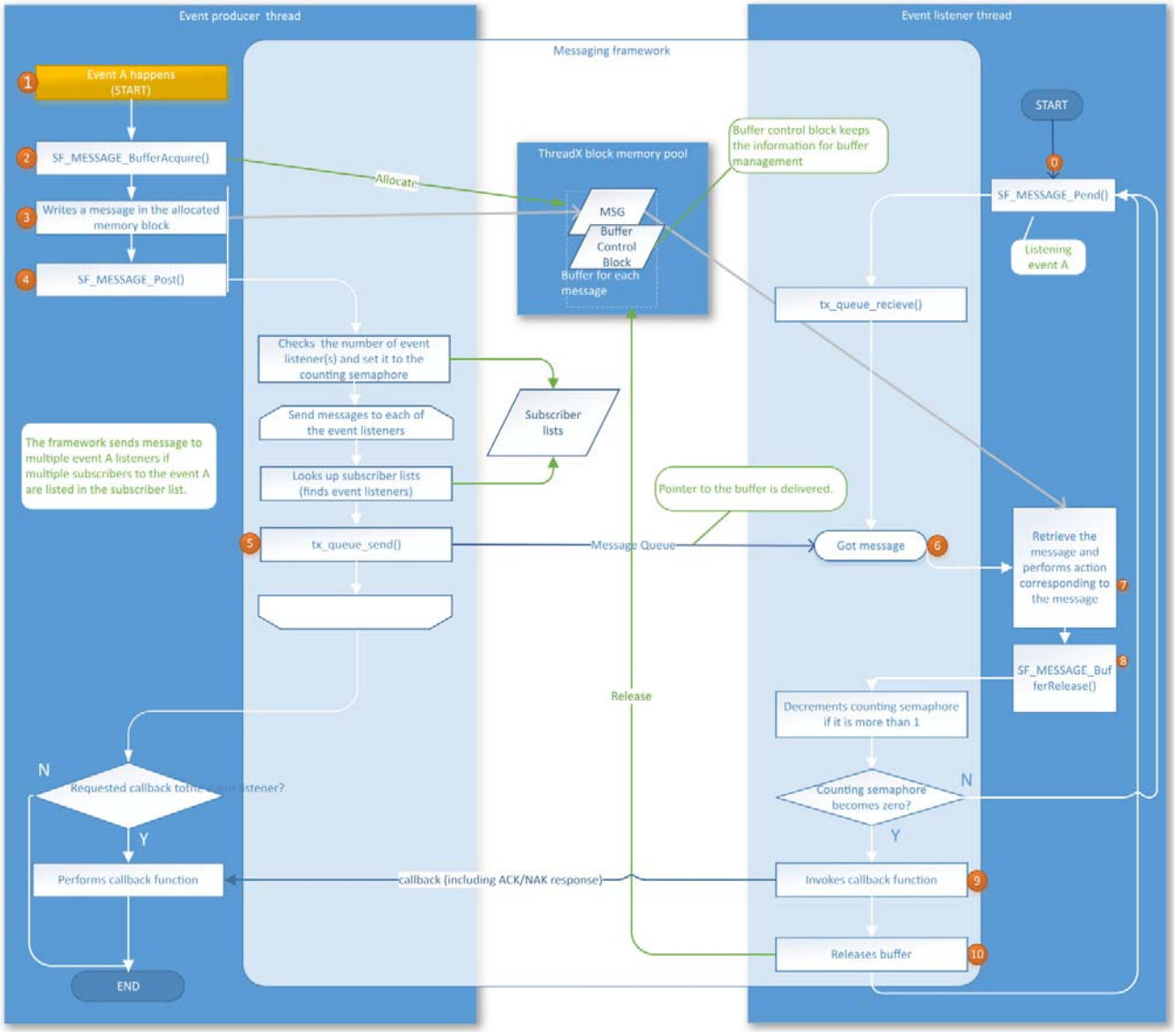


図 203: メッセージフレームワーク - データフロー

以下で、メッセージ送受信手順の各ステージについて説明します。

注: システムのスレッドが open API を使用して呼び出されており、さらにメッセージサブスクライバスレッドが pend API を呼び出して、イベントクラスのメッセージを待っています。

1) イベント（イベント A) がメッセージ作成元スレッドで発生します。
2) メッセージ作成元スレッドは、bufferAcquire を呼び出して、メッセージフレームワークモジュールが管理する ThreadX メモリプールからバッファを取得します。bufferAcquire は割り当てられたバッファのアドレスを返します。
3) メッセージ作成元が、割り当てられたバッファにメッセージを書き込みます。
4) メッセージ作成元が、 post を呼び出してメッセージをポストします。
5) メッセージフレームワークモジュールは、イベントサブスクライバリストを検索し、ThreadX メッセージキュープリミティブを使用して、メッセージサブスクライバスレッドのメッセージキューにメッセージを送信します。フレームワークは、バッファのポインタのみを送信し、メッセージ全体を送信せず、軽量なメッセージ送受信を行います。
6) メッセージがメッセージサブスクライバスレッドのメッセージキューに到達し、メッセージサブスクライバスレッドが pend から戻ります。 API 関数は、メッセージが格納されているバッファアドレスをメッセージサブスクライバスレッドに返します。
7) メッセージサブスクライバースレッドは、メッセージを受信し、イベントに応じたアクションを実行します。
8) メッセージサブスクライバースレッドは、bufferRelease を呼び出して、メッセージに割り当てられたバッファを解放しようとします。そのメッセージサブスクライバスレッドが、メッセージにサブスクライブしている最後のスレッドでない場合、フレームワークはバッファを解放しません。なぜなら、すべてのサブスクライバがメッセージを受信するまでメッセージをバッファ内に保持する必要があるためです。
9) メッセージフレームワークモジュールは、そのメッセージサブスクライバスレッドが、メッセージサブスクライバグループの最後のスレッドである場合、イベント作成元スレッドによって指定されたユーザーコールバック関数を呼び出します。
10) メッセージングフレームワークモジュールは、そのメッセージサブスクライバースレッドが、メッセージサブスクライバーグループの最後のスレッドである場合、バッファを解放します。（バッファを解放しないオプション 'SF_MESSAGE_ACQUIRE_OPTION_KEEP' があります。）

メッセージフレームワークモジュールのメッセージ作成元とサブスクライバ

メッセージフレームワークモジュールは、パブリッシュ / サブスクライブモデルに基づくスレッド間メッセージングシステムです。メッセージは、イベント作成元スレッドによって、イベントクラスコードとともにポストされます。メッセージサブスクライバスレッドは、イベントクラスにサブスクライブする保留中のメッセージを確認できます。サブスクライバは、フレームワークによって参照されるサブスクライバリストに登録されます。フレームワークは、サブスクライバリストを使用することで、メッセージを複数のサブスクライバに配信できます。

メッセージングフレームワークモジュールシステムネットワークに参加するすべてのスレッドと、ネットワーク内のすべてのスレッドがメッセージを受信待ちできます。

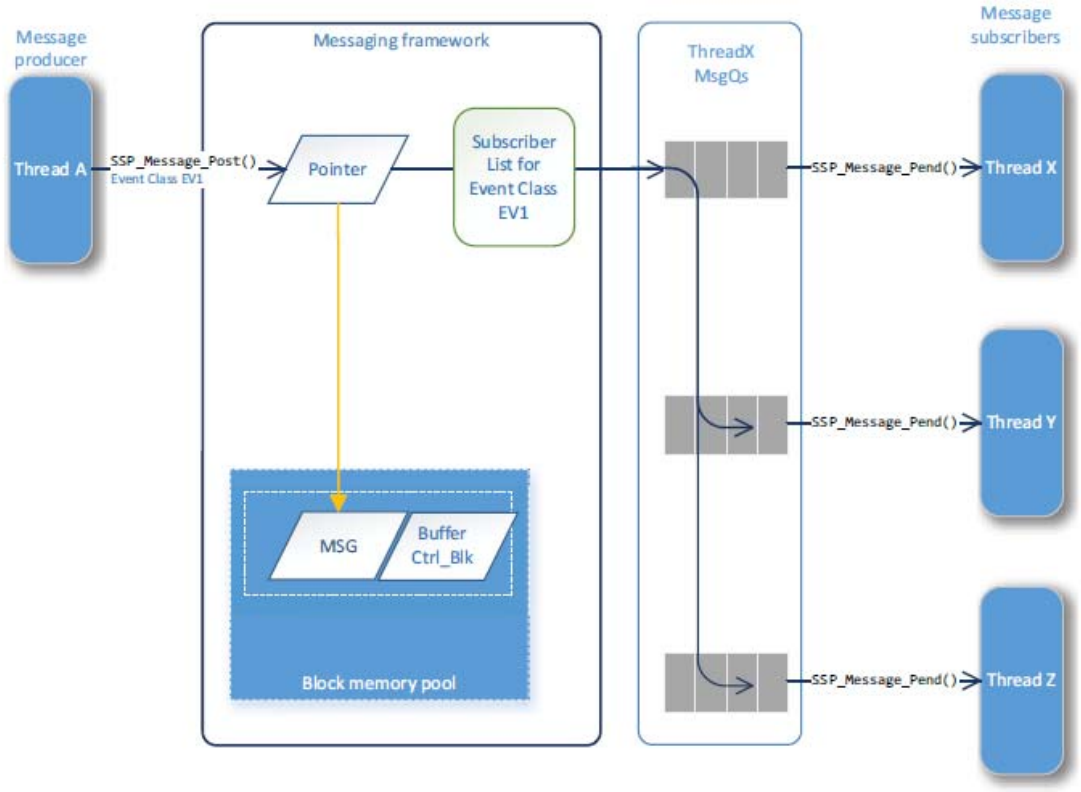


図 204:　メッセージフレームワーク - サブスクライブ

メッセージフレームワークモジュールのイベント、サブスクライバ、メッセージ

イベントクラスコードは、メッセージモジュールの最も重要な定義です。メッセージモジュールは、メッセージ作成元をサブスクライバに接続するメカニズムとしてイベントクラスコードを使用します。イベントクラスコードは、アプリケーションで発生するイベントのクラス定義です。イベントクラスの分類はユーザー定義に依存しますが、サブシステム内で発生する可能性がある特定のイベントのグループ名であることが想定されています。たとえば、以下のイベントクラスを使用できます。

- タッチサブシステムの一部である「touch」イベントクラス。 Touch イベントクラスは、イベントクラスのウィンドウに自動的にロードされます。このウィンドウは、Synergy プロジェクトにタッチパネルフレームワークを追加するときにプロジェクトコンフィギュレータの **[Messaging]** タブで使用できます。
- 「time」イベントクラスは、時間関連のアプリケーションを管理するサブシステムの一部です。

イベントクラスコードは、 sf_message_event_class_t 列挙で定義され、プレフィックス SF_MESSAGE_EVENT_CLASS_XXX. を持ちます。イベントクラスコードの定義はシステムごとに異なりますが、フレームワークでは具体的なイベントクラスコードが用意されておらず、代わりに一連のイベントクラスコードが例として提供されています。(「メッセージフレームワークモジュールの構成」セクションを参照してください。) イベントクラスの最大数は 255 です。

アプリケーションは、イベントクラスコードを以下のように使用できます。

- メッセージ作成元スレッドは、イベントクラスコードを sf_message_header_t 型の共通メッセージヘッダの event_b.class_code ビットフィールドに設定してからメッセージをポストします。
- メッセージサブスクライバスレッドは、メッセージを受信した後、メッセージヘッダに設定されているイベントクラスコードに従ってイベント処理を分岐します。
- イベントクラスコードのサブスクライバは、メッセージフレームワークがメッセージをサブスクライバに配信できるように、グループ化されてサブスクライバリストに登録されている必要があります。

次の図は、ISDE を使用してイベントクラスを構成する方法を示しています。

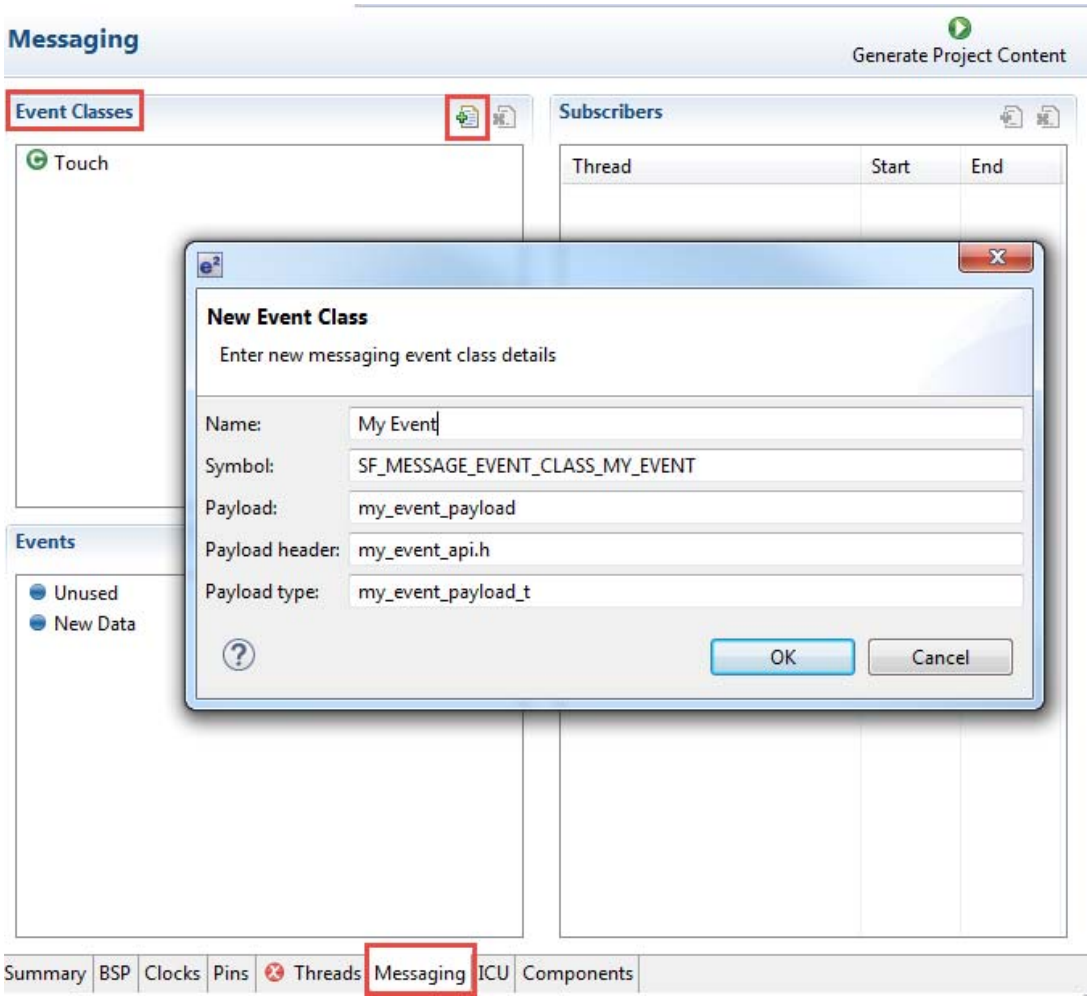


図 205: メッセージング –ISDE イベントクラスの構成

メッセージフレームワークモジュールのイベントクラスインスタンス番号

イベントクラスインスタンス番号は、アプリケーションが異なるイベントクラスインスタンスを必要とするときに使用されます。たとえば、オーディオストリーミングイベントクラスでは、ストリーミングチャネル

N を示すインスタンス *N* を使用できます。メッセージサブスクライバは、メッセージの共通ヘッダにあるイベントクラスインスタンス番号が、所有する番号と一致する場合にのみメッセージを受信できます。

言い換えれば、イベントクラスインスタンス番号がサブスクライバの範囲外であるメッセージはフィルタ処理で除外され、イベントクラスサブスクライバであっても、そのサブスクライバには配信されません。イベントクラスインスタンス番号の最大値は 255 です。

注: *タッチパネルフレームワークには通常、1 つしかインスタンスがないため、イベントクラスインスタンス番号は 0 となり、サブスクライバリストの開始値と終了値が 0 に設定されます。*

アプリケーションは、イベントクラスインスタンス番号を以下のように使用できます。

- メッセージ作成元スレッドは、イベントクラスインスタンス番号を sf_message_header_t 型の共通メッセージヘッダの event_b.class_instance ビットフィールドに設定してからメッセージをポストします。
- サブスクライバリスト内の各サブスクライバインスタンスは、メッセージを受信するにはイベントクラスインスタンス番号の範囲を指定する必要があります (sf_message_subscriber_t:: instance_range.start および sf_message_subscriber_t::instance_range.end).
- イベントクラスに複数インスタンスが必要でない場合は、サブスクライバリストのサブスクライバインスタンスで sf_message_header_t::event_b.class_instance, sf_message_subscriber_t::instance_range.start, sf_message_subscriber_t::instance_range.end にゼロを指定します。

メッセージフレームワークモジュールのイベントコード

イベントコードには、イベント定義の詳細が含まれています。たとえば、オーディオ再生イベントクラスのイベントコードは、「再生開始」と「再生停止」です。もう 1 つの例は、「time」イベントクラスの「set」または「get」です。イベントコードは sf_message_event_t で列挙され、プレフィックス SF_MESSAGE_EVENT_XXX を持ちます。イベントコードの定義は、ユーザーコードとイベントクラスコードに依存します。フレームワークには、いくつかのコードが例として用意されています。イベントコードの構成については、「イベントクラスコードとイベントコードの構成」を参照してください。イベントクラスインスタンス番号の最大値は 65535 です。

タッチパネルフレームワークは、イベントコードとしてのみ新しいデータを使用します。

アプリケーションは、イベントコードを以下のように使用できます。

- メッセージ作成元スレッドは、イベントコードを sf_message_header_t 型の共通メッセージヘッダの event_b.code ビットフィールドに設定してからメッセージをポストします。
- メッセージサブスクライバスレッドは、メッセージを受信した後、メッセージヘッダに設定されているイベントコードに従ってアクションを実行します。

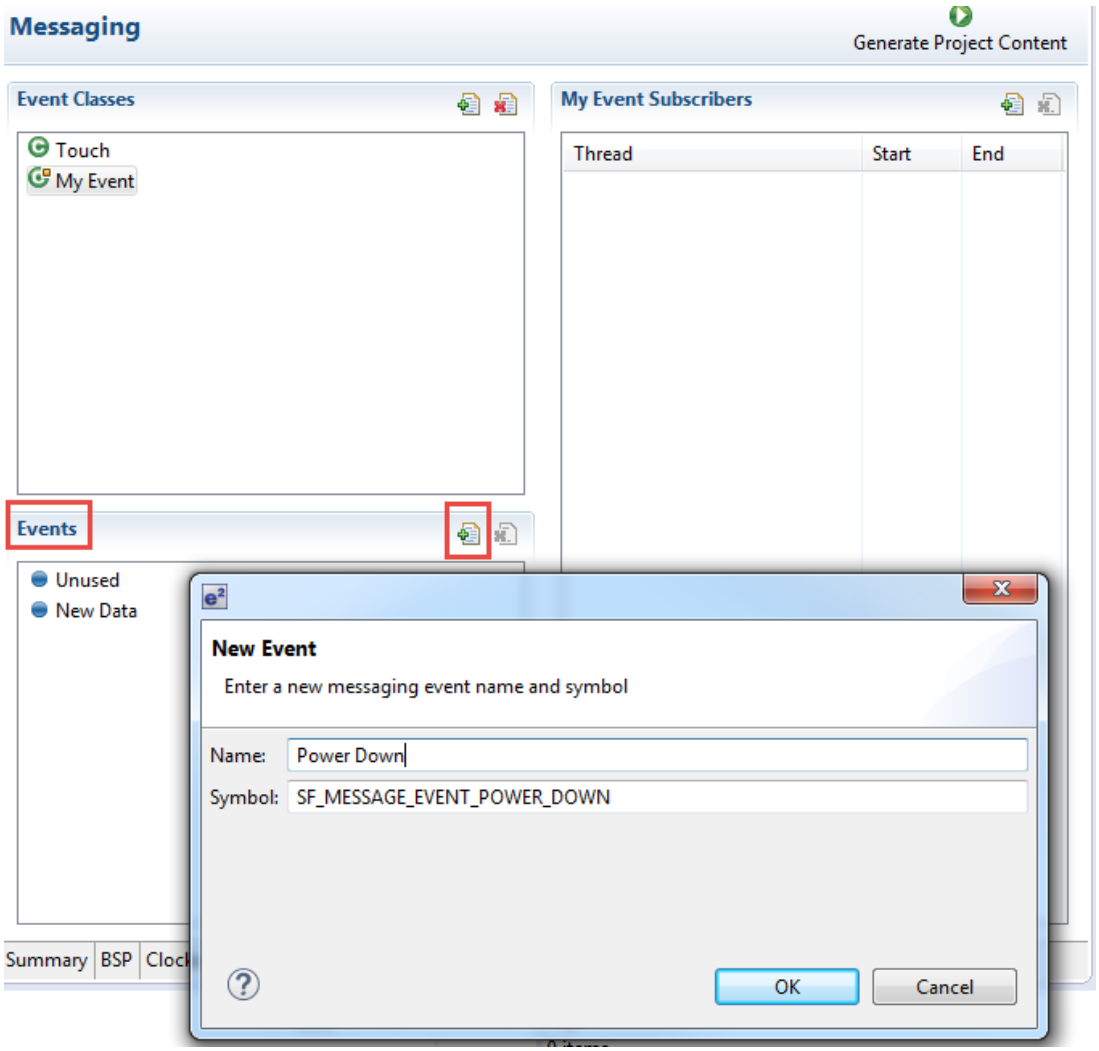


図 206: メッセージング -ISDE イベントの構成

メッセージフレームワークモジュールのサブスクライバリスト

サブスクライバリストは、フレームワークによるメッセージ配信と検索に使用されます。

フレームワークは、 sf_message_subscriber_list_t インスタンスへのポインタ配列の先頭にリストされたサブスクライバグループから、各サブスクライバスレッドのメッセージキューの検索を開始します。サブスクライバリストがイベントクラスコード (event_class). によりグループ化されるという点は重要です。

フレームワークが実行時に post API 関数のサブスクライバリストを検索する際に、メッセージペイロードデータに含まれているメッセージヘッダ (sf_message_header_t::event_b.class_code), のイベントクラスコードが、サブスクライバグループインスタンス (event_class). のものと比較されます。一致する場合、フレームワークは、次のレベルに移動し、繰り返し回数が number_of_nodes). に達するまでメッセージキューインスタンス (sf_message_subscriber_t::queue) を取得します。一致しない場合、フレームワークは次のサブスクライバグル

ープを検索し、sf_message_subscriber_list_t インスタンスへのポインタ配列に NULL が見つかるまで繰り返します。

検索手順では、サブスクライバリストの先頭にあるサブスクライバグループのメッセージングスループットが最大になりますが、下位のサブスクライバグループは不利となり、メッセージングスループットが低下します。

サブスクライバリストは、すべてのメッセージサブスクライバのルックアップテーブルです。サブスクライバリストはコンパイル時に構成されます。 post API 関数が呼び出されると、メモリに静的にマップされ、フレームワークにより検索されます。フレームワークは、サブスクライバリストを使用することで、メッセージの配信先のメッセージキューを決定できます。サブスクライバリストは、下図に示す 2 つの構造体（sf_message_subscriber_list_t と sf_message_subscriber_t）からなります。

- サブスクライバースレッドのキューは、 sf_message_subscriber_t インスタンスに登録されます。
- 同じイベントクラスコードのための上記インスタンスは、グループ化され、ポインタ配列に格納されます。
- サブスクライバグループのポインタ配列は、sf_message_subscriber_list_t インスタンスに登録されます。
- サブスクライバリストは、サブスクライバグループ構造体へのポインタ配列です。サブスクライバは、イベントクラスコードによってグループ化されます。
- ポインタ配列は、 NULL で終端している必要があります。

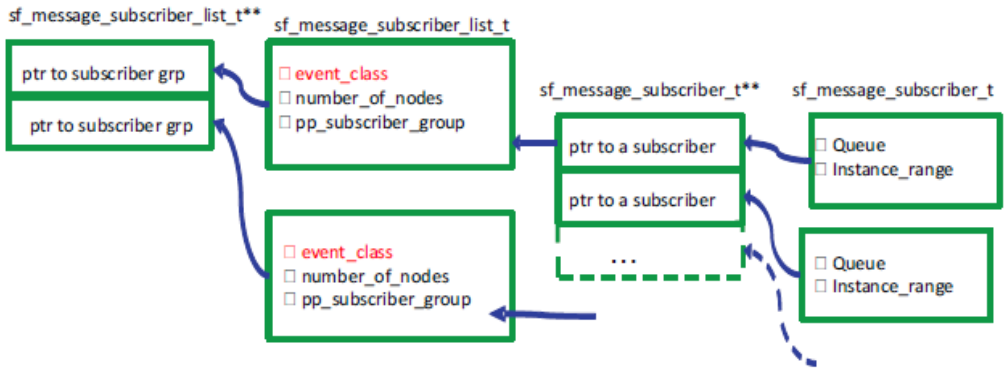


図 207:　メッセージフレームワーク - サブスクライバリスト

ISDE では、 **[Threads]** タブで名付けたスレッドのイベントクラス別にグループ化されたサブスクライバを構成できます。以下の例では、スレッドは **[Threads]** タブで「マイスレッド」と名付けられています。開始値と終了値は、このスレッドが受け入れるイベントクラスのインスタンス数を反映しています。

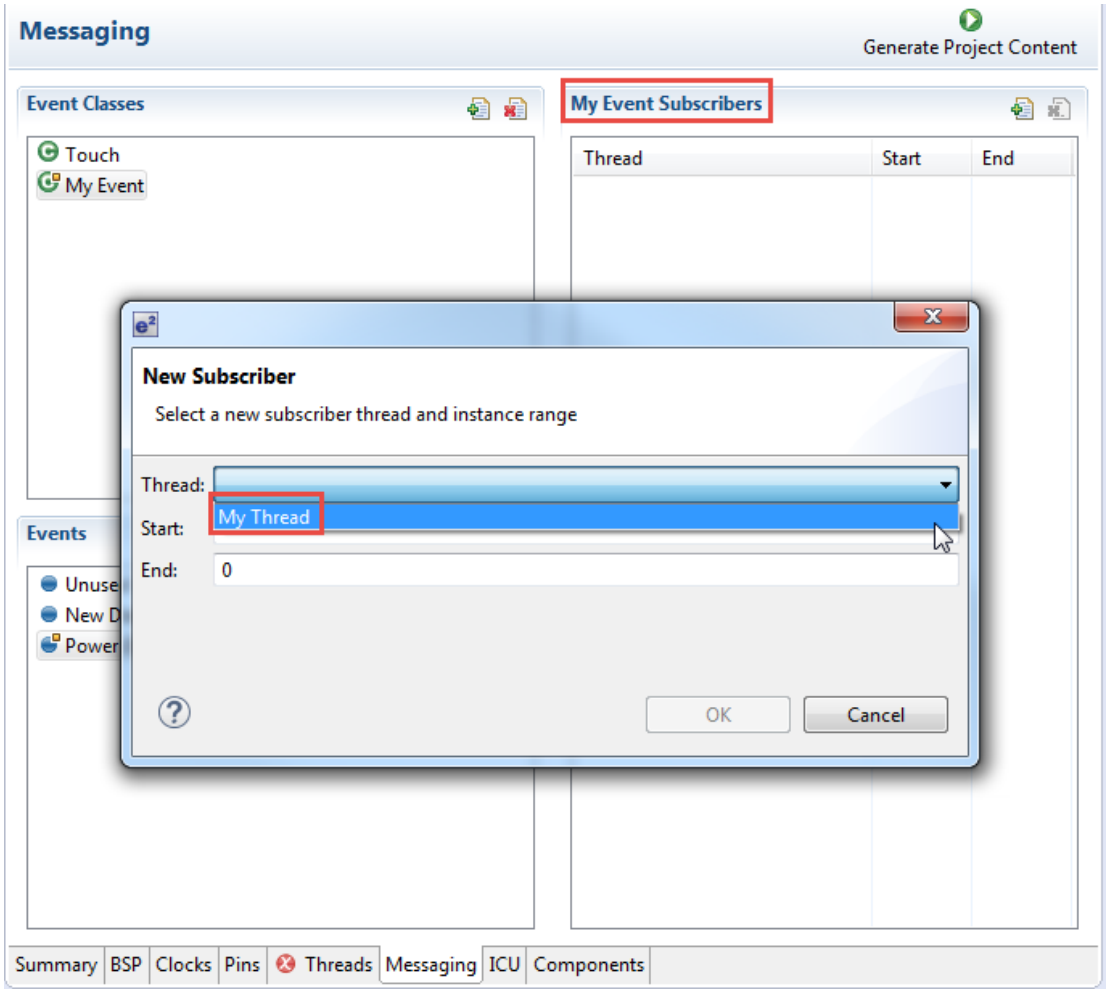


図 208: メッセージング -ISDE サブスクライバ構成

メッセージフレームワークモジュールのメッセージペイロード

メッセージペイロードは、メッセージ作成元とメッセージサブスクライバが互いに通信するために使用する構造化データです。メッセージペイロードには、メッセージ作成元がサブスクライバに発生したイベントについて通知するメッセージをポストできるよう、イベントクラスとイベントコードが共通ヘッダに含まれます（sf_message_header_t 型のデータ、イベントクラスとイベントコードを参照）。

タッチパネルフレームワークモジュールやオーディオ再生モジュールなど、事前に定義された構造体が SSP によって提供されるモジュール以外では、システム固有のメッセージペイロード構造体を定義する必要があります。メッセージペイロードには、共通ヘッダに加え、イベント処理に必要な追加データを含むことができます。

SSP には、以下の事前定義されたメッセージペイロード構造体が含まれています。

- sf_touch_panel_payload_t タッチパネルフレームワークモジュールの型
- sf_audio_playback_data_t オーディオ再生フレームワークモジュールの型

これらのフレームワークモジュールは、内部でメッセージフレームワークを使用して、最適なメッセージペイロード構造体を定義します。タッチパネルフレームワークモジュールからのタッチイベントメッセージをサブスクライブするアプリケーションスレッドは、ヘッダーファイルsf_touch_panel_api.hを含むことにより、事前定義されたメッセージペイロードを使用できます。同様に、オーディオイベントメッセージをオーディオ再生再生フレームワークモジュールにポストするアプリケーションスレッドは、sf_audio_playback_api.h を使用できます。

各イベントクラスコードに対しメッセージペイロード構造体を定義する必要があります。ただし、前述のSSP事前定義ペイロード、または共通ヘッダのみを必要とするペイロードは例外です。新しいメッセージペイロード構造体を作成するには、共通のメッセージヘッダ（sf_message_header_t 型の構造体）を、ユーザー固有のメッセージペイロード構造体の先頭に追加します。ヘッダーのサイズは 4 バイトです。

注: *ペイロードは、バッファサイズを超えてはなりません。*

このバッファサイズによる制限はきわめて重要であり、バッファを超えて大量のデータを書き込むと、ThreadX カーネルに必要なブロックメモリプール内のデータが破壊される可能性があります。サイズの制限に違反すると、ハード故障例外が発生します。バッファサイズは sf_message_ctrl_t::buffer_size. で構成できます。

メッセージフレームワークモジュールの動作に関する重要な注意事項と制限事項

メッセージングフレームワークと OS メッセージキューサービス

メッセージフレームワークモジュールは、 ThreadX のプリミティブメッセージキューとカーネルサービスを使用し、ThreadX RTOS の機能に対するいくつかの拡張をサポートしています。このため、メッセージフレームワークモジュールは、 ThreadX のメッセージキューサービスと全く同じように動作するわけではありません。ただし、メッセージングフレームワークモジュールを使用したメッセージングシステムは、アプリケーション内で ThreadX メッセージキューサービスと同時に使用できます（2 つのメッセージングシステムが分離されている場合）。

API コールのコンテキスト

- open API は、1 つのスレッドからのみ呼び出すことができます。メッセージフレームワーク制御ブロックインスタンスあたり 1 回だけ呼び出すことができます。この関数を 2 回呼び出したときの動作は不定です。
- close API は、 1 つのスレッドからのみ呼び出すことができます。
- bufferAcquire API は、 1 つのスレッドおよび 1 つの ISR から呼び出すことができます。
- bufferRelease API は、 1 つのスレッドからのみ呼び出すことができます。
- post API は、 1 つのスレッドおよび 1 つの ISR から呼び出すことができます。
- pend API は、 1 つのスレッドおよび 1 つの ISR から呼び出すことができます。

バッファ数の見積もり

メッセージングシステムを設計する際には、作業メモリ内で割り当てることができるバッファの数を正しく見積もる必要があります。バッファ数は以下のように見積もります。

$$N \approx \frac{W_m}{M_b + B_{cb} + T_x} = \frac{W_m}{M_b + 12\ bytes + 4\ bytes}$$

図 209: 作業メモリで使用可能なバッファ数の見積もり

意味 :

N - バッファ数

*W*m - 作業メモリサイズ （バイト単位）

*M*b - メッセージバッファサイズ （バイト単位）

*B*cb = 12 bytes - バッファ制御ブロックのサイズ

*T*x = 4 bytes - ThreadX の予約バイト。

同時に割り当てることができるバッファの最大数は、システム内のメッセージキューの深さの総数に等しくなります。理想的には、堅牢なシステムのバッファ数は、システム内のメッセージキューの深さの合計にする必要があります。

メッセージキューのサイズと深さの設定

メッセージフレームワークモジュールは、メッセージペイロードが格納されたバッファのポインタを配信するため、メッセージキュー上に 4 バイトのメモリブロックを必要とします。このため、メッセージキューのサイズは 4 バイトに固定する必要があります。サイズが 4 バイトよりも大きいと、メッセージフレームワーク API 関数によって内部的に実行される ThreadX メッセージキューサービスで余分なメモリがコピーされるため、性能に悪影響を与えます。

メッセージキューの深さは任意ですが、実行時にキューに格納されるメッセージ数を収容できるようにする必要があります。ガイドラインとして、次のように値を見積もります。

$$D \approx \frac{P_{avg}}{S_{avg}}$$

図 210: メッセージの深さ値の見積もり

意味 :

D - キューの深さ

*P*avg - 作成元からの平均メッセージ配信速度

*S*avg - サブスクライバでの平均イベントループ完了時間。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.19.3 アプリケーションへのメッセージフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにメッセージフレームワークモジュールを組み込む方法について説明します。メッセージフレームワークは、単一スレッドに追加される場合はすべてのスレッドで使用できます。

注 : *このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

メッセージフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。(メッセージフレームワークのデフォルト名は g_sf_message0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

メッセージフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_message0 Messaging Framework on sf_message	Threads	New Stack> Framework> Services> Messaging Framework on sf_message

次の図に示すように、 sf_message のメッセージフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、 1 回のみ追加すれば複数のスタックで使用できます。

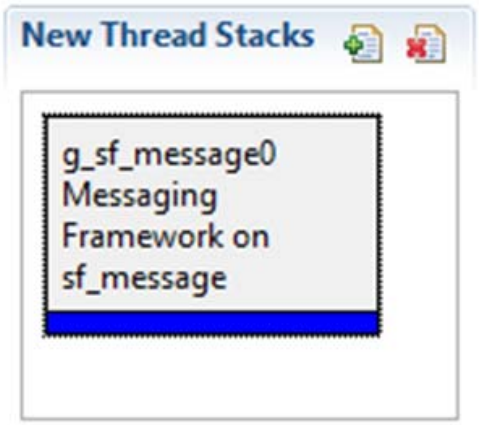


図 211: メッセージフレームワークモジュールのスタック

5.1.19.4 メッセージフレームワークモジュールの構成

必要な動作に合わせてメッセージフレームワークモジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、 ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、 SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウで使用できます。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

sf_message 上のメッセージフレームワークモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	メッセージサブスクライバーの ThreadX メッセージキューのサイズをバイト単位で指定します。この値は、すべてのメッセージキューに適用されます。
Name	g_sf_message0	メッセージングフレームワークモジュール制御ブロックインスタンスの名前。
Work memory size in bytes	2048	作業メモリサイズをバイト単位で指定します。小さい数を選択すると、同時に割り当てることができるバッファの数が少なくなります（総バッファ数は sf_message_ctrl_t::number_of_buffers). で確認できます）。この値が、同時にポストできるピークメッセージ数よりも小さい場合、フレームワークでバッファ割り当てが失敗し、システムのパフォーマンスに影響を与えます。
Pointer to subscriber list array	p_subscriber_lists	サブスクライバーリスト配列へのポインタ名を指定します。

ISDE Property	Value	説明
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	フーレムワークが制御ブロック内に作成するメモリブロックメモリの名前です。このパラメータは、デバッグ目的で有用な可能性がありますが、メモリを節約するために NULL を指定できます。

***注:** 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

メッセージキューを作成するメッセージフレームワークモジュール

メッセージングコンフィギュレータは、サブスクライバのメッセージキューを自動的に作成します。

イベントクラスとイベントを構成するメッセージフレームワークモジュール

独自のイベントクラスでメッセージフレームワークを使用するには、ISDE のプロジェクトコンフィギュレータの **[Threads]** タブと **[Messaging]** タブを使用します。

[Threads] タブで、次の操作を実行します。

1) **[Threads]** ウィンドウに新しいスレッドを追加し、一意の名前を付けます。
2) **[Threads]** ウィンドウの **[Thread Stacks]** パネルに**メッセージングフレームワーク**コンポーネントを追加します。
3) **[Messaging]** タブ（イベントクラスコードを参照）で、次の操作を実行します。 A.**[Event Class]** ウィンドウで、クラスイベントを追加します。 B. サブスクライブするスレッドのイベントクラスの名前を **[New Event]** ダイアログボックスに入力します。

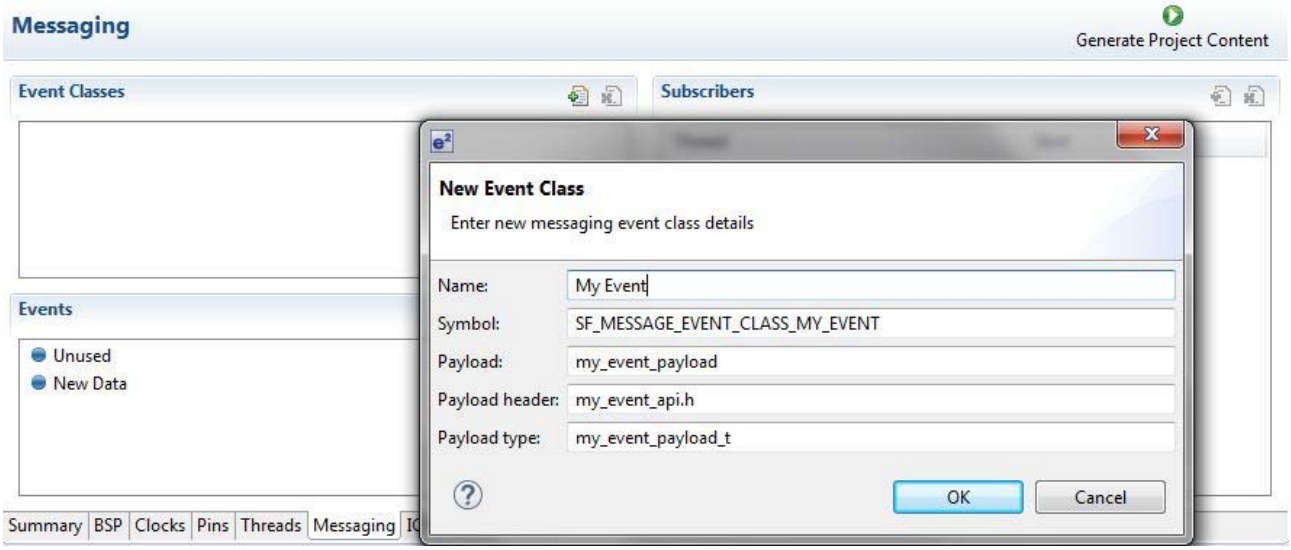


図 212: メッセージング -e² studio の新しいイベントクラスの構成

4) **[Events]** ウィンドウで、アプリケーションがサポートする可能性のあるイベントを追加します（イベントコードを参照）。

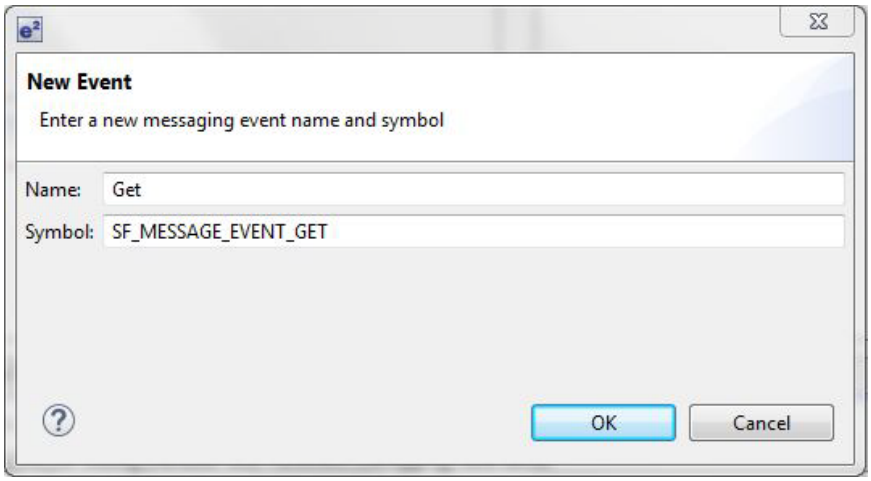


図 213: メッセージング –e² studio ISDE 新しいイベントの設定

カスタムイベントクラスコードとイベントコードはsf_message_port.h.というファイルに格納されます。オーディオ再生クラスとタッチイベントクラスは、SSP で事前定義されている 2 つのイベントクラスです。タッチイベントクラスは、新しいデータイベント SF_MESSAGE_EVENT_NEW_DATA. のみを使用します。

サブスクライバリストを構成するメッセージフレームワークモジュール

[Messaging] タブ（サブスクライバリストを参照）で、次の操作を実行します

1) **[Event Classes]** ウィンドウでイベントクラスを選択し、イベントクラスの **[Subscribers]** ウィンドウでサブスクライバリストのスレッドを構成します。
2) **[Threads]** ダイアログボックスのドロップダウンリストからスレッドを選択します。
3) [start] の横にイベントクラス instance(s) の開始番号を入力します。システムでこのイベントクラスに対し複数のイベントクラスインスタンスが使用されない場合、または指定する番号が不明な場合は、デフォルトの番号（0）のままにしておきます。使用可能な値範囲は 0 ～ 255 です。
4) [End] の横にイベントクラス instance(s) の終了番号を入力します。システムでこのイベントクラスに対し複数のイベントクラスインスタンスが使用されない場合、または指定する番号が不明な場合は、デフォルトの番号（0）のままにしておきます。使用可能な値範囲は 0 ～ 255 です。
5) [OK] をクリックします。指定したイベントのサブスクライバーがサブスクライバーリストに追加されます。
6) イベントクラスインスタンスが複数ある場合は、すべてでこの手順を繰り返します。

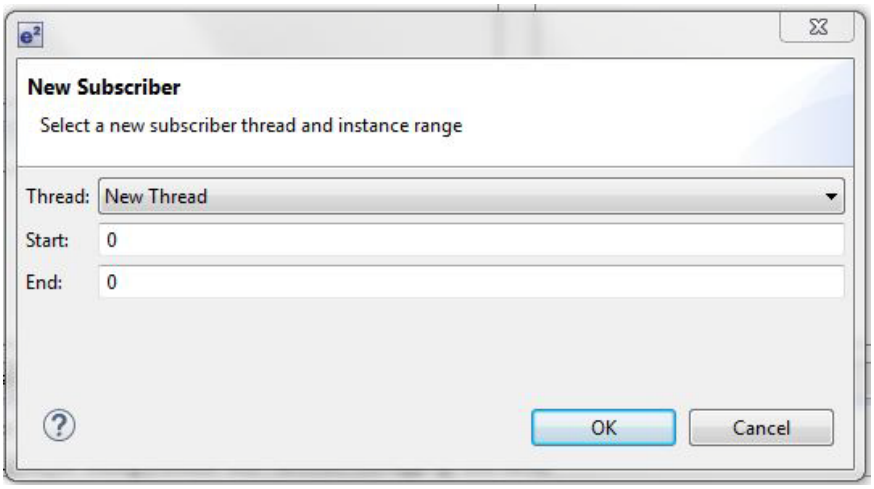


図 214: メッセージング -e² studio の新しいサブスクライバの構成

イベントクラスコードとイベントコードを構成するメッセージフレームワークモジュール

独自のメッセージペイロード構造体を定義できます。ユーザー定義のメッセージ構造体には必ず、メンバーの 1 つとして sf_message_header_t 型の構造体が含まれている必要がありますが、他のメンバーは完全にユーザー定義が可能です。メッセージフレームワークでは、メッセージペイロード構造体がどこで定義されているかは認識しません。メッセージ作成元とサブスクライバーのスレッドのソースファイル内に独自のメッセージペイロード構造体を定義するファイルを含むことができます。

sf_message_cfg_t 型構成パラメータを、システムに合わせて構成します。構成構造体用のコードは、Synergy 構成ツールを使用して生成できます。[Threads] タブで、スレッドスタックに [Messaging Framework] コンポーネントを追加し、[Properties] ウィンドウでメッセージフレームワークモジュールのプロパティを変更します。[Generate Project Content] ボタンを押すと、メッセージングフレームワークモジュールのコードがスレッドコードに生成されます。

メッセージをポストする前に、イベント作成元スレッドは、メッセージングフレームワークモジュールからメッセージ用のバッファを取得する必要があります。イベント作成元スレッドは、bufferAcquire を呼び出すことでバッファを取得できます。

API 関数が SSP_SUCCESS, を返す場合、Synergy 構成ツールで構成されたメッセージバッファサイズ（バイト単位）のバッファは、メッセージフレームワークによって管理されるメモリプールに割り当てられます。割り当て可能な最大数は、Synergy 構成ツールで指定された **[Work memory size in bytes]** の構成によって異なります。最大数の見積もりについては、「バッファ数の見積もり」を参照してください。

bufferAcquire API には、メッセージ送受信の動作を変えるためのいくつかのオプションがあります。

- **バッファキープ：**このオプションを使用すると、API 関数 bufferRelease により解放されないバッファをアプリケーションスレッドが保持できます（true に設定した場合）。一般に、バッファは、メッセージが渡された後で bufferRelease によって解放されます。ただし、スレッド間の定期的なメッセージや繰り返されるメッセージがある場合は、バッファを何度も割り当てて解放することなく同じバッファをメッセージングに再利用できます。このオプションを有効にすると、バッファの割り当て / 解放操作のオーバーヘッドを低減してシステムスループットを向上できます。

- **wait_options:** これは、待ち時間オプションであり、すべてのバッファが既に取得されている場合に有効です。任意のスレッドティックカウント、TX_WAIT_FOREVER, TX_NO_WAIT をこのオプションに設定できます。

メッセージサブスクライバスレッドは、イベント作成元によってポストされたメッセージを受信した後、バッファをフレームワークに解放する必要があります。バッファの解放は bufferRelease を呼び出すことで実行します。システムに複数のイベントサブスクライバーがある場合、API 関数は複数回呼び出される可能性があるため、実際のバッファ解放は、イベントサブスクライバーの中の最後のメッセージサブスクライバースレッドによってのみ実行されます。たとえば、イベントクラスのサブスクライバグループ中に 3 つのサブスクライバがある場合、bufferRelease を呼び出す 1 番目と 2 番目のスレッドはバッファを開放しません。3 番目のスレッドのみがバッファを解放します。 bufferAcquire により **buffer keep** オプションが指定されている場合、オプション SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE が API 関数の引数 option に渡される場合を除いて、バッファが解放されることはありません。(bufferRelease API関数の使用方法については、「メッセージフレームワークのコールバック」も参照してください。)

この API は、ユーザーコールバック関数を呼び出して、イベント作成元スレッドとメッセージサブスクライバスレッドの間でハンドシェイクを確立するためにも使用されます。

メッセージのポスト

1) bufferAcquireでバッファを取得した後、イベント作成元はメッセージペイロードデータをバッファに書き込むことができます。
2) **注意:** データをバッファに書き込むのはユーザーの責任であり、バッファサイズを超えてデータを書き込むと、メッセージフレームワークモジュールで致命的なエラーとなります。
3) ペイロード構造体の sf_message_header_t::event_b.class_code にイベントクラスコードを書き込みます。
4) ペイロード構造体の sf_message_header_t::event_b.code にイベントコードを書き込みます。この指定は必須ではありませんが、多くの場合に必要となります。
5) sf_message_header_t::event_b.class_instance. にイベントクラスインスタンス番号を書き込みます。システムに特定のイベントクラスのインスタンスが複数ある場合は、0 ～ 255 の番号を指定します。イベントクラスのインスタンスがシステムに 1 つしかない場合は、 0 を指定します。
6) post APIによってメッセージをポストします。バッファへのポインタはAPIに渡す際にsf_message_header_t * 型でキャストする必要があります。メッセージは、メッセージサブスクライバリストに登録されているメッセージサブスクライバに配信されます。post API には、メッセージ送受信の動作を変えるためのいくつかのオプションがあります。
   - **メッセージプライオリティ:** メッセージには、 SF_MESSAGE_PRIORITY_NORMAL と SF_MESSAGE_PRIORITY_HIGH. の 2 つのレベルのメッセージプライオリティがあります。SF_MESSAGE_PRIORITY_HIGH が指定された場合、メッセージは、メッセージサブスクライバーのメッセージキューの先頭に格納されます。一般にこれは、緊急メッセージに使用され、メッセージサブスクライバーは、メッセージキューに格納済みのイベントよりも前にそのイベント処理するようになります。
   - **ユーザーコールバック関数:** この関数は、メッセージフレームワークモジュールのバッファ制御ブロックに登録されます。コールバック関数は、 bufferRelease によって呼び出されます。この関数は、イベント作成元スレッドとメッセージサブスクライバースレッドの間でのハンドシェイクに使用できます。

- **Wait_options:** これは、待ち時間オプションであり、メッセージサブスクライバスレッドのメッセージキューが一杯の場合に有効です。任意の ThreadX ティックカウント、TX_WAIT_FOREVER、 TX_NO_WAIT をこのオプションに設定できます。

保留メッセージの確認

1) メッセージフレームワークモジュールがオープンされた後、メッセージサブスクライバスレッドは、pend を呼び出すことで、メッセージを待つことができます。通常、2 つ目の API 引数では、**[Messaging]** タブの[Thread Subscribers]ペインでメッセージサブスクライバスレッドに対して構成したメッセージキューへのポインタを指定しますが、必要に応じて他のメッセージキューを指定することもできます。
2) イベント作成元からメッセージが配信されると、スレッドは pend から戻ります。
3) API は、 API の第 3 引数を介してスレッドへのメッセージを含むバッファへのポインタを返します。
4) メッセージサブスクライバは、ユーザーカスタムメッセージペイロード構造体に上記のポインタ型のポインタをキャストし、イベントクラスコード sf_message_header_t::event_b.class_code, イベントコード sf_message_header_t::event_b.code、メッセージ内にあるユーザー定義の任意のデータに応じてイベントを処理します。

注:** pend には、API 関数の動作を変更する wait_option オプションがあります。PEND** の 4 つ目の引数は待ち時間オプションであり、メッセージサブスクライバスレッドのメッセージキューが空の場合にのみ有効です。任意の ThreadX ティックカウント、TX_WAIT_FOREVER, TX_NO_WAIT をこのオプションに設定できます。詳細については、『ThreadX ユーザーガイド』の ThreadX サービス呼び出し tx_queue_receive() の説明を参照してください。

メッセージフレームワークモジュールの割り込み

メッセージフレームワークモジュールは割り込みを使用しません。

5.1.19.5 アプリケーションでのメッセージフレームワークモジュールの使用

アプリケーションでメッセージフレームワークモジュールを使用する際の一般的な手順では、まず次のようにすべての必要な設定を構成します。

- メッセージキューの作成
- イベントクラスとイベントの構成
- サブスクライバリストの構成
- イベントクラスコードとイベントコードの構成

構成が完了したら、次のようにモジュールの API をターゲットアプリケーションで使用できます。

1) open API でメッセージフレームワークを初期化する
2) bufferAcquire API でバッファを取得する
3) post API でメッセージをポストする
4) pend API で保留メッセージを確認する
5) bufferRelease API を使用してバッファを解放する
6) close API でメッセージフレームワークを閉じる

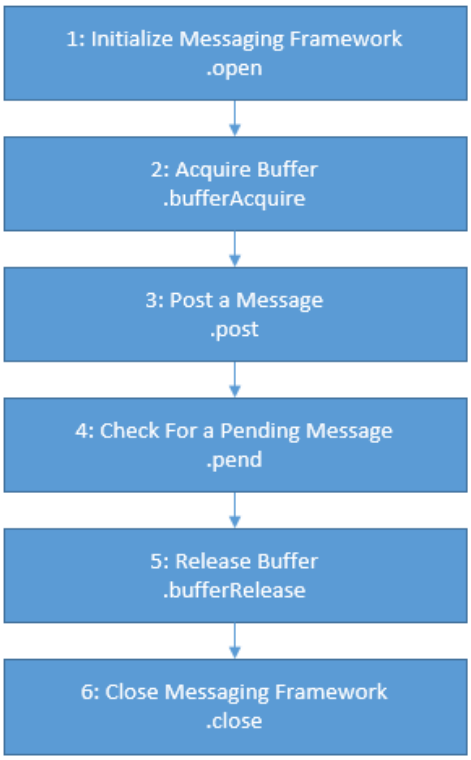


図 215:　通常のメッセージフレームワークモジュールアプリケーションのフロー図

5.1.20 パワープロファイルフレームワーク

「パワープロファイル」セクションでは、オープン、スリープ、クローズなどの一般的な機能の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表も提供します。

パワープロファイルフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_power_profiles0.p_api->open(g_sf_power_profiles0.p_ctrl, g_sf_power_profiles0.p_cfg); パワープロファイルフレームワークを初期化します。

Function Name	API の呼び出し例と説明
sleep	g_sf_power_profiles0.p_api->sleep(g_sf_power_profiles0.p_ctrl); MCU をソフトウェアスタンバイモードにします。
close	g_sf_power_profiles0.p_api->close(g_sf_power_profiles0.p_ctrl); パワープロファイルフレームワークを閉じます。
versionGet	g_sf_power_profiles0.p_api->versionGet(&p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	フレームワークが既に初期化されています。
SSP_ERR_INVALID_HW_CONDITION	互換性のないシステムクロック構成。
SSP_ERR_NOT_OPEN	デバイスは開かれていません。
SSP_ERR_UNSUPPORTED	モジュールでサポートされない機能です。
SSP_ERR_INTERNAL	内部エラー。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、関連モジュールの『Renesas Synergy Software Package Reference』を参照してください。*

5.1.20.1 パワープロファイルフレームワークモジュールの動作の概要

パワープロファイルフレームワークは、Synergy マイクロコントローラハードウェアの RTC、LPM、IOPORT および CGC を使用するもので、ローパワー動作モードにアクセスする際に使いやすいソフトウェアインタフェースとなっています。パワープロファイルフレームワークにより、事前に定義された状態でソフトウェアスタンバイモードの開始と終了を実行できるように、システムが設定されます。パワープロファイルフレームワークは、スレッド環境と非スレッド環境の両方で使用できます。

パワープロファイルモジュールは、ユーザーが行う初期構成により動作モード（RTC または外部割り込み）が決定されるという形で動作します。パワープロファイルモジュールでは、ユーザーは 2 つのピン構成表（ソフトウェアスタンバイおよび復帰）を定義できます。構成表により、ソフトウェアスタンバイモード中とその後の復帰時の MCU ポートピンの状態が定義されます。両方の表は、ともに bsp_pin_cfg.h ファイルをベースとして使用し、これに従って表の名前などを含む全体を修正することにより生成できます。

それらの表へのポインタは、復帰およびソフトウェアスタンバイのピン設定に関するパワープロファイルのプロパティに記載されています。これはピン設定表の定義にあたって必須ではありません。これらのポインタの一方または両方に NULL を指定することにより、それぞれのスリープ / 復帰状態に関するピンの再設定が防止されます。

パワープロファイルフレームワークモジュールの動作に関する注意事項

パワープロファイルフレームワークを使用すると、MCU をソフトウェアスタンバイモードに設定できます。ソフトウェアスタンバイモードでは、消費電力を非常に低く抑えながら、すべての RAM および操作パラメータを保持できます。外部割り込みモードは、主に RTC に加えて LOCO およびサブクロックのクロックソースがオフになっているため、最小限の電力しか使用しないモードです。

RTC モードでは、RTC は動作を許可されているため、結果的により多くの電力を使用します。通常 MCU がソフトウェアスタンバイモード中に動作の継続を許可しているペリフェラルまたはクロックは、ソフトウェアスタンバイモードに入る前にパワープロファイルモジュールによりオフにされ、その後復帰時にオンになります。

この動作対象から除外する動作が他にある場合は（LOCO をソフトウェアスタンバイ中も動作させる場合など）、SF_POWER_PROFILES_EVENT_PRE_SLEEP または SF_POWER_PROFILES_EVENT_POST_SLEEP コールバックイベントを使用して特定の動作を含めることで、この動作を構成できます。

***注：** パワープロファイルモジュールは番号付きの IRQ（IRQ0-IRQ15...）によってウェイクアップできないため、追加の作業が必要です。番号付きの IRQ を使用して MCU からウェイクアップするには、アプリケーションで r_lpmAPI 関数 wupenGet と wupenSet を使用して、番号付きの IRQ によるスタンバイモードからのウェイクアップを有効にする必要があります。*

パワープロファイルフレームワークでサポートされるオペレーティングシステム

ThreadX と併用した場合、このフレームワークはミューテックスのような ThreadX に固有のオブジェクトを使用します。 ThreadX を利用した動作はオプションです。

スリープモードの開始と終了

次のブロック図は、スリープ開始（ソフトウェアスタンバイ）モードの処理フローを示したものです。

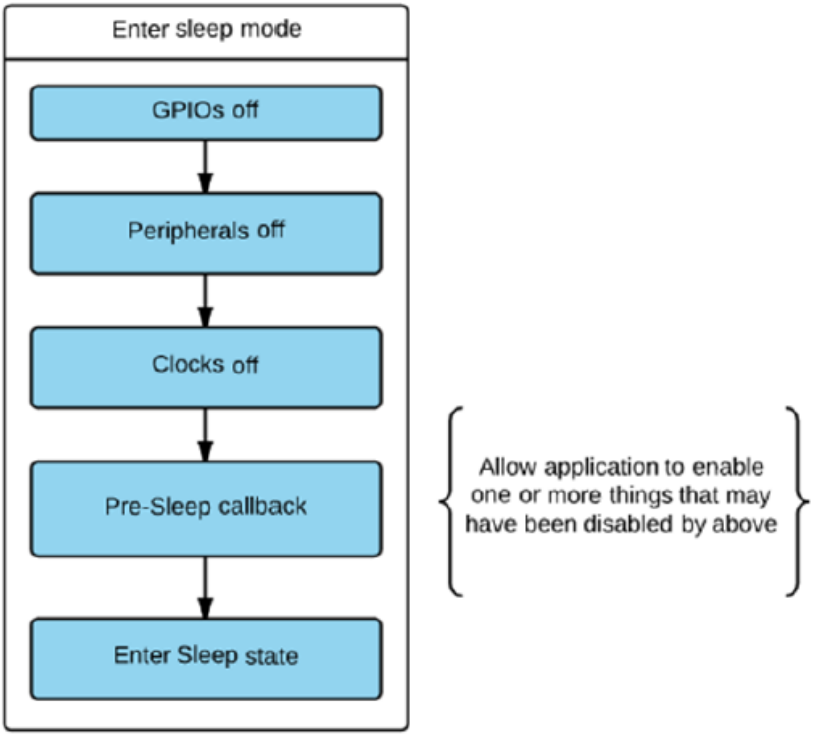


図 216:　スリープモードの開始

次のブロック図は、スリープ（ソフトウェアスタンバイ）モードからの復帰に関する処理フローを示したものです。

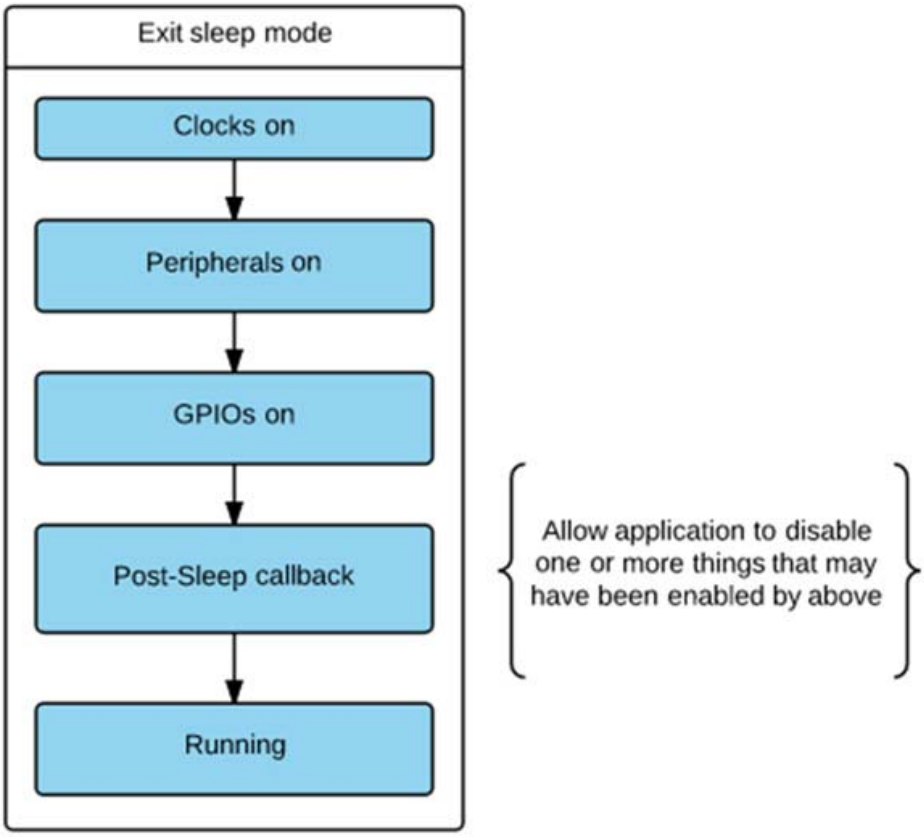


図 217:　スリープモードの終了

パワープロファイルフレームワークモジュールの制限事項

パワープロファイルフレームワークは、現在ソフトウェアスタンバイ以外のどのスリープモードもサポートしていません。

SF_POWER_PROFILES_V2 フレームワークは SF_POWER_PROFILES フレームワークに優先します。プロジェクトでは SF_POWER_PROFILES の使用を継続できますが、SF_POWER_PROFILES_V2. に更新することをお勧めします。新しいプロジェクトでは、SF_POWER_PROFILES. ではなく SF_POWER_PROFILES_V2 を使用する必要があります。SF_POWER_PROFILES は R_LPM で使用する必要があり、SF_POWER_PROFILES_V2 は R_LPM_V2. で使用する必要があります。相違点と特長については、SF_POWER_PROFILES V2 の使用上の注意を参照してください。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.20.2 アプリケーションへのパワープロファイルフレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してパワープロファイルをアプリケーションに組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

パワープロファイルフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（パワープロファイルのデフォルト名は g_sf_touch_button0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

パワープロファイルフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_power_profiles0 Power Profiles Framework on sf_power_profiles	**Threads**	**New Stack > Framework > Services > Power Profiles Framework on sf_power_profiles**

次の図に示すように sf_power_profiles のパワープロファイルフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、テキストボックスが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

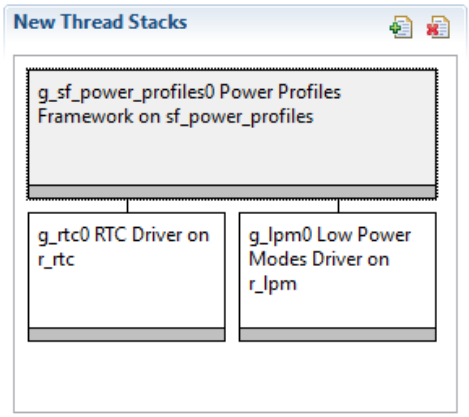


図 218: パワープロファイルフレームワークモジュールのスタック

5.1.20.3 パワープロファイルフレームワークモジュールの構成

必要な動作に合わせてパワープロファイルモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能

な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_power_profiles 上のパワープロファイルフレームワークの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効化または無効化します
RTC Support	Disabled, Enabled (Default: Disabled)	RTC サポートを有効化または無効化します
Name	g_sf_power_profiles0	モジュール名
Callback	3	システムの他のスレッドのプライオリティに基づいてユーザーが決定します。高いプライオリティを使用することを推奨します。
Wakeup pin config table	NULL	復帰時のピン構成表の場所
Sleep pin config table	NULL	スリープ時のピン構成表の場所
Operating mode	Run, RTC, External (Default: Run)	動作モード

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : 低レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

パワープロファイルフレームワークの低レベルモジュールの構成設定

通常は、ローレベルドライバーの少数の設定のみをデフォルトから変更する必要があり、これらは [Thread Stack] ブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

r_rtc 上の RTC ドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking Enable	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効化または無効化します
Name	r_rtc0	モジュールの名前
Clock Source	Sub Clock, LOCO, (Default: LOCO)	RTC モジュールのクロックソースを選択します
Error Adjustment Value [DEPRECATED]	0	エラー調整値を選択します
Error Adjustment Type [DEPRECATED]	None	調整タイプを選択します
Callback	NULL	コールバック名
Alarm Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	アラームイベントのプライオリティレベルを設定します

ISDE Property	Value	説明
Period Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	周期イベントのプライオリティレベルを設定します
Carry Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	キャリーイベントのプライオリティレベルを設定します

r_lpm 上のローパワーモードドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_lpm0	モジュール名
Operating power mode	High speed operating mode, Middle speed operating mode, Low speed operating mode, low voltage operating mode (Default: High speed operating mode)	動作モードを選択します
Sub oscillator mode	Sub oscillator mode not enabled, Sub oscillator mode enabled (Default: Sub oscillator mode not enabled)	サブ発振器モードを選択します

注：上記の設定例とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイルフレームワークモジュールのクロック構成

パワープロファイルフレームワークは、固有のクロック設定を必要としません。

パワープロファイルフレームワークモジュールのピン構成

このアプリケーションでは、ソフトウェアスタンバイ開始前とソフトウェアスタンバイからの復帰後のポートピン状態の設定に使用する、ソフトウェアスタンバイと復帰時のピン設定表を任意に利用できます。

5.1.20.4 アプリケーションでのパワープロファイルフレームワークモジュールの使用

モジュールの構成が完了し、ISDE ファイルが生成されたら、以下の手順に従ってパワープロファイルフレームワークをスレッドで使用します。パワープロファイルは、任意にスタンドアロン（非スレッド環境）アプリケーションで使用できます。

パワープロファイルフレームワークフィールド p_callback. に構成したコールバック関数の本体を定義します。MCU がソフトウェアスタンバイモードに移行する直前と、ソフトウェアスタンバイモードから復帰した直後に、コールバック関数によってアプリケーションに通知が行われます。コールバックの使用は任意ですが、パワープロファイルのプロパティにコールバックを 1 つ定義している場合は、それに対する定義が必要です。

注：パワープロファイルは番号付きの IRQ（IRQ0-IRQ15...）によってウェイクアップできません。番号付きの IRQ を使用して MCU からウェイクアップするには、アプリケーションで r_lpm API 関数 wupenGet と wupenSet を使用して、番号付きの IRQ によるスタンバイモードからのウェイクアップを有効にする必要があります。

1) アプリケーションスレッドで、 open 関数呼び出しを使用してフレームワークを初期化します。
2) sleep 関数によりソフトウェアスタンバイモードに入ります。
3) close 関数を呼び出してフレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

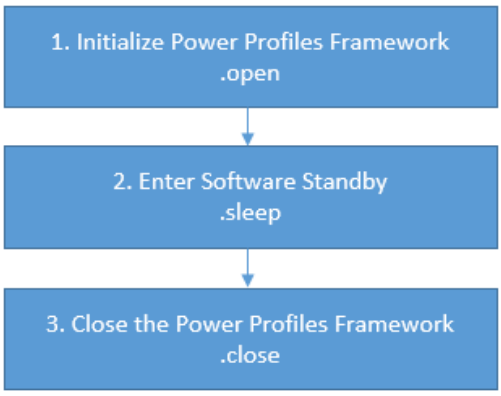


図 219: 通常のパワープロファイルフレームワークモジュールアプリケーションのフロー図

5.1.21 パワープロファイル V2 フレームワーク

- カスタマイズ可能なクロックドメインでさまざまなMCU電力制御モードを設定するための構成可能なオプション。
- 異なる IO ポートまたはピン構成でさまざまな MCU ローパワーモードを設定するための構成可能なオプション。
- スレッド動作と非スレッド動作の両方をサポート

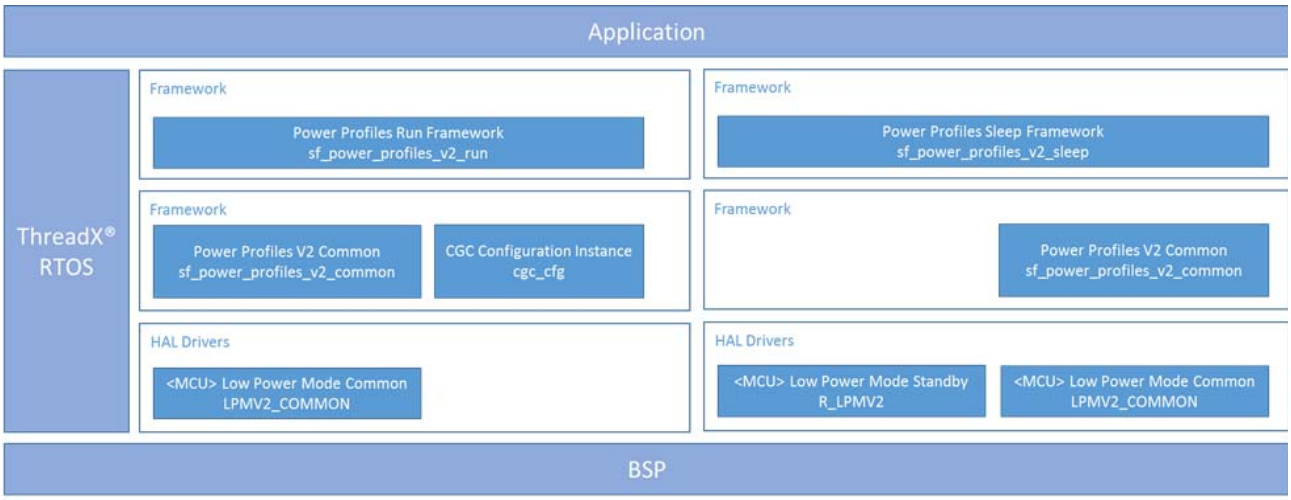


図 220: パワープロファイル V2 フレームワークの編成、オプション、スタックの実装

低レベル LPMV2 HAL モジュールには多くのオプションがあり、それらはさまざまな MCU で使用できるため、上記の図にはすべてが示されているわけではありません。簡略化した <MCU> インジケータは実際のフレームワーク実装の CPU 名（S7G2 など）で置換されます。

5.1.21.1 パワープロファイル V2 フレームワークモジュール

ターゲット MCU に応じてフレームワークで使用されるさまざまな低レベル LPM V2 HAL モジュールがあります。これらを次の表に示します。

MCU	Driver
S124	S124 Low Power Mode Sleep on r_lpmv2
S124	S124 Low Power Mode Standby on r_lpmv2
S128	S128 Low Power Mode Sleep on r_lpmv2
S128	S128 Low Power Mode Standby on r_lpmv2
S3A3	S3A3 Low Power Mode Sleep on r_lpmv2

MCU	Driver
S3A3	S3A3 Low Power Mode Standby on r_lpmv2
S3A7	S3A7 Low Power Mode Sleep on r_lpmv2
S3A7	S3A7 Low Power Mode Standby on r_lpmv2
S5D9	S5D9 Low Power Mode Sleep on r_lpmv2
S5D9	S5D9 Low Power Mode Standby on r_lpmv2
S5D9	S5D9 Low Power Mode Deep Standby on r_lpmv2
S7G2	S7G2 Low Power Mode Sleep on r_lpmv2
S7G2	S7G2 Low Power Mode Standby on r_lpmv2
S7G2	S7G2 Low Power Mode Deep Standby on r_lpmv2

使用可能な低レベル LPM V2 HAL モジュールは数多くあるため、それぞれの API と構成設定を識別するのは困難です。このモジュールガイドでは、S7G2 MCU の詳細のみを示します。すべての API と構成設定は同じであり（ディープスタンバイをサポートしない MCU を除く）、ターゲット MCU に簡単に推定されます。

「パワープロファイル」では、オープン、スリープ、クローズなどの一般的な機能の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表も提供します。

パワープロファイル V2 フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_power_profiles_v2_common.p_api->open(g_sf_power_profiles_v2_low_power_0.p_ctrl, g_sf_power_profiles0.p_cfg); パワープロファイル V2 フレームワークを初期化します。
.runApply	g_sf_power_profiles_v2_common.p_api->runApply(g_sf_power_profiles_v2_common.p_ctrl, &g_sf_power_profiles_v2_run_0); 選択した実行パワープロファイルを適用します。

Function Name	API の呼び出し例と説明
.lowPowerApply	g_sf_power_profiles_v2_common.p_api->lowPowerApply(g_sf_power_profiles_v2_common.p_ctrl, &g_sf_power_profiles_v2_low_power_0); 選択したローパワープロファイルを適用します。
.close	g_sf_power_profiles_v2_common.p_api->close(g_sf_power_profiles_v2_common.p_ctrl); モジュールを閉じます。
.versionGet	g_sf_power_profiles_v2_common.p_api->versionGet(&p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_IN_USE	フレームワークが既に初期化されています。
SSP_ERR_INVALID_HW_CONDITION	互換性のないシステムクロック構成。
SSP_ERR_NOT_OPEN	デバイスは開かれていません。
SSP_ERR_UNSUPPORTED	モジュールでサポートされない機能です。
SSP_ERR_INTERNAL	内部エラー。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.21.2 パワープロファイル V2 フレームワークモジュールの動作の概要

PPv2 フレームワークは、LPM v2 ドライバー、CGC ドライバー、IO ポートドライバーとともに使用される場合に、Synergy MCU でのローレベルパワープロファイルのサポート用の汎用 API を提供します。これは MCU の電力消費の高度な制御インタフェースと見なすことができ、ThreadX RTOS の有無を問わずアプリケーションで使用できます。内部的には、SSP の LPM v2、IOPORT、CGC ドライバーに依存し、MCU のパワーモードを制御するための使いやすいソフトウェアインタフェースを提供します。

PPv2 フレームワーク実行プロファイル

実行プロファイルは、CGC クロック構成と IO ポートピン構成を使用して、通常の実行モードの MCU のシステムクロックと IO ポートピンを設定します。その機能は RunApply() API で実装され、以下のタスクを指定の順序で実行します。

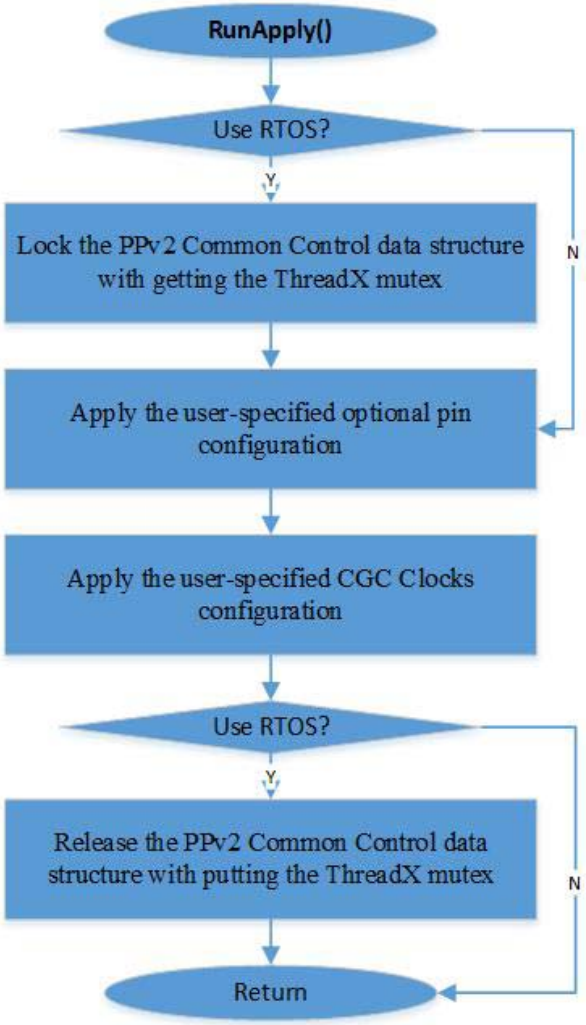


図 221: PPv2 RunApply() プロセス

PPv2 フレームワークのローパワープロファイル

ローパワープロファイルは、構成されたローパワーモードを開始する前とローパワーモードからの復帰後に LPM v2 構成と LPM 前および LPM 後の IO ポート構成を使用してローパワーモードと IO ポートピンを設定します。使用可能なローパワーモードの詳細については、『*SSP ユーザーズマニュアル*』および対応する Synergy Microcontroller Group のユーザーズマニュアルを参照してください。

内部関数 LowPowerApply() API は、次のタスクを順番に実行します。

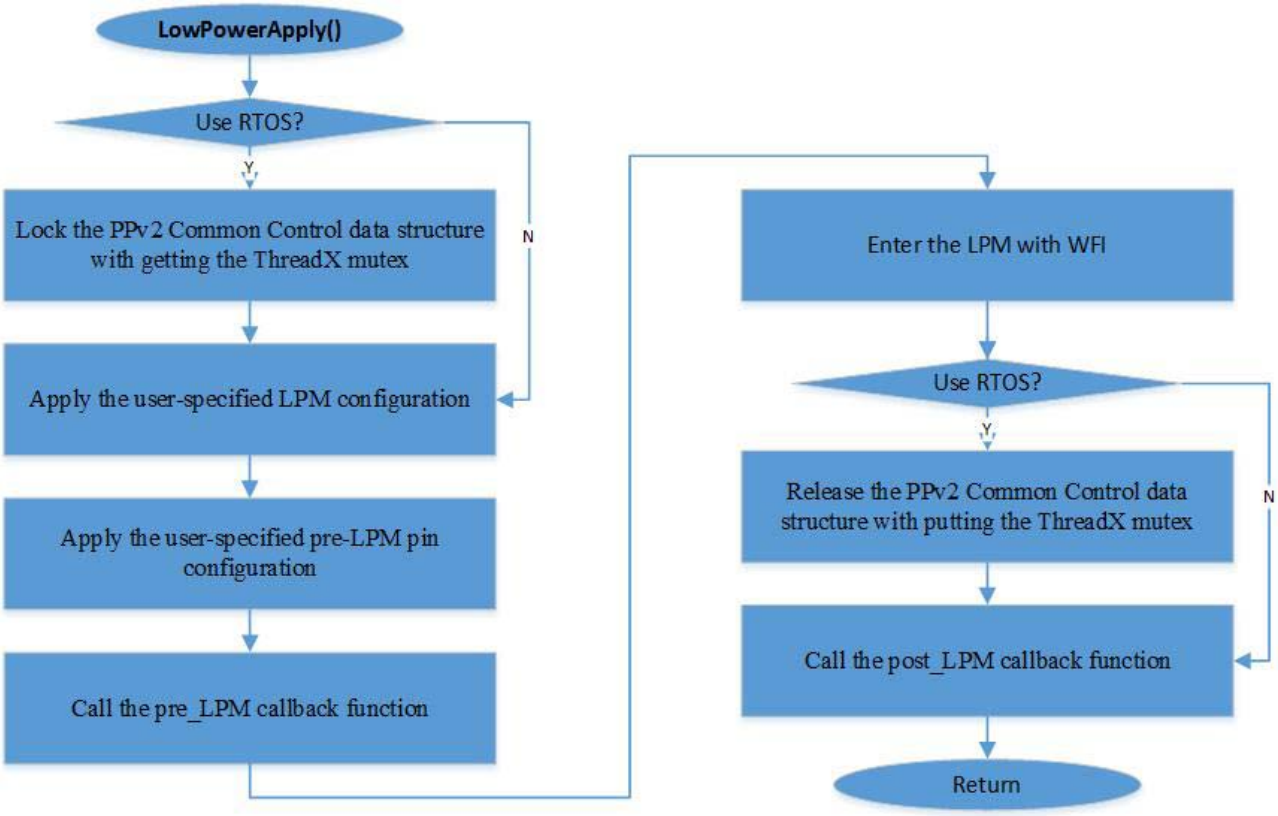


図 222: PPv2 LowPowerApply() プロセス

PPv2 フレームワークの動作に関する注意事項

以下の動作に関する注意事項は、PPv2 フレームワークおよびドライバーの最新リリースの観察に基づいています。これは多くのユーザーフィードバックが寄せられると更新され、パワープロファイルパッケージの新しいバージョンがリリースされます。

パワープロファイル v1 および PPv2

パワープロファイル v1 と PPv2 のフレームワークは互換性がないため、PPv1 フレームワークと PPv2 フレームワークを同じプロジェクトで使用しないでください。すべての新規プロジェクトでは、アプリケーションで PPv2 フレームワークを使用することをお勧めします。

LPM v2 ドライバーインスタンスはデフォルトで PPv2 アプリケーションに追加される

この MCU 固有の LPM v2 ドライバーは、lpmv2_mcu_cfg_t および lpmv2_api_t の構造体で LPM 動作を構成、有効化、無効化するための構成と API を定義し、自動的に生成されるファイル common_data.c でインスタンス化されます。したがって、PPv2 フレームワークインスタンスは LPM v2 ドライバーインスタンスに基づきます。

PPv2 runApply() 関数を使用するための追加の CGC ドライバーインスタンス

デフォルトの CGC ドライバーはすべての Synergy アプリケーションプロジェクトに含まれてファイル common_data.c, にインスタンス化され、プロジェクトのコードサイズを増やしません。 PPv2 runApply() 関数の CGC クロック構成には CGC ドライバーの別のインスタンスが必要です。

さまざまなピン構成の I/O ポートドライバーインスタンス

PPv2 パワープロファイルではさまざまなピン構成表を定義できますが、自動的に生成されるファイル common_data.c にインスタンス化されるのは 1 つの I/O ポートドライバーインスタンス ioport_instance_t のみです。

ThreadX での動作

上記の RunApply() および LowPowerApply() プロセスで示したように、PPv2 フレームワーク API は、ThreadX で使用される場合にマルチスレッドアプリケーションにミューテックスなどの ThreadX 固有のオブジェクトを使用します。

マルチスレッドアプリケーションでの特別な考慮事項

ソフトウェアスタンバイ、ディープソフトウェアスタンバイ、スヌーズの LPM モードでは、 Systick のソースクロックは無効にして構成できます。 PPv2 LPM モードでマルチスレッド RTOS プロジェクトを実装する場合は特別な考慮が必要です。

LPM モードでのデバッガの使用方法

デフォルトでは、 Arm® Core は CoreSight™ デバッグ要素によってアクセスされるため、デバッガは MCU が LPM モードにならないようにします。それらの接続を解除する 2 つのアプローチがあります。

- JTAG の USB 接続を外してから差し込むことで、完全なパワーオンリセットサイクルを実行します。
- 次に示すように、Synergy e2 studio デバッガでローパワー処理を有効化し、**[Debugger] タブ > [Connection Settings]** で提供されている J-LINK スクリプトを利用します。

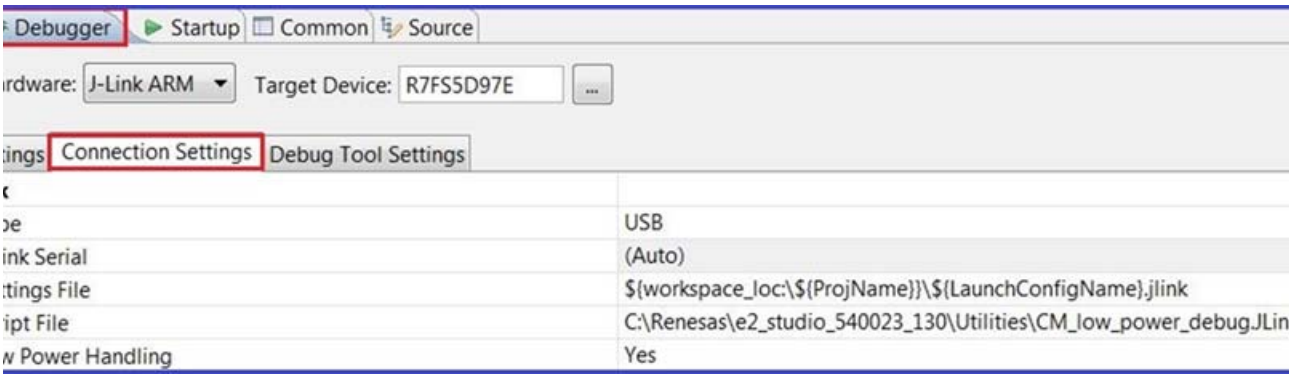


図 223: Synergy e2 studio デバッガでのローパワー処理の有効化

スクリプト CM_low_power_debug.JLinkScript は要求に応じて提供されます。

PPv2 フレームワークの制限事項

- パワープロファイル V2 フレームワークの open 関数は、プロジェクトが ThreadX を使用していない場合は main の前に自動的には呼び出されません。g_common_init() を呼び出すか API 関数 open() を明示的に呼び出して初期化を明示的に行う必要があります。これは PPv2 の制限事項ではなく、ThreadX RTOS を使用せずにフレームワークモジュールを初期化する場合の一般的なアプローチです。
- PPv2 フレームワークは MCU ペリフェラルの開始や停止を処理せず、LPM モードで停止する必要のあるペリフェラルを選択するために使用可能なプロパティビューはないため、ユーザーが手動で停止する必要があります。
- PPv2 フレームワークの最新バージョンでは、Synergy MCU S124、S128、S3A7、S5D9、S7G2 のみがサポートされます。
- PPv2 フレームワークの最新バージョンでは、LPM スタンバイモードのスヌーズからノーマルへの遷移はサポートされません。

5.1.21.3 アプリケーションへのパワープロファイル V2 フレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してパワープロファイルをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

パワープロファイルフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（パワープロファイルのデフォルト名は g_sf_power_profiles_low_power_0 です。この名前は、対応する [Properties] ウィンドウで変更できます。）

パワープロファイル V2 フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_power_profiles_v2_low_power_0 Power Profiles V2 Low Power Profile	Threads	New Stack > Framework > Services > Power Profiles V2 Low Power Profile
g_sf_power_profiles_v2_run_0 Power Profiles V2 Run Profile	Threads	New Stack > Framework > Services > Power Profiles V2 Run Profile
g_sf_power_profiles_v2_common Power Profiles V2 Common	Threads	New Stack> Framework> Services> Power Profiles V2 Common

次の図に示すようにパワープロファイル V2 ローパワーまたは実行モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

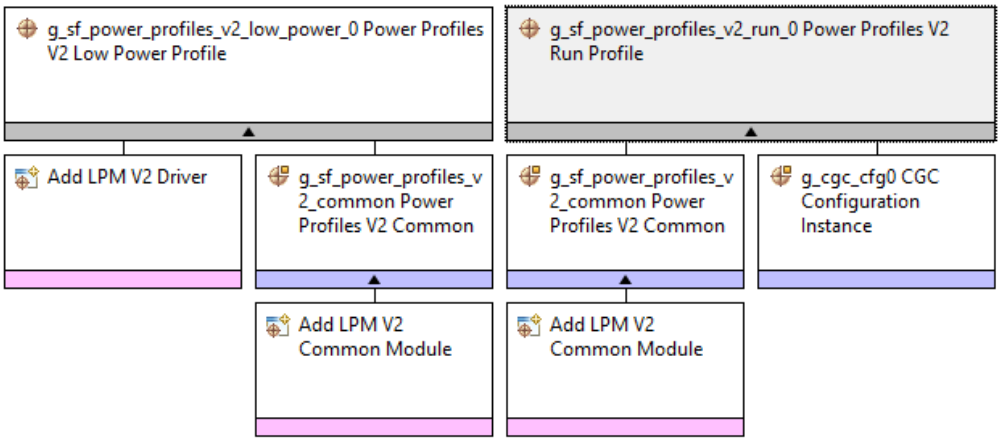


図 224: パワープロファイル V2 フレームワークモジュールのスタック

5.1.21.4 パワープロファイル V2 フレームワークモジュールの構成

必要な動作に合わせてパワープロファイルV2実行およびローパワーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

パワープロファイル V2 フレームワーク、実行プロファイル、ローパワープロファイルには 2 つの異なるスタック選択項目があります。それぞれの構成設定について、以降のセクションで別々に説明します。

パワープロファイル V2 実行プロファイルの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

パワープロファイル V2 実行プロファイルの構成設定

ISDE Property	Value	説明
Name	g_sf_power_profiles_v2_run_0	モジュール名
Pin configuration table	NULL	ピン構成表の選択

ピン構成表を電力モードに対して作成し、実行プロファイルのプロパティにリンクできます。そうしない場合、電力モードでは g_bsp_pin_cfg. で指定されたデフォルトのピン配置が使用されます。

カスタムピン構成表を作成する基本手順は次のとおりです。

- マウスの右ボタンをクリックし、**[Copy]** 操作を選択することで S5D9-PK.pincfg などの特定のボードピン構成ファイルをコピーしてから、同じプロジェクトに **[Paste]** を行います。
- この新しいピン構成ファイル S5D9-PK_RUN.pincfg. の名前を変更します。

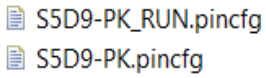


図 225: コピーとペーストによる新しいピン構成ファイルの作成

- SSP コンフィギュレータの [Pins] タブのプルダウンメニューから新しいピン構成を選択し、次に示すように生成するピン構成名を指定します。

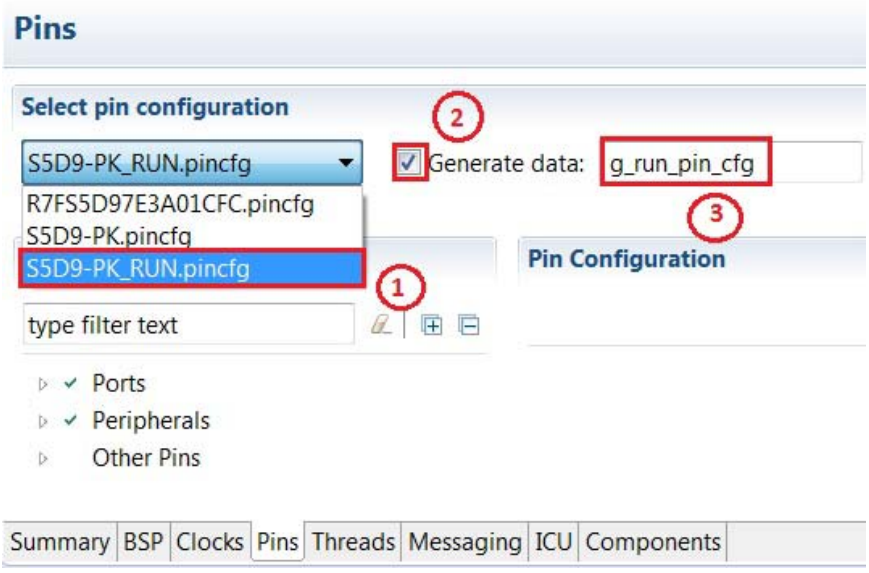


図 226: 割り当てる新しいピン構成ファイルの選択

- 各ピンの I/O 機能を構成してから、『*Synergy SSP ユーザーズマニュアル』で示されているように SSP コンフィギュレータの **[Generate Project Content]** ボタンを押して新規ピン構成を生成します。

この実行プロファイルの電力制御モードは、CGC 構成インスタンスのプロパティビューを使用して定義できます。

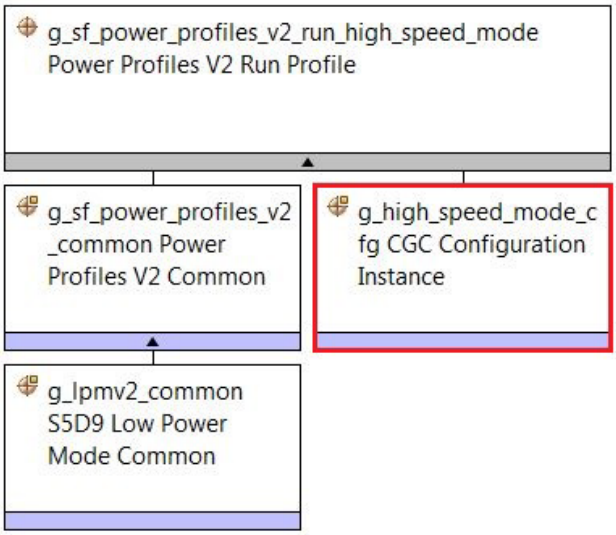


図 227: 実行プロファイルモジュールの CGC 構成インスタンス

CGC 構成プロパティビューのパラメータを次に示します。

実行プロファイルモジュールの CGC 構成設定

ISDE Property	Value	説明
Name	g_cgc_cfg0	モジュール名
System Clock	HOCO, MOCO, LOCO, Main Oscillator, Sub Clock, PLL Default: HOCO	システムクロックソースを設定します
LOCO State Change	None, Start, Stop Default: None	LOCO 状態変更の選択
MOCO State Change	None, Start, Stop Default: None	MOCO 状態変更の選択

ISDE Property	Value	説明
HOCO State Change	None, Start, Stop Default: None	HOCO 状態変更の選択
Sub-Clock State Change	None, Start, Stop Default: None	サブクロック状態変更の選択
Main Clock State Change	None, Start, Stop Default: None	メインクロック状態変更の選択
PLL State Change	None, Start, Stop Default: None	PLL ソースクロックの選択
PLL Source Clock	HOCO, MOCO, LOCO, Main Oscillator, Sub Clock, PLL Default: HOCO	PLL ソースを設定します
PLL Divisor	1, 2, 3, 4 Default: 1	PLL 出力周波数の分周
PLL Multiplier	10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0 , 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 31.0 Default: 10.0	PLL 出力周波数の増倍
PCLKA Divisor	1, 2, 4, 8, 16, 64 Default: 1	周辺クロック A の分周

ISDE Property	Value	説明
PCKLB Divisor	1, 2, 4, 8, 16, 64 Default: 1	周辺クロック B の分周
PCLKC Divisor	1, 2, 4, 8, 16, 64 Default: 1	周辺クロック C の分周
PCLKD Divisor	1, 2, 4, 8, 16, 64 Default: 1	周辺クロック D の分周
BCLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	外部バスクロックの分周
FCLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	フラッシュクロックの分周
ICLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	システムクロックの分周

注: *これらの CGC パラメータの割り当ては、各電力制御モードでの発振器の可用性と各動作クロックに想定される周波数で実現する必要があります。この関係は、対応する『Synergy MCU ユーザーズマニュアル』の CGC ブロック図に示されています。*

これらのクロックの最大動作周波数範囲は、対応する『MCU ユーザーズマニュアル』で指定された範囲外であってはなりません。『*S5D9 マイクロコントローラグループユーザーズマニュアル*』に、次の表に示すこの仕様の一覧が示されています。

Synergy S5D9 MCU グループ内部クロックの最大動作周波数範囲

Parameter	クロックソース	クロック供給	仕様
Peripheral module clock A (PCLKA)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	周辺モジュール P（ETHERC、EDMAC、USBHS、QSPI、SPI、SCI、SCE7、GLCDC、SDHI、CRC、JPEG エンジン、DRW、irDA、GPY バスクロック）	最大 120MHz*2 分周率: 1、2、4、8、16、32、64
Peripheral module clock B (PCLKB)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	周辺モジュール（IIC、SSIE、SRC、DOC、CAC、CAN、DAC12、POEG、CTSU、AGT、スタンバイ SCRAM、ELC、I/O ポート、RTC、WDT、IWDT、ADC12、KINT、USBFS、ACMPHS、TSN、PDC）	最大 60MHz 分周率: 1、2、4、8、16、32、64
Peripheral module clock C (PCLKC)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	周辺モジュール（ADC12 変換クロック）	最大 60MHz 分周率: 1、2、4、8、16、32、64
Peripheral module clock D (PCLKD)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	周辺モジュール（GPT カウントクロック）	最大 120MHz 分周率: 1、2、4、8、16、32、64
Flash interface clock (FCLK)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	フラッシュインタフェース	最大 60MHz（P/E） 最大 60MHz（リード）*1 分周率: 1、2、4、8、16、32、64
External bus clock (BCLK)	MOSC、SOSC、HOCO、MOCO、LOCO、PLL	外部バス	最大 120MHz 分周率: 1、2、4、8、16、32、64

Parameter	クロックソース	クロック供給	仕様
EBCLK pin output (EBCLK)	BCLK または ½ BCLK	EBCLK ピン	最大 60MHz 分周率 : 1、 2
SDCLK pin output (SDCLK)	BCLK	SDCLK ピン	最大 120MHz
USB clock (CABMCLK)	PLL	USB	48MHz 分周率 : 3、 4、 5
USB-PHY clock (USBMCLCK)	MOSC	USB-PHY	12、 20、 25MHz
Can CLOCK (CANMCLK)	MOSC	CAN	8 ～ 24MHz
LCD_CLCK pin output (LCD_CLK) and graphic LCD pixel clock (PXCLK)	LCD_EXTCLK, PLL 出力	LCD_CLK ピン、周辺モジュール（グラフィックス LLCD コントローラ）	最大 54MHz （パラレル RGB） 最大 60MHz （シリアル RGB） LCD_CLK: PXCLK = 1.1（パラレル RGB）

現在のプロパティビューは割り当てを自動的に検証しないため、SSP コンフィギュレータの CGC パネルを使用して、最初に割り当てを確認できます。 CGC 構成の次のスクリーンショットは、高速モードの例です。

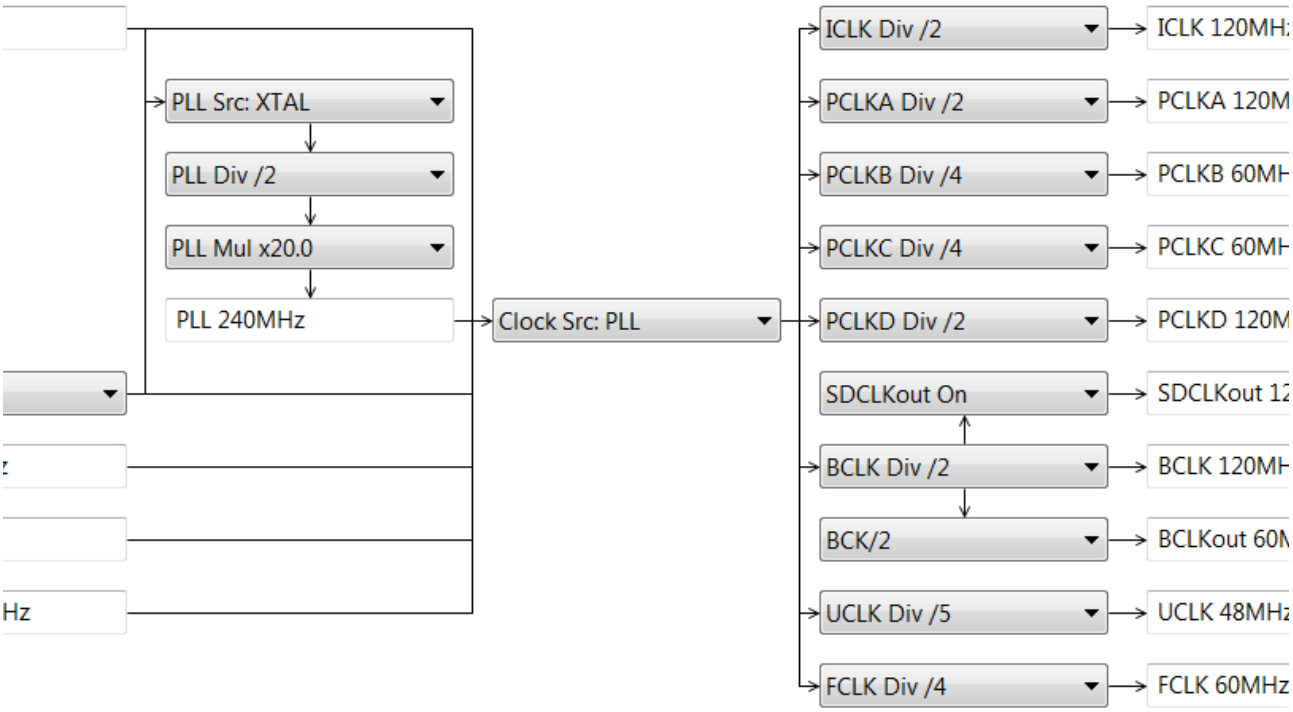


図 228: 電力制御モードのクロック設定の確認

同様に、他の電力制御モードの CGC 構成を定義できます。

PPv2 フレームワークローパワープロファイルの構成

選択した Synergy MCU グループに応じて、ローパワープロファイルモジュールのプルダウンメニューで使用可能な 3 つの LPM モード（スリープ、ソフトウェアスタンバイ、ディープソフトウェアスタンバイ）があります。

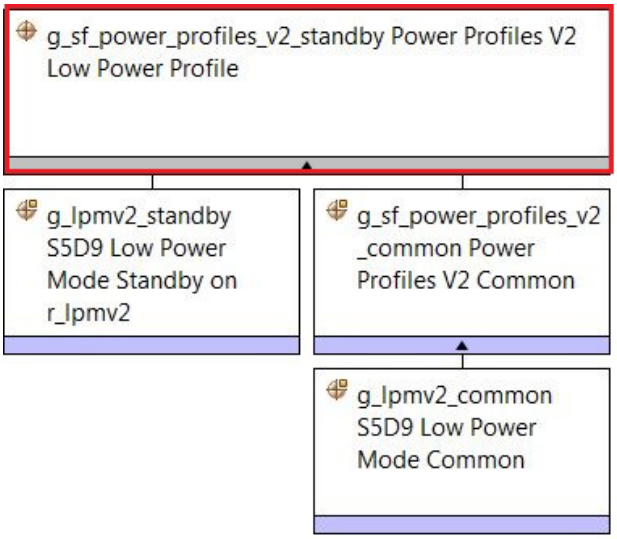


図 229: PPv2 ローパワープロファイルスタンバイモジュール

ローパワープロファイルモジュールがスレッドに既に追加されていると想定した場合は、次のような構成になります。

PPv2 ローパワープロファイルモジュールの構成設定

ISDE Property	Value	説明
Name	g_sf_power_profiles_v2_low_power_0	モジュール名
Callback (Low Power Exit Event N/A when using Deep Software Standby)	NULL	コールバックの選択
Low power entry pin configuration table	NULL	ローパワー開始ピン構成表の選択
Low power exit pin configuration table	NULL	ローパワー終了ピン構成表の選択

コールバック関数は、次のイベントの処理に使用できます。

- SF_POWER_PROFILES_V2_EVENT_PRE_LOW_POWER
- SF_POWER_PROFILES_V2_EVENT_POST_LOW_POWER

次の 2 つのピン構成表で、 IO ポート機能を変更できます。

- Low power entry pin configuration table
- Low power exit pin configuration table

これらは、 PPv2 API 関数 LowPowerApply() で内部的に使用されます。

割り込みが MCU をスリープモードから復帰させるため、LPM スリープモードの構成は単純です。そのため、モジュール名のみを変更できます。

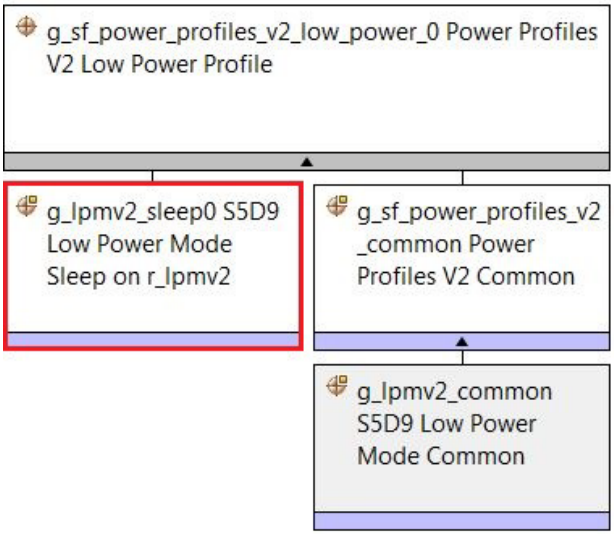


図 230: PPv2 ローパワープロファイルスリープドライバー r_lpmv2

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します
Name	g_lpmv2_standby0	モジュール名
Choose the low power mode	Standby, Standby with snooze Enabled Default: Standby	ローパワーモードの選択
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	出力ポート状態の選択
IRQ1-15	Enabled, Disabled Default: Disabled	IRQ1-15 の選択

ISDE Property	Value	説明
IWDT	Enabled, Disabled Default: Disabled	IWDT の選択
Key Interrupt	Enabled, Disabled Default: Disabled	キー割り込みの選択
LVD1 Interrupt	Enabled, Disabled Default: Disabled	LVD1 割り込みの選択
LVD2 Interrupt	Enabled, Disabled Default: Disabled	LVD2 割り込みの選択
Analog Comparator High-speed 0 Interrupt	Enabled, Disabled Default: Disabled	アナログコンパレータ高速 0 割り込みの選択
RTC Alarm	Enabled, Disabled Default: Disabled	RTC アラームの選択
RTC Period	Enabled, Disabled Default: Disabled	RTC 周期の選択
USB High-speed	Enabled, Disabled Default: Disabled	USB 高速の選択
USB Full-speed	Enabled, Disabled Default: Disabled	USB フルスピードの選択

ISDE Property	Value	説明
AGT1 underflow	Enabled, Disabled Default: Disabled	AGT1 アンダーフローの選択
AGT1 Compare Match A	Enabled, Disabled Default: Disabled	AGT1 比較一致 A の選択
AGT1 Compare Match B	Enabled, Disabled Default: Disabled	AGT1 比較一致 B の選択
12C 0	Enabled, Disabled Default: Disabled	12C 0 の選択
Snooze Entry Source	RXD0 falling edge, IRQ0-IRQ15, KINT, ACMPHS0, RTC Alarm, RTC Period, AGT1 Underflow, AGT1 Compare Match A, AGT1 Compare Match B Default: RXD0 falling edge	スヌーズ開始ソースの選択
AGT1 Underflow	Enabled, Disabled Default: Disabled	AGT1 アンダーフローの選択
DTC Transfer Completion	Enabled, Disabled Default: Disabled	DTC 転送完了の選択
DTC Transfer Completion Negated Signal	Enabled, Disabled Default: Disabled	DTC 転送完了ネゲート信号の選択
ADC0 Compare Match	Enabled, Disabled Default: Disabled	ADC0 比較一致の選択

ISDE Property	Value	説明
ADC0 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC0 比較不一致の選択
ADC1 Compare Match	Enabled, Disabled Default: Disabled	ADC1 比較一致の選択
ADC1 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC1 比較不一致の選択
SCI0 Address Match	Enabled, Disabled Default: Disabled	SCI0 アドレス一致の選択
DTC state in Snoooze Mode	Enabled, Disabled Default: Disabled	スヌーズモードの DTC の状態の選択

PPv2 プロファイルスリープモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します
Name	g_lpmv2_sleep0	モジュール名

スタンバイモード構成は、終了トリガと、PPv2 の現在のリリースのスタンバイモードの特別なケースとして扱われるスタンバイモードとスヌーズモード間のいくつかの遷移条件を設定します。スヌーズ構成については次のセクションで詳細に説明します。

例として、 S5D9 MCU スタンバイモジュールを次の図に示します。

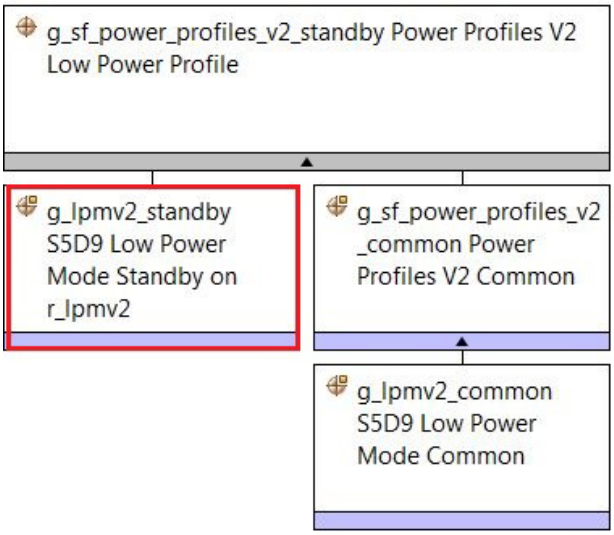


図 231: PPv2 ローパワープロファイルスタンバイドライバー r_lpmv2

PPv2 プロファイルスタンバイモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します
Name	g_lpmv2_standby0	モジュール名
Choose the low power mode	Standby, Standby with snooze Enabled Default: Standby	ローパワーモードの選択
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	出力ポート状態の選択
IRQ1-15	Enabled, Disabled Default: Disabled	IRQ1-15 の選択

ISDE Property	Value	説明
IWDT	Enabled, Disabled Default: Disabled	IWDT の選択
Key Interrupt	Enabled, Disabled Default: Disabled	キー割り込みの選択
LVD1 Interrupt	Enabled, Disabled Default: Disabled	LVD1 割り込みの選択
LVD2 Interrupt	Enabled, Disabled Default: Disabled	LVD2 割り込みの選択
Analog Comparator High-speed 0 Interrupt	Enabled, Disabled Default: Disabled	アナログコンパレータ高速 0 割り込みの選択
RTC Alarm	Enabled, Disabled Default: Disabled	RTC アラームの選択
RTC Period	Enabled, Disabled Default: Disabled	RTC 周期の選択
USB High-speed	Enabled, Disabled Default: Disabled	USB 高速の選択
USB Full-speed	Enabled, Disabled Default: Disabled	USB フルスピードの選択

ISDE Property	Value	説明
AGT1 underflow	Enabled, Disabled Default: Disabled	AGT1 アンダーフローの選択
AGT1 Compare Match A	Enabled, Disabled Default: Disabled	AGT1 比較一致 A の選択
AGT1 Compare Match B	Enabled, Disabled Default: Disabled	AGT1 比較一致 B の選択
12C 0	Enabled, Disabled Default: Disabled	12C 0 の選択
Snooze Entry Source	RXD0 falling edge, IRQ0-IRQ15, KINT, ACMPHS0, RTC Alarm, RTC Period, AGT1 Underflow, AGT1 Compare Match A, AGT1 Compare Match B Default: RXD0 falling edge	スヌーズ開始ソースの選択
AGT1 Underflow	Enabled, Disabled Default: Disabled	AGT1 アンダーフローの選択
DTC Transfer Completion	Enabled, Disabled Default: Disabled	DTC 転送完了の選択
DTC Transfer Completion Negated Signal	Enabled, Disabled Default: Disabled	DTC 転送完了ネゲート信号の選択
ADC0 Compare Match	Enabled, Disabled Default: Disabled	ADC0 比較一致の選択

ISDE Property	Value	説明
ADC0 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC0 比較不一致の選択
ADC1 Compare Match	Enabled, Disabled Default: Disabled	ADC1 比較一致の選択
ADC1 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC1 比較不一致の選択
SCI0 Address Match	Enabled, Disabled Default: Disabled	SCI0 アドレス一致の選択
DTC state in Snoooze Mode	Enabled, Disabled Default: Disabled	スヌーズモードの DTC の状態の選択

現在、スヌーズモードは、次の図に示すように **[Choose the low power mode]** のフィールドで **[Standby with Snooze Enabled]** を選択することで、 PPv2 ソフトウェアスタンバイモードのプロパティビューで有効化されます。

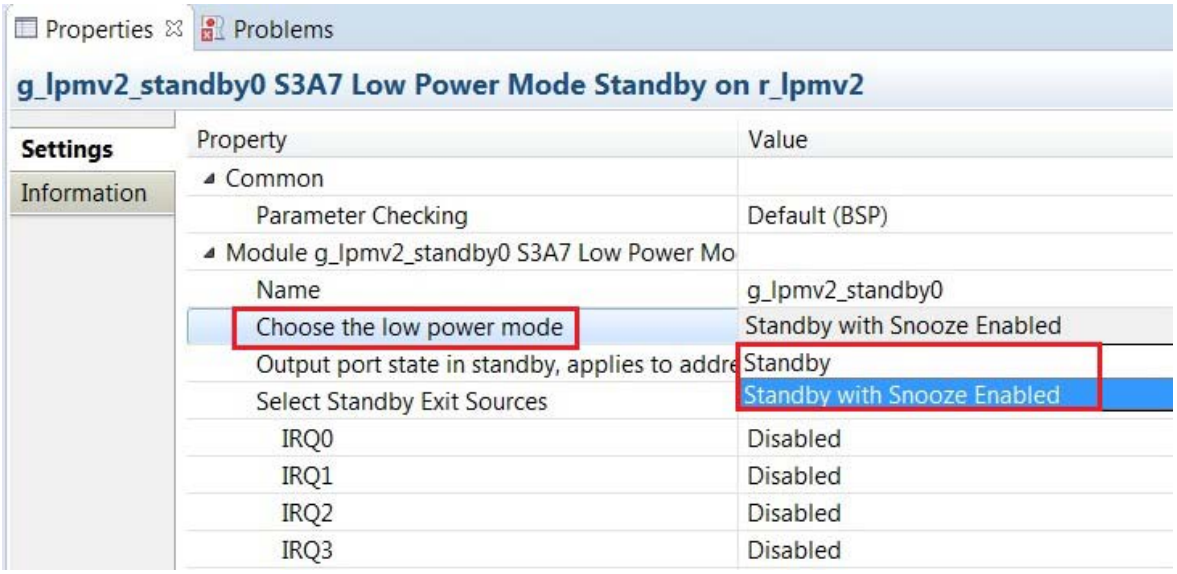


図 232: ソフトウェアスタンバイモードでのスヌーズモードの作成

ディープソフトウェアスタンバイモードの構成では、ディープソフトウェアスタンバイモードの終了トリガと内部電力供給オプションを設定します。可能な構成を示す例として、S5D9 で PPv2 フレームワークを次のように使用します。

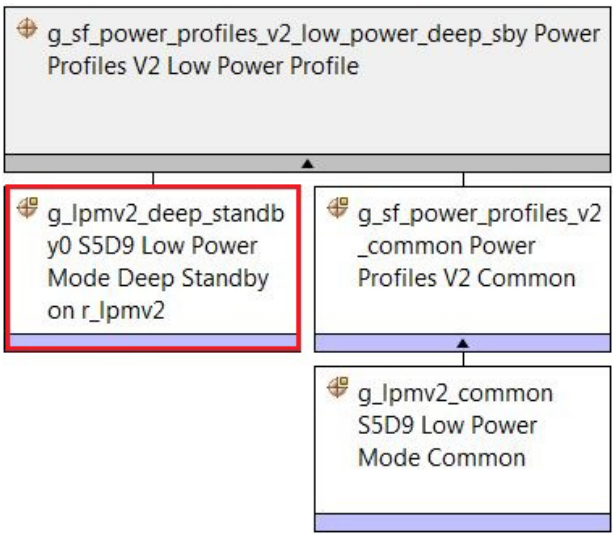


図 233: PPv2 ローパワープロファイルディープソフトウェアスタンバイドライバー r_lpmv2

PPv2 プロファイルディープソフトウェアスタンバイモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します
Name	g_lpmv2_deep_standby	モジュール名
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	スタンバイおよびディープソフトウェアスタンバイの出力ポート状態設定
Maintain or reset the IO port states on exit from deep standby mode	Maintain the IO port states, Reset the IO port states Default: Maintain the IO port states	出力ポート状態設定終了
Internal power supply control in deep standby mode	Maintain the internal power supply, Cut the power supply to standby RAM, low-speed on-chip oscillator, AGTn, and USPFS/HS resume detecting unit, Cut the power supply to LVDn, standby RAM, low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit Default: Maintain the internal power supply	ディープソフトウェアスタンバイモードの内部電力供給制御の設定
IRQ0-15	Enabled, Disabled Default: Disabled	IRQ0-15 の選択
IRQ0-15 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	IRQ0-15 エッジの選択
LVD1	Enabled, Disabled Default: Disabled	LVD1 の選択

ISDE Property	Value	説明
LVD1 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD1 エッジの選択
LVD2	Enabled, Disabled Default: Disabled	LVD2 の選択
LVD2 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD2 エッジの選択
RTC Interval	Enabled, Disabled Default: Disabled	RTC インターバルの選択
RTC Alarm	Enabled, Disabled Default: Disabled	RTC アラームの選択
NMI	Enabled, Disabled Default: Disabled	NMI の選択
NMI Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	NMI エッジの選択
USBFS	Enabled, Disabled Default: Disabled	USBFS の選択
UBSHS	Enabled, Disabled Default: Disabled	UBSHS の選択

ISDE Property	Value	説明
AGT11	Enabled, Disabled Default: Disabled	AGT11 の選択

注：PPv2 フレームワークの [property] ダイアログには、ディープソフトウェアスタンバイモードの内部電力供給制御用の 2 つの定義済み構成があります。内部コンポーネントへの電力供給の停止に関する他の選択肢の一覧は、『Synergy S5D9 マイクロコントローラグループユーザーズマニュアル』の表 11.2 など、『Synergy マイクロコントローラグループユーザーズマニュアル』に記載されています。

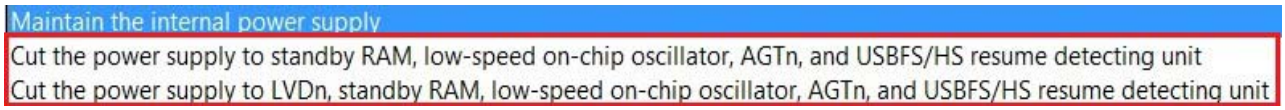


図 234: PPv2 フレームワークディープソフトウェアスタンバイモードの定義済み内部電力供給オプション

PPv2 フレームワーク共通モジュールの構成

PPv2 フレームワーク実行プロファイルモジュールとローパワープロファイルモジュールで共有される 2 つの共通モジュールがあります。

- **自動的に生成されるパワープロファイル V2 共通モジュール**

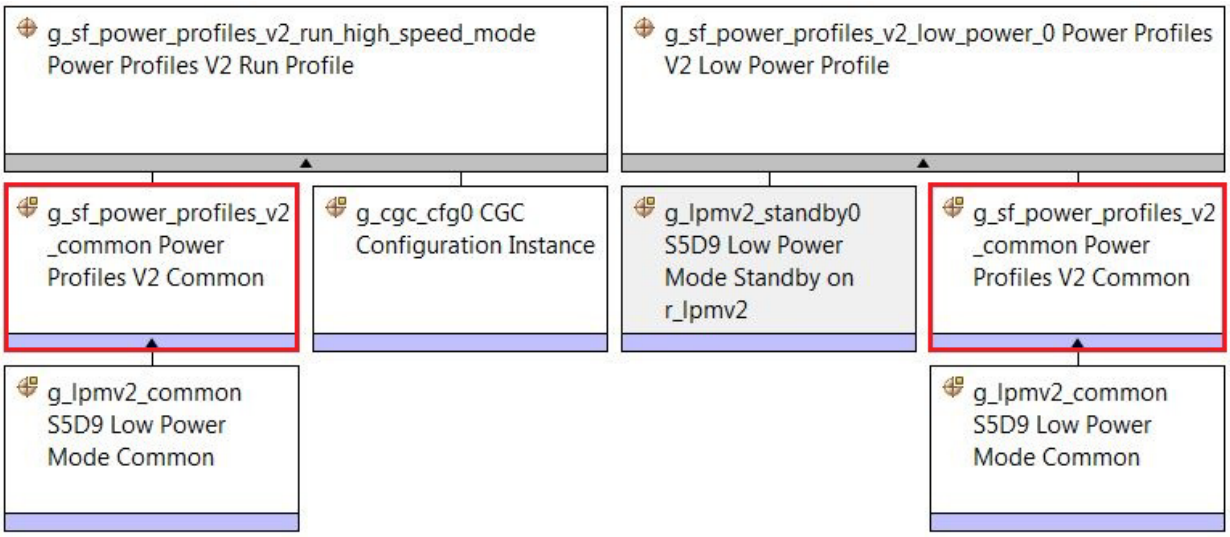


図 235: PPv2 実行プロファイルモジュールとローパワープロファイルモジュールの PPv2 共通モジュール

構成は次のように表示されます。

パワープロファイル V2 共通の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_sf_power_profiles_v2_common	モジュール名

- ユーザーが追加する MCU 固有のローパワーモード共通モジュール

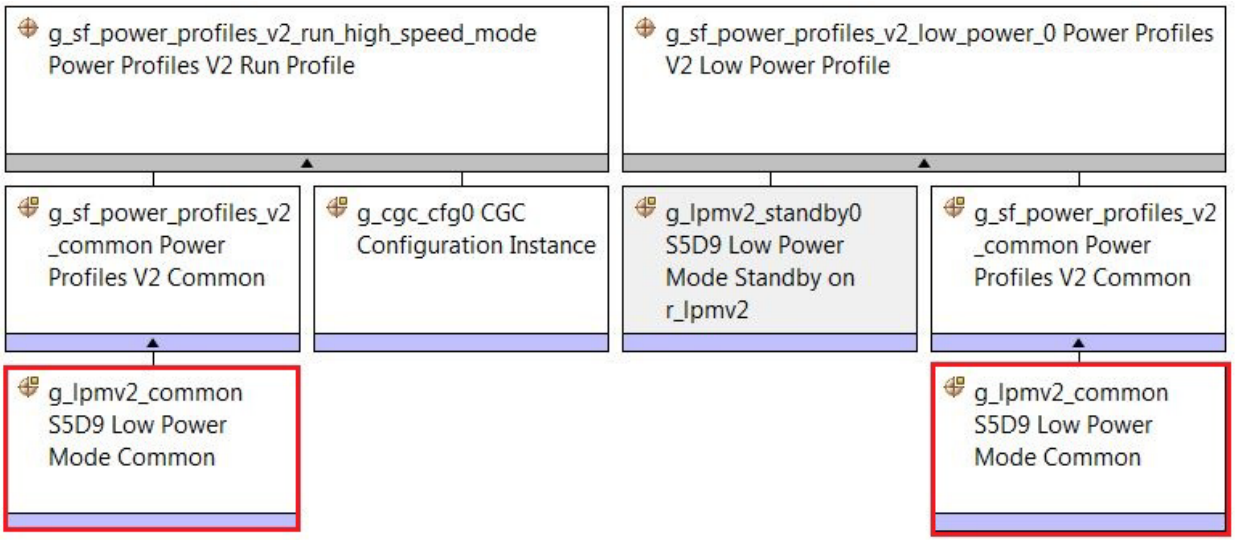


図 236: PPv2 実行プロファイルモジュールとローパワープロファイルモジュールの LPM 共通モジュール

構成は次のように表示されます。

ローパワーモード共通の構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_lpmv2_common	モジュール名。

注: *Synergy MCU S5D9 LPM モジュールは、 PPv2 フレームワーク実行プロファイルモジュールを追加するときに図13 に既に追加されています。*

パワープロファイル V2 フレームワークは、固有のクロック設定を必要としません。

アプリケーションは、ローパワーモード中に IO ポート状態を維持することもできます。

5.1.21.5 アプリケーションでのパワープロファイル V2 フレームワークモジュールの使用

モジュールを構成し、ISDE ファイルが生成されたら、以下の手順に従ってパワープロファイル V2 フレームワークをスレッドで使用します。以下の手順は、パワープロファイル V2 フレームワーク共通インスタンスのユーザー定義名が g_sf_power_profiles_v2_common. であることを前提とします。以下の手順は、パワープロファイル V2 フレームワークローパワーおよび実行プロファイルのユーザー定義名が g_sf_power_profiles_v2_low_power_0 および g_sf_power_profiles_v2_run_0. であることを前提とします。

API を使用する前に、ローパワープロファイルで構成されたコールバック関数の本体を定義する必要があります。MCU がローパワーモードに移行する直前と、ローパワーモードから復帰した直後に、コールバック関数によってアプリケーションに通知が行われます。コールバックの使用は任意ですが、パワープロファイル V2 のプロパティにコールバックを 1 つ定義している場合は、それに対する定義が必要です。

1) ThreadX が使用されている場合は、ユーザーアプリケーションコードに到達する前に、Synergy で生成されるコードによってパワープロファイル V2 フレームワークの open API 関数が呼び出されます。ThreadX が使用されていない場合は、アプリケーションが open 関数を呼び出す必要があります。

2) runApply() API を使用して任意の時点で実行プロファイルを適用します。 runApply() API は実行プロファイルを第 2 パラメータとして受け入れます。パラメータは任意の有効な実行プロファイルにすることができ、アプリケーションで実行プロファイル間を簡単に切り替えることができます。

```
g_sf_power_profiles_v2_common.p_api->runApply(g_sf_power_profiles_v2_common.p_ctrl,
&g_sf_power_profiles_v2_run_0);
```

3) lowPowerApply() API を使用してローパワープロファイルを適用します。lowPowerApply() API はローパワープロファイルを第 2 パラメータとして受け入れます。パラメータは任意の有効なローパワープロファイルにすることができ、アプリケーションでローパワープロファイル間を簡単に切り替えることができます。

```
g_sf_power_profiles_v2_common.p_api->lowPowerApply(g_sf_power_profiles_v2_common.p_ctrl,
&g_sf_power_profiles_v2_low_power_0);
```

4) *オプション*: close 関数を呼び出してフレームワークを閉じます。

```
g_sf_power_profiles_v2_common.p_api->close(g_sf_power_profiles_v2_common.p_ctrl);
```

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

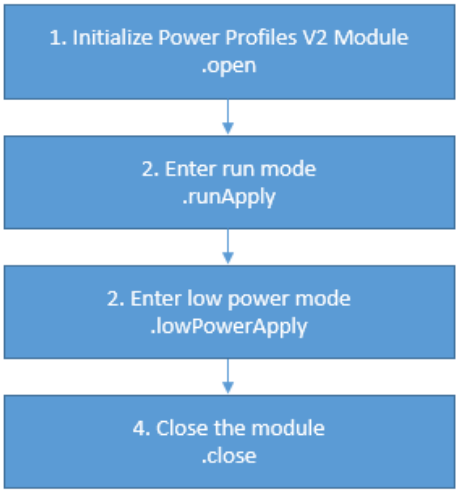


図 237: 通常のパワープロファイル V2 フレームワークモジュールアプリケーションのフロー図

5.1.22 SPI フレームワーク

SPI フレームワークモジュールは、SPI モードの SCI の SPI モード（SCI 共通低レベルモジュールとともに）、または RSPI ローレベルドライバーモジュールを使用して、Synergy マイクロコントローラ上の SPI と通信します。

- バス上の複数のデバイスをサポート
- モジュールの初期化、転送、クローズ用の高レベル API を提供
- 同期転送をサポート
- チップ選択操作をサポート
- バスロックをサポート

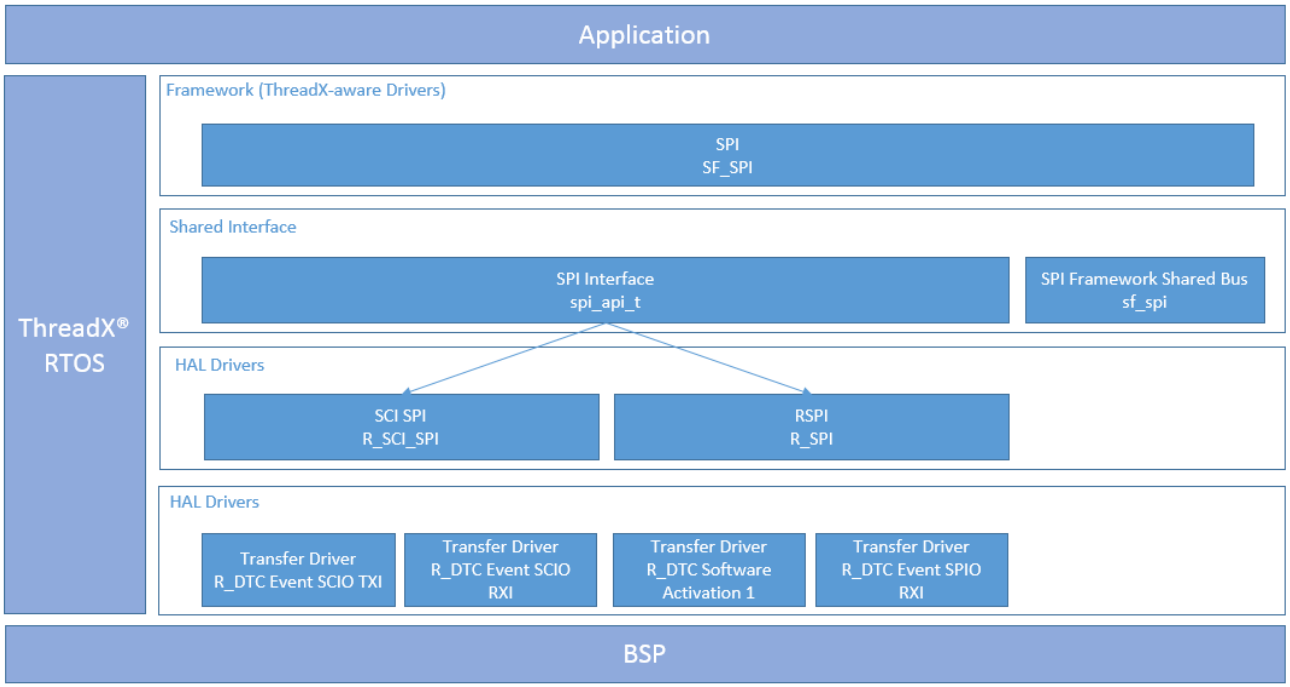


図 238: SPI フレームワークモジュールのブロック図

5.1.22.1 SPI フレームワークモジュール API の概要

SPI フレームワークモジュールは、オープン、クローズ、リード、ライト、その他の有用な機能の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

SPI フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_spi_device.p_api->open(g_sf_spi_device.p_cntl, g_sf_spi_device.p_cfg); 指定された SPI デバイスをバス上で開きます。
read	g_sf_spi_device.p_api->read(g_sf_spi_device.p_cntl, dst8, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); SPI デバイスからデータを受信します。

Function Name	API の呼び出し例と説明
write	g_sf_spi_device.p_api->write(g_sf_spi_device.p_cntl, src8, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); SPI デバイスにデータを送信します。
writeRead	g_sf_spi_device.p_api->writeRead (g_sf_spi_device.p_cntl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); SPI デバイスとの間で、データの送信と受信を同時に実行します（全二重通信）。この読み書き API は、ミューテックスオブジェクトを取得し、 SPI HAL レイヤで SPI データ送信を処理するとともに、SPI HAL レイヤからデータを受信します。またこの API は、イベントフラグ待ちを使用して、データ転送の完了と同期します。
close	g_sf_spi_device.p_api->close(g_sf_spi_device.p_cntl); 制御ハンドルで指定された SPI デバイスを無効化します。またバスにデバイスが接続されていない場合は、バスで使用される RTOS サービスを終了します。この関数は、ハンドルで指定された SPI チャネルへの電源を遮断し、関連付けられた割り込みを無効にします。
lock	g_sf_spi_device.p_api->lock(g_sf_spi_device.p_cntl); バスをデバイスにロックします。ロックすると、一定時間（ロックとロック解除の間など）、デバイスがバスを予約できるようになります。これにより、デバイスがバスに対する複数のリードとライトを割り込みなしで完了することができます。
unlock	g_sf_spi_device.p_api->unlock(g_sf_spi_device.p_cntl); 特定のデバイスのバスをロック解除し、他のデバイスがそのバスを使用できるようにします。
version	g_sf_spi_device.p_api->version(&version); バージョンポインタを使用して API バージョンを取得します。

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に完了しました
SSP_ERR_INVALID_MODE	無効なモード
SSP_ERR_INVALID_CHANNEL	無効なチャネル
SSP_ERR_IN_USE	使用中エラー
SSP_ERR_INVALID_ARGUMENT	無効な引数
SSP_ERR_QUEUE_UNAVAILABLE	キューが使用不能
SSP_ERR_INVALID_POINTER	無効なポインタ
SSP_ERR_INTERNAL	内部エラー
SSP_ERR_TRANSFER_ABORTED	転送中断
SSP_ERR_MODE_FAULT	モード障害
SSP_ERR_READ_OVF	リードオーバーフロー
SSP_ERR_PARITY	パリティエラー
SSP_ERR_OVERRUN	オーバーランエラー
SSP_ERR_UNDEF	不明なエラー
SSP_ERR_TIMEOUT	タイムアウトエラー
SSP_ERR_NOT_OPEN	デバイスが開かれていない

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.22.2 SPI フレームワークモジュールの動作の概要

SPI フレームワークモジュールは、SSP の階層化されたドライバーアーキテクチャに準拠します。また、SPI モジュール上の SCI または RSPI モジュールを使用して、Synergy マイクロコントローラ上の SPI と通信します。

フレームワークドライバーアーキテクチャは、「バス」と「バス上のデバイス」によるアーキテクチャを使用します。低レベルには一度に 1 つのデバイスのみを構成でき、残りのデバイスは必要に応じてリードまたはライト動作時に再構成されます。デバイスはバスがロックされているときにも再構成できます。どのデバイスも、接続先のバスに関連付けられています。ユーザーは、バス、フレームワーク、バスに接続されている各デバイスの低レベルモジュールを構成する必要があります。ユーザーは、複数のデバイスを同じバス上で構成するときに既存のフレームワーク共有バスを追加することができます。各SPIフレームワークは、ISDEコンフィギュレータで一意の名前を使用して構成する必要があります。

共通の開始および停止手順が、すべての SPI データ転送動作（リード、ライト、読み書き）に使用されます。開始プロセスの際に、フレームワークモジュールは再構成が必要かどうかを確認します。バスがロックされていない場合、チップ選択は、転送開始処理の中でアサートされ、転送終了処理の中でアサート解除されます。ユーザーは、チップ選択 IO ピンとチップ選択アクティブレベルを構成する必要があります。

SPI フレームワークモジュールではバスロック機能がサポートされるため、ユーザーが特定のペリフェラルのバスをロックできます。ロックすると、一定時間（ロックとロック解除の間）、デバイスがバスを予約できるようになります。これにより、一部のニーズに対応して、デバイスが複数の読み書き操作を中断することなく完了できるようになります。これにより、一部のニーズに対応して、デバイスが複数の読み書き操作を中断することなく完了できるようになります。ロックとロック解除の間にライトとリードを行っても、チップ選択ラインは変更されません。

SPI フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 複数の SPI デバイスが共通バスを共有するように構成できます。SPI フレームワークバスモジュールを構成したら、各種の SPI（デバイス）をそのバスに接続できます。
- バスに接続している SPI デバイスごとに、1 つの SPI HAL モジュールと 1 つの SPI フレームワークデバイスモジュールを追加する必要があります。
- フレームワークによって内部的に処理されるため、ユーザー定義コールバックは不要です。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。
- SPI バスの互換性については、MCU の仕様に関するマニュアルを参照してください。SPI バスとのデバイスの互換性はフレームワークでチェックされないため、互換性のない SPI デバイスによって不適切な動作が発生することがあります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.22.3 アプリケーションへの SPI フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して SPI フレームワークモジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

SPI フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（SPI フレームワークのデフォルト名は g_sf_spi_device0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

SPI フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_spi_device0 on sf_spi	**Threads**	**New Stack> Framework> Connectivity> SPI Framework Device on sf_spi**

次の図に示すように、sf_spi の SPI フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に **Add** というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、**[New]** アイコンが表示され、可能な選択肢が表示されます。

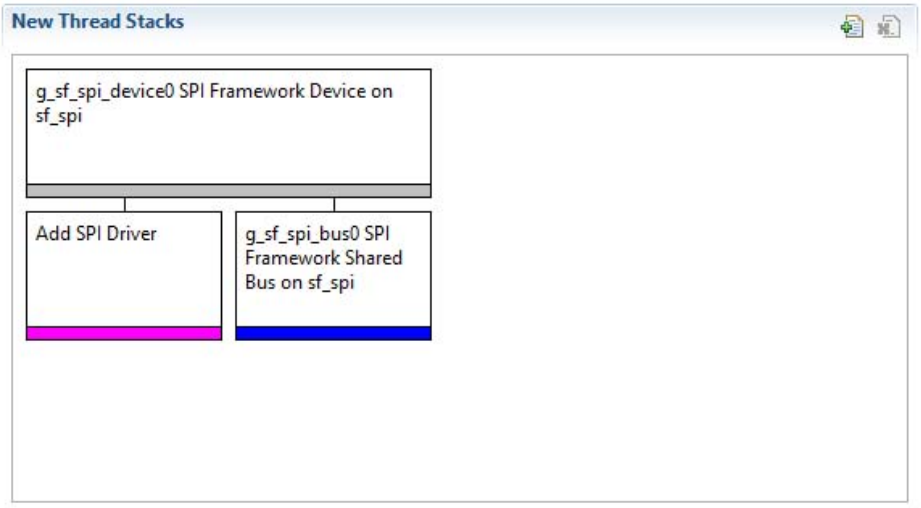


図 239: SPI フレームワークモジュールのスタック （ドライバー選択なし）

必要な低レベルモジュールが選択されると、モジュールスタックが次の図のように表示されます。

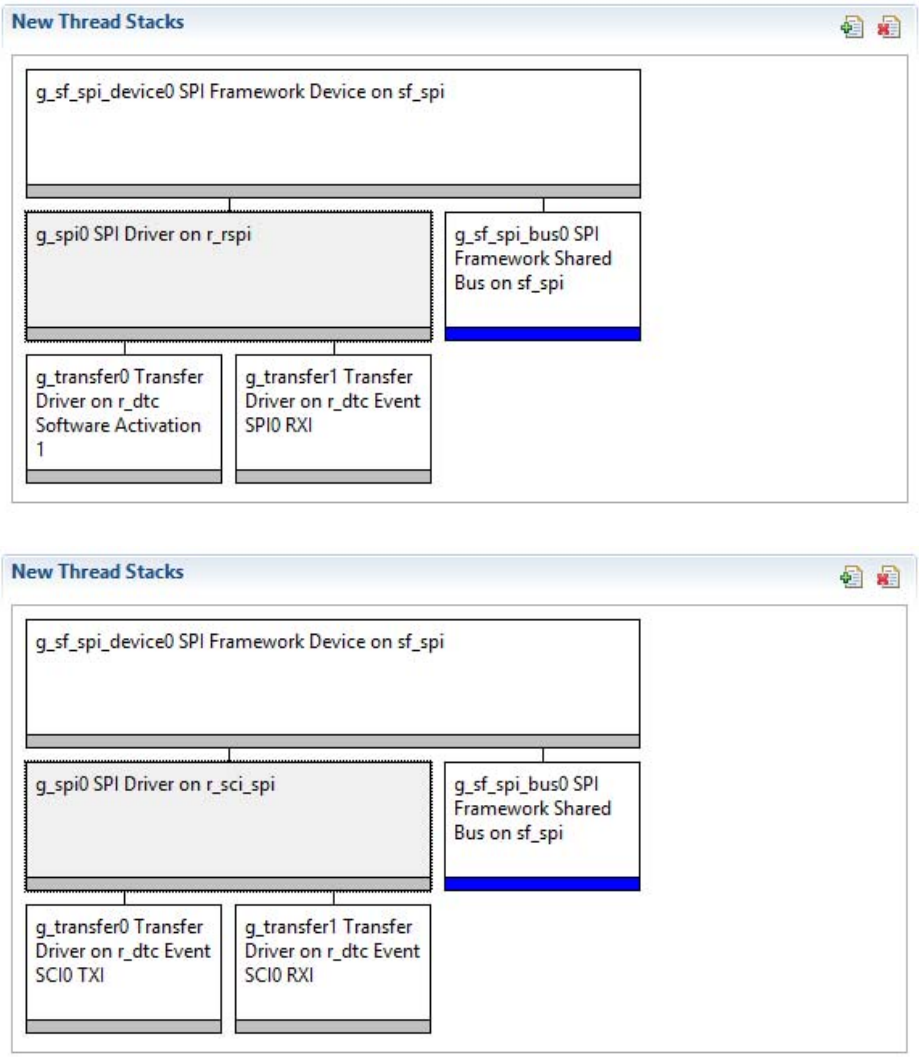


図 240: SPI フレームワークの実装

5.1.22.4 SPI フレームワークモジュールの設定

必要な動作に合わせて SPI フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。「ロック」されている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の **[Properties]** ウィンドウで「ロック」されているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性が示されることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

sf_spi 上の SPI フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定によって、パラメータが適切な制限範囲内にあることをチェックするコードが追加されるかどうかが決定されます。
Name	g_sf_spi_device0	モジュール名
Chip Select Port	00 thru 11 (Default: 00)	チップ選択に使用される GPIO ポートを選択します。
Chip Select Pin	00 thru 15 (Default: 00)	チップ選択に使用される GPIO ピンを選択します。
Chip Select Active Level	Low, High (Default: Low)	チップ選択信号の極性、アクティブハイまたはロー。

注: *例の値とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

スタックモジュールのデフォルト以外の値が望ましい場合もあります。たとえば、異なるチップ選択 GPIO やレベルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する* **[PROPERTIES]** *ウィンドウを調べることで決定されます。*

低レベルモジュールの SPI フレームワークモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_rspi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定によって、パラメータが適切な制限範囲内にあることをチェックするコードが追加されるかどうかが決定されます。
Name	g_spi0	モジュール名
Channel	0	チャネル番号
Operating Mode	Master, Slave (Default: Master)	動作モードの選択
Clock Phase	Data sampling on odd edge/ data variation on edge, Data sampling on even edge, data variation on odd edge (Default: odd/even)	クロック位相の選択
Clock Polarity	Low when idle, high when idle (Default: Low when idle)	クロック極性の選択
Mode Fault Error	Enable, Disable (Default: Disable)	モード障害エラーの選択
Bit Order	MSB First, LSB First (Default: MSB First)	ビット順序の選択
Bitrate	500,000	ビットレートの選択
Callback	NULL	コールバック関数名

ISDE Property	Value	説明
SPI Mode	SPI Operation, Clock Synchronous operation (Default: SPI Operation)	SPI モードの選択
SPI Communication Mode	Full Duplex, Transmit Only (Default: Full Duplex)	SPI 通信モードの選択
Slave Select Polarity (SSL0)	Active Low, Active High (Default: Active Low)	スレーブ選択極性の選択 0
Slave Select Polarity (SSL1)	Active Low, Active High (Default: Active Low)	スレーブ選択極性の選択 1
Slave Select Polarity (SSL2)	Active Low, Active High (Default: Active Low)	スレーブ選択極性の選択 2
Slave Select Polarity (SSL3)	Active Low, Active High (Default: Active Low)	スレーブ選択極性の選択 3
Select Loopback 1	Normal, Inverted (Default: Normal)	ループバック 1 の選択
Select Loopback 2	Normal, Inverted (Default: Normal)	ループバック 2 の選択
Enable MOSI Idle	Enable, Disable (Default: Disable)	MOSI アイドルの有効化の選択

ISDE Property	Value	説明
MOSI Idle State	MOSI Low, MOSI High (Default: MOSI Low)	MOSI アイドル状態の有効化の選択
Enable Parity	Enable, Disable (Default: Disable)	パリティ有効化の選択
Parity Mode	Parity Odd, Parity Even (Default: Parity Odd)	パリティモードの有効化の選択
Select SSL (Slave Select)	SSL0, SSL1, SSL2, SSL3 (Default: SSL0)	SSL 選択の選択
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep (Default: Do Not Keep)	転送後の SSL レベル選択の選択
Clock Delay Enable	Enable, Disable (Default: Disable)	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 RSPCK thru Clock Delay 8 RSPCK (Default: Clock Delay 1 RSPCK)	クロック遅延カウントの選択
SSL Negation Delay Enable	Enable, Disable (Default: Disable)	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 RSPCK thru Negation Delay 8 RSPCK (Default: Negation Delay 1 RSPCK)	ネゲート遅延カウントの選択

ISDE Property	Value	説明
Next Access Delay Enable	Enable, Disable (Default: Disable)	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 RSPCK thru Next Access Delay 8 RSPCK (Default: Next Access Delay 1 RSPCK)	次アクセス遅延カウントの選択
Receive Interrupt Priority	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	送信割り込み優先順位の選択。
Error Interrupt Priority	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	エラー割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy SK-S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択

ISDE Property	Value	説明
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Link section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンクセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation 1	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

注: *設定例とデフォルトは、Synergy S7 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI 時の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Link section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンクセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode)	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

sf_spi 上の SPI フレーム共有バスの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_sf_spi_bus0	モジュール名
SPI Implementation	SCI SPI	SPI 実装の選択
Channel	0	チャネルの選択

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定によって、パラメータが適切な制限範囲内にあることをチェックするコードが追加されるかどうかが決定されます。
Name	g_spi0	モジュール名
Channel	0	チャネル番号
Operating Mode	Master, Slave (Default: Master)	動作モードの選択
Clock Phase	Data sampling on odd edge/ data variation on even edge, Data sampling on even edge, data variation on odd edge (Default: odd/even)	クロック位相の選択
Clock Polarity	Low when idle, high when idle (Default: Low when idle)	クロック極性の選択

ISDE Property	Value	説明
Mode Fault Error	Enable, Disable (Default: Disable)	モード障害エラーの選択
Bit Order	MSB First, LSB First (Default: MSB First)	ビット順序の選択
Bitrate	100,000	ビットレートの選択
Callback	NULL	コールバック関数名
Receive Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	送信終了割り込み優先順位の選択。
Error Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 2)	エラー割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 TXI の DTC HAL モジュール

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Link section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンクセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

***注:** 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI の DTC HAL モジュール

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Link section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンクセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

***注:** 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SPI フレームワークモジュールのクロック構成

SPI 周辺モジュールは PCLKB をそのクロックソースとして使用します。 PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、ランタイムで CGC インタフェースを使用します。

SPI フレームワークモジュールのピン構成

SPI 周辺モジュールは、 MCU のピンを使用して外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は SPI ピンの選択例を示しています。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

SPI フレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select **Peripherals > Connectivity: SCI> SCI1**
RSPI	Pins	Select **Peripherals > Connectivity: SPI > SPI0**

注: *上の選択シーケンスでは SCI1 と SPI0 がドライバーに必要なハードウェアターゲットであることを想定し、下の選択シーケンスでは SPI0 が必要なターゲットであることを想定しています。*

SPI フレームワークモジュールのピン構成設定

Property	設定値	説明
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard (Default: Disabled)	SCI 上の SPI の動作モードとして簡易 SPI を選択します
Pin Group Selection	_A_Only, Mixed (Default: Mixed)	Pin Group Selection
CTS0_RTS0_SS0	None, P101 (Default: None)	SS0 ピンの選択

Property	設定値	説明
RXD0_SCL0_MISO0	None, P212 (Default: None)	MISO0 ピンの選択
SCK0	None, P100 (Default: None)	SCK0 ピンの選択
TXD1_SDA1_MOSI0	None, P213 (Default: None)	MOSI0 ピンの選択

SPI 通信フレームワークモジュールのその他の設定

外部チップ選択が使用されている場合は、チップ選択ピンを GPIO 出力として構成します。

5.1.22.5 アプリケーションでの SPI フレームワークモジュールの使用

アプリケーションで SPI フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して SPI フレームワークモジュールを初期化する
2) lock API を使用して連続転送のためにバスをロックする（必要な場合）
3) read API を使用してデータを読み取る
4) write API を使用してデータを書き込む
5) writeRead API を使用してデータのライトとリードを同時に行う
6) unlock API を使用してバスを連続転送からロック解除する（必要な場合）
7) close API を使用してモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

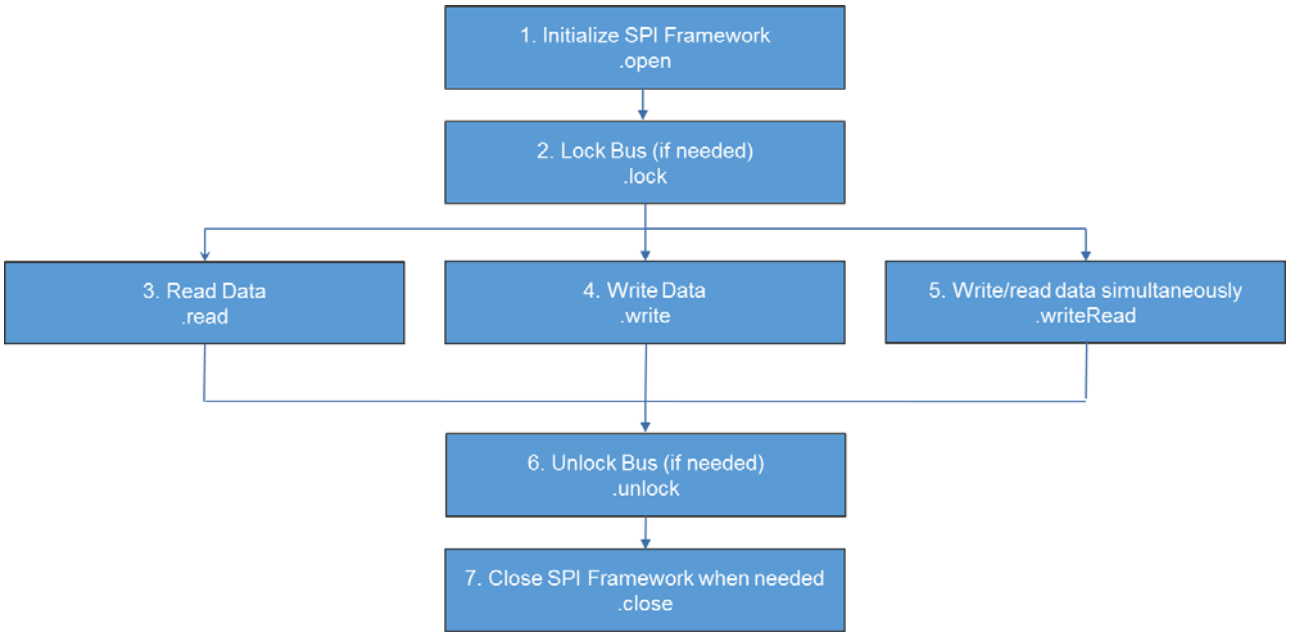


図 241: 通常の SPI フレームワークアプリケーションのフロー図

5.1.23 スレッド監視フレームワーク

スレッド監視フレームワークは、以下の機能を実装します。

- スレッド監視フレームワークインタフェースは、ウォッチドッグタイマを使用して RTOS スレッドを監視します。スレッド間メッセージングは、監視対象のスレッドが想定外の動作を行った場合、ウォッチドッグによるマイクロコントローラのリセットを強制的に実行します。
- スレッド間メッセージングは、API への変更なしに、WDT または IWDT および HAL モジュールを使用して、すべての Synergy デバイスをサポートするように設計されています。
- プロファイリングモードでは、登録されたスレッドのカウンタの最小値および最大値を決定できます。プロファイリングモード中は、ウォッチドッグタイマが常にリフレッシュされるため、デバイスはリセットされません。
- このフレームワークモジュールは、WDT と IWDT HAL モジュールの両方をサポートしています。

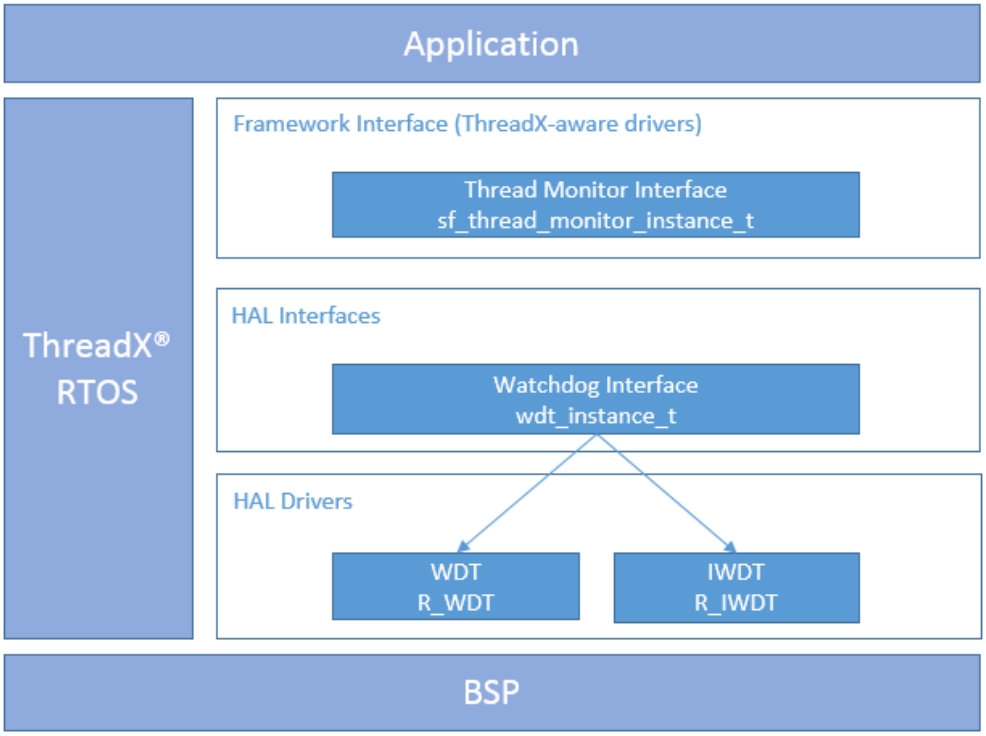


図 242:　スレッド監視フレームワークモジュールのブロック図

5.1.23.1 スレッド監視フレームワークモジュール API の概要

スレッド監視フレームワークは、フレームワークのオープンとクローズ、および監視用スレッドの登録と登録解除用の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。戻りステータス値の表は API 要約表の後にあります。

スレッド監視フレームワーク API 要約

Function Name	API の呼び出し例と定義
open	g_sf_thread_monitor.p_api->open (g_sf_thread_monitor.p_ctrl,g_sf_thread_monitor.p_cfg); WDT または IWDT モジュールを構成します。構成データから、WDT または IWDT のタイムアウト期間が決定されます。登録されたスレッドを監視するスレッドが作成されます。

Function Name	API の呼び出し例と定義
close	g_sf_thread_monitor.p_api->close (g_sf_thread_monitor.p_ctrl); スレッド監視スレッドをサスペンドします。ウォッチドッグはリフレッシュされなくなります。
threadRegister	g_sf_thread_monitor.p_api-> threadRegister (g_sf_thread_monitor.p_ctrl, &p_min_max_struct); 監視対象のスレッドを登録します。
threadUnregister	g_sf_thread_monitor.p_api-> threadUnregister (g_sf_thread_monitor.p_ctrl); スレッドを監視から除外します。
countIncrement	g_sf_thread_monitor.p_api-> countIncrement (g_sf_thread_monitor.p_ctrl); 監視対象のスレッドのカウント値を安全にインクリメントします。
versionGet	g_sf_thread_monitor.p_api-> versionGet(&version); API バージョンを取得し、バージョンポインタに格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	ポインタが NULL です。
SSP_ERR_IN_USE	スレッド監視が既に開かれています。

Name	説明
SSP_ERR_INVALID_MODE	オープン時に低レベルウォッチドッグがエラーを返しました。
SSP_ERR_UNSUPPORTED	データ構造体が割り当てられませんでした。
SSP_ERR_INVALID_ARGUMENT	ローレベルドライバーの1つ以上の構成オプションが無効です。
SSP_ERR_NOT_OPEN	SF_THREAD_MONITOR_Open が呼び出されていないか、正常に呼び出されませんでした。
SSP_ERR_INSUFFICIENT_SPACE	監視対象のスレッドの配列に十分なエントリが存在しないため、このスレッドに追加できません。sf_thread_moinitor_cfg.h の THREAD_MONITOR_CFG_MAX_NUMBER_OF_THREADS の値を増やしてください

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

5.1.23.2 スレッド監視フレームワークモジュールの動作の概要

スレッド監視は、スレッドがスレッド監視にカウンタ変数と、このカウンタ変数の予測される最小値と最大値を登録する、という形で動作します。監視されているスレッドは、動作中にカウンタ変数をインクリメントします。ウォッチドッグタイムアウト期間の半分の期間で、スレッド監視は登録されたスレッドのカウンタ変数をチェックします。最小値から最大値の範囲外となった場合、ウォッチドッグタイマはマイクロコントローラをリセットできるようになります。すべてが予測範囲内の場合は、ウォッチドッグタイマがリフレッシュされ、カウンタ変数はクリアされてゼロになります。

スレッド監視フレームワークモジュールの動作に関する重要な注意事項と制限事項

次の図は、スレッド監視フレームワークモジュールの動作に関するフローチャートを示しています。

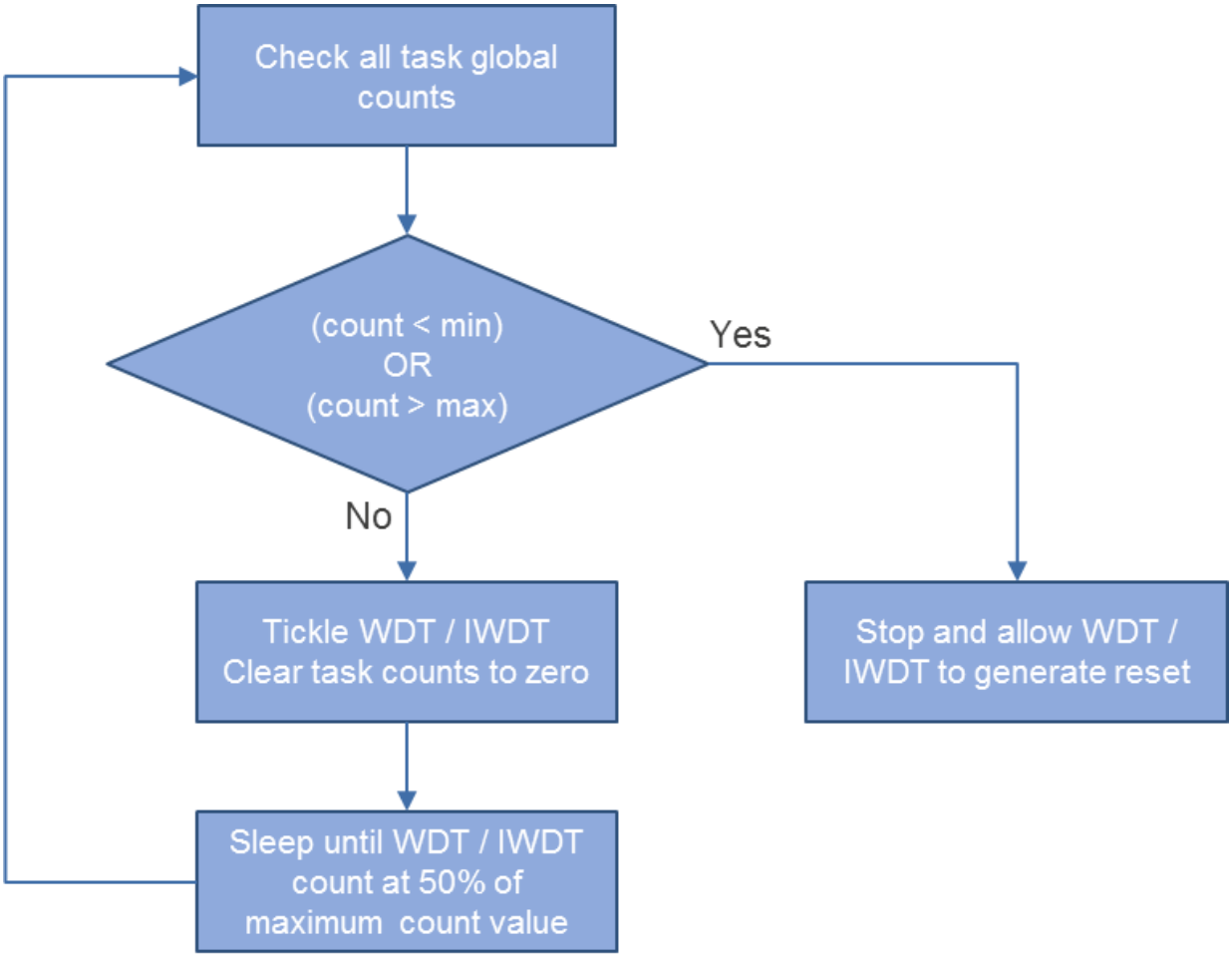


図 243: スレッド監視フレームワークの動作図

次の図は、WDT または IWDT がリフレッシュされた場合を示しています。有効なリフレッシュ期間は、カウント期間の中心にあたる 50%、すなわち 50% のカウント値の両側 25% にわたるものであることに注意してください。

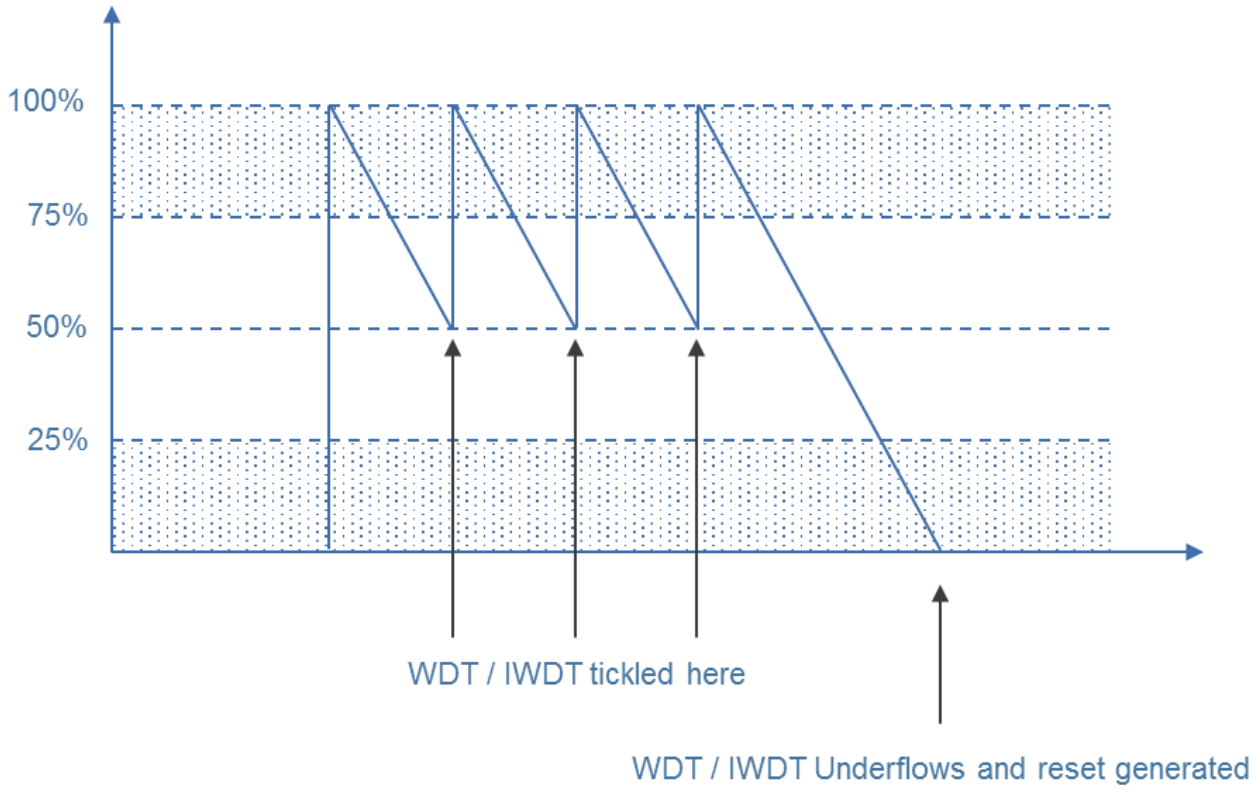


図 244: WDT または IWDT の動作図

- IWDT には、安全性を向上させるための独自のクロックソースがあります。
- WDT はアプリケーションから開始することができます。
- スレッドがスリープモードを実行していない場合は、WDT の stop-control プロパティを >[WDT Count Enabled in Low Power Mode] に設定します。スレッドが実行されていない場合（スリープ状態）、ThreadX® は WFI 命令を実行してデバイスを事実上ソフトスリープにしますが、これは WDT がカウントを停止する原因となります。

注: *監視対象スレッドファイルで WDT を開いたりリフレッシュしたりしないでください。これはスレッド監視フレームワークによって自動的に行われます。*

- 内部的に、スレッド監視フレームワークはウォッチドッグタイマのリセット期間の半分の速度で動作します。この速度により、スレッド監視フレームワークはウォッチドッグカウント値の 50% で動作するため、有効なリフレッシュウィンドウの範囲内に十分収まります。スレッド間メッセージングは、ローレベルウォッチドッグドライバーにクエリを実行することにより、内部的にリセット期間を計算します。
- スレッド監視フレームワークには、close API コールがあります。WDT と IWDT が使用されている場合は、これらを停止することはできません。スレッド監視フレームワークが閉じられた場合は、ウォッチドッグをリフレッシュするために他のプロビジョニングを実行する必要があり、このようなプロビジョニングを実行しなければデバイスはリセットされます。

- 特定のデバイスで JLink を使用して動作させている場合は、デバッグモードサポートが必要です。J-Link デバッグハードウェアを使用していると、WDT または IWDT のカウントは実行されません。スレッド監視フレームワークスレッドは通常、WDT または IWDT カウンタと同期されますが、JLink を使用して動作している場合はこのような同期化が省略されます。
- スレッド監視フレームワークスレッドには高プライオリティ（ThreadX で低い数値）を割り当てます。スレッド監視フレームワークの実行中に遅延が発生すると、有効なリフレッシュウィンドウの範囲外でウォッチドッグがリフレッシュされることがあり、マイクロコントローラがリセットされる原因となります。スレッド監視フレームワークスレッドは長時間にわたる動作はしないため、システムの性能に影響を及ぼすことはありません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

5.1.23.3 アプリケーションへのスレッド監視フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにスレッド監視フレームワークを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらのタスクに精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

スレッド監視フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（スレッド監視フレームワークのデフォルト名は g_thread_monitor0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

スレッド監視フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_thread_monitor0 Thread Monitor Framework	Threads	New Stack> Framework> Services> Thread Monitor Framework on sf_thread_monitor

sf_thread_monitor上のスレッド監視フレームワークが次の図に示すようにスレッドスタックに追加されると、コンフィギュレータは、スタックを完了するには少なくとも 1 つのウォッチドッグタイマが必要であることを（**[Add WDT]** ブロックを介して）レポートします。追加の構成情報を必要とする低レベルモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、低レベルモジュールの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。低レベルモジュールの追加が必要な場合は、モジュールの説明に「**Add**」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、**[New]** アイコンが表示され、可能な選択肢が表示されます。

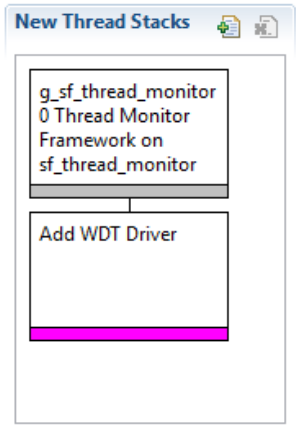


図 245:　スレッド監視フレームワークモジュールのスタック

5.1.23.4 スレッド監視フレームワークモジュールの構成

ユーザーは必要な動作に合わせてスレッド監視フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。

ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

sf_thread_monitor 上のスレッド監視フレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	API パラメータチェック用のコードを含めるかどうかを制御します。

Parameter	Value	説明
Maximum Number of Monitored Threads	5	監視可能な最大スレッド数。
Name	g_sf_thread_monitor0	モジュール名。
Profiling Mode	Enabled, Disabled, (Default: Disabled)	プロファイリングモード有効の要否。
Thread Monitor Thread Priority	1	スレッド監視の内部スレッドのプライオリティ。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、監視対象のスレッドの最大数を目的の値に設定することが役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

スレッド監視フレームワークの低レベルモジュールの構成設定

通常、低レベルモジュールでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_iwdt 上の独立ウォッチドッグタイマの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックコードを含みます
Name	g_wdt0	モジュール名
NMI Callback	NULL	コールバック名

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_wdt 上のウォッチドッグタイマの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックコードを含みます

ISDE Property	Value	説明
Name	g_wdt0	モジュール名
Start Mode	Register, Auto (Default: Register)	スタート モードをレジスタ スタートまたはオートスタートに設定します。
Start Watchdog After Configuration	True, False (Default: True)	WDTが初期化の際に開始するかどうかを制御します
Timeout	1024 cycles, 4096 cycles, 8192 cycles, 16384 cycles (Default: 16384 cycles)	WDT タイムアウト期間。
Clock Division Ratio	PCLK/4, PCLK/64, PCLK/128, PCLK/512, PCLK/2048, PCLK/8192 (Default: PCLK/8192)	WDT クロック分周器
Window Start Position	100% (Window Position Not Specified); 75%, 50%, 25% (Default: 100%)	許可されたリフレッシュ周期の開始位置。
Window End Position	0% (Window Position Not Specified; 25%, 50%, 75% (Default: 0%)	許可されたリフレッシュ周期の終了位置。
Reset Control	Reset Output, NMI Generated (Default: Reset Output)	WDT が MCU をリセットするか、NMI を生成するかを選択します。
Stop Control	WDT Count Enabled in Low Power Mode, WDT Count Disabled in Low Power Mode (Default: Disabled)	WDT が低電力モードでカウントを停止するかどうかを選択します。
NMI Callback	NULL	コールバック。ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、IRQnがトリガーされるたびに割り込みサービスルーチン(ISR)から呼び出されます。注意：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないようにしてください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのプロパティ設定の大半は直観的であり、通常はSSPコンフィギュレータから対応する[Properties]ウィンドウを調べることで決定されます。

スレッド監視フレームワークモジュールのクロック構成

ISDE を使用して、 **[Clocks]** タブを使用して WDT クロックを構成します。

初期設定では、 WDT は PCLKB 周波数に基づいて動作します。クロックコンフィギュレータの [Configuring Clocks]を使用するか実行時にCGCインタフェースを使用して、e^2 studio統合ソリューション開発環境（ISDE）で PCLKB 周波数を設定します。

PCLKB が 60 MHz で動作している状態での WDT の最大タイムアウト期間は約 2.24 秒です。

IWDT クロックが 15 kHz で動作している場合の最大タイムアウト期間は、 35 秒をわずかに下回ります。

スレッド監視フレームワークモジュールのピン構成

スレッド監視フレームワークは、動作にあたってピンを必要としません。

スレッド監視フレームワークモジュールアプリケーションのスレッド

スレッド監視フレームワークにより監視されるスレッドが監視モジュールと連携するには、実装されている必要があります。監視するスレッドがスレッド間メッセージングに登録されていると、予測された最小および最大カウント値が、これらの値を含む sf_thread_monitor_counter_min_max_t 型の構造体へのポインタを通じて渡されます。

スレッドのカウンタと最小値および最大値は、g_sf_thread_monitor.p_api->threadRegister() を呼び出してスレッド監視フレームワークに登録する必要があります。

監視対象のスレッドのループが完了するたびに、 g_sf_thread_monitor.p_api->countIncrement() を呼び出してカウンタの値を更新する必要があります。

その他のスレッド監視フレームワークモジュール設定

スレッド監視フレームワークは割り込みを使用しません。WDT または IWDT はリセットを生成するように構成する必要があります。

5.1.23.5 アプリケーションでのスレッド監視フレームワークモジュールの使用

アプリケーションでスレッド監視フレームワークを使用する際の一般的な手順は次のとおりです。

1) open API を使用してスレッド間メッセージングを初期化する
2) threadRegister API で監視する必要のあるスレッドを登録する
3) countIncrement API を使用して、登録されたスレッドが実行されるたびにカウントをインクリメントする。
4) 登録されたスレッドを監視する必要がなくなったら、threadUnregister API を使用してスレッドを登録解除する。
5) close API でスレッド間メッセージングを閉じる（スレッド間メッセージングがアプリケーションで不要になった場合のみ）。

多くの場合、カウントのインクリメント手順は周期的に繰り返します。これらの一般的な手順を、次の図の通常の動作フローに示します。

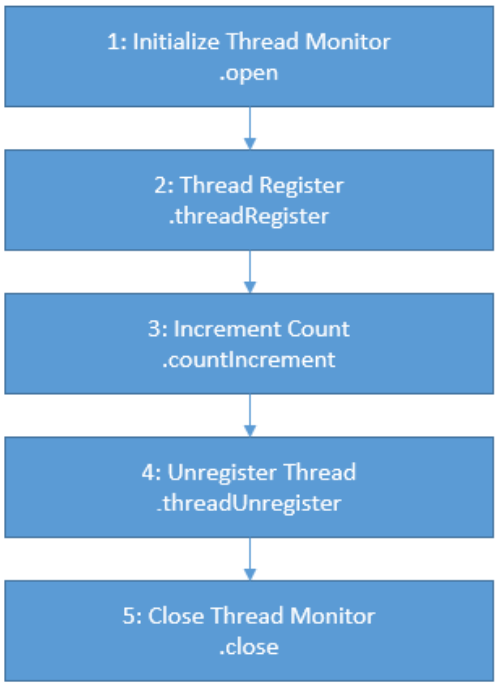


図 246: 通常のスレッド監視フレームワークモジュールアプリケーション

5.1.24 タッチパネルフレームワーク

タッチパネルフレームワークは、以下の機能をサポートします。

- タッチコントローラからデータを読み取り、メッセージフレームワークのタッチイベントにサブスクライブされたキューに対してタッチメッセージをパブリッシュする
- 位置データ（X および Y 座標）を提供する
- タッチイベントタイプ（下、上、移動、保持、無効）を提供する
- 外部割り込み要求による I2C ベースのタッチパネルの実装をサポートする

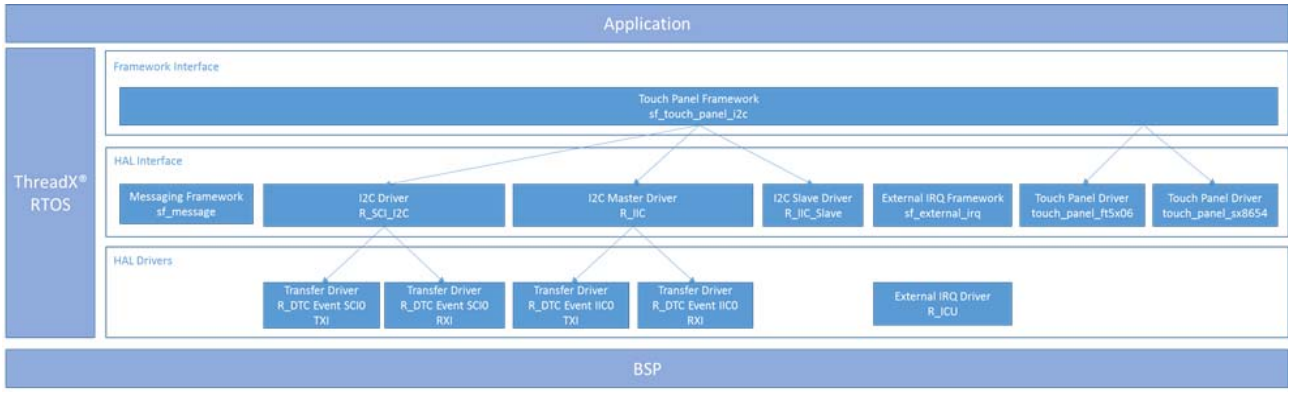


図 247:　タッチパネルフレームワークモジュールのブロック図

5.1.24.1 タッチパネルフレームワークモジュール API の概要

タッチパネルフレームワークモジュールは、オープン、較正、開始、停止、クローズなどの主要機能の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

タッチパネルフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_touch_panel_i2c0.p_api->open(g_sf_touch_panel_i2c0.p_ctrl, g_sf_touch_panel_i2c0.p_cfg); 必要な RTOS オブジェクトを作成し、ハードウェア固有の初期化のための低レベルモジュールを呼び出し、メッセージキューにタッチデータをポストするスレッドを作成します。
calibrate	g_sf_touch_panel_i2c0.p_api->calibrate(g_sf_touch_panel_i2c0.p_ctrl, &expected, &actual, timeout); 指定された期待座標に基づいて、較正ルーチンを開始します。許容誤差が期待タッチポイントと実測タッチポイントの差よりも大きい場合のみ、SSP_SUCCESS が返されます（以下の式を使用：p_calibrate->tolerance_pixels \^ 2 > (p_calibrate->x - x_measured) \^ 2 + (p_calibrate->y - y_measured) \^ 2）

Function Name	API の呼び出し例と説明
start	g_sf_touch_panel_i2c0.p_api->start(g_sf_touch_panel_i2c0.p_ctrl); タッチイベントのスキャンを開始します。
stop	g_sf_touch_panel_i2c0.p_api->stop(g_sf_touch_panel_i2c0.p_ctrl); タッチイベントのスキャンを停止します。
reset	g_sf_touch_panel_i2c0.p_api->reset(g_sf_touch_panel_i2c0.p_ctrl); リセットピンが提供されている場合は、タッチコントローラをリセットし、I2C バスをリセットします。
close	g_sf_touch_panel_i2c0.p_api->close(g_sf_touch_panel_i2c0.p_ctrl); タッチスレッドを終了し、HAL レイヤーでチャネルを閉じます。
versionGet	g_sf_touch_panel_i2c0.p_api->versionGet(&version); API バージョンを取得し、バージョンポインタに格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	ポインタパラメータが NULL であったか、低レベルドライバーがこのエラーをレポートしました。

Name	説明
SSP_ERR_INTERNAL	タッチパネルスレッドまたはイベントフラグが作成できなかったか、低レベルドライバーがこのエラーをレポートしました。
SSP_ERR_CALIBRATE_FAILED	タッチの実測値が期待範囲外です。
SSP_ERR_NOT_OPEN	タッチパネルが設定されていません。SF_TOUCH_PANEL_I2C_Open を呼び出してください。
SSP_ERR_IN_USE	ミューテックスが使用できなかったか、低レベルドライバーがこのエラーをレポートしました。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.1.24.2 タッチパネルフレームワークモジュールの動作の概要

タッチパネルフレームワークモジュールは、タッチコントローラからデータを読み取り、メッセージフレームワークのタッチイベントにサブスクライブされたキューに対してタッチメッセージをパブリッシュします。タッチイベントには、位置データ（X および Y 座標）や、タッチイベントの種類（上、下、移動、保持、または無効）が含まれます。タッチパネルフレームワークは、外部割り込みを使用する I2C ベースのタッチパネル実装をサポートしています。タッチパネルフレームワークは、内部的にスレッドを作成し、タッチコントローラをリードします。

タッチフレームワークを使用する場合の通常のフローは次のとおりです。

1) タッチメッセージを保留にします。
2) タッチメッセージを解析します。

タッチパネルフレームワークモジュールの動作に関する重要な注意事項と制限事項

初期構成

タッチパネルフレームワークをプロジェクトに追加すると、e^2 studio ISDE はタッチイベントクラスと新しいデータイベントをプロジェクトコンフィギュレータの [Messaging] タブに自動的に生成します。プロジェクトでタッチイベントメッセージ subscriber(s) を構成するには、次の手順を使用します。

- e^2^ studio Synergy コンフィギュレータの [Messaging] タブの [Event Classed] ペインで、タッチイベントクラスを選択します。
- [Messaging] タブの [Touch Subscriber] ペインでサブスクライバースレッドのチェックボックスを選択します（スレッドがペインに表示されていない場合は、[Touch Subscriber] ペインの右上にある [New] アイコンをクリックしてスレッドを追加します）。

カスタムタッチパネルチップドライバーの作成

カスタムタッチチップドライバーを作成するには、synergy/ssp_supplemental/ touch_drivers にある SSP の既存のドライバーコードを参照し（このディレクトリは既存のタッチチップドライバーが Synergy コンフィギュレータで選択されている場合にのみ表示され、この操作を行うには Synergy コントローラに移動して [New] >

[Framework] > [Input] > [Touch Panel Framework on sf_touch_panel_i2c] > [Add Touch Driver] を選択し、参照用の既存のタッチドライバーを選択してプロジェクトコンテンツを生成します）、次の手順と擬似コードを使用してタッチパネルフレームワークに接続します。

- タッチチップドライバーインスタンスを実装し、ドライバーをアタッチするタッチパネルフレームワークを有効化します。
- payloadGet 関数を実行し、タッチパネルコントローラデバイスから I2C インタフェースを介してタッチイベントとタッチ座標を取得します。
- reset 関数を実行し、関連付けられた GPIO ピンと I2C インタフェースを使用してタッチパネルコントローラデバイスをリセットします。

カスタムタッチパネルチップドライバーの構成

プロジェクトに対してカスタムタッチチップドライバーを構成するには、以下の手順を使用します。

1) **Synergy** プロジェクトを作成します。
2) 以下の手順に従って既存のタッチドライバー XML をカスタムタッチドライバー用に更新して、既存のタッチ XML を変更します。 A. プロジェクトルートフォルダの下にある .module_descriptions フォルダに移動します。 B. タッチドライバー XML: Renesas##HAL Drivers##touch panel##touch_panel_i2c_ft5x06####<version> または Renesas##HAL Drivers##touch panel##touch_panel_i2c_sx8654####<version> を編集します。 C. **value** フィールドのインスタンス構造体の名前をカスタムドライバーインスタンス構造体の名前に変更します。 D. 新しいインスタンス宣言を [property] タブの後に追加します。E.XML 内のすべてのタッチチップ番号をカスタムタッチチップ番号に変更します。 F.XML ファイルを保存して閉じます。
3) プロジェクトコンフィギュレータを開き、 [New] > [Framework] > [Input] > [Touch Panel Framework on sf_touch_panel_i2c] の順に選択してタッチパネルフレームワークを追加します（コンフィギュレータが既に開いていた場合は、いったん閉じてから再び開く必要があります）。
4) すべてのコンポーネントを構成します。
5) カスタムタッチパネルチップドライバーを **[Add Touch Driver]** ボックスに追加します。（上記の説明に従って XML を正しく変更した場合は、開いているカスタムタッチドライバーが表示されます）。
6) プロジェクトコンテンツを生成し、新規ディレクトリ（たとえば、 touch_panel_i2c_xxxxxx) 構造体を synergy/ ssp_supplemental/ touch_drivers, の下に追加します。ディレクトリ構造体は次のようになります : synergy/ ssp_supplemental/ touch_drivers/ touch_panel_i2c_xxxxxx （xxxxxx はタッチコントローラのパーツ番号で置換します）。
7) カスタムドライバーコードをこのディレクトリに追加します（上記の「**カスタムタッチパネルチップドライバーの作成**」セクションの説明に従って作成したコードをコピーして、 ssp_supplemental/ touch_drivers/ touch_panel_i2c_xxxxxx). に貼り付けます）。
8) ビルドに既存のドライバーが存在する場合は除外します（.c ファイルを右クリック > [Exclude from build…] > [Select all] > [OK]）。
9) コードをビルドします。

- ユーザーコールバックは、 I2C バス通信手順では使用できません。

- reset API は、以下の条件が満たされた場合にのみ使用できます : (1) stop API が呼び出されている。(2) タッチイベントが発生し、タッチパネルフレームワークがタッチイベントメッセージをポストしている。
- タッチパネルドライバーのアドオンディレクトリは （SSPv1.2.0-b.1 の） renesas_sybd から ssp_supplemental （SSPv1.2.0）に変更されました。 SSPv1.2.0-b.1 と SSPv1.2.0 の両方が開発環境にインストールされている場合は、両方のタッチパネルモジュールが [component] タブで選択されます。プロジェクトで、ビルドから renesas_sybd フォルダを除外します。 renesas_sybd フォルダを右クリック > [Exclude from build] -> [Select all] -> [OK]。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.24.3 アプリケーションへのタッチパネルフレームワークワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してタッチパネルフレームワークモジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

タッチパネルフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（タッチパネルフレームワークのデフォルト名は sf_touch_panel_i2c. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

タッチパネルフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_panel_i2c0 Touch Panel Framework on sf_touch_panel_i2c	Threads	New Stack> Framework> Input> Touch Panel Framework on sf_touch_panel_i2c

次の図に示すように、 sf_touch_panel_i2c のタッチパネルフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、 1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に 「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、 [New] アイコンが表示され、可能な選択肢が表示されます。

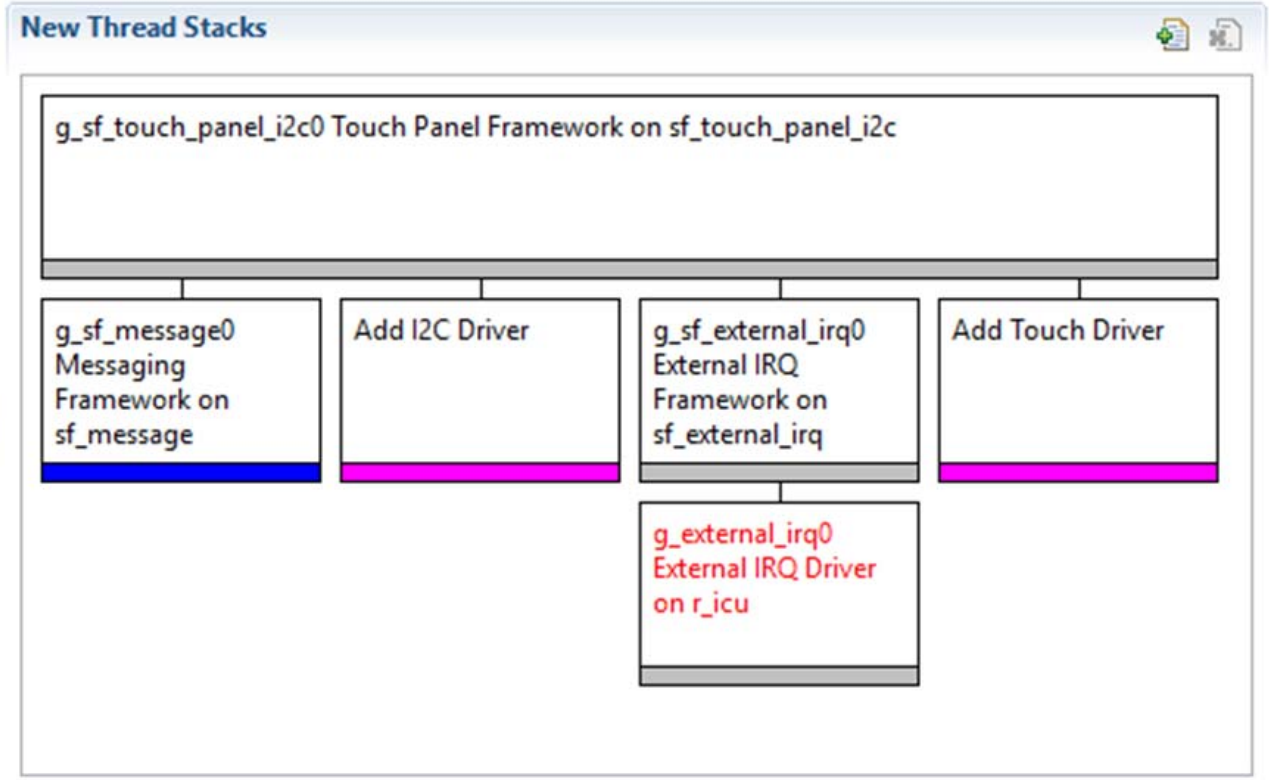


図 248:　タッチパネルフレームワークモジュールのスタック

5.1.24.4 タッチパネルフレームワークモジュールの構成

ユーザーは必要な動作に合わせてタッチパネルフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。**ロック**されている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウで**ロック**されているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の**手動**での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: ISDE を開き、タッチパネルフレームワークを作成し、次の構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。

sf_touch_panel_i2c 上のタッチパネルフレームワークモジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled ;Default: BSP	パラメータチェックを有効または無効にします。
Name	g_sf_touch_panel_i2c0	モジュール名。
Thread Priority	Default: 3	スレッドプライオリティの選択。
Hsize Pixels	Default: 800	幅のサイズピクセルの選択。
Vsize Pixels	Default: 480	高さのサイズピクセルの選択。
Update Hz	Default: 10	更新 hz の選択。
Reset Pin	IOPORT_PORT_10_PIN_02	リセットピンの選択。
Touch Event Class Instance Number	0	タッチイベントクラスインスタンス番号の選択。
Touch Coordinate Rotation Angle (Clockwise)	0, 90 (select this if 'Screen Rotation Angle' in GUIX Port is '270'), 180, 270 (select this if 'Screen Rotation Angle' in GUIX Port is '90') Default: 0	タッチ座標のローテーション角度の選択。

注: 上記の設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、異なる画面サイズの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: 下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

タッチパネルフレームワークの低レベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。タッチパネルフレームワークは、以下のローレベルドライバーから構成されます。

- sf_message 上のメッセージフレームワーク
- I2C ドライバー : riic 上の I2C マスタドライバーまたは r_sci_i2c 上の I2C マスタドライバー
- sf_external_irq 上の外部 IRQ フレームワーク
- タッチパネルドライバー

次の表は、モジュールの [properties] セクション内のローレベルドライバーのすべての設定を示しています。

sf_message 上のメッセージフレームワークの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	メッセージサブスクライバーの ThreadX メッセージキューのサイズをバイト単位で指定します。この値は、すべてのメッセージキューに適用されます。
Name	g_sf_message0	メッセージングフレームワークモジュール制御ブロックインスタンスの名前。
Work memory size in bytes	2048	作業メモリサイズをバイト単位で指定します。小さい数を選択すると、同時に割り当てることができるバッファの数が少なくなります（総バッファ数は sf_message_ctrl_t::number_of_buffers). で確認できます）。この値が、同時にポストできるピークメッセージ数よりも小さい場合、フレームワークでバッファ割り当てが失敗し、システムのパフォーマンスに影響を与えます。
Pointer to subscriber list array	p_subscriber_lists	サブスクライバーリスト配列へのポインタ名を指定します。
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	フーレムワークが制御ブロック内に作成するメモリブロックメモリの名前です。このパラメータは、デバッグ目的で有用な可能性がありますが、メモリを節約するために NULL を指定できます。

r_sci_i2c 上の I2C ドライバーの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_i2c0	モジュール名。
Channel	0 to 9 Default: 0	この設定で使用する SCI チャネルを指定します。 SCI には以下のチャネルがあります : Series S7 : 0 1 2 3 4 5 6 7 8 9; Series S3 : 0 1 2 3 4 - - - - 9; Series S1 : 0 1 - - - - - - - 9。
Rate	Standard, Fast-mode Default: Standard	標準および高速。
Slave Address	0x00	スレーブデバイスのアドレス。
Address Mode	7-Bit, 10-Bit Default: 7-Bit	現在、7ビットアドレスのみがサポートされています。
SDA Output Delay (nano seconds)	300	SDA 出力遅延 （ナノ秒単位）。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数を有効化します。
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 i2c_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン (ISR) から呼び出されます。 警告 : コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

ISDE Property	Value	説明
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信終了割り込み優先順位の選択。

r_dtc イベント SCI0 TXI 時の DTC HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled / /Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled / /Default: Enabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。

ISDE Property	Value	説明
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

r_dtc イベント SCI0 RXI 時の DTC HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Value	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

r_riic 上の I2C マスタドライバーの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_i2c0	モジュール名。
Channel	0, 1, or 2	この設定で使用するIICチャネルを指定します。
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	標準、高速および高速プラス。（「IICレート計算」を参照。）
Slave Address	0x00	I2C マスターが通信するスレーブデバイスのアドレスを設定します。
Address Mode	7-Bit, 10-Bit Default: 7-Bit	現在、7ビットアドレスのみがサポートされています。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 i2c_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信終了割り込み優先順位の選択。

ISDE Property	Value	説明
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	エラー割り込み優先順位の選択。

r_dtc IIC0 TXI 時の DTC HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択

ISDE Property	Value	説明
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IIC0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

r_dtc IIC0 RXI 時の DTC HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。

ISDE Property	Value	説明
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IIC0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

sf_external_irq 上の外部 IRQ フレームワークの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
Name	g_sf_external_irq0	フレームワーク名。
Event	None, Semaphore Put Default: Semaphore Put	イベントの選択。

r_icu 上の外部 IRQ HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック設定は、パラメータチェックコードの追加を有効化または無効化します。
Name	g_external_irq0	モジュール名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	エッジトリガまたはレベルトリガを設定します。
Digital Filtering	Enabled, Disabled Default: Disabled	デジタルフィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	True, False Default: True	初期化の直後に割り込みが有効化されるかどうかを決定します。
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、IRQn がトリガーされるたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

ISDE Property	Value	説明
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位を irq_ipl で登録できます。

touch_panel_ft5x06 上のタッチパネルドライバーの構成

ISDE Property	Value	説明
Name	g_sf_touch_panel_i2c_chip_ft5x06	モジュール名。

touch_panel_sx8654 上のタッチパネルドライバーの構成

ISDE Property	Value	説明
Name	g_sf_touch_panel_i2c_chip_sx8654	モジュール名。

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

タッチパネルフレームワークのクロック構成

ここで説明する I2C インタフェースの実装例では、SCI を使用しています。その他の実装にはさまざまな選択肢がありますが、次の例から推測できます。SCI 周辺モジュールは PCLKB をそのクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、ランタイムで CGC インタフェースを使用します。構成時に、I2C 転送レートは、ユーザー選択 PCLB レートとユーザー選択転送レートに基づいて、ドライバーにより内部で計算および設定されます。ユーザー選択レートを実現できないような構成が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

タッチパネルフレームワークのピン構成

ここで説明する I2C インタフェースの実装例では、SCI を使用しています。その他の実装にはさまざまな選択肢がありますが、次の例から推測できます。Synergy キット固有のピン設定を次のセクションに示します。SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。次の設定例は、Synergy S7G2 および SK-S7G2

キットを使用するプロジェクトを示しています。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

注: *動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

SCI 上の I2C マスタドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI0	Pins	Select **Peripherals > Connectivity: SCI> SCI0**

注: *上記の選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

SCI 上の I2C マスタドライバーのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard (Default: Disabled) Simple I2C	SCI 上の SPI の動作モードとして簡易 I2C を選択します
RXD1_SCL1_MISO1	None, P212, P708 (Default: None)	SCL ピン
TXD1_SDA1_MOSI1	None, P213, P709 (Default: None)	SDA ピン

注: *前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

タッチパネルフレームワークモジュールの Synergy キット固有の設定

上記の選択項目に加えて、使用されるタッチチップに基づいていくつかのキット固有の設定が必要な場合があります。これらの設定を次の例にまとめます。この例では、独自のカスタムタッチコントローラデバイスに対してこれらを構成する方法が示されます。

次の外部 IRQ 設定を構成します。

DK-S7G2 （タッチチップ SX8654）では、以下を選択します。

- チャネル : 7
- トリガ : Falling

SK-S7G2 （タッチチップ SX8654) では、以下を選択します。

- チャネル : 9
- トリガ : Falling

PE-HMI1-S7G2 （タッチチップ FT5x06) では、以下を選択します。

- チャネル : 12
- トリガ : Falling

すべてのボードの共通設定 :

- デジタルフィルター設定 : Any
- 初期化後に割り込みを有効にする : True
- コールバック : NULL

次の I2C ドライバー設定を構成します。

DK-S7G2 の場合は、 r_sci_i2c: を選択します。

- チャネル : 7
- 速度 : Standard
- スレーブアドレス : 0x48
- アドレスモード : 7-bit

SK-S7G2 では、 r_riic を選択します。

- チャネル : 2
- 速度 : Standard
- スレーブアドレス : 0x48
- アドレスモード : 7-bit

PE-HMI1-S7G2 では、 r_riic を選択します。

- チャネル : 1
- 速度 : 高速モード
- スレーブアドレス : 0x38
- アドレスモード : 7-bit

すべてのボードの共通設定 :

- コールバック : NULL

次のタッチパネルフレームワークモジュール設定を構成します。

DK-S7G2 の場合 :

- タッチチップ : g_sf_touch_panel_i2c_chip_sx8654
- 幅のサイズピクセル : 480
- 高さのサイズピクセル : 272
- リセットピン : IOPORT_PORT_07_PIN_11

SK-S7G2 の場合 :

- タッチチップ : g_sf_touch_panel_i2c_chip_sx8654
- 幅のサイズピクセル : 240

- 高さのサイズピクセル : 320
- リセットピン : IOPORT_PORT_06_PIN_09

PE-HMI1-S7G2 の場合 :

- タッチチップ : g_sf_touch_panel_i2c_chip_ft5x06
- 幅のサイズピクセル : 800
- 高さのサイズピクセル : 480
- リセットピン : IOPORT_PORT_10_PIN_02

ボードの共通設定 :

- スレッドのプライオリティ : Any

SX8654 および FT5x06 のタッチチップドライバーは SSP のビルトインドライバーで、モジュールを構成するときに選択できます。

5.1.24.5 アプリケーションでのタッチパネルフレームワークモジュールの使用

モジュールを構成してファイルの生成が済むと、タッチパネルフレームワークはアプリケーションで使用できる状態になります。タッチパネルフレームワークは、アプリケーションが開始されると、Synergy で生成されたファイルから自動的に開かれます。

タッチパネルフレームワークモジュールの使用例 :

1) タッチパネルフレームワークの start API を使用してタッチパネルフレームワークを開始します。
2) pend API を使用してタッチメッセージを保留にします
3) アプリケーションコードでタッチメッセージを解析します
4) 受信したデータをアプリケーションの必要に応じて操作します。手順 2 に進みます。

次の図では、これらの一般的な手順を通常の動作フロー図に示します。

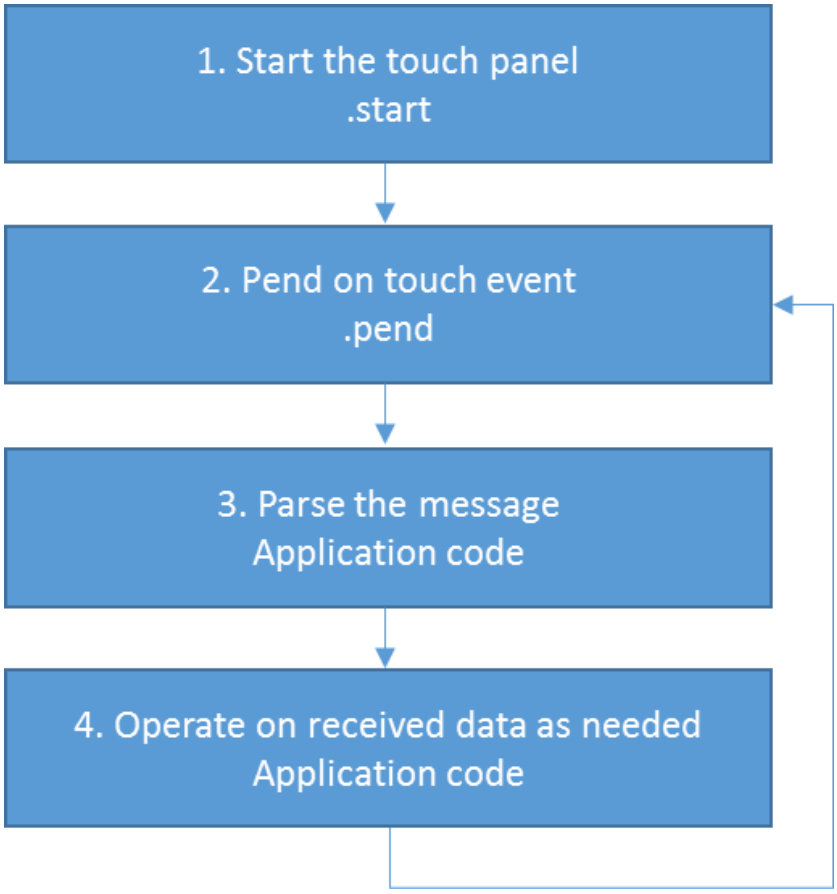


図 249: 通常のタッチパネルフレームワークモジュールアプリケーションのフロー図

5.1.25 WiFi フレームワークモジュール

WiFi フレームワークモジュールには、以下の主要な特長があります。

- WiFi モジュールの構成とプロビジョニングを行うために使用できる汎用 API インタフェースをアプリケーションに提供する
- WiFi モジュールデバイスドライバーの抽象化を提供する
- WiFi モジュールに対して NetX または NetX-Duo ドライバーを実装する
- WiFi モジュールで実行されているアクセスポイントにどのデバイスが接続できるかをアプリケーションで制御できるようにするアクセス制御リスト管理を提供する *
- アプリケーションがマルチキャストグループへの参加や離脱を行えるようにするマルチキャストフィルタリスト管理を提供する *
- 802.11 規格ファミリによって指定されたさまざまな構成パラメータを提供する *
- オンチップネットワークスタックを使用する BSD ソケット API インタフェースを提供する

*注: * このオンチップネットワークスタックは、WiFi モジュールまたは WiFi モジュールドライバーでサポートされている場合にのみサポートされます。*

次の図に、 WIFI レームワークモジュールの編成、オプション、スタックの実装を示します。

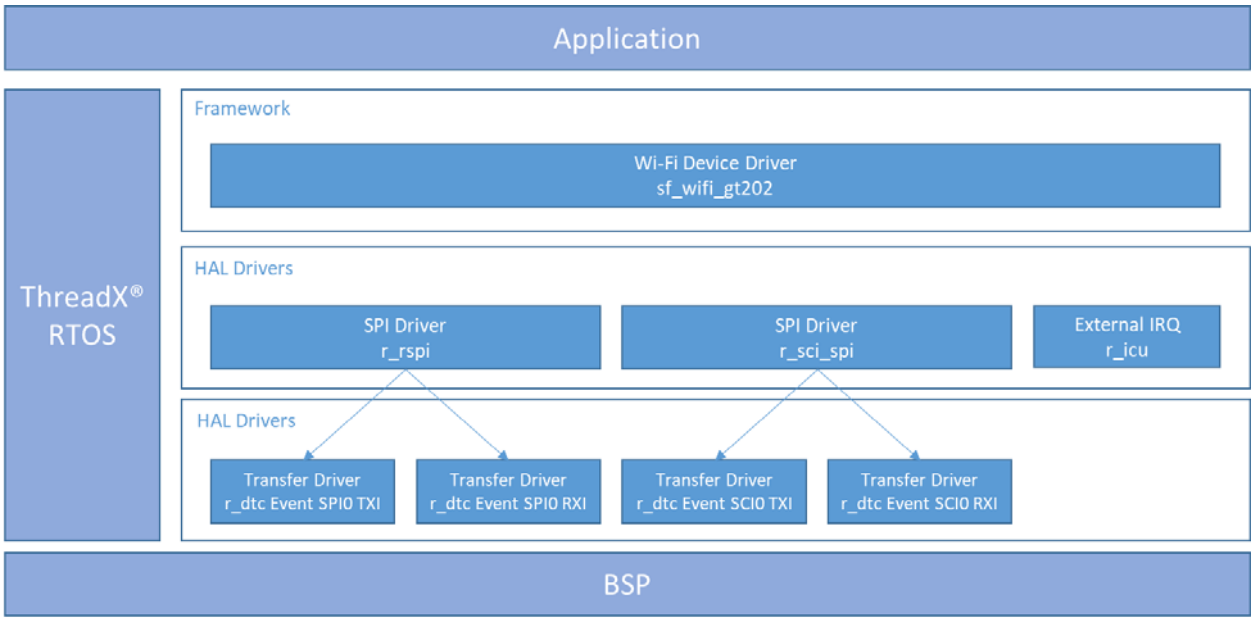


図 250: WiFi デバイスドライバーの実装

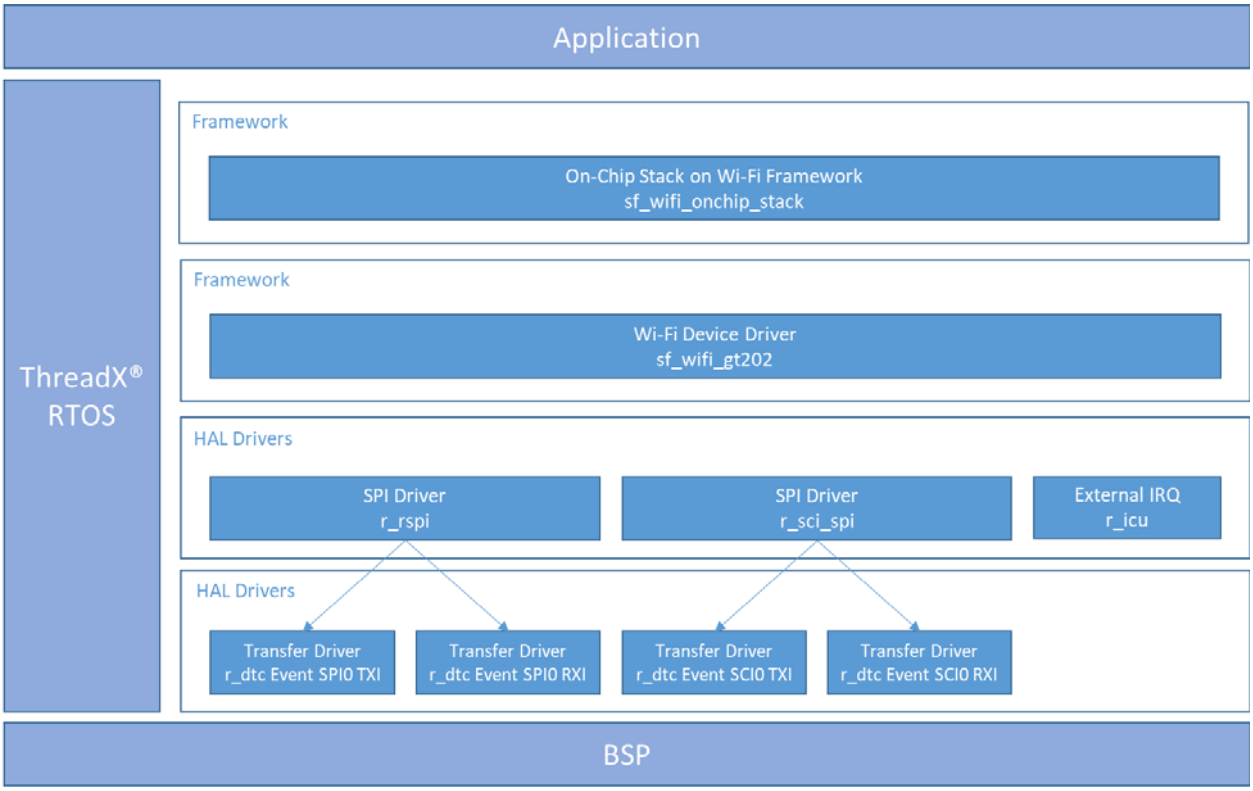


図 251: WiFi オンチップスタックの実装

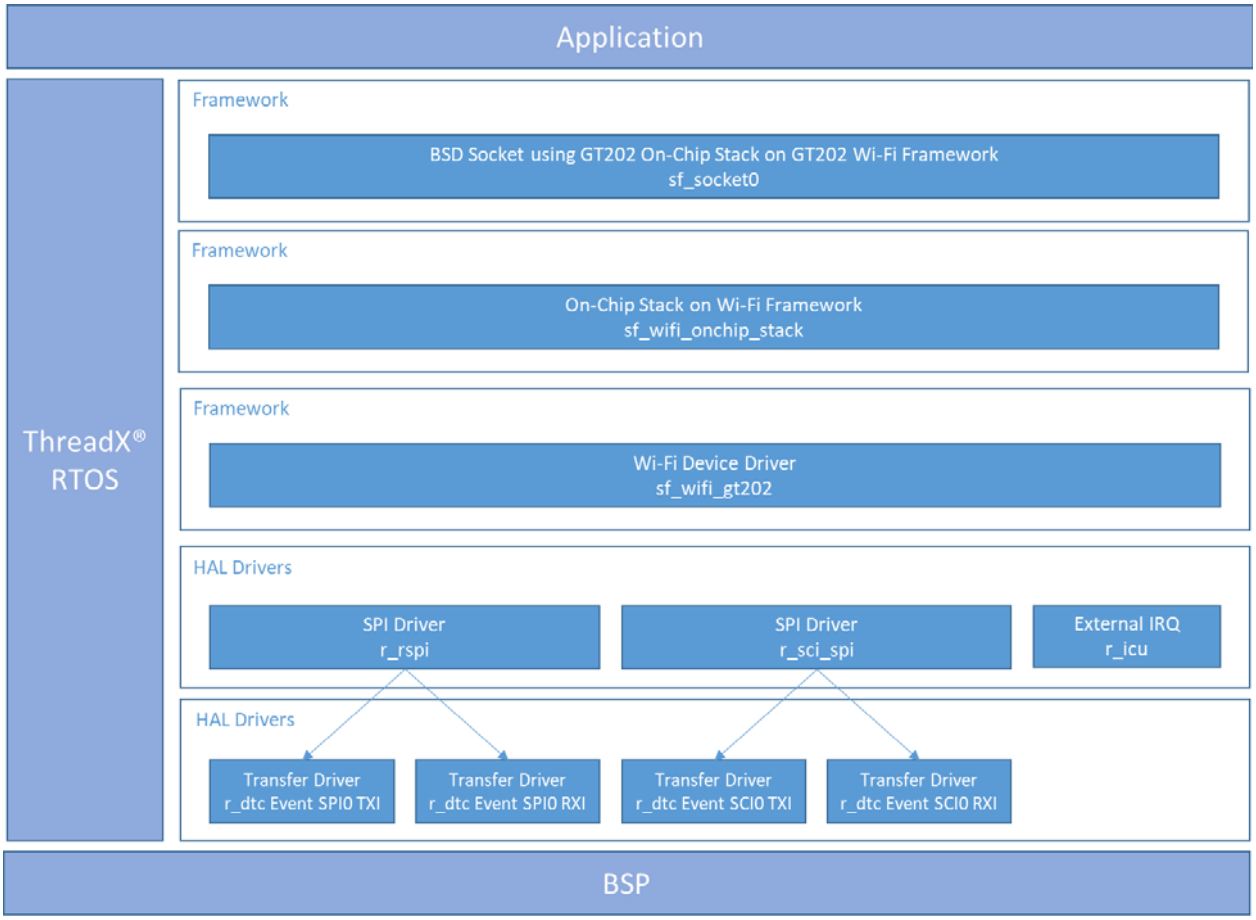


図 252: WiFi BSD ソケットの実装

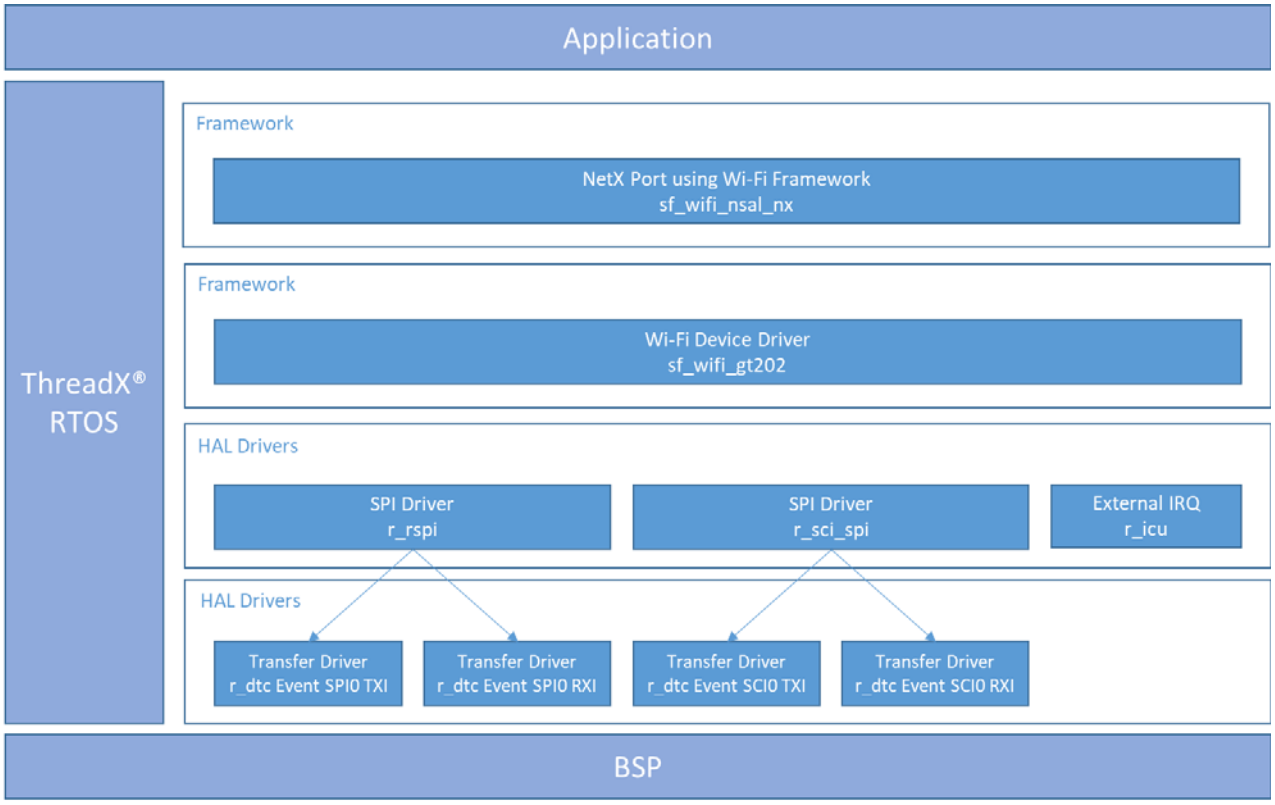


図 253: NSAL での WiFi NetX ポートの実装

5.1.25.1 WIFI フレームワークモジュール API の概要

WiFi フレームワークは、各関連モジュールの API を定義します。このセクションでは各 API の動作について説明します。一般的な戻りコードの表は API 要約表の後にあります。詳細については、『*SSP ユーザーズマニュアル*』で対応するモジュールの API リファレンスセクションを参照してください。

5.1.25.2 WiFi フレームワークモジュールの API

WiFi フレームワークモジュールによって、WiFi モジュールに関係なくネットワークスタックとアプリケーションの汎用インタフェースが提供されます。次の表に、フレームワークで公開されている API の一覧を示します。

WiFi フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_wifi0.p_api->open (g_sf_wifi0.p_ctrl, g_sf_wifi0.p_cfg); この API は、 WIFI モジュールを初期化して有効にします。open 関数は、WiFi フレームワークのインスタンスを一意に特定する WiFi 制御構造体を返します。
.close	g_sf_wifi0.p_api->close(g_sf_wifi0.p_ctrl); この API は、 WiFi モジュールの初期化を解除してオフにします。
.infoGet	g_sf_wifi0.p_api->infoGet (g_sf_wifi0.p_ctrl, wifi_info); この API は、WiFi 制御構造体を引数として受け取り、WiFi モジュールから取得した以下の情報を返します。 • チップセットまたはドライバーの情報文字列 • RSSI 値 （無符号整数 16 ビット） • ノイズレベル （無符号整数 16 ビット） • リンク品質 （無符号整数 16 ビット）
.statisticsGet	g_sf_wifi0.p_api->statisticsGet (g_sf_wifi0.p_ctrl, p_stats); この API は WiFi モジュールからデータ統計を取得します。これは WiFi 制御構造体を引数として受け取り、以下の統計を返します。 • 受信パケット数 （無符号整数 32 ビット） • 送信パケット数 （無符号整数 32 ビット） • 送信パケットエラー数（無符号整数 32 ビット）
.transmit	g_sf_wifi0.p_api->transmit (g_sf_wifi0.p_ctrl, p_buffer, length); この API はデータすなわちパケットを送信します。この関数は WiFi 制御構造体を引数として受け取ります。ネットワークパケットバッファとネットワークパケットバッファ長を引数として受け取ります。 WiFi フレームワークの transmit 関数は、パケットバッファを WiFi ドライバーに渡して送信します。

Function Name	API の呼び出し例と説明
.receiveCallback	これはコールバック API であり、 WiFi モジュールでデータやパケットが受信されたときに呼び出されます。この関数は、パケットバッファとパケット長を引数として受け取ります。
.provisioningGet	g_sf_wifi0.p_api->provisioningGet (g_sf_wifi0.p_ctrl, &sf_wifi_provisioninfo); この API は、 WiFi 制御構造体を引数として受け取り、以下のパラメータを返します。 • モード（列挙、つまり AP またはクライアント） • チャネル （無符号整数 8 ビット） • SSID （文字列） • セキュリティタイプ （列挙） • 暗号化タイプ （列挙） • セキュリティキー （文字列）
.provisioningSet	g_sf_wifi0.p_api->provisioningSet (g_sf_wifi0.p_ctrl, &g_sf_wifi_provisioninfo); この API は、特定のモード （AP またはステーション）で WiFi モジュールを設定します。 provisioningSet 関数は、次のパラメータを使用して WiFi モジュールをプロビジョニングします。 • モード（列挙、つまり AP またはクライアント） • AP モードでのみ使用されるチャネル（無符号整数 8 ビット） • SSID （文字列） • セキュリティタイプ （列挙） • 暗号化タイプ （列挙） • セキュリティキー （文字列） • AP リンクステータス通知コールバック関数 : ステーションモードでプロビジョニングされたときにAPリンクステータス変更通知を取得するために使用されます。

Function Name	API の呼び出し例と説明
.scan	g_sf_wifi0.p_api->scan (g_sf_wifi0.p_ctrl, p_scan, p_count); この API は、範囲内の使用可能な SSID（アクセスポイント）をスキャンします。この関数は WiFi 制御構造体を引数として受け取り、WiFi モジュールによってスキャンされた SSID のリストを次のパラメータで返します。 • HW モード（列挙 a、b、g、n） • RSSI（無符号整数 16 ビット） • SSID（文字列） • BSSID（6 バイトのサイズのバイト配列） • チャネル（無符号整数 8 ビット） • セキュリティタイプ（列挙） • 暗号化 type(enumeration) • BSS タイプ（列挙） WiFi フレームワークの scan 関数は SSID 数を引数として受け取り、この引数は入力パラメータおよび出力パラメータとして作動します。これはスキャン結果配列のサイズを指定し、WiFi フレームワークが、配列に格納されるスキャン結果の数を示す値に設定します。
.ACLAdd	g_sf_wifi0.p_api->ACLAdd (g_sf_wifi0.p_ctrl, peer_device_mac); この API は、特定の MAC アドレスをアクセス制御リストに追加します。この関数は WiFi 制御構造体と MAC アドレスを引数として受け取ります。
.ACLDelete	g_sf_wifi0.p_api->ACLDelete (g_sf_wifi0.p_ctrl, peer_device_mac); この API は、特定の MAC アドレスをアクセス制御リストから削除します。この関数は WiFi 制御構造体と MAC アドレスを引数として受け取ります。

Function Name	API の呼び出し例と説明
.multicastListAdd	g_sf_wifi0.p_api->multicastListAdd (g_sf_wifi0.p_ctrl, p_mac_addr); この API は、特定のマルチキャスト IP アドレスをマルチキャストファイラリストに追加します。この関数は WiFi 制御構造体と MAC アドレスを引数として受け取ります。
.multicastListDelete	g_sf_wifi0.p_api->multicastListDelete (g_sf_wifi0.p_ctrl, p_mac_addr); この API は、特定のマルチキャスト IP アドレスをマルチキャストファイラリストから削除します。この関数は WiFi 制御構造体と MAC アドレスを引数として受け取ります。
.macAddressGet	g_sf_wifi0.p_api->macAddressGet (g_sf_wifi0.p_ctrl, p_mac); この API は、WiFi モジュールから MAC アドレスを読み取ります。この関数は、WiFi 制御構造体を引数として受け取り、WiFi モジュールから読み取った MAC アドレスを返します。
.macAddressSet	g_sf_wifi0.p_api->macAddressSet (g_sf_wifi0.p_ctrl, p_mac); この API は、 WiFi モジュールの MAC アドレスを設定します。この関数は WiFi 制御構造体と MAC アドレスを引数として受け取ります。

オンチップネットワークスタックサポート API

これらの API を使用して、オンチップネットワークスタックを使用するときに WiFi モジュールを構成できます。これは、インタフェースの IP アドレスの構成と、 DHCP サーバーの起動と停止（AP モードで構成されている場合）に役立ちます。

オンチップネットワークスタックサポート WiFi フレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
.open	g_sf_wifi_onchip_stack0.p_api->open (g_sf_wifionchip_stack0.p_ctrl, g_sf_wifi_onchip_stack0.p_cfg); この API は、WiFi モジュールを初期化する WiFi フレームワークの open API を呼び出します。
.close	g_sf_wifi_onchip_stack0.p_api->close(g_sf_wifi_onchip_stack0.p_ctrl); この API は、WiFi モジュールを初期化解除する WiFi フレームワークの close API を呼び出します。
.ipAddressCfg	g_sf_wifi_onchip_stack0.p_api->ipAddressCfg (g_sf_wifi_onchip_stack0.p_ctrl, p_cfg); この API は、オンチップネットワークスタックを使用してインタフェースの IP アドレスを構成します。DHCP を使用して静的 IP アドレスを構成する機能を提供します。
.dhcpServerStart	g_sf_wifi_onchip_stack0.p_api->dhcpServer (g_sf_wifi_onchip_stack0.p_ctrl, p_start_ip, p_end_ip); この API は、オンチップネットワークスタックを使用して（AP モードで構成されている場合に）インタフェースの DHCP サーバーを起動します。DHCP サーバーで使用する IP アドレスの範囲を受け取ります。
.dhcpServerStop	g_sf_wifi_onchip_stack0.p_api->dhcpServerStop (g_sf_wifi_onchip_stack0.p_ctrl); この API は DHCP サーバーを停止します。
.versionGet	g_sf_wifi_onchip_stack0.p_api->versionGet (&version); バージョンポインタを使用して API バージョンを取得します。

BSD ソケット API

これらの API は、 GT202 オンチップスタック実装を使用した BSD ソケットサポートに使用できます。

GT202 オンチップスタック WIFI フレームワークモジュール API を使用した BSD ソケットの要約

Function Name	API の呼び出し例と説明
.open	g_sf_socket0.p_api->open (g_sf_socket0.p_ctrl, g_sf_socket0.p_cfg); この API はネットワークインタフェースを初期化します。
.close	g_sf_socket0.p_api->close(g_sf_socket0.p_ctrl); この API はネットワークインタフェースを閉じます。
.versionGet	g_sf_socket0.p_api->versionGet (&version); バージョンポインタを使用して API バージョンを取得します。

また、この実装には BSD API に準拠するソケット API が含まれます。これらの API は、アプリケーションがソケットを使用したデータ転送を実行するために使用できます。次の API を使用できます。

- socket
- close
- bind
- listen
- accept
- connect
- send
- recv
- recvfrom
- sendto
- setsockopt
- getsockopt
- select

これらの API の詳細については、SSP ページの Synergy ギャラリーから入手できる『*NetX ユーザーガイド*』で「NetX BSD 4.3 Sockets API Compliancy Wrapper」を参照してください（[documentation] タブで Renesas Synergy の X-Ware™ コンポーネントドキュメントの zip ファイルを選択します）。

WiFi NSAL

Synergy WiFi フレームワークは、NetX または NetX-Duo ネットワークサービス抽象化レイヤーをサポートします。これらには、NetX または NetX-Duo ドライバー、パケット送信および受信コールバック関数の実装が含まれます。

NetX または NetX-Duo ドライバー関数

NetX または NetX-Duo ドライバー関数は NetX IP インスタンス、WiFi フレームワークインスタンス、NSAL 構成を引数として受け取ります。 NSAL 構成は、送信および受信コールバック関数の動作を制御します。NSAL 構成には、ゼロコピーサポートが送信および受信パスで有効か無効かを示すフラグが含まれます。NetX または NetX-Duo ドライバー関数は、NetX または NetX-Duo で使用される各種 IP ドライバーコマンドを実装します。interface attach コマンドは、WiFi フレームワークの open API を呼び出して WiFi モジュールを初期化します。initialize コマンドは、WiFi フレームワークの macAddressGet API を呼び出して、WiFi モジュールから MAC アドレスを読み取ります。un-initialize コマンドは、WiFi フレームワークの close API を呼び出して WiFi モジュールを初期化解除します。 multicast-join コマンドは、 WiFi フレームワークの multicastListAdd API を呼び出して、特定の MAC アドレスをマルチキャストリストに追加します。multicast-leave コマンドは、WiFi フレームワークの multicastListDelete API を呼び出して、特定の MAC アドレスをマルチキャストリストから削除します。Send および Broadcast コマンドは、WiFi フレームワークの transmit API を呼び出してパケットを送信します。

NSAL 送信関数

NSAL transmit 関数は NetX IP インスタンス、NetX パケット、WiFi フレームワークインスタンス、NSAL 構成を引数として受け取ります。ゼロコピーサポートが有効になっている場合、 NetX パケットは NetX から WiFi ドライバーに転送されます。ゼロコピーがサポートされていない場合は、 NetX パケットからドライバーバッファにデータをコピーします。バッファやパケットを WiFi ドライバーに渡してさらに送信する WiFi フレームワークの transmit API を呼び出します。

NSAL 受信コールバック

NSAL 受信コールバック関数は、NetX IP インスタンス、パケットバッファ、パケットバッファ長、NSAL 構成を引数として受け取ります。このコールバックは WiFi デバイスドライバーから呼び出されます。ゼロコピーサポートが有効になっている場合、NetX パケットは WiFi ドライバーから NetX に転送されます。ゼロコピーがサポートされていない場合は、ドライバーバッファから NetX パケットにデータをコピーし、さらに処理するために NetX スタックに渡します。バッファを WiFi ドライバーに渡してさらに送信する WiFi フレームワークの transmit API を呼び出します。

これらの API の詳細については、 SSP ページの Synergy ギャラリーから入手できる『*NetX ユーザーガイド*』を参照してください（[documentation] タブで Renesas Synergy の X-Ware™ コンポーネントドキュメントの zip ファイルを選択します）。

WiFi フレームワークのエラーコード

次の表に、 WiFi フレームワーク固有のエラーコードを示します。これらのエラーコードは ssp_err_t. に含まれます。

WiFi フレームワークのエラーコード

Error Codes	Value
SSP_ERR_WIFI_CONFIG_FAILED	70000

Error Codes	Value
SSP_ERR_WIFI_INIT_FAILED	70001
SSP_ERR_WIFI_TRANSMIT_FAILED	70002
SSP_ERR_WIFI_INVALID_MODE	70003
SSP_ERR_WIFI_FAILED	70004

5.1.25.3 WiFi フレームワークモジュールの動作の概要

WiFi フレームワークモジュールの動作について

WiFi フレームワークは、WiFi モジュールの構成とプロビジョニングを行ってデータ転送を実行するアプリケーションのために汎用インタフェースを提供します。これにより、エンドユーザーは WiFi モジュールドライバーの詳細を知らなくてもアプリケーションコードを開発できます。また、同じアプリケーションコードを異なる WiFi モジュールに使用できます。

Synergy WiFi フレームワークは以下の論理ブロックで構成されています。

- SF WiFi API
- ネットワークスタック抽象化レイヤー
- SSP HAL インタフェース
- WiFi デバイスドライバー（ベンダが提供するドライバー）。

これには、オンチップネットワークスタックを利用する BSD ソケット API をサポートするための次のブロックが含まれます。

- オンチップスタック API
- ソケット API。

次の図に、 Synergy WiFi フレームワーク階層化アーキテクチャの概要を示します。

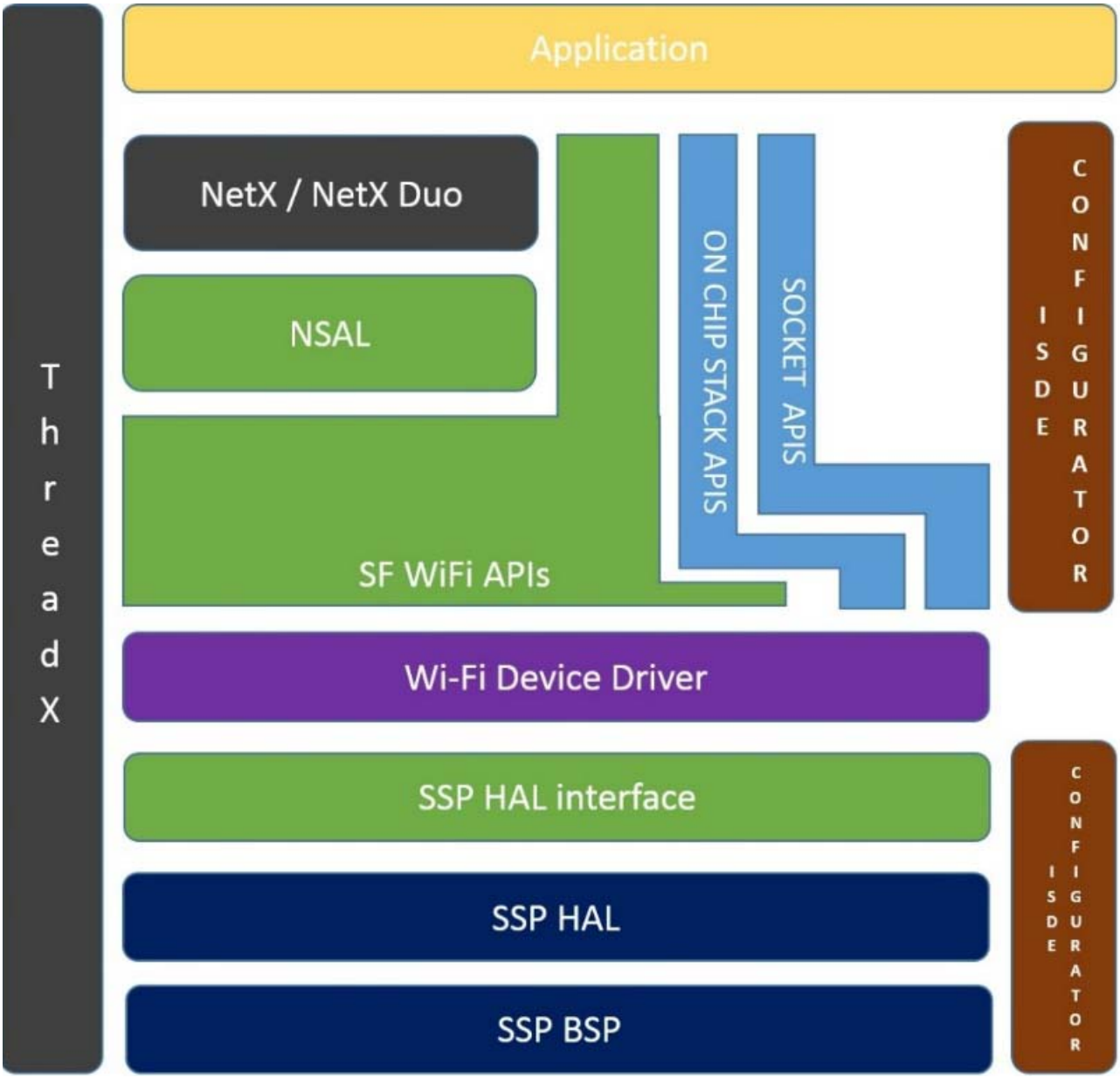


図 254: WIFI 階層化アーキテクチャ

Synergy WiFi フレームワーク API

WiFi フレームワークは、WiFi モジュールの構成とプロビジョニングを行い、データ転送を実行するための汎用 API を提供します。WiFi モジュールには、802.11 規格ファミリによって指定されたさまざまなパラメータがあります。個別のデバイスドライバーまたは WiFi チップセット（あるいは両方）ですべての関数の構成がサポートされるとは限りません。

WiFi フレームワークは、WiFi インタフェースをアクセスポイント（AP）またはクライアントとして構成するためのプロビジョニングインタフェースも提供します。インタフェースがアクティブになるためには、少なくともチャネル、サービスセット識別子（SSID）、セキュリティスキーム、セキュリティ資格情報を構成する必要があります。

ネットワークスタック抽象化レイヤー

WiFi フレームワークは、ネットワークスタック抽象化レイヤー（NSAL）を提供します。NSAL は、WiFi フレームワーク API を利用する MAC レイヤーを実装します。これにより、ネットワークスタック（NetX または NetX-Duo）の抽象化が作成されるため、ネットワークデバイスドライバー（MAC レイヤー実装）を再利用できるようになります。現在の NSAL 実装には、NetX（IPv4）と NetX-Duo（IPv6）のサポートが含まれます。新しいネットワークスタックのサポートを追加するには、適切な NSAL の実装が必要です。

SSP HAL Interface

HAL インタフェースは、Synergy MCU との低レベル通信のために WiFi モジュールで使用される SSP HAL コンポーネントのインタフェースを実装します。この実装は WiFi モジュールに固有です。WiFi モジュールは、SPI、ICU、IOPORT、SDMMC などのさまざまな HAL コンポーネントを利用します。

ソケット API

ソケット API は、BSD ソケット API を利用するためのインタフェースをアプリケーションに提供します。これには、WiFi モジュールまたはドライバーが、オンチップネットワークスタックと BSD ソケット API のサポートを提供する必要があります。アプリケーションは、これらの API を使用するときに、WiFi チップセットに存在するオンチップネットワークスタックを使用し、NSAL（または Synergy MCU で実行しているネットワークスタック）は使用しません。

オンチップスタック API

オンチップスタック API は、モジュールの IP アドレスの構成と、DHCP サーバーの起動と停止（AP モードで構成されている場合）を行うためのインタフェースをアプリケーションに提供します。これらの API は、WiFi チップセットで実行されているネットワークスタックを利用します。ソケット API と同様に、これらの API と NSAL の使用は相互に排他的です。

WiFi フレームワークモジュールの動作の説明

次の図に、Synergy WiFi フレームワークで可能な WiFi の使用の 2 つのパスを示します。

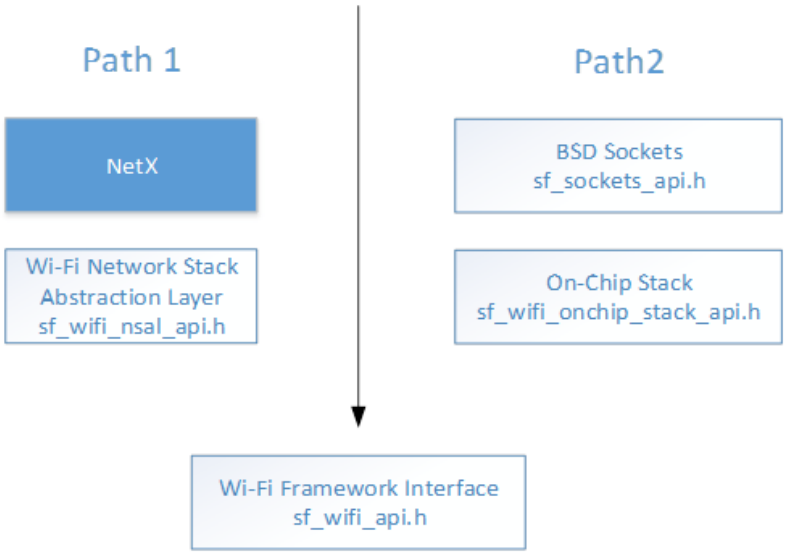


図 255: SF WiFi の使用のパス

最初のオプションは、 NetX での Synergy WiFi フレームワークの使用です。 Synergy WiFi フレームワークには、NetX または NetX-Duo デバイスドライバーインタフェースを実装するネットワークスタック抽象化レイヤー（NSAL）が含まれます。これは、NetX がイーサネットで使用されるアプリケーションでは、ユーザーが Synergy WiFi フレームワークに対してイーサネットドライバーを sf_el_nx, でスワップアウトできることを意味します。次の図に、 [SSP Configuration] ページでの NetX とイーサネットを示します。

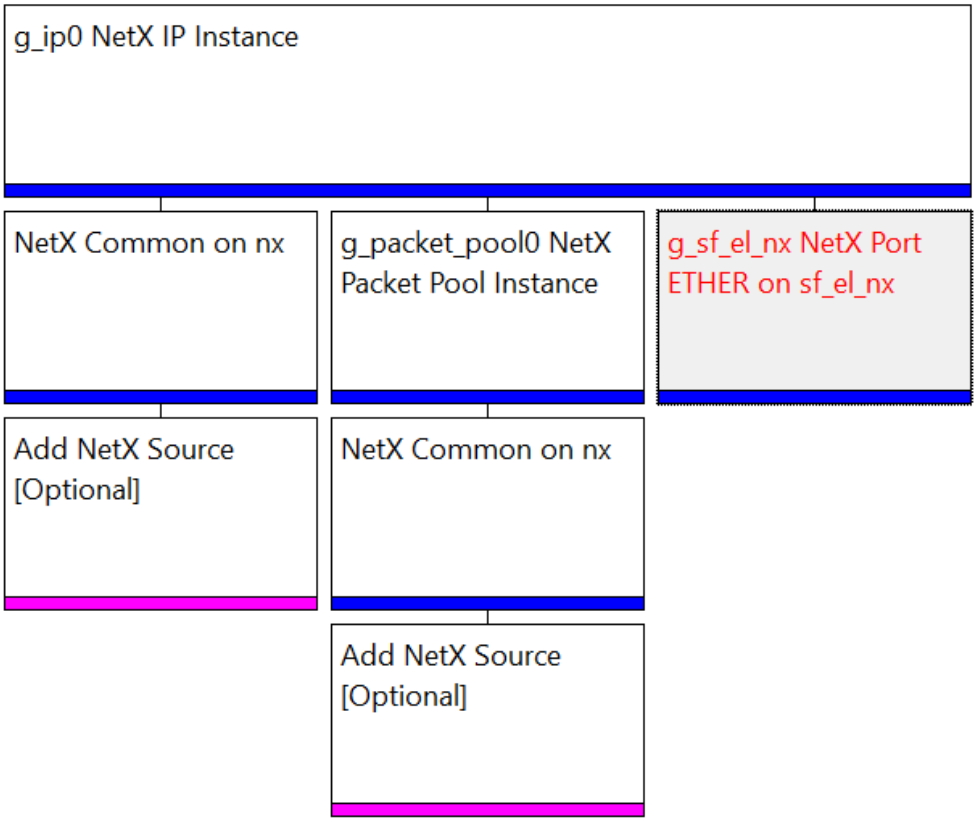


図 256: NetX IP インスタンス

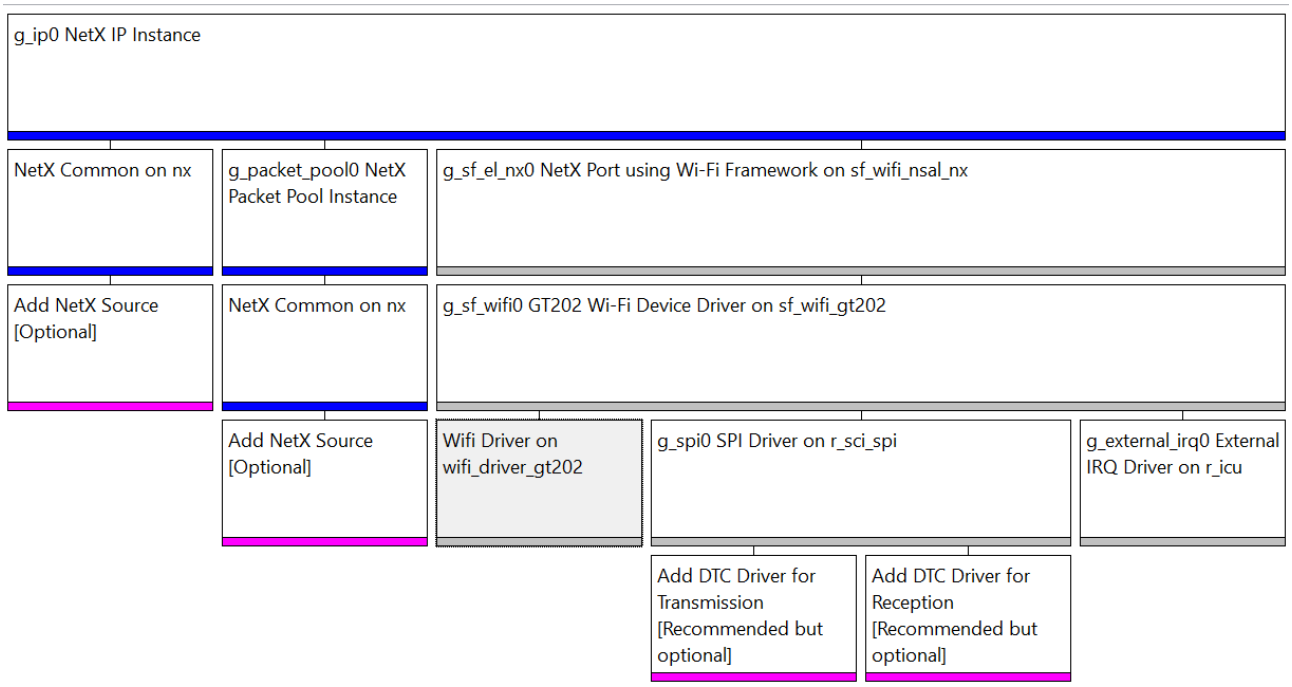


図 257: sf_wifi_nsal_nx 上で WiFi フレームワークを使用する NetX

NetX デバイスドライバーがスワップアウトされた後、既存の NetX アプリケーションは以前と同様に動作する必要があります。ユーザーは、アプリケーションからの呼び出しを追加して、WiFi プロビジョニングを設定する必要があります。

他のオプションでは BSD ソケット API を使用します。このパスでは、特定の WiFi チップセットのオンチップスタック機能も使用します。すべての WiFi チップセットが追加のオンチップサポートを提供するわけではありません。 BSD ソケット API の詳細については、次の場所を参照してください。
http://pubs.opengroup.org/onlinepubs/7908799/xns/syssocket.h.html.

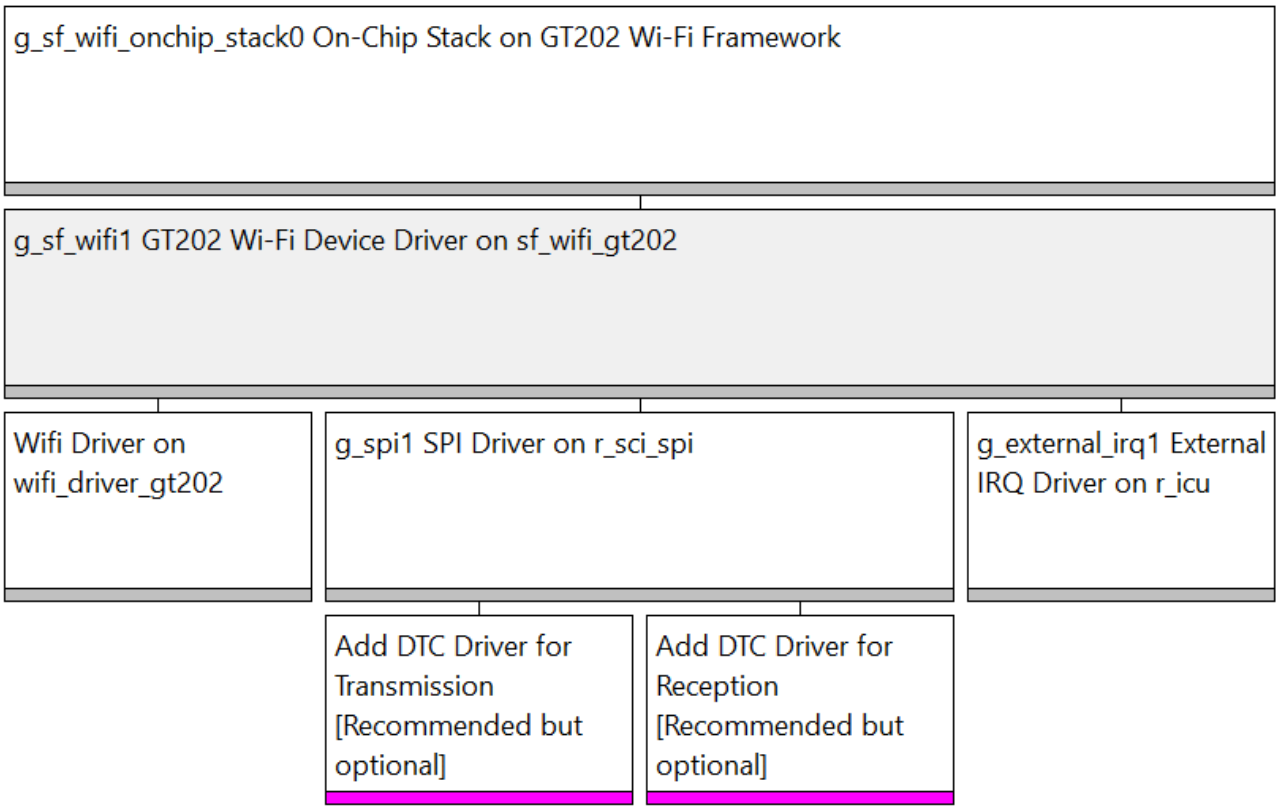


図 258: GT202 モジュールを使用したオンチップスタックのスレッド構成

WiFi フレームワークモジュールの動作に関する注意事項

アプリケーションは、WiFi フレームワークインスタンスを使用する前に定義する必要があります。WiFi フレームワークインスタンスは、WiFi モジュール固有の制御データ、構成データ、API を参照します。アプリケーションは、このインスタンスを使用して WiFi モジュール上で操作を実行します。インスタンスは、制御構造体、構成構造体、フレームワーク API 構造体へのポインタを含む構造体です。

Synergy WiFi フレームワークインスタンスのメンバは次のとおりです。

制御構造体

この構造体はすべての WiFi フレームワーク API で使用されます。ドライバーハンドルへのポインタが含まれます。ドライバーハンドルは、WiFi モジュールのデバイスドライバーでの動作中にフレームワークが使用します。

構成構造体

この構造体は open 呼び出しに引数として渡され、以下に示す WiFi モジュールの各種パラメータの構成に使用されます。

MAC アドレス

- ハードウェアモード （802.11、 b、 g、 n など）

- 送信電力
- RTS/CTS ハンドシェイク有効 / 無効フラグ
- フラグメンテーションしきい値
- 配信トラフィック指示マップ周期 （DTIM など）
- 高スループットモード有効 / 無効フラグ
- プリアンブル長 （short または long）
- WiFi マルチメディアモード有効 / 無効フラグ
- バッファプール Rx （パケットの受信に使用される NetX パケットプール） [applicable only when driver supports zero copy]
- 許可される最大ステーション数（関連付けることができるステーションの最大数）[applicable only in AP Mode]
- SSID ブロードキャスト有効 / 無効フラグ（SSID を非表示にするかどうか）[applicable only in AP Mode]
- アクセス制御リストモード（AP に関連付けることのできるステーションの制御リスト）[applicable only in AP Mode]
- ビーコンインターバル [applicable only in AP Mode]
- ステーション非アクティブタイムアウト値 （ステーションがこの期間にわたって非アクティブな状態が続くと関連付けが解除されます） [applicable only in AP Mode]
- WDS （ワイヤレス配信システム）有効 / 無効フラグ [applicable only in AP Mode]
- 必須高スループット有効 / 無効フラグ [Only allow HT mode. AP mode only]

フレームワーク API 構造体

この構造体には、 WiFi モジュール固有のフレームワーク API へのポインタが含まれます。

オンチップスタック API 構造体

この構造体には、所定のモジュール固有のオンチップスタック API へのポインタが含まれます。

BSD ソケット API 構造体

この構造体には、 BSD ソケットインタフェース API へのポインタが含まれます。

Synergy WiFi フレームワークの制限事項と注意事項

1) メモリに制約があるため、WiFi フレームワークは S124（CM0+ コアなど）をサポートしていません。
2) WiFi モジュールドライバーを RSPI に使用するときは、 WiFi モジュールの SPI ドライバーでの制限のため、依存関係に自動的に設定される DTC コンポーネントを削除する必要があります。制限とは、DTC を RSPI で使用するときは 32 ビット転送が必要ですが、GT202 ベンダーコード（つまり、WiFi モジュールの SPI ドライバー）は 8 ビット転送のみを使用することです。

その他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.25.4 アプリケーションへの WiFi フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに WiFi フレームワークモジュールを組み込む方法について説明します。

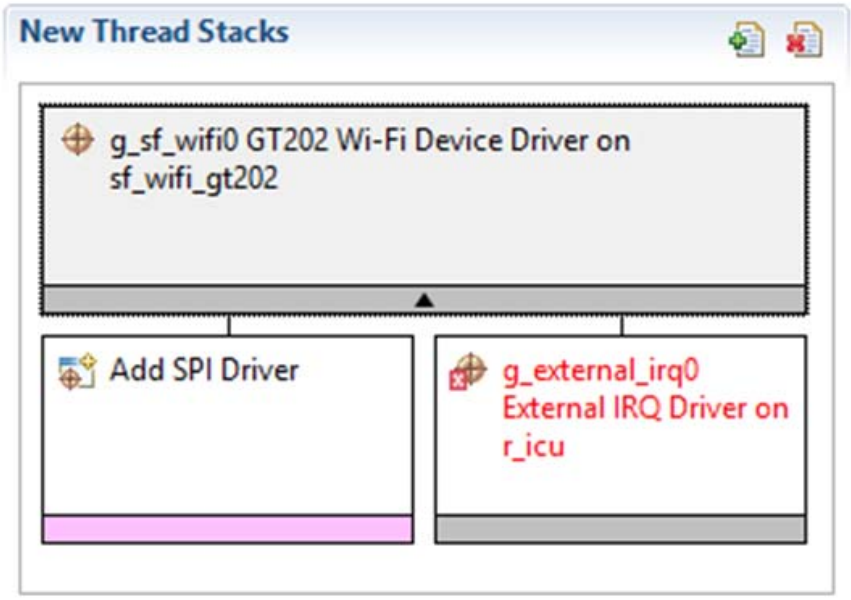
***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

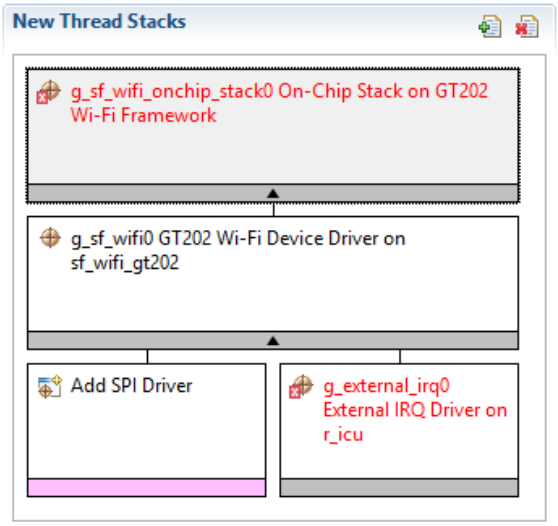
WiFi フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。4 つの異なるオプションがあるので、4 つのシーケンスを示してあります。

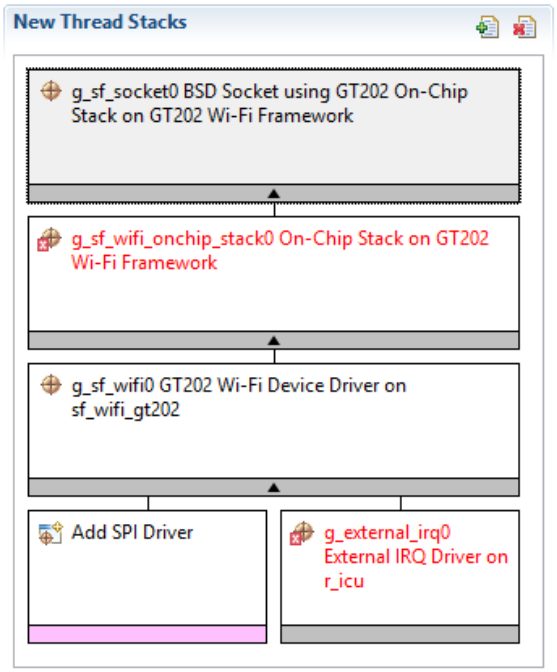
WiFi フレームワークの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_wifi0 GT202 Wi-Fi Device Driver on sf_wifi_gt202	Threads	New Stack> Framework> Networking> WiFi> GT202 Wi-Fi Device Driver on sf_wifi_gt202
g_sf_wifi_onchip_stack0 On-Chip Stack on Gt202 Wi-Fi Framework	Threads	New Stack> Framework> Networking> WiFi> On-Chip Stack on Gt202 Wi-Fi Framework
g_sf_socket0 BSD Socket using On-Chip Stack on Gt202 Wi-Fi Framework	Threads	New Stack> Framework> Networking> WiFi> BSD Socket using On-Chip Stack on Gt202 Wi-Fi Framework
g_sf_el_nx0 NetX Port using Wi-Fi Framework on sf_wifi_nsal_nx	Threads	New Stack> Framework> Networking> WiFi> NetX Port using Wi-Fi Framework on sf_wifi_nsal_nx

次の図に示すように WiFi フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。ピンクの帯で示されるモジュールでは実装オプションのの選択が必要です。


New Thread Stacks
g_sf_wifi0 GT202 Wi-Fi Device Driver on sf_wifi_gt202
Add SPI Driver
g_external_irq0 External IRQ Driver on r_icu


New Thread Stacks
g_sf_wifi_onchip_stack0 On-Chip Stack on GT202 Wi-Fi Framework
g_sf_wifi0 GT202 Wi-Fi Device Driver on sf_wifi_gt202
Add SPI Driver
g_external_irq0 External IRQ Driver on r_icu



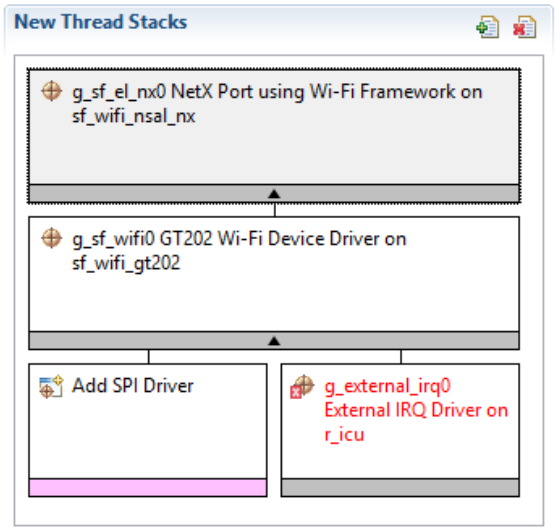


図 259: 4 つの WiFi フレームワークモジュールスタックオプション

SF_WIFI_NSAL_NX の選択

sf_wifi_nsal_nx の実装は、NetX IP のインスタンスと組み合わせて使用できます。このケースでは、これは IP インスタンススタックの下にオプションとして追加されます。次の表の選択手順を使用して、NetX IP のインスタンスを追加します。

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_wifi0 GT202 Wi-Fi Device Driver on sf_wifi_gt202	Threads	New Stack> X-Ware> NetX> NetX IP Instance

使用可能なオプションを使用して、sf_wifi_nsal_nx を NetX IP インスタンスに NetX ネットワークドライバーとして追加します。

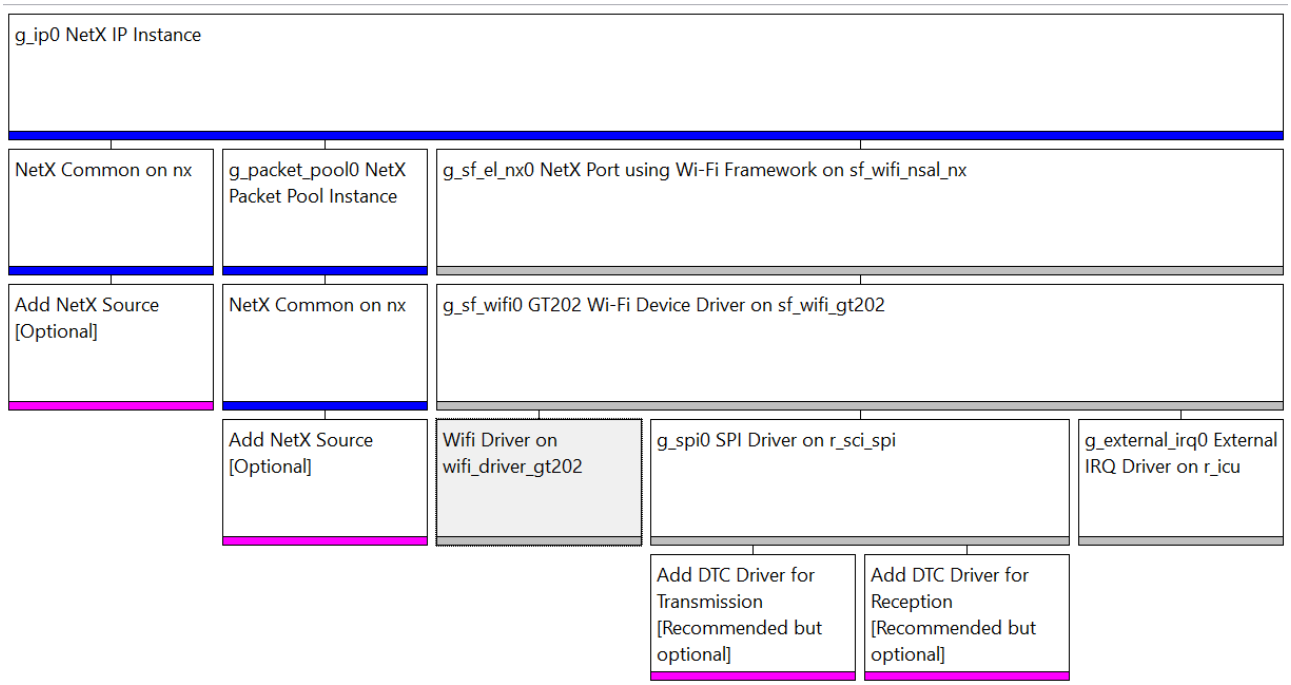


図 260: sf_wifi_nsal_nx の NetX IP インスタンスの使用

次の図に示すように、下位レベルモジュールは自動的に設定されます。

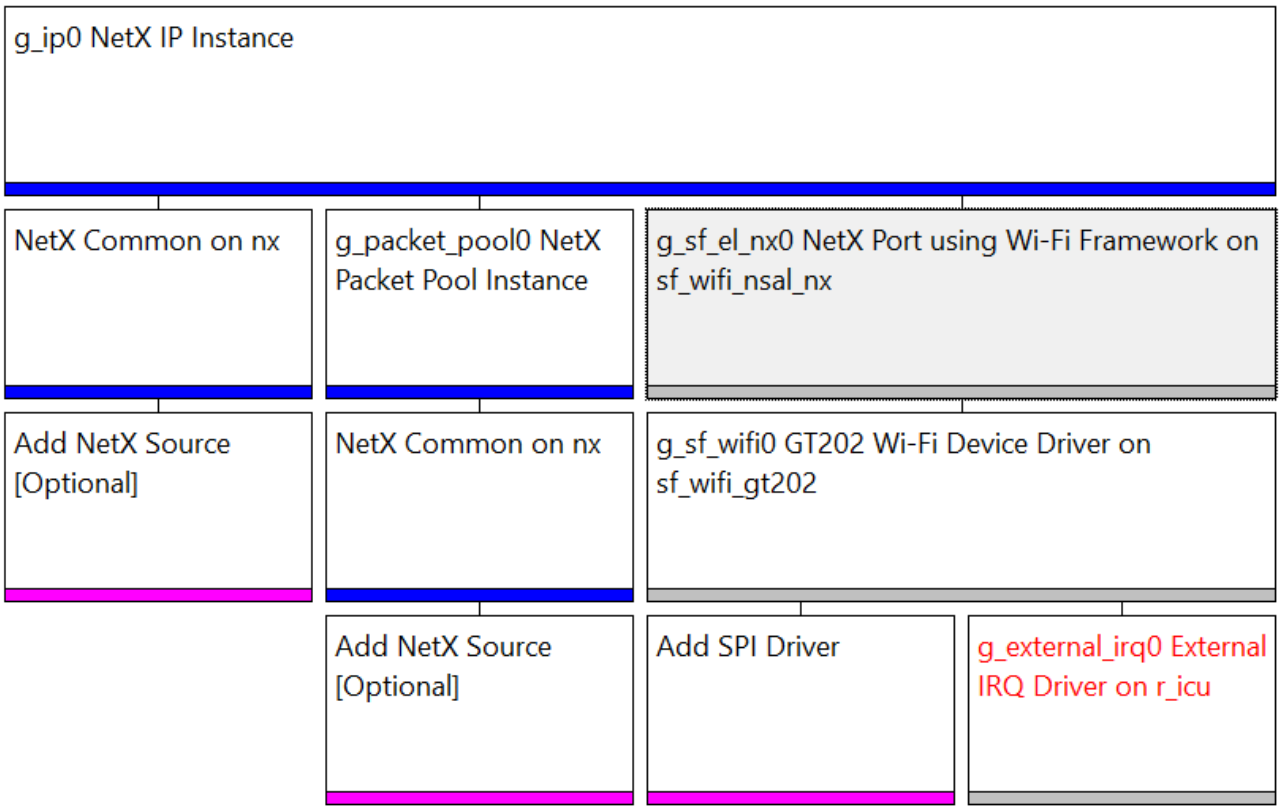


図 261: sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

5.1.25.5 WIFI フレームワークモジュールの構成

ユーザーは必要な動作に合わせて WIFI フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注：統合ソリューション開発環境を開き、ローパワーモードドライバーを作成し、次に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

以下の 4 つのセクションでは、WiFi フレームワークによってサポートされる WiFi の 4 つの各実装に対する構成プロパティを示します。

GT202 Wi-Fi デバイスドライバーの構成

このセクションでは、WiFi フレームワークの GT202 実装上のオンチップスタックの構成設定を示します。

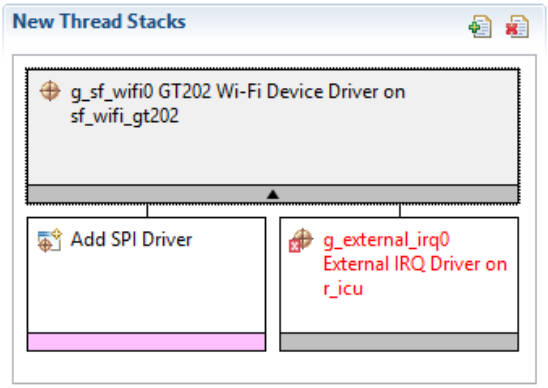


図 262: GT202 Wi-Fi デバイスドライバーのスレッドスタック

sf_wifi_gt202 上の WiFi デバイスドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled (Default: Disabled)	オンチップスタックサポートの選択
Driver Heap Size in Bytes (Minimum 8192 bytes)	8192	ドライバーのヒープサイズの選択
Name (Must be a valid C symbol)	g_sf_wifi0	モジュール名
Hardware Mode	802.11a, 802.11b, 802.11g, 802.11n (Default: 802.11n)	ハードウェアモードの選択
Transmit (TX) Power (Valid Range 1-17)	10	送信電力の選択
Ready/Clear to Send (RTS/CTS) Flag	Enabled, Disabled (Default: Enabled)	Ready または Clear の送信の選択

ISDE Property	Setting	説明
Delivery Traffic Indication Message (DTIM) Interval (Valid Range: 1-255)	3	配信トラフィック指示メッセージインターバルの選択
Broadcast SSID (AP mode only)	Enabled, Disabled (Default: Enabled)	ブロードキャスト SSID の選択
Beacon Interval in Microseconds (AP mode only and must be greater than 1023)	1024	ビーコンインターバル （マイクロ秒単位）の選択
Station inactivity timeout in seconds (AP mode only and must be greater than 0)	100	ステーション非アクティブタイムアウトの選択
Requested High Throughput	Enabled, Disabled (Default: Disabled)	高スループット要求の選択
Reset Pin (must be a valid C symbol)	IOPORT_PORT_06_PIN_00	リセットピンの選択
Slave Select Pin (SSL)(Must be a valid C symbol)	IOPORT_PORT_01_PIN_03	スレーブ選択ピンの選択
GT202 Driver Task Thread Priority (Modifying Task Thread Priority may cause Driver to malfunction)	5	GT202 ドライバータスクスレッドプライオリティの選択
Callback	NULL	コールバックの選択
Support NetX Packet Chaining	Enabled, Disabled (Default: Enabled)	NetX パケットチェーンのサポートの選択

***注：** 例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_rspi での SPI HAL ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。

ISDE Property	Setting	説明
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注:SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	500000	送信または受信レート。1秒あたりのビット。
Callback	NULL	オプションのコールバック関数ポインタ。
SPI Mode	Clock synchronous operation	動作モードを選択します（SPI またはクロック同期）。
SPI Communication Mode	Full Duplex	通信方式を選択します（全二重または送信のみ）。
Slave Select Polarity(SSL0)	Active Low	SSL0 の信号極性を選択します
Slave Select Polarity(SSL1)	Active Low	SSL1 の信号極性を選択します
Slave Select Polarity(SSL2)	Active Low	SSL2 の信号極性を選択します
Slave Select Polarity(SSL3)	Active Low	SSL3 の信号極性を選択します
Select Loopback1	Normal	loopback1 を選択します
Select Loopback2	Normal	loopback2 を選択します
Enable MOSI Idle State	Disable	MOSI アイドル固定値および選択肢を選択します
MOSI Idle State	MOSI Low	MOSI アイドル固定値および選択肢を選択します

ISDE Property	Setting	説明
Enable Parity	Disable	パリティを有効または無効にします
Parity Mode	Parity Even	パリティを選択します
Select SSL(Slave Select)	SSL0	使用するスレーブを選択します (0-SSL0、1-SSL1、2-SSL2、3-SSL3)
Select SSL Level After Transfer	SSL Level Do Not Keep	転送完了後の SSL レベルを選択します（0- ネゲート、 1- 維持）
Clock Delay Enable	Clock Delay Disable	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 RSPCK	クロック遅延カウントの選択
SSL Negation Delay Enable	Negation Delay Disable	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 RSPCK	ネゲート遅延カウントの選択
Next Access Delay Enable	Next Access Delay Disable	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 RSPCK	次アクセス遅延カウントの選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX®), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	受信割り込み優先順位の選択
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。***注：*** *SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSB ファーストと LSB ファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。***注：*** *これは SCI SPI に対してのみ適用できます。*

ISDE Property	Setting	説明
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。***注:*** *SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。

ISDE Property	Setting	説明
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。***注：****これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注：*例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定は、パラメータチェックコードの追加を有効化または無効化します。
Name	g_external_irq0	モジュール名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling	トリガイベントモードの選択
Digital Filtering	Disabled	デジタル フィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	TRUE	初期化の直後に割り込みが有効化されるかどうかを決定します。
Callback	custom_hw_irq_isr	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、IRQn がトリガーされるたびに割り込みサービスルーチン (ISR) から呼び出されます。警告 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GT202 Wi-Fi フレームワークでのオンチップスタックの構成

このセクションでは、 WiFi フレームワークの GT202 実装上のオンチップスタックの構成設定を示します。

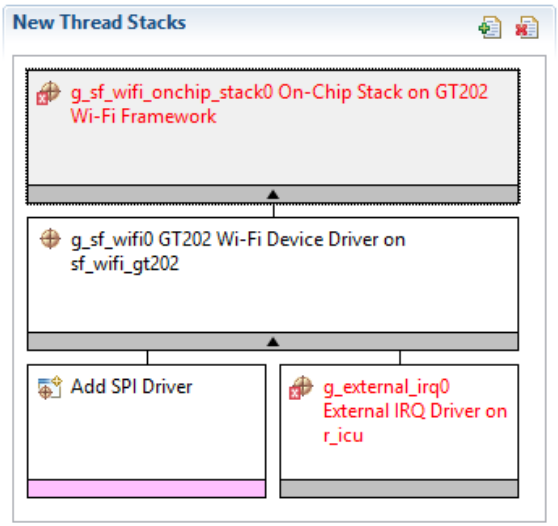


図 263: GT202 Wi-Fi フレームワークでのオンチップスタックのスレッドスタック

GT202 Wi-Fi フレームワークでのオンチップスタックの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Name (Must be a valid C Symbol)	g_sf_wifi_onchip_stack0	モジュール名

注: *例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

WiFi デバイスドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled (Default: Disabled)	オンチップスタックサポートの選択
Driver Heap Size in Bytes (Minimum 8192 bytes)	8192	ドライバーのヒープサイズの選択
Name (Must be a valid C symbol)	g_sf_wifi0	モジュール名

ISDE Property	Setting	説明
Hardware Mode	802.11a, 802.11b, 802.11g, 802.11n (Default: 802.11n)	ハードウェアモードの選択
Transmit (TX) Power (Valid Range 1-17)	10	送信電力の選択
Ready/Clear to Send (RTS/CTS) Flag	Enabled, Disabled (Default: Enabled)	Ready または Clear の送信の選択
Delivery Traffic Indication Message (DTIM) Interval (Valid Range: 1-255)	3	配信トラフィック指示メッセージインターバルの選択
Broadcast SSID (AP mode only)	Enabled, Disabled (Default: Enabled)	ブロードキャスト SSID の選択
Beacon Interval in Microseconds (AP mode only and must be greater than 1023)	1024	ビーコンインターバル（マイクロ秒単位）の選択
Station inactivity timeout in seconds (AP mode only and must be greater than 0)	100	ステーション非アクティブタイムアウトの選択
Requested High Throughput	Enabled, Disabled (Default: Disabled)	高スループット要求の選択
Reset Pin (must be a valid C symbol)	IOPORT_PORT_06_PIN_00	リセットピンの選択
Slave Select Pin (SSL)(Must be a valid C symbol)	IOPORT_PORT_01_PIN_03	スレーブ選択ピンの選択
GT202 Driver Task Thread Priority (Modifying Task Thread Priority may cause Driver to malfunction)	5	GT202 ドライバータスクスレッドプライオリティの選択
Callback	NULL	コールバックの選択
Support NetX Packet Chaining	Enabled, Disabled (Default: Enabled)	NetX パケットチェーンのサポートの選択

***注:** 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_rspi での SPI HAL ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。

ISDE Property	Setting	説明
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSB ファーストと LSB ファーストのいずれかの送信順序を選択します
Bitrate	500000	送信または受信レート。1秒あたりのビット。
Callback	NULL	オプションのコールバック関数ポインタ。
SPI Mode	Clock synchronous operation	動作モードを選択します（SPI またはクロック同期）。
SPI Communication Mode	Full Duplex	通信方式を選択します（全二重または送信のみ）。
Slave Select Polarity(SSL0)	Active Low	SSL0 の信号極性を選択します
Slave Select Polarity(SSL1)	Active Low	SSL1 の信号極性を選択します
Slave Select Polarity(SSL2)	Active Low	SSL2 の信号極性を選択します
Slave Select Polarity(SSL3)	Active Low	SSL3 の信号極性を選択します
Select Loopback1	Normal	loopback1 を選択します
Select Loopback2	Normal	loopback2 を選択します

ISDE Property	Setting	説明
Enable MOSI Idle State	Disable	MOSI アイドル固定値および選択肢を選択します
MOSI Iidle State	MOSI Low	MOSI アイドル固定値および選択肢を選択します
Enable Parity	Disable	パリティを有効または無効にします
Parity Mode	Parity Even	パリティを選択します
Select SSL(Slave Select)	SSL0	使用するスレーブを選択します (0-SSL0、1-SSL1、2-SSL2、3-SSL3)
Select SSL Level After Transfer	SSL Level Do Not Keep	転送完了後の SSL レベルを選択します（0- ネゲート、 1- 維持）
Clock Delay Enable	Clock Delay Disable	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 RSPCK	クロック遅延カウントの選択
SSL Negation Delay Enable	Negation Delay Disable	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 RSPCK	ネゲート遅延カウントの選択
Next Access Delay Enable	Next Access Delay Disable	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 RSPCK	次アクセス遅延カウントの選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	受信割り込み優先順位の選択
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Setting	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Setting	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注：例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数の有効化または無効化

ISDE Property	Setting	説明
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。*注：これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注：例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc イベント SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。

ISDE Property	Setting	説明
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。***注：****SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。***注：****これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択

ISDE Property	Setting	説明
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択

ISDE Property	Setting	説明
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定は、パラメータチェックコードの追加を有効化または無効化します。
Name	g_external_irq0	モジュール名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling	トリガイベントモードの選択
Digital Filtering	Disabled	デジタルフィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	TRUE	初期化の直後に割り込みが有効化されるかどうかを決定します。

ISDE Property	Setting	説明
Callback	custom_hw_irq_isr	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 IRQn がトリガーされるたびに割り込みサービスルーチン (ISR) から呼び出されます。 ***注:*** *注意: コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択。

注: *例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

GT202 Wi-Fi フレームワークでの GT202 オンチップスタックを使用した BSD ソケットの構成

このセクションでは、WiFi フレームワークの GT202 オンチップスタック実装を使用した BSD ソケットの構成設定を示します。

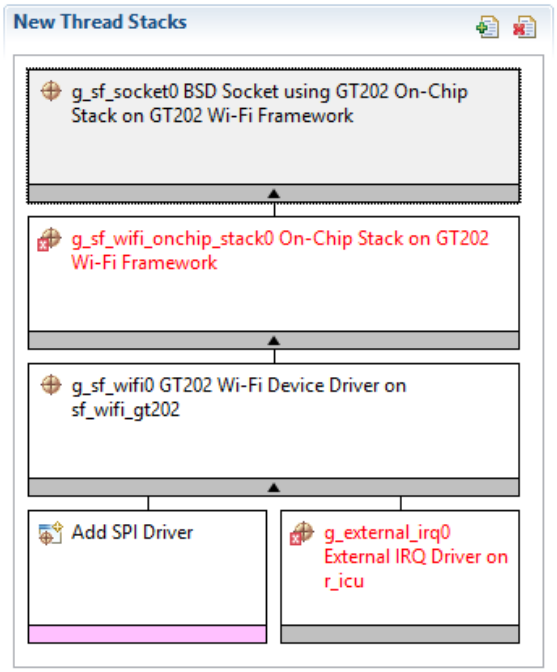


図 264: GT202 Wi-Fi フレームワークでの GT202 オンチップスタックを使用した BSD ソケット

GT202 Wi-Fi フレームワークでの GT202 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Name (Must be a valid C Symbol)	g_sf_socket0	モジュール名

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

GT202 Wi-Fi フレームワークでのオンチップスタックの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Name (Must be a valid C Symbol)	g_sf_wifi_onchip_stack0	モジュール名

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

WiFi デバイスドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled (Default: Disabled)	オンチップスタックサポートの選択
Driver Heap Size in Bytes (Minimum 8192 bytes)	8192	ドライバーのヒープサイズの選択
Name (Must be a valid C symbol)	g_sf_wifi0	モジュール名
Hardware Mode	802.11a, 802.11b, 802.11g, 802.11n (Default: 802.11n)	ハードウェアモードの選択
Transmit (TX) Power (Valid Range 1-17)	10	送信電力の選択
Ready/Clear to Send (RTS/CTS) Flag	Enabled, Disabled (Default: Enabled)	Ready または Clear の送信の選択
Delivery Traffic Indication Message (DTIM) Interval (Valid Range: 1-255)	3	配信トラフィック指示メッセージインターバルの選択
Broadcast SSID (AP mode only)	Enabled, Disabled (Default: Enabled)	ブロードキャスト SSID の選択
Beacon Interval in Microseconds (AP mode only and must be greater than 1023)	1024	ビーコンインターバル（マイクロ秒単位）の選択
Station inactivity timeout in seconds (AP mode only and must be greater than 0)	100	ステーション非アクティブタイムアウトの選択
Requested High Throughput	Enabled, Disabled (Default: Disabled)	高スループット要求の選択
Reset Pin (must be a valid C symbol)	IOPORT_PORT_06_PIN_00	リセットピンの選択
Slave Select Pin (SSL)(Must be a valid C symbol)	IOPORT_PORT_01_PIN_03	スレーブ選択ピンの選択
GT202 Driver Task Thread Priority (Modifying Task Thread Priority may cause Driver to malfunction)	5	GT202 ドライバータスクスレッドプライオリティの選択

ISDE Property	Setting	説明
Callback	NULL	コールバックの選択
Support NetX Packet Chaining	Enabled, Disabled (Default: Enabled)	NetX パケットチェーンのサポートの選択

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_rspi での SPI HAL ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。***注:*** *SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSB ファーストと LSB ファーストのいずれかの送信順序を選択します
Bitrate	500000	送信または受信レート。1秒あたりのビット。
Callback	NULL	オプションのコールバック関数ポインタ。
SPI Mode	Clock synchronous operation	動作モードを選択します（SPI またはクロック同期）。

ISDE Property	Setting	説明
SPI Communication Mode	Full Duplex	通信方式を選択します（全二重または送信のみ）。
Slave Select Polarity(SSL0)	Active Low	SSL0 の信号極性を選択します
Slave Select Polarity(SSL1)	Active Low	SSL1 の信号極性を選択します
Slave Select Polarity(SSL2)	Active Low	SSL2 の信号極性を選択します
Slave Select Polarity(SSL3)	Active Low	SSL3 の信号極性を選択します
Select Loopback1	Normal	loopback1 を選択します
Select Loopback2	Normal	loopback2 を選択します
Enable MOSI Idle State	Disable	MOSI アイドル固定値および選択肢を選択します
MOSI Iidle State	MOSI Low	MOSI アイドル固定値および選択肢を選択します
Enable Parity	Disable	パリティを有効または無効にします
Parity Mode	Parity Even	パリティを選択します
Select SSL(Slave Select)	SSL0	使用するスレーブを選択します(0-SSL0、1-SSL1、2-SSL2、3-SSL3)
Select SSL Level After Transfer	SSL Level Do Not Keep	転送完了後の SSL レベルを選択します（0- ネゲート、 1- 維持）
Clock Delay Enable	Clock Delay Disable	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 RSPCK	クロック遅延カウントの選択
SSL Negation Delay Enable	Negation Delay Disable	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 RSPCK	ネゲート遅延カウントの選択
Next Access Delay Enable	Next Access Delay Disable	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 RSPCK	次アクセス遅延カウントの選択

ISDE Property	Setting	説明
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	受信割り込み優先順位の選択
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択

ISDE Property	Setting	説明
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択

ISDE Property	Setting	説明
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。***注:*** *SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*

ISDE Property	Setting	説明
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。***注：****これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注：例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc イベント SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。*注：これは SCI SPI に対してのみ適用できます。*

ISDE Property	Setting	説明
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc イベント SCI0 RXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択

ISDE Property	Setting	説明
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定は、パラメータチェックコードの追加を有効化または無効化します。
Name	g_external_irq0	モジュール名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling	トリガイベントモードの選択
Digital Filtering	Disabled	デジタルフィルター有効 / 無効。

ISDE Property	Setting	説明
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	TRUE	初期化の直後に割り込みが有効化されるかどうかを決定します。
Callback	custom_hw_irq_isr	ユーザーコールバック関数をopenで登録できます。このコールバック関数が指定されている場合、IRQn がトリガーされるたびに割り込みサービスルーチン (ISR) から呼び出されます。警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択。

注: *例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

Wi-Fi フレームワークを使用した NetX ポートの構成

このセクションでは、 WiFi フレームワークの NetX ポート実装の構成設定を示します。

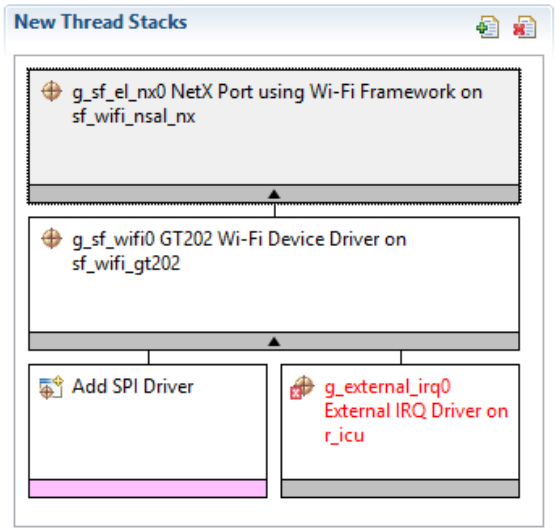


図 265: Wi-Fi フレームワークを使用する NetX ポート

sf_wifi_nsal_nx で Wi-Fi フレームワークを使用する NetX ポートの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Name (Must be a valid C Symbol)	g_sf_el_nx0	モジュール名

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

WiFi デバイスドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
On-Chip Stack Support	Enabled, Disabled (Default: Disabled)	オンチップスタックサポートの選択
Driver Heap Size in Bytes (Minimum 8192 bytes)	8192	ドライバーのヒープサイズの選択
Name (Must be a valid C symbol)	g_sf_wifi0	モジュール名

ISDE Property	Setting	説明
Hardware Mode	802.11a, 802.11b, 802.11g, 802.11n (Default: 802.11n)	ハードウェアモードの選択
Transmit (TX) Power (Valid Range 1-17)	10	送信電力の選択
Ready/Clear to Send (RTS/CTS) Flag	Enabled, Disabled (Default: Enabled)	Ready または Clear の送信の選択
Delivery Traffic Indication Message (DTIM) Interval (Valid Range: 1-255)	3	配信トラフィック指示メッセージインターバルの選択
Broadcast SSID (AP mode only)	Enabled, Disabled (Default: Enabled)	ブロードキャスト SSID の選択
Beacon Interval in Microseconds (AP mode only and must be greater than 1023)	1024	ビーコンインターバル（マイクロ秒単位）の選択
Station inactivity timeout in seconds (AP mode only and must be greater than 0)	100	ステーション非アクティブタイムアウトの選択
Requested High Throughput	Enabled, Disabled (Default: Disabled)	高スループット要求の選択
Reset Pin (must be a valid C symbol)	IOPORT_PORT_06_PIN_00	リセットピンの選択
Slave Select Pin (SSL)(Must be a valid C symbol)	IOPORT_PORT_01_PIN_03	スレーブ選択ピンの選択
GT202 Driver Task Thread Priority (Modifying Task Thread Priority may cause Driver to malfunction)	5	GT202 ドライバータスクスレッドプライオリティの選択
Callback	NULL	コールバックの選択
Support NetX Packet Chaining	Enabled, Disabled (Default: Enabled)	NetX パケットチェーンのサポートの選択

***注:** 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_rspi での SPI HAL ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。

ISDE Property	Setting	説明
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロックレベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSB ファーストと LSB ファーストのいずれかの送信順序を選択します
Bitrate	500000	送信または受信レート。1秒あたりのビット。
Callback	NULL	オプションのコールバック関数ポインタ。
SPI Mode	Clock synchronous operation	動作モードを選択します（SPI またはクロック同期）。
SPI Communication Mode	Full Duplex	通信方式を選択します（全二重または送信のみ）。
Slave Select Polarity(SSL0)	Active Low	SSL0 の信号極性を選択します
Slave Select Polarity(SSL1)	Active Low	SSL1 の信号極性を選択します
Slave Select Polarity(SSL2)	Active Low	SSL2 の信号極性を選択します
Slave Select Polarity(SSL3)	Active Low	SSL3 の信号極性を選択します
Select Loopback1	Normal	loopback1 を選択します
Select Loopback2	Normal	loopback2 を選択します

ISDE Property	Setting	説明
Enable MOSI Idle State	Disable	MOSI アイドル固定値および選択肢を選択します
MOSI Iidle State	MOSI Low	MOSI アイドル固定値および選択肢を選択します
Enable Parity	Disable	パリティを有効または無効にします
Parity Mode	Parity Even	パリティを選択します
Select SSL(Slave Select)	SSL0	使用するスレーブを選択します (0-SSL0、1-SSL1、2-SSL2、3-SSL3)
Select SSL Level After Transfer	SSL Level Do Not Keep	転送完了後の SSL レベルを選択します（0- ネゲート、 1- 維持）
Clock Delay Enable	Clock Delay Disable	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 RSPCK	クロック遅延カウントの選択
SSL Negation Delay Enable	Negation Delay Disable	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 RSPCK	ネゲート遅延カウントの選択
Next Access Delay Enable	Next Access Delay Disable	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 RSPCK	次アクセス遅延カウントの選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	受信割り込み優先順位の選択
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Setting	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: *例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Setting	説明
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注：例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータ エラー チェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master	マスター デバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロック レベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	ビットレート変調関数の有効化または無効化

ISDE Property	Setting	説明
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Priority 2)	ビットレート変調関数の有効化または無効化。*注：これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

***注：** 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。

ISDE Property	Setting	説明
Operating Mode	Master	マスターデバイスまたはスレーブデバイスとして設定します。*注：SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on even edge, data variation on odd edge	奇数または偶数クロック エッジのいずれかのデータ サンプリングを選択します。
Clock Polarity	High when idle	アイドル時のクロック レベル。
Mode Fault Error	Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	100000	送信または受信レート。1秒あたりのビット。
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	ビットレート変調関数の有効化または無効化
Callback	NULL	オプションのコールバック関数ポインタ。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	ビットレート変調関数の有効化または無効化。*注：これは SCI SPI に対してのみ適用できます。*
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	送信終了割り込み優先順位の選択

ISDE Property	Setting	説明
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	エラー割り込み優先順位の選択

***注:** 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI 時の転送ドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択

ISDE Property	Setting	説明
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	ELC ソフトウェアイベント割り込み優先順位の選択。

注: 例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック設定は、パラメータチェックコードの追加を有効化または無効化します。
Name	g_external_irq0	モジュール名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger	Falling	トリガイベントモードの選択
Digital Filtering	Disabled	デジタルフィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	TRUE	初期化の直後に割り込みが有効化されるかどうかを決定します。

ISDE Property	Setting	説明
Callback	custom_hw_irq_isr	ユーザーコールバック関数をopenで登録できます。このコールバック関数が指定されている場合、IRQn がトリガーされるたびに割り込みサービスルーチン (ISR) から呼び出されます。警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	割り込み優先順位の選択。

***注:** 例の値とデフォルトは、 Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

WiFi フレームワークモジュールのクロック構成

WiFi フレームワークモジュールは、選択された特定の低レベルモジュール （SPI など）に必要なクロックを使用します。

WiFi フレームワークモジュールのピン構成

WiFi フレームワークモジュールは、選択された低レベルモジュール（SPI や IRQ など）に依存する入力ピンと出力ピンを使用します。

5.1.25.6 アプリケーションでの WiFi フレームワークモジュールの使用

WiFi フレームワークは、実装ごとにターゲットアプリケーションによる扱いが異なります。初期化、NetX/NetX Duo を使用したパケット送信、NetX/NetX Duo を使用したパケット受信、およびオンチップネットワークスタックの使用に関する一般的な制御フローは、多くのアプリケーションにおいて重要です。以下の図では、いくつかの一般的な実装の機能フローの例を示します。

ドライバーの初期化

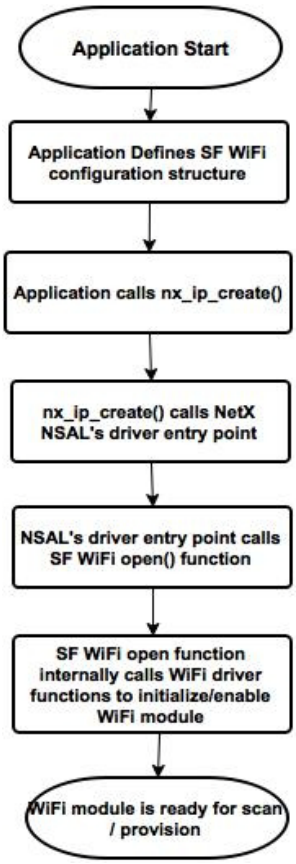


図 266: WiFi モジュールの初期化を行うアプリケーションの制御フロー

NetX/NetX-Duo を使用するパケット送信

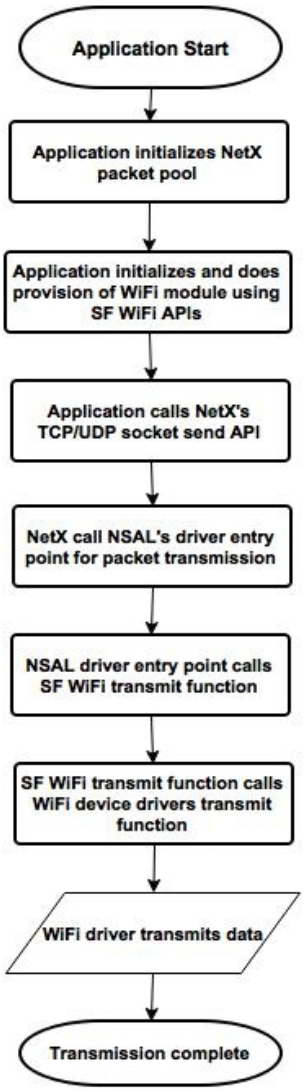


図 267: NetX を使用してパケット送信を行うアプリケーションの制御フロー

NetX/NetX-Duo を使用するパケット受信

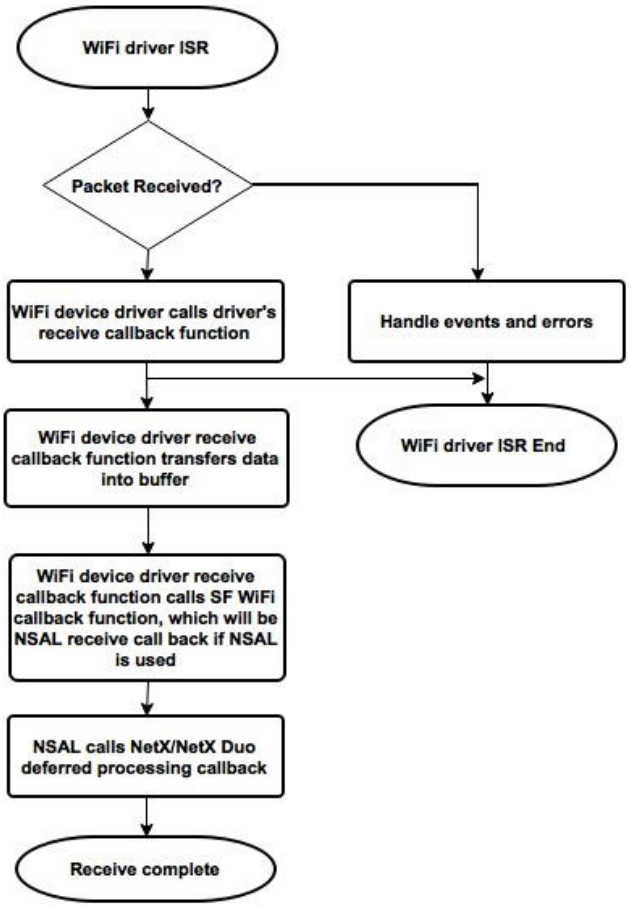


図 268: NetX を使用してパケットを受信するアプリケーションの制御フロー

オンチップネットワークスタックを使用するアプリケーション

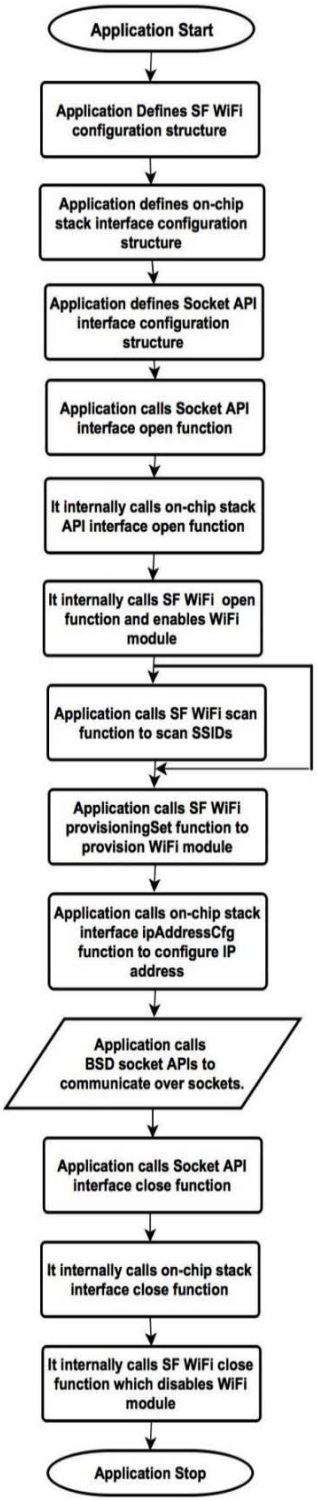


図 269: オンチップネットワークスタックを使用するアプリケーションの制御フロー

5.1.26 GUIX Synergy ポートフレームワークモジュール

GUIX Synergy ポートフレームワークモジュールには、以下の主要機能が含まれます。

- GUIX を SSP フレームワークに適応させる
- SSP ディスプレイインタフェースドライバーを GUIX ディスプレイドライバーインタフェースに設置する
- Synergy D2W (2DG) エンジンにより、 GUIX が高速でウィジェットを描画できるようにする
- Synergy JPEG エンジンにより、 GUIX が高速でウィジェットを描画できるようにする
- 切れ目が発生することなく画面の移行を可能とする、ダブルバッファ切り替え制御をサポートする
- 画面のローテーションをサポートする （90/180/270 度）
- 各種の色出力形式をサポートする
  - 32bpp （ARGB8888、 RGB-888）
  - 16bpp （RGB565）
  - 8bpp （8 ビットパレット （CLUT））
- ユーザーコールバック関数をサポートする

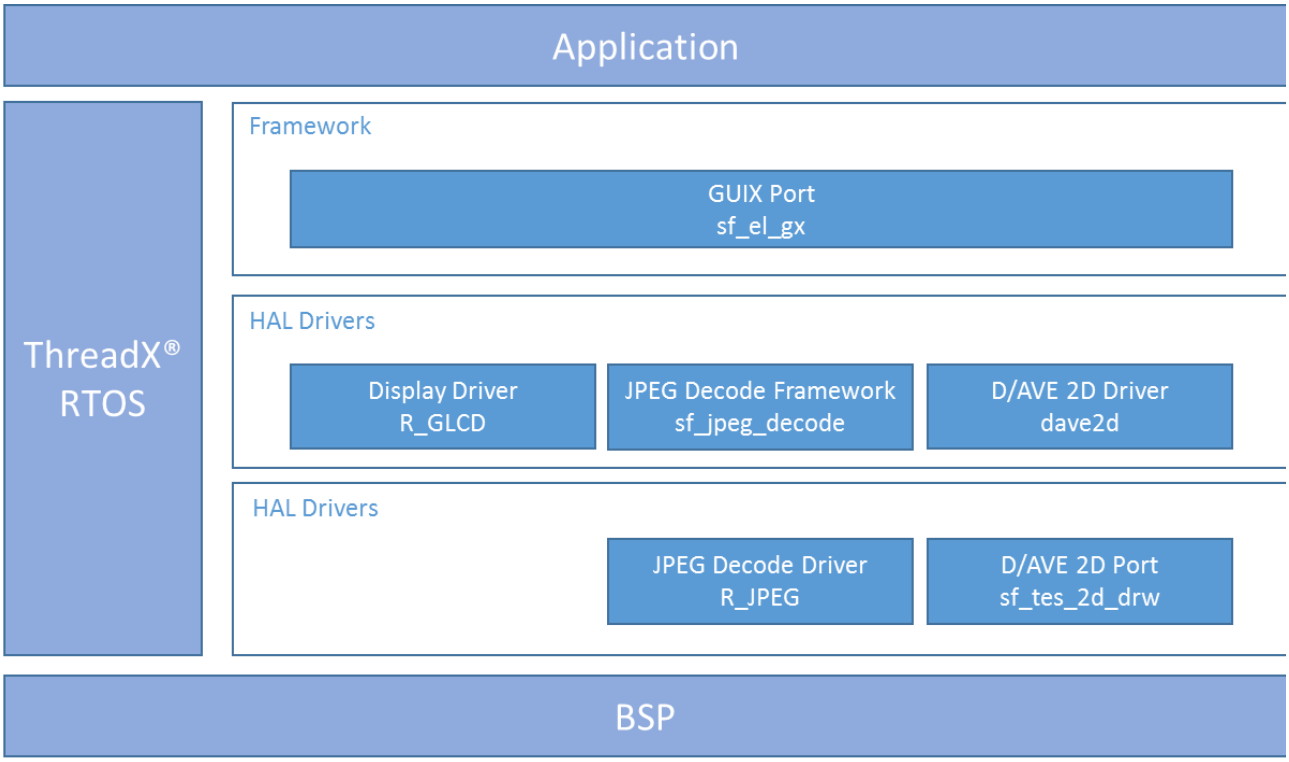


図 270: GUIX Synergy ポートフレームワークの編成、オプション、スタックの実装

5.1.26.1 GUIX Synergy ポートフレームワークモジュール API の概要

GUIX Synergy ポートフレームワークは、オープン、クローズ、設定、初期化の API を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

GUIX Synergy ポートフレームワークモジュール API の要約

Function Name	API の呼び出し例と説明
open	g_sf_el_gx.p_api->open(g_sf_el_gx.p_ctrl, g_sf_el_gx.p_cfg); SF_EL_GX モジュールを開きます。この API は、1 つのスレッドからのみ呼び出すことができます。この API は、構成ポインタを渡してローレベルグラフィックスデバイスドライバーとフレームバッファを定義し、ユーザーコールバック関数を登録します。この関数は実際には低レベルドライバーを初期化しません。ローレベルドライバーの初期化は API setup によって行われます。その理由については、下記の setup の説明を参照してください。
close	g_sf_el_gx.p_api->close(g_sf_el_gx.p_ctrl); SF_EL_GX モジュールを閉じます。この API は低レベルドライバーを閉じます。通常、GUIX は初期化後に閉じられないため、この API は呼び出されません。
versionGet	g_sf_el_gx.p_api->versionGet(&version); モジュールのバージョンをバージョンポインタで返します。

Function Name	API の呼び出し例と説明
setup	gx_studio_display_configure (MAIN_DISPLAY, g_sf_el_gx.p_api->setup, LANGUAGE_ENGLISH, MAIN_DISPLAY_THEME_1, &p_window_root); これはローレベルグラフィックスデバイスドライバーを初期化するためのインタフェースであり、GUIX（Studio）サービス呼び出し gx_studio_display_configure() を介してこれを関数ポインタとして GUIX に渡す必要があります。続いて GUIX はこの API をコールバックし、そのときに open によって渡された構成に基づいて SSP デバイスドライバーが構成されます。このような手順でローレベルドライバーを初期化する理由は、この API は GUIX 準拠の引数 (GX_DISPLAY*) 型を持っており、この API では e^2 studio から生成された SSP グラフィックスデバイスドライバーの詳細な構成を適用できないためです。
canvasInit	g_sf_el_gx.p_api->canvasInit(g_sf_el_gx.p_ctrl, p_window_root); これは GUIX キャンバスのメモリアドレスを決定する GUIX ヘルパー API です。この API には (GX_WINDOW_ROOT*) 型の引数があり、キャンバスメモリの開始アドレスを GUIX に提供します。このアドレスは、低レベルグラフィックスデバイスドライバーがイメージを描画 / 表示するために必要となります。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

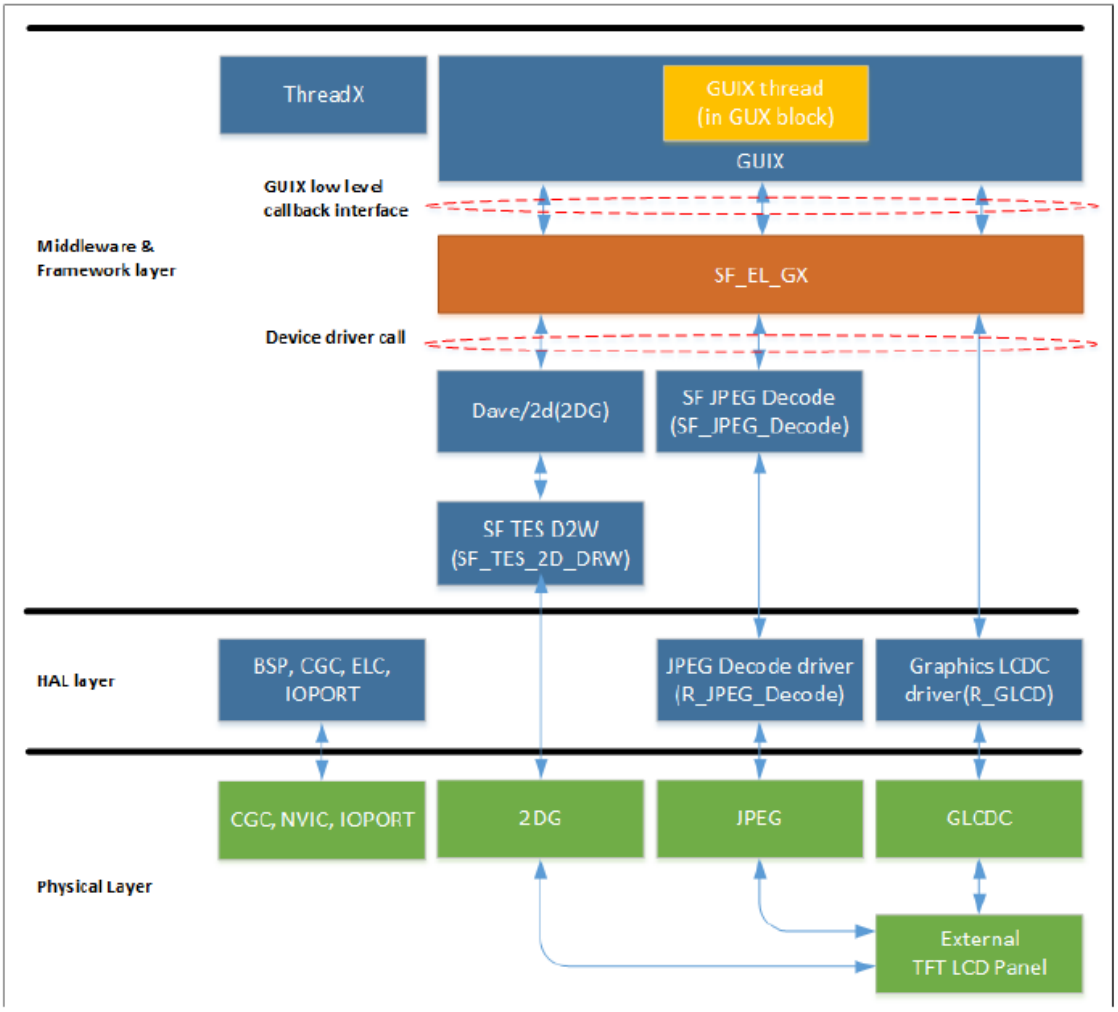
Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	NULL ポインタのエラーが発生しました。

Name	説明
SSP_ERR_IN_USE	SF_EL_GX が使用中です。
SSP_ERR_INTERNAL	カーネルサービス呼び出し中にエラーが発生しました。
SSP_ERR_NOT_OPEN	SF_EL_GX が開かれていません。
SSP_ERR_TIMEOUT	タスクがディスプレイドライバーで完了する前にタイムアウトになりました（またはリトライ制限を超えました）。
SSP_ERR_D2D_ERROR_DEINT	D/AVE 2D ドライバーでエラーが発生しました。
GX_SUCCESS	デバイスドライバーのセットアップが正常に完了しました。
GX_FAILURE	デバイスドライバーのセットアップが失敗しました。
SSP_ERR_INVALID_CALL	ドライバーが SF_EL_GX_CONFIGURED 状態でないときに、関数呼び出しが実行されます。
SSP_ERR_D2D_RENDERING	D/AVE 2D は表示リストバッファを開くときにエラーを返します。
SSP_ERR_INVALID_ARGUEMENT	無効な非ポインタ（たとえばパラメータ）入力です
SSP_ERR_UNSUPPORTED	指定されたカラーフォーマットはサポートされていません
SSP_ERR_D2D_ERROR_INIT	D/AVE 2D は初期化でエラーを返します。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.1.26.2 GUIX Synergy ポートフレームワークモジュールの概要

下図に、Synergy グラフィックスソリューションのコンポーネントと、このソリューションでのグラフィックスデータの流れを示します。



- モジュールの初期化

SF_EL_GX は、GUIX システムの実行に必要な Synergy グラフィックスハードウェア設定をサポートします。モジュールは、 Express Logic GUIX^{TM} および GUIX Studio^{TM} で生成されたコードに依存します。 GUIX システムの初期化では、一般的なガイダンスとして以下のシーケンスに従う必要があります。

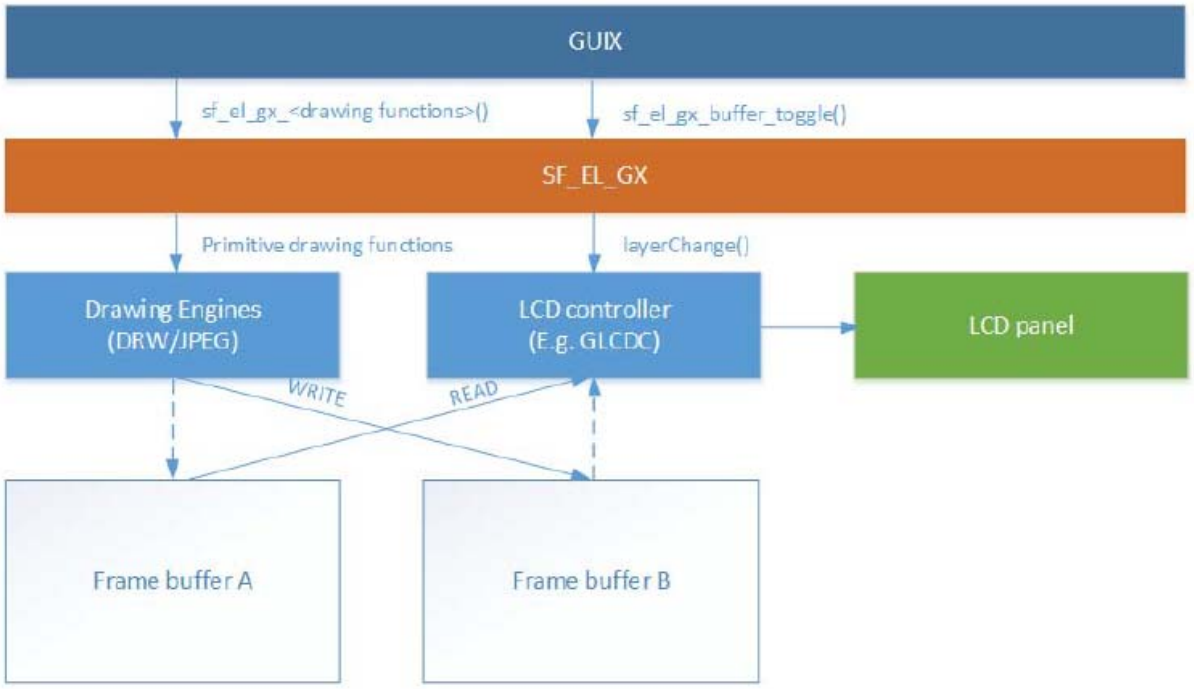
1) SF_EL_GX モジュールの 'open' を実行して SF_EL_UX コントロールブロックを初期化し、モジュール構成を渡します。
2) GUIX Studio で生成された API gx_studio_display_configure. で GUIX ディスプレイオブジェクトを初期化します。この API を通じて、SF_EL_GX 'setup' API が GUIX に入力され、Synergy グラフィックスハードウェア設定が完了します。また、GUIX で初期化されるルートウィンドウは、ユーザーアプリケーションに出力されます。
3) canvasInit API で GUIX キャンバスバッファのプライマリメモリアドレスを初期化します。
4) GUIX Studio で生成された API gx_studio_named_widget_create. でルートウィンドウを作成します。

5) GUIX API gx_window_show API でルートウィンドウを表示します。

6) GUIXAPI gx_system_start. で GUIX システムを起動します。

- ピンポンフレームバッファ管理

SF_EL_GX モジュールは、ピンポンフレームバッファを備えたグラフィックスシステムでバッファ切り替え動作を管理します。下図に、SF_EL_GX モジュールによって管理されるピンポンバッファグラフィックスシステムを示します。このモジュールは、GUIX とローレベルディスプレイドライバーの機能を使用してイメージの描画（2D 描画エンジン（DRW）または JPEG）とイメージの表示（ディスプレイモジュール、たとえば GLCDC など）を行います。SF_EL_GX 構成では、単一のフレームバッファを持つ設計も可能です。しかし、2 つのフレームバッファを使用して、単一フレームバッファシステムで起こり得る切れ目の問題を回避することをお勧めします。



GUIX Synergy ポートフレームワークモジュールの動作に関する注意事項

Synergy 2D 描画エンジンのサポート

モジュールは、2D 描画エンジン（DRW）で加速されるグラフィックスイメージを描画して、グラフィックス性能を向上させることができます。ユーザーは、以下の構成で 2D 描画エンジンを有効化できます。この構成は、Synergy コンフィギュレータを通じて利用できます。

- gx_user.h で GX_USE_SYNEGY_DRW （1）を定義する
- DRW (SF_TES_2D_DRW) 割り込み優先順位を設定する

Express Logic GUIX Studio（v5.2.8 以降）の [Configure Project] ウィンドウから、[Target CPU] 設定で [Renesas Synergy] が選択され、[Synergy Advanced Settings] ウィンドウの [Enable Graphics Accelerator] がチェックされていることを確認してください。[Edit Pixel map]ウィンドウのPixelmap出力フォーマットについては（[pixelmap]

-> [edit settings] を右クリックするとウィンドウが開きます）、 [Compress Output] を選択してください。 [Raw Output] を選択しないでください。このように設定することで、 GUIX Studio により Targa RLE フォーマットでエンコードされた画像リソースデータが生成されます。2D 描画エンジンハードウェアはこのフォーマットを読み取れるうえに、フレームバッファに画像のデコードと描画を実行できます。

Synergy JPEG のサポート

JPEG でエンコードされたイメージが GUIX イメージリソースとして使用されている場合、モジュールは、JPEG エンジンで加速されるグラフィックスイメージを描画して、グラフィックス性能を向上させることができます。ユーザーは、以下の構成で JPEG エンジンを有効化できます。この構成は、Synergy コンフィギュレータを通じて利用できます。

- gx_user.h で GX_USE_SYNEGY_JPEG （1）を定義する
- JPEG (R_JPEG_DECODE) 割り込み優先順位を設定する

Express Logic GUIX Studio（v5.2.8 以降）の [Configure Project] ウィンドウから、 [Target CPU] 設定で [Renesas Synergy] が選択され、 [Decoder Types JPEG] ドロップダウンメニューの [Hardware JPEG Decoder] がチェックされていることを確認してください。 [Exit Pixel map] ウィンドウの Pixelmap 出力フォーマットについては（[pixelmap] -> [edit settings] を右クリックするとウィンドウが開きます）、[Raw Format] を選択してください。このように設定することで、GUIX Studio により、圧縮されていない JPEG エンコードされた画像リソースデータが生成されます。JPEG ハードウェアはこのフォーマットをリードできるうえに、フレームバッファに画像のデコードと描画を実行できます。

GUIX キャンバスバッファ

モジュールでは、グラフィックス画面イメージを回転するために使用される GUIX キャンバスバッファがサポートされます。 GUIX キャンバスバッファのサイズは、ディスプレイモジュールのフレームバッファとまったく同じでなければなりません。 GUIX キャンバスバッファを使用すると、追加のグラフィックイメージ処理が必要なため、グラフィックス性能に影響します。このため、 GUIX キャンバスバッファは画面のローテーションが必要な場合にのみ使用する必要があります。それ以外の場合は、p_canvas に NULL を設定して、GUIX が画像をフレームバッファに直接描画するようにします。

画面のローテーション

このモジュールは、画面のローテーションをサポートしています。GUIX は GUIX Studio で設計されたとおりに画面イメージを GUIX キャンバスバッファに描画しますが、 SF_EL_GX モジュールは画面のローテーションを実行して、回転された画面イメージをフレームバッファに描画できます。たとえば、GUIX Studio で横長イメージとして設計された GUI 画面を回転し、横長形状のディメンションの LCD パネルに表示できます。サポートされているローテーション角度は反時計回りに 90、 180、 270 度で、これは open で設定できます。動的な画面ローテーションはサポートされていません。画面のローテーション機能を有効にするには、GUIX キャンバスバッファを使用する必要があります。GUIX はまずキャンバスに画面更新を描画し、その後 GUIX ポートがフレームバッファへの画面コピーを反時計回りにイメージを回転させて処理します。 *Synergy 2D 描画エンジンサポート*が有効になっている場合 (GX_USE_SYNEGY_DRW = 1)、ローテーションは 2D 描画エンジンによってテクスチャマッピングで処理されます。有効になっていない場合 (GX_USE_SYNEGY_DRW = 0)、ローテーションはソフトウェアコピーによって処理されます。

注: *（rotation_angle = 0）かつ （p_canvas = NULL 値以外）という構成は禁止されてはいませんが、行わないでください。この構成は、GUIX キャンバスバッファからフレームバッファへの画面イメージコピーに余分なバス帯域を消費します。したがって、p_canvas に NULL を設定し、GUIX キャンバスバッファを使用しないようにします。*

JPEG 作業バッファのサイズ

JPEG 作業バッファと JPEG のデコード速度は、バッファサイズに対してトレードオフの関係にあります。画面のウェジェットが JPEG でフォーマットされている場合は、JPEG 作業バッファは一時ストレージメモリとして使用され、デコードされた画像が作成されます。画像全体をデコードするにあたってバッファサイズの大きさが不足している場合は、JPEG デコーディングは出力バッファストリーミングモードで実行されます。2D 描画エンジンの BitBLT 操作では、バッファの JPEG ラスタ画像のピースがデコードされ、その後フレームバッファに転送されます。JPEG 作業バッファの最小サイズは、{（水平ラインのピクセル数）x（ディスプレイフォーマットの bpp（1 ピクセルあたりのバイト数））x8（ライン）} です。たとえば、デコードした画像が水平ライン 800 ピクセルで RGB565 フォーマットの場合では、この数は 800 x 2 x 8= 12 800（バイト）となります。バッファサイズがこの数値より小さい場合、JPEG デコーディングは実行されません。スループットを向上させるには、[Size of the JPEG Work Buffer] のパラメータをもっと大きくする必要があります。それにより、JPEG デコードのスループットが向上します。JPEG 出力バッファストリーミングモードは部分的な JPEG デコード操作を繰り返し、補充がオーバーヘッドになります。

単一のバッファ設計における画面の切れ目

画面の切れ目は、ディスプレイデバイスが複数フレームからの情報を単一の画面描画で表示するビデオ表示の視覚的アーティファクトです。一般的に、単一フレームバッファを使用するシステムは、LCD パネルにおける画面の切れ目問題の原因となります。単一フレームバッファを使用することはできますが（p_framebuffer_b に NULL を設定）、画面の切れ目がモジュールによって対処されることはありません。2 つのフレームバッファから構成されるピンポンフレームバッファシステムを使用することをお勧めします。

GUIX Synergy ポートフレームワークモジュールの制限事項

- SF_EL_GX は、GLCDC を搭載した Synergy MCU（必須）、2D 描画エンジン、JPEG エンジン（オプション）にのみ適用できます
- SF_EL_GX は、3 つ以上のフレームバッファを持つシステムをサポートしていません。
- SF_EL_GX は、GUIX キャンバスシステムを 1 つだけサポートします。
- SF_EL_GX は、ディスプレイモジュールのレイヤーを 1 つだけ利用します。
- GUIX が TES D/AVE 2D モジュールおよび TES D/AVE 2D ポートモジュールを使用する場合、これらのモジュールに直接アクセスしないでください。
- GUIX が JPEG デコードフレームモジュールおよび JPEG デコード HAL モジュールを使用する場合、これらのモジュールに直接アクセスしないでください。
- このモジュールを使用する際のその他の制限事項については、SSP の最新のリリースノートを参照してください。

5.1.26.3 JPEG デコード HAL モジュールの動作の概要

JPEG デコード HAL モジュールは、r_jpeg. に実装されている JPEG デコード処理のための高レベルの API です。このモジュールは、Synergy JPEG コーデックをサポートします。以下の特長があります。

- JPEG 解凍をサポート。
- JPEG　デコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直サブサンプル値、水平ストライド、デコードされたピクセルフォーマット、入力および出力データフォーマット、色空間などのパラメータを設定。

- イメージをデコードする前にそのサイズを取得。
- デコードされたイメージフレームの保管のため、コード化されたデータを入力バッファーおよび出力バッファーに格納する操作をサポート。
- JPEG　デコーダーモジュールへのコード化されたデータのストリーム転送をサポート。この機能により、アプリケーションは、ファイルやネットワークからのコード化された JPEG イメージの読み取りを、イメージ全体をバッファーリングすることなく実行できます。
- デコードするイメージ行数を設定。この機能により、アプリケーションは、デコードされたイメージの処理を、フレーム全体をバッファーリングすることなくオンザフライで実行可能。
- 入力値のデコードされたフォーマットとして YCbCr444、YCbCr422、YCbCr420、YCbCr411 をサポート。
- 出力フォーマットとして ARGB8888、 RGB565 をサポート。
- JPEG イメージのサイズ、高さ、および幅が要件を満たさない場合にエラーを返すことが可能。

JPEG デコーダ HAL モジュールは、入力バッファストリーミングモードまたは JPEG 出力バッファストリーミングモードで使用できます。

入力バッファストリーミングモードの動作の説明

このシナリオでは、JPEG イメージデータはファイル上に存在し、ネットワークから受信されます。 HAL レイヤドライバーは、最初に入力データをメモリに格納しなくても、このシナリオを処理できます。

出力バッファストリーミングモードの動作の説明

このシナリオでは、アプリケーションはデコードされたイメージデータをファイルまたはネットワークに書き込む必要があります。HAL レイヤドライバーは、アプリケーションがフレーム全体のメモリを割り当てることを必要としません。代わりに、アプリケーションは一度に 1 ライン以上のデコードを選択できます。この特長により、出力データに必要なメモリ量が大幅に削減されます。

JPEG デコード HAL モジュールの動作に関する注意事項

JPEG デコードコールバック

ユーザーコールバック関数を open API で登録できます。ユーザーコールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービスルーチン (ISR) からコールバック関数が呼び出されます。コールバック関数の引数 status は、デコーディングプロシージャにおいて発生したイベントをユーザーが識別できるように、以下に示す列挙値を受け取ることができます。

Event Name	イベントの条件
JPEG_DECODE_STATUS_ERROR JPEG	デコードモジュールでエラーが発生した。
JPEG_DECODE_STATUS_IMAGE_SIZE_READY JPEG	デコードが、デコードするデータのイメージサイズを取得して、一時停止した。
JPEG_DECODE_STATUS_INPUT_PAUSE JPEG	デコードが追加の入力データを待つために一時停止した。
JPEG_DECODE_STATUS_OUTPUT_PAUSE JPEG	デコードが、ユーザーにより指定された行数をデコードした後に一時停止した。

Event Name	イベントの条件
JPEG_DECODE_STATUS_DONE JPEG	デコード操作が正常に完了した。

***注：** ユーザーコールバック関数は ISR から呼び出されるため、ブロッキング呼び出しを使用することや、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*

JPEG デコード HAL モジュールの制限事項

- JPEG デコード HAL モジュールドライバーは、 JPEG エンコード処理をサポートしていません。
- このモジュールの最新の制限事項については、 SSP の最新のリリースノートを参照してください。

5.1.26.4 アプリケーションへの GUIX フレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用して GUIX フレームワークモジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

GUIX フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（GUIX フレームワークのデフォルト名は g_sf_el_gx0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

GUIX フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_gx0 GUIX Port on sf_el_gx	**Threads**	**New Stack> Framework> Graphics> GUIX Port on sf_el_gx**

下図に示すように sf_el_gx の GUIX Synergy ポートフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、 1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に 「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、 [New] アイコンが表示され、可能な選択肢が表示されます。

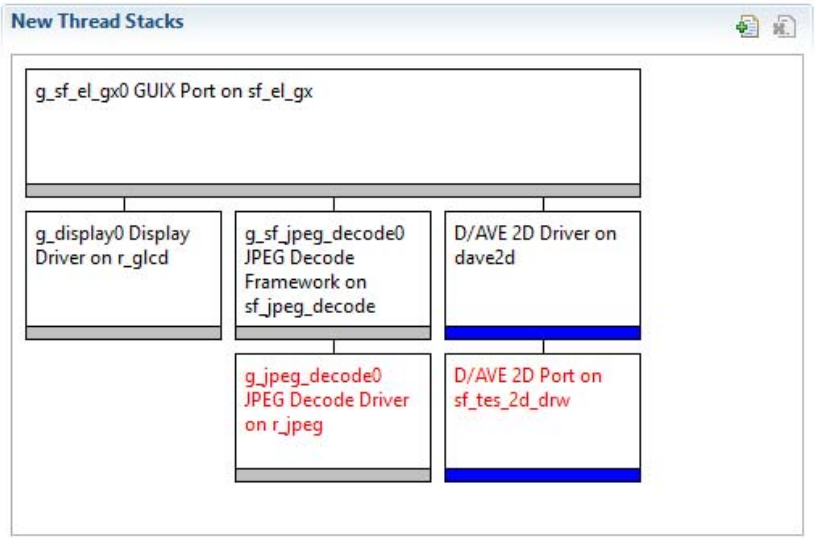


図 271: GUIX Synergy ポートフレームワークモジュールのスタック

5.1.26.5 GUIX Synergy ポートフレームワークモジュールの構成

ユーザーは必要な動作に合わせて、GUIX Synergy ポートフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_el_gx 上の GUIX Synergy ポートフレームワークモジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
Name	g_sf_el_gx0	ISDE が生成する SF_EL_GX インスタンスの名前。このモジュールのインスタンス名を指定します。名前は有効な C シンボルである必要があります。
Name of Display Driver Run-time Configuration (Must be a valid symbol)	g_display0_runtime_cfg_bg	Synergy 構成で指定したディスプレイモジュールのランタイム設定名を指定します。名前には必ず有効な C シンボルを使用してください。NULL を設定することはできません。
Name of Frame Buffer A (Must be a valid symbol)	g_display0_fb_background[0]	フレームバッファの名前を指定します。 Synergy 設定のディスプレイモジュール設定には、作成するフレームバッファの名前が含まれています。フレームバッファの名前をここに設定します。名前には必ず有効な C シンボルを使用してください。NULL を設定することはできません。
Name of Frame Buffer B (NULL allowed if consisting a single frame buffer system)	g_display0_fb_background[1]	もう 1 つのフレームバッファの名前を指定します。シングルフレームバッファを使用するグラフィックシステムを設定する場合は、このパラメータに NULL を設定するか、またはパラメータ「フレームバッファ A の名前」を使用して同じフレームバッファ名を設定します。「単一のバッファ設計における切れ目」を参照してください。 Jlc、この値を NULL 以外に設定されたこの定義済みのタイプから変更できるかどうかが不明です
Name of User Callback function	NULL	イベント発生時にモジュールによって呼び出されたユーザーコールバック関数の名前。必ず有効な C シンボルを使用してください。 NULL も使用できます。

ISDE Property	Value	説明
Screen Rotation Angle (Clockwise)	0, 90, 180, 270 Default: 0	画面のローテーション角度（度）。0 以外の値が選択された場合、画面のローテーションが有効になり、GUIX は画面イメージを反時計回りに指定された角度回転させてフレームバッファに描画します。
GUIX Canvas Buffer (required if rotation angle is not zero)	Not used; Used Default: Not used	画面のローテーションを有効にする場合、キャンバスバッファを使用する必要があります。キャンバスバッファのサイズは、ディスプレイモジュールのフレームバッファとまったく同じでなければなりません。
Size of JPEG Work Buffer (valid if JPEG hardware acceleration enabled)	768000	JPEG 作業バッファサイズ（バイト単位）。値には必ず有効な整数値を使用してください。JPEG 高速化を使用しない場合はゼロも設定できます。バッファサイズが大きいほど描画時間が短縮されます。「JPEG 作業バッファのサイズ」を参照してください
Memory section for GUIX Canvas Buffer	sdram, bss, … Default: sdram	GUIX キャンバスバッファを割り当てるメモリセクション名。リンカースクリプトファイルで定義されている有効なセクション名を入力します。名前は有効な C シンボルである必要があります。
Memory section for JPEG Work Buffer	sdram, bss, … Default: sdram	JPEG 作業バッファを割り当てるメモリセクション名。リンカースクリプトファイルで定義されている有効なセクション名を入力します。名前は有効な C シンボルである必要があります。

***注：** 上記の設定例とデフォルトは、Synergy S7G2 ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、異なる画面サイズや入力フォーマットの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

***注：** 下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する* **[PROPERTIES]** *ウィンドウを調べることで決定されます。*

GUIX Synergy ポートフレームワークモジュールのローレベルドライバーの構成設定

通常は、ローレベルドライバーの少数の設定のみをデフォルトから変更する必要があり、これらは [Thread Stack] ブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動する

ために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表は、モジュールのプロパティセクションのすべての設定を示しています。

r_glcd 上の GLCD HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
Name	g_display0	GLCDC モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Name of display callback function to be defined by user	NULL	名前は有効な C シンボルである必要があります。
Input - Panel clock source select	Internal clock(GLCDCLK), External clock(LCD_EXTCLK) Default: Internal clock (GLCDCLK)	システムに応じてパネルクロックソースを選択します。
Input - Graphics screen1	Used, Not used Default: Used	グラフィックス画面 N を使用する場合は「使用」を指定します。これにより、グラフィックス画面 1 用のフレームバッファ「display_fb_background」と、グラフィックス画面 2 用のフレームバッファ「display_fb_foreground」が、ISDE により自動生成されます。いずれのグラフィックス画面も使用しない場合は「未使用」を指定します。その場合、フレームバッファは作成されません。「未使用」を指定した場合、フレームバッファに対するメモリ読み取りアクセスがなく、バス帯域幅の消費量が減ることに注意してください。
Input - Graphics screen1 frame buffer name	fb_background	フレームバッファのカスタム名。
Input - Number of Graphics screen1 frame buffer	2	グラフィック数の選択。

ISDE Property	Value	説明
Input - section where Graphics screen1 frame buffer allocated	sdram	フレームバッファを割り当てるセクション名を指定します。これは、[Input – Graphics screen1] が [Used] に設定されている場合に有効です。
Input - Graphics screen1 input horizontal size	800	幅のピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります
Input - Graphics screen1 vertical size	480	高さのピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input - Graphics screen1 input horizontal stride (not bytes but pixels)	800	水平ラインのメモリストライドを指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。一般に、このパラメータは、パラメータ「入力幅」と同じ数に設定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input - Graphics screen1 input format	32bits ARGB888, 32bits RGB888, 16bits RGB565, 16bits ARGB1555, 16bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16 bits RGB565	グラフィックス画面の入力形式を指定します。CLUT フォーマットを選択する場合は、clut を使用して CLUT データを書き込んでから start を実行する必要があります。デフォルト設定では、RGB565 形式の画像がサポートされます。
Input - Graphics screen1 input line descending	On, Off Default: Off	画像データが、フレームバッファの一番下のラインから一番上のラインに下降する場合に「オン」を指定します。通常は「オフ」です。
Input - Graphics screen1 input line repeat	On, Off Default: Off	LCD パネルのサイズよりも小さいラスター画像を繰り返し読み取ることが期待される場合は「オン」を指定します。通常は「オフ」です。詳細については、ラインリピート機能の説明を参照してください。
Input - Graphics screen1 input line repeat times	0	フレームに繰り返し読み込まれるラスター画像の繰り返し回数を指定します。
Input - Graphics screen1 layer coordinate X	0	グラフィックス画面の背景画面からの水平オフセットをピクセル単位で指定します。

ISDE Property	Value	説明
Input - Graphics screen1 layer coordinate Y	0	グラフィックス画面の背景画面からの垂直オフセットをピクセル単位で指定します。
Input - Graphics screen1 layer background color alpha	255	アルファ値に基づいて、グラフィックス画面 2（前景グラフィックス画面）がグラフィックス画面 1（背景グラフィックス画面）にブレンドされるか、グラフィックス画面 1 がモノクロ背景画面にブレンドされます。
Input - Graphics screen1 layer background color Red	255	グラフィックス画面 N の背景色を指定します。
Input - Graphics screen1 layer background color Green	255	グラフィックス画面 N の背景色を指定します。
Input - Graphics screen1 layer background color Blue	255	グラフィックス画面 N の背景色を指定します。
Input - Graphics screen1 layer fading control	None, Fade-in, Fade-out Default: None	グラフィックス画面のフェードインを行うには「Fade-in」を指定します。透明の画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを行うには「オフ」を指定します。不透明の画面が徐々に透明に変化します。この処理は、GLCDC ハードウェアによってアクセラレートされ、いったん開始すると停止できないことに注意してください。遷移ステータスは、statusGet で監視できます。
Input - Graphics screen1 layer fade speed	0	フェード遷移が完了するフレーム数を指定します。

ISDE Property	Value	説明
Input - Graphics screen2	Used, Not used Default: Not used	グラフィックス画面 N を使用する場合は「使用」を指定します。これにより、グラフィックス画面 1 用のフレームバッファ「display_fb_background」と、グラフィックス画面 2 用のフレームバッファ「display_fb_foreground」が、ISDE により自動生成されます。いずれのグラフィックス画面も使用しない場合は「未使用」を指定します。その場合、フレームバッファは作成されません。「未使用」を指定した場合、フレームバッファに対するメモリ読み取りアクセスがなく、バス帯域幅の消費量が減ることに注意してください。
Input - Graphics screen2 frame buffer name	fb_foreground	フレームバッファのカスタム名。
Input - Number of Graphics screen2 frame buffer	2	グラフィック数の選択。
Input - section where Graphics screen2 frame buffer allocated	sdram	フレームバッファを割り当てるセクション名を指定します。これは、[Input – Graphics screen1] が [Used] に設定されている場合に有効です。
Input - Graphics screen2 input horizontal size	800	幅のピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります
Input - Graphics screen2 vertical size	480	高さのピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input - Graphics screen2 input horizontal stride (not bytes but pixels)	800	水平ラインのメモリ ストライドを指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。一般に、このパラメータは、パラメータ「入力幅」と同じ数に設定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。

ISDE Property	Value	説明
Input - Graphics screen2 input format	32bits ARGB888, 32bits RGB888, 16bits RGB565, 16bits ARGB1555, 16bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16 bits RGB565	グラフィックス画面の入力形式を指定します。 CLUT フォーマットを選択する場合は、 clut を使用して CLUT データを書き込んでから start を実行する必要があります。デフォルト設定では、 RGB565 形式の画像がサポートされます。
Input - Graphics screen2 input line descending	On, Off Default: Off	画像データが、フレーム バッファの一番下のラインから一番上のラインに下降する場合に 「オン」を指定します。通常は 「オフ」です。
Input - Graphics screen2 input line repeat	On, Off Default: Off	LCD パネルのサイズよりも小さいラスター画像を繰り返し読み取ることが期待される場合は「オン」を指定します。通常は「オフ」です。詳細については、ライン リピート機能の説明を参照してください。
Input - Graphics screen2 input line repeat times	0	フレームに繰り返し読み込まれるラスター画像の繰り返し回数を指定します。
Input - Graphics screen2 layer coordinate X	0	グラフィックス画面の背景画面からの水平オフセットをピクセル単位で指定します。
Input - Graphics screen2 layer coordinate Y	0	グラフィックス画面の背景画面からの垂直オフセットをピクセル単位で指定します。
Input - Graphics screen2 layer background color alpha	255	アルファ値に基づいて、グラフィックス画面 2（前景グラフィックス画面）がグラフィックス画面 1（背景グラフィックス画面）にブレンドされるか、グラフィックス画面 1 がモノクロ背景画面にブレンドされます。
Input - Graphics screen2 layer background color Red	255	グラフィックス画面 N の背景色を指定します。
Input - Graphics screen2 layer background color Green	255	グラフィックス画面 N の背景色を指定します。
Input - Graphics screen2 layer background color Blue	255	グラフィックス画面 N の背景色を指定します。

ISDE Property	Value	説明
Input - Graphics screen2 layer fading control	None, Fade-in, Fade-out Default: None	グラフィックス画面のフェードインを行うには「Fade-in」を指定します。透明の画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを行うには「オフ」を指定します。不透明の画面が徐々に透明に変化します。この処理は、GLCDC ハードウェアによってアクセラレートされ、いったん開始すると停止できないことに注意してください。遷移ステータスは、statusGet で監視できます。
Input - Graphics screen2 layer fade speed	0	フェード遷移が完了するフレーム数を指定します。
Output - Horizontal total cycles	1024	水平ラインの合計サイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Horizontal active video cycles	800	水平ラインのアクティブビデオサイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Horizontal back porch cycles	46	水平ラインのバックポーチサイクル数を指定します。バックポーチは、Hsync サイクルの始点から開始します。つまり、バックポーチサイクルには Hsync サイクルが含まれます。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Horizontal sync signal cycles	20	Hsync 信号アサーションサイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。

ISDE Property	Value	説明
Output - Horizontal sync signal polarity	Low active, High active Default: Low active	システムに合わせて、Hsync 信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Vertical total lines	525	1 フレームの合計ライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Vertical active video lines	480	1 フレームのアクティブビデオライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Vertical back porch lines	23	1 フレームのバックポーチライン数を指定します。バックポーチは、Vsync ラインの始点から開始します。つまり、バックポーチラインには Vsync ラインが含まれます。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Vertical sync signal lines	10	1 フレームの Vsync 信号アサーションライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Vertical sync signal polarity	Low active, High active Default: Low active	システムに合わせて、Vsync 信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Format	24bits RGB888, 18 bits RGB666, 16 bits RGB565, 8bits serial Default: 24 bits RGB888	LCD パネルに合ったグラフィックス画面の出力形式を指定します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。

ISDE Property	Value	説明
Output - Endian	Little endian, Big endian Default: Little endian	LCD パネルへの出力信号のデータ エンディアンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Color order	RGB, BGR Default: RGB	LCD パネルへの出力信号のデータオーダーを選択します。赤と青の順序は、必要に応じて入れ替えることができます。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Data Enable Signal Polarity	Low active, High active Default: High active	システムに合わせて、データ有効信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Sync edge	Rising Edge, Falling Edge Default: Rising Edge	システムに合わせて、同期信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output - Background color alpha channel	255	背景画面の背景色を指定します。
Output - Background color R channel	0	背景画面の背景色を指定します。
Output - Background color G channel	0	背景画面の背景色を指定します。
Output - Background color B channel	0	背景画面の背景色を指定します。
CLUT	Used, Not used Default: Not used	グラフィックス画面の入力形式に CLUT 形式を選択する場合は、「使用」を指定します。その場合、CLUT ソースデータ用の「CLUT_buffer」という名前のバッファが、ISDE で自動生成されるソースファイル中に生成されます。
CLUT - CLUT buffer size	256	CLUT ソースデータバッファのエントリ数を指定します。各エントリは 4 バイト（1 ワード）を消費します。このパラメータで指定された CLUT ソースデータのワードは、ISDE で自動生成されるソースファイル中に生成されます。

ISDE Property	Value	説明
TCON - Hsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON0	システムに合わせて、Hsync 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON - Vsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON1	システムに合わせて、Vsync 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON - DataEnable pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON2	システムに合わせて、DataEnable 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON - Panel clock division ratio	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/12, 1/16, 1/24, 1/32 Default: 1/8	クロックソースの除算値を選択します。ピクセルクロックのソースクロックについては、この表の下部にある注記を参照してください。
Color correction - Brightness	Off, On Default: Off	明るさ制御を行う場合は「オン」を指定します。「オフ」を指定した場合、以下の設定は出力色に影響を与えません。
Color correction - Brightness R channel	512	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル +/ - 512。R、G、B チャネルのそれぞれの値を設定します。
Color correction - Brightness G channel	512	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル +/ - 512。R、G、B チャネルのそれぞれの値を設定します。
Color correction - Brightness B channel	512	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル +/ - 512。R、G、B チャネルのそれぞれの値を設定します。

ISDE Property	Value	説明
Color correction - Contrast	Off, On Default: Off	コントラスト制御を行う場合は「オン」を指定します。「オフ」を指定した場合、以下の設定は出力色に影響を与えません。
Color correction - Contrast(gain) R channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction - Contrast(gain) G channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction - Contrast(gain) B channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction - Gamma correction(Red)	Off, On Default: Off	各チャネルの R/G/B の制御。赤チャネルのガンマ補正を行う場合は「オン」を指定します。「オフ」を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。
Color correction - Gamma gain R[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]）<<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます:出力色レベル = 入力色レベル / 1024 (/128)。
Color correction - Gamma threshold R[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。

ISDE Property	Value	説明
Color correction - Gamma correction(Green)	Off, On Default: Off	各チャネルの R/G/B の制御。緑チャネルのガンマ補正を行う場合は 「オン」を指定します。「オフ」を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。
Color correction - Gamma gain G[0-15]	0	ガンマ補正カーブ上のエリア N 内の緑チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]） <<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。
Color correction - Gamma threshold G[0-15]	0	ガンマ補正カーブ上のエリア N 内の緑チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。
Color correction - Gamma correction(Blue)	Off, On Default: Off	各チャネルの R/G/B の制御。青チャネルのガンマ補正を行う場合は 「オン」を指定します。「オフ」を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。
Color correction - Gamma gain B[0-15]	0	ガンマ補正カーブ上のエリア N 内の青チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]） <<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。

ISDE Property	Value	説明
Color correction - Gamma threshold B[0-15]	0	ガンマ補正カーブ上のエリア N 内の青チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。
Dithering	Off, On Default: Off	ディザリング有効。出力形式 RGB666 または RGB565 を選択する場合に、ディザ効果を適用しバンディングを減らす場合は「オン」を指定します。ディザリングは変換時に適用できます。「オフ」を指定した場合、以下のディザリング設定は出力に影響を与えません。ディザ効果の詳細については、ハードウェア マニュアルの出力制御ブロック パネル ディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering - Mode	Truncate, Round off, 2x2 Pattern Default: Truncate	ディザ モードを指定します。詳細については、ハードウェア マニュアルの出力制御ブロック パネル ディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering - Pattern A	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	2X2 パターン モードのディザ パターンを指定します。詳細については、ハードウェア マニュアルの出力制御ブロック パネル ディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering - Pattern B	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	2X2 パターン モードのディザ パターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHB) を参照してください。
Dithering - Pattern C	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	2X2 パターン モードのディザ パターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHC) を参照してください。

ISDE Property	Value	説明
Dithering - Pattern D	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	2X2 パターン モードのディザ パターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロック パネルディザ補正レジスタ (OUT_PDTHD) を参照してください。
Misc - Correction Process Order	Brightness and Contrast then Gamma, Gamma then Brightness and Contrast Default: Brightness and Contrast then Gamma	必要に応じて色補正処理の順序を指定します。
Line Detect Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ドライバーには有効な割り込み優先順位設定が必要です。
Underflow 1 Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ドライバーには有効な割り込み優先順位設定が必要です。
Underflow 2 Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ドライバーには有効な割り込み優先順位設定が必要です。

sf_jpeg_decode 上の JPEG デコードフレームワークモジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効または無効にします。
Name	g_sf_jpeg_decode0	JPEG デコード フレームワーク モジュール インスタンスに使用する名前。

r_jpeg 上の JPEG デコード HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_jpeg_decode0	JPEG デコードモジュールインスタンスに使用する名前。

ISDE Property	Value	説明
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Byte-Swap (8)(7)(6)(5)(4)(3)(2)(1) Default: Normal Byte order	入力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。

ISDE Property	Value	説明
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Byte-Swap (8)(7)(6)(5)(4)(3)(2)(1) Default: Normal Byte order	出力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format Default: Pixel Data RGB565 format	出力データ フォーマットを指定します。
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	出力データフォーマットのアルファ値を指定します（ARGB8888 フォーマットに対してのみ有効）。
Name of user callback function	NULL	ユーザーコールバック関数の名前を指定します。

ISDE Property	Value	説明
Decompression Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	解凍割り込み優先順位の選択。
Data Transfer Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	データ転送割り込み優先順位の選択。

dave2d 上の D/AVE 2D ドライバーの構成

ISDE Property
No configurable settings

sf_tes_2d_drw 上の D/AVE 2D ポートの構成

ISDE Property	Value	説明
Work memory size for display lists in bytes	32,768	表示リストの作業メモリサイズの選択
DRW Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	DRW INT の選択

注: 上記の設定例とデフォルトは、Synergy S7G2 ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GUIX Synergy ポートフレームワークモジュールのクロック構成

GUIX Synergy ポートモジュールは、論理モジュールであるため、Arm Cortex-M コア SysTick タイマ設定を除くハードウェア設定は不要です。

GUIX Synergy ポートフレームワークモジュールのピン構成

GUIX Synergy ポートモジュールは、論理モジュールであるため、ピン設定は不要です。

5.1.26.6 アプリケーションでの GUIX フレームワークモジュールまたは JPEG デコード HAL モジュールの使用

アプリケーションで GUIX フレームワークモジュールを使用する際の一般的な手順は次のとおりです（GUIX ドライバーのインスタンスが g_sf_el_ux0 で、LCD の SPI ドライバーのインスタンスが g_rspi_lcdc): であると想定しています）。

手順 0: gx_system_initialize 関数で GUIX を初期化する

手順 1: open API (g_sf_el_ux0.p_api -> open) を使用して GUIX ドライバーを初期化する

手順 2: setup API を使用して gx_studio_display_configure 関数で GUIX システムを構成する

手順 3: canvasInit API (g_sf_el_ux0.p_api -> canvasInit) でキャンバスのメモリアドレスを初期化する

手順 4: gx_studio_named_widget_create 関数でプライマリ画面を作成する

手順 5: gx_widget_show 関数を使用してルートウィンドウを表示する

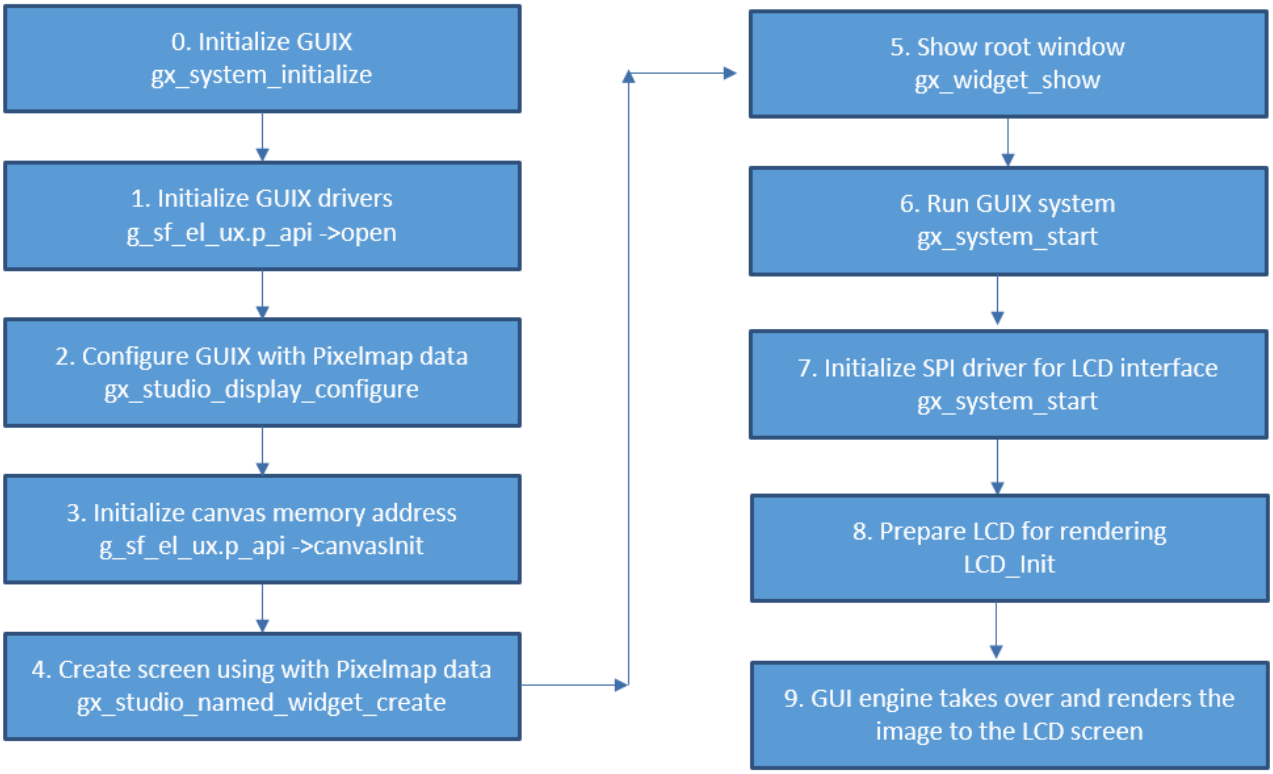
手順 6: gx_system_start 関数で GUIX システムを起動する

手順 7: gx_system_start API で LCD ディスプレイとの SPI インタフェースを初期化する

手順 8: LCD_Init 呼び出しでで LCD ハードウェアを初期化する

手順 9: GUI エンジンが引き継ぎ、 LCD 画面にイメージをレンダリングする

これらの一般的な手順を、次の図の通常の動作フロー図に示します。



JPEG デコード HAL モジュールを構成してファイルが生成されたら、JPEG デコードをアプリケーションで使用する準備ができます。アプリケーションで JPEG デコーダ HAL モジュールを使用する際の一般的な手順では、 open API を使用した JPEG デコードを初期化し、水平ストライド、イメージサブサンプル、入力バッファ、出力バッファを構成します。入力バッファと出力バッファが設定されると JPEG コーデックによってデコード操作がトリガされてデコードされたイメージが出力バッファに格納され、 statusGet API を使用して JPEG 操作のステータスをポーリングできます。

アプリケーションで JPEG デコード HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して JPEG デコード HAL モジュールを初期化します。
2) horizontalStrideSet API を使用して水平ストライドを設定します。
3) imageSubsampleSet API を使用して垂直および水平イメージサブサンプルを設定します。
4) inputBufferSet API を使用して入力バッファアドレス（JPEG イメージを含む）を設定します。
5) outputBufferSet API を使用して出力バッファを設定します（未加工イメージデータを十分に保持できる大きさである必要があります）。
6) statusGet API を使用して JPEG 操作を取得でき、statusGet API は列挙値（上記で説明）を返してユーザーに通知します。statusGet API からの状態 JPEG_DECODE_STATUS_DONE は、デコード操作が完了したことを示します。
7) 受信した未加工イメージデータをアプリケーションの必要に応じて操作します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

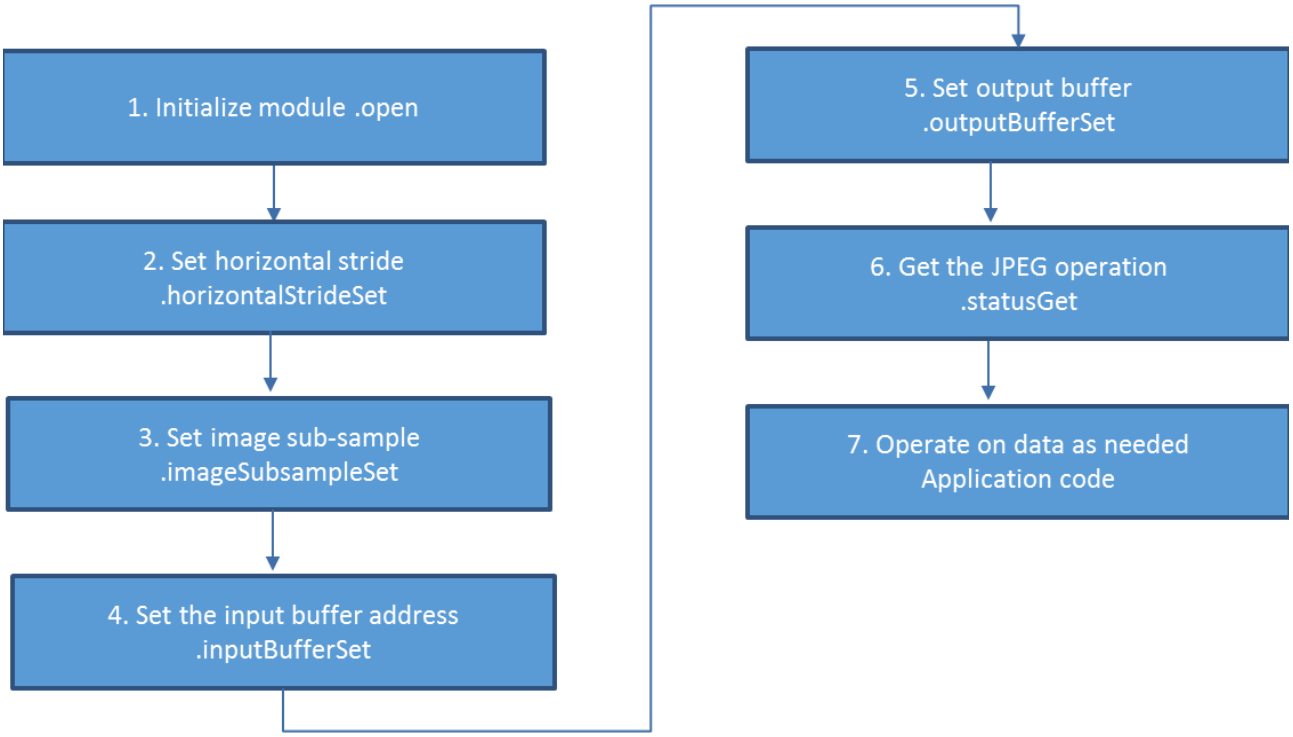


図 272: 通常の JPEG デコード HAL モジュールアプリケーションのフロー図

5.1.27 Express Logic 社の NetX ポート ETHER モジュール

- NetX サービスはライブラリとして実装されているので、必要なコードだけがプロジェクトに追加されます。
  - これにより、ほとんどのアプリケーションでは、命令イメージは 5k バイトから 30k バイトであり、IPV6 および ICMPv6 を有効にするとサイズは 30k バイトから 45k バイトです。
- NetX は次のようなさまざまな RFC をサポートしています。
  - RFC 1112 IP マルチキャスト用ホスト拡張機能 （IGMPv1）
  - RFC 1122 インターネットホストに関する要件 - 通信レイヤー
  - RFC 2236 インターネットグループ管理プロトコル、バージョン 2
  - RFC 768 ユーザーデータグラムプロトコル (UDP)
  - RFC 791 インターネットプロトコル (IP)
  - RFC 792 インターネット制御通知プロトコル (ICMP)
  - RFC 793 伝送制御プロトコル (TCP)
  - RFC 826 イーサネットアドレス解決プロトコル （ARP）

– RFC 903 逆アドレス解決プロトコル (RARP)
– RFC 2460 インターネットプロトコル v6 （IPv6）仕様 （NetX Duo のみ）
– RFC 4443 インターネット制御通知プロトコル （ICMPV6）（NetX Duo のみ）
– RFC 4861 IPv6 近隣探索 （NetX Duo のみ）
– RFC 4862 IPv6 ステートレスアドレス自動構成 （NetX Duo のみ）

- TCP/IP のパケットベースのゼロコピー実装 （バッファはスタック内またはスタックとユーザーアプリケーションの間を移動するので NetX の内部ではコピーされず、たとえばメモリや処理サイクルが解放されて、送信速度が大幅に向上します）
- ファスト UDP 処理

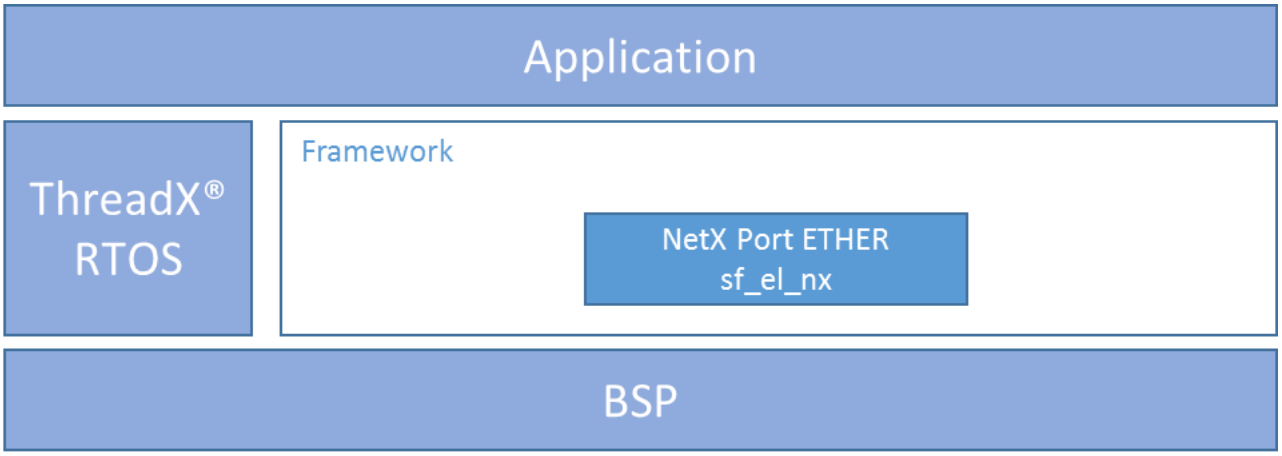


図 273: NetX ポート ETHER モジュールの編成、オプション、スタックの実装

5.1.27.1 NetX ポート ETHER モジュールの API

NetX ポート ETHER モジュールには狭い範囲の API があり、NetX およびモジュール自体によって使用されます。イーサネットドライバーのエントリポイント (nx_ether_driver_eth0, nx_ether_driver_eth1), イーサネット割り込みハンドラ、モジュールによって内部的に使用されるが外部から見ることのできる他の関数などが含まれます。

5.1.27.2 NetX ポート ETHER モジュールの動作の概要

NetX ポート ETHER モジュールは、 Renesas Synergy ソフトウェアおよび Synergy イーサネット IP 用の NetX イーサネットドライバーの高性能なリアルタイムの実装です。

***注:** NetX は ThreadX の存在を前提にしており、そのスレッド実行、サスペンション、定期タイマ、相互排除の機能に依存します。*

ネットワークドライバー

NetX の各 IP インスタンスには、nx_ip_create サービスで指定されているデバイスドライバーによって識別されるプライマリインタフェースがあります。ネットワークドライバーは、パケット送信、パケット受信、状態と制御の要求など、 NetX のさまざまな要求を処理します。

マルチホームシステムの場合、複数のインタフェース用に IP インスタンスを構成することができ、それぞれのインタフェースに対するこれらのタスクを実行するネットワークドライバーがインタフェースごとに関連付けられます。ネットワークドライバーは、メディアで発生する非同期イベントの処理も行う必要があります。メディアで発生する非同期イベントとしては、パケットの受信、パケットの送信の完了、状態の変化などがあります。NetX には、さまざまなイベントを処理するための複数のアクセス関数を含むネットワークドライバーが用意されています。これらの関数は、ネットワークドライバーの割り込みサービスルーチン部分から呼び出されるように設計されています。IP ネットワークのネットワークドライバーは、受信したすべての ARP パケットを _nx_arp_packet_deferred_receive 内部関数に転送する必要があります。すべての RARP パケットは、_nx_rarp_packet_deferred_receive 内部関数に転送される必要があります。IP パケットに関しては 2 つのオプションがあります。

IP パケットの高速なディスパッチが必要な場合は、受信 IP パケットは _nx_ip_packet_receive に転送されてすぐに処理されます。これにより、 NetX による IP パケットの処理性能が大きく向上します。それ以外の場合は、IP パケットは _nx_ip_packet_deferred_receive. に転送されます。このサービスは、IP パケットを遅延処理キューに格納します。このキューの IP パケットは内部 IP スレッドによって処理されて、ISR の処理時間が最小限になります。

ネットワークドライバーは、割り込み処理を延期して、IP スレッドのコンテキスト外で実行することもできます。このモードでは、 ISR は必要な情報を保存し、内部関数 _nx_ip_driver_deferred_processing, を呼び出して、割り込みコントローラにアクノリッジする必要があります。このサービスは、割り込みを発生させる、デバイスドライバーに対するイベント処理完了のコールバックをスケジュールするように、IP スレッドに通知します。

一部のネットワークコントローラは、貴重な CPU リソースを使わずに、ハードウェアで TCP/UDP/IP のヘッダーチェックサムの計算と検証を実行する機能を備えています。ハードウェアの機能を利用できるよう、NetX には、コンパイル時にさまざまなソフトウェアチェックサム計算を有効または無効にするオプション、および実行時にチェックサム計算をオンまたはオフにするオプションがあります。NetX ネットワークドライバーの作成方法の詳細については、『NetX ユーザーガイド』の 「NetX ネットワークドライバー」セクションを参照してください。

NetX ポート ETHER モジュールの動作についての注意

以下は動作に関する重要な注意事項の要約です。これらの各トピックの詳細については、『*NetX User Guide for the Renesas Synergy™ Platform*』および 『*NetX Duo User Guide for the Renesas Synergy™ Platform*』を参照してください。以下の各機能の NetX ソースのプロパティに加えて、ソースコードシンボルが提供されています。たとえば、物理ネットワークインタフェースの数を変更するには、NetX ソース要素または NetX Duo ソース要素の *Maximum Physical Interfaces* プロパティを設定するか、ソースコードシンボル NX_MAX_PHSYCIAL_INTERFACES を直接定義することができます。いずれの場合も、 NetX および NetX Duo のソースコンポーネントを組み込み、プロジェクトファイルを生成して、NetX ライブラリを再ビルドする必要もあります。

NetX のソースレベルの設定

以下のオプションは、 NetX および NetX Duo のソースコンポーネントにあります。

- **Multiple Network Interface Support**

NetX は、 1 つの IP インスタンスを使用して複数の物理デバイスに接続されているシステムをサポートします。各物理インタフェースは、IP インスタンスのインタフェース制御ブロックに割り当てられます。マルチ

ホームシステムを使用するアプリケーションでは、*[Maximum Physical Interfaces]* プロパティ（または NX_MAX_PHSYCIAL_INTERFACES) の値にシステムに接続される物理デバイスの数を設定し、NetX ライブラリを再ビルドする必要があります。ライブラリを再ビルドしないと有効になりません。デフォルト値は 1 です。

- **Loopback Interface**

ループバックインタフェースは、物理リンクが接続されていない特殊なネットワークインタフェースです。ループバックインタフェースを使用すると、アプリケーションは IP ループバックアドレス 127.0.0.1 を使用して通信できます。論理ループバックインタフェースを利用するには、[Loopback] プロパティを有効（デフォルト設定）に設定するか、シンボル NX_DISABLE_LOOPBACK_INTERFACE を定義されていない状態にします。

- **Enabling IPv6（NetX Duo のみ）**

NetX Duo の IP インスタンスでは、IPv6 がデフォルトで有効になります。NetX Duo のソースコンポーネントで *[NetX Duo IPV6 Support]* プロパティを設定することで、IP インスタンスの IPv6 を無効または有効にできます。IPv6 関連の他のプロパティを有効にすることもできます（ICMPv6 パケットでのチェックサム、重複アドレスの検出など）。IPv6 ネットワークの詳細については、『*NetX Duo User Guide for the Renesas Synergy™ Platform*』を参照してください。

[Interface Control Blocks]: IP インスタンスのインタフェース制御ブロックの数は、物理インタフェースの数（NX_MAX_PHYSICAL_INTERFACES) で定義されている値）に、有効になっている場合はループバックインタフェースを加えた値です。インタフェースの合計数はシンボル NX_MAX_IP_INTERFACES. で定義されています。この値は直接変更**しないでください**。NetX は、NX_MAX_PHSYCIAL_INTERFACES と NX_DISABLE_LOOPBACK_INTERFACE の設定に基づいて、このシンボルを内部的に定義します。

NetX ポート ETHER の設定

- **チャネル**

これは、イーサネットドライバーが適用されるインタフェースを決定します。デフォルト値の 0 を変更することが必要な場合があります。セクション 5「NetX ポート ETHER の構成」を参照してください。

- **Channel 0/1 PHY Reset Pin**

このプロパティは、上のチャネルプロパティの値に基づいて設定されます。デフォルト値の変更が必要な場合があります。セクション 5「NetX ポート ETHER の構成」を参照してください。

- **Channel 0/1 MAC Address High Bits**
- **Channel 0/1 MAC Address Low Bits**

これらのプロパティは、コンパイル時に該当するチャネルインタフェースのデバイス MAC アドレスを設定します。実行時に MAC アドレスを設定するには、後の *[Callback]* プロパティの説明を参照してください。

- **コールバック**

このプロパティは、実行時に MAC アドレスを設定するユーザー定義のコールバック関数を定義します（ネットワークリンクの初期化）。デフォルト値は NULL であり、MAC アドレスはチャネルの MAC アドレスの高ビットおよび低ビットの設定を使用して割り当てられます。

- **名前**

このプロパティは、NetX ポート ETHER ドライバーのインスタンスの名前を指定します。複数のネットワークインタフェースが構成されている IP インスタンスでは、アプリケーションはドライバーのインスタンスを

追加して、一意の名前を指定する必要があります（そうしないと、コンパイルエラーになります）。デフォルトの名前は g_sf_el_nx. です。

- **イーサネット割り込みの優先順位**

このプロパティは、ドライバーの割り込み優先順位を設定します。デフォルトのプライオリティは 4 です。ドロップダウンリストでは、MCU ターゲットに基づいて有効なプライオリティと無効なプライオリティが示されます。

- **受信バッファ記述子の数**
- **送信バッファ記述子の数**

これらのプロパティは、パケットを受信および送信するためのバッファ記述子（BD）の数を設定します。受信 BD を初期化するときに、ドライバーは各 BD に対してパケットを割り当てます。したがって、受信 BD の数は、パケット割り当て元の IP インスタンスパケットプールを使い切らない値にする必要があります。

NetX ポート ETHER ドライバーは、受信パケットと送信パケットの両方について、パケットチェーンをサポートします（パケットがアプリケーションレイヤーでチェーンされている場合）。IP のデフォルトパケットプールペイロードのサイズを超えるパケットをネットワークで受信した場合、ドライバーは追加のパケットを割り当てて、パケットチェーンとして受信パケットを処理できます。

- **パラメータチェック**

これは、プロジェクトハードウェアレイヤーの各コンポーネントに対する低レベルのエラーチェックです。デフォルトでは [Default (BSP)] に設定され、これはこのプロパティが BSP の同じプロパティを継承することを意味します。

NetX ポート ETHER モジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.27.3 アプリケーションへの NetX ポート ETHER モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX ポート ETHER モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

アプリケーションの NetX ポート ETHER フレームワークを使用するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。別のコンポーネントがプロジェクトコンフィギュレータで既に追加されていて、ドライバーが必要な場合は、ドライバーを選択する必要があることを示すピンクの帯が付いた [Add NetX Network Driver]（NetX Duo プロジェクトの場合は [Add NetX Duo Network Driver]）が表示されます。

次の表で示すように、NetX ポート ETHER フレームワークのデフォルトの名前は g_sf_el_nx です。この名前は、対応する [Properties] ウィンドウで変更できます。次の図に示すように、ドライバーが追加されるとスレッドに表示されます。

NetX ポート ETHER フレームワークの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_nx NetX Port ETHER on sf_el_nx	**Threads**	**New Stack> Framework> Networking> NetX Port ETHER on sf_el_nx**

次の図で示すように、NetX ポート ETHER フレームワークがスレッドスタックに追加されるとき、低レベルのモジュールは必要ありません。このモジュールは NetX システムの基礎となるものなので、役に立つように、常に NetX と組み合わせて使用されます。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。

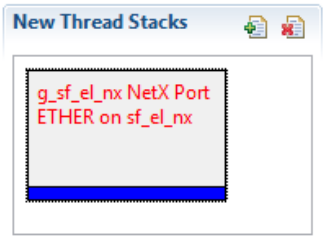


図 274: NetX ポート ETHER フレームワークのスタック

以下のように構成すると、NetXおよびNetX DuoのAutoIPは、IPインスタンスのパケットプールg_packet_pool0. を自動的に使用します。ドライバーは、 NetX ポート ETHER ドライバー g_sf_el_nx. に設定されます。

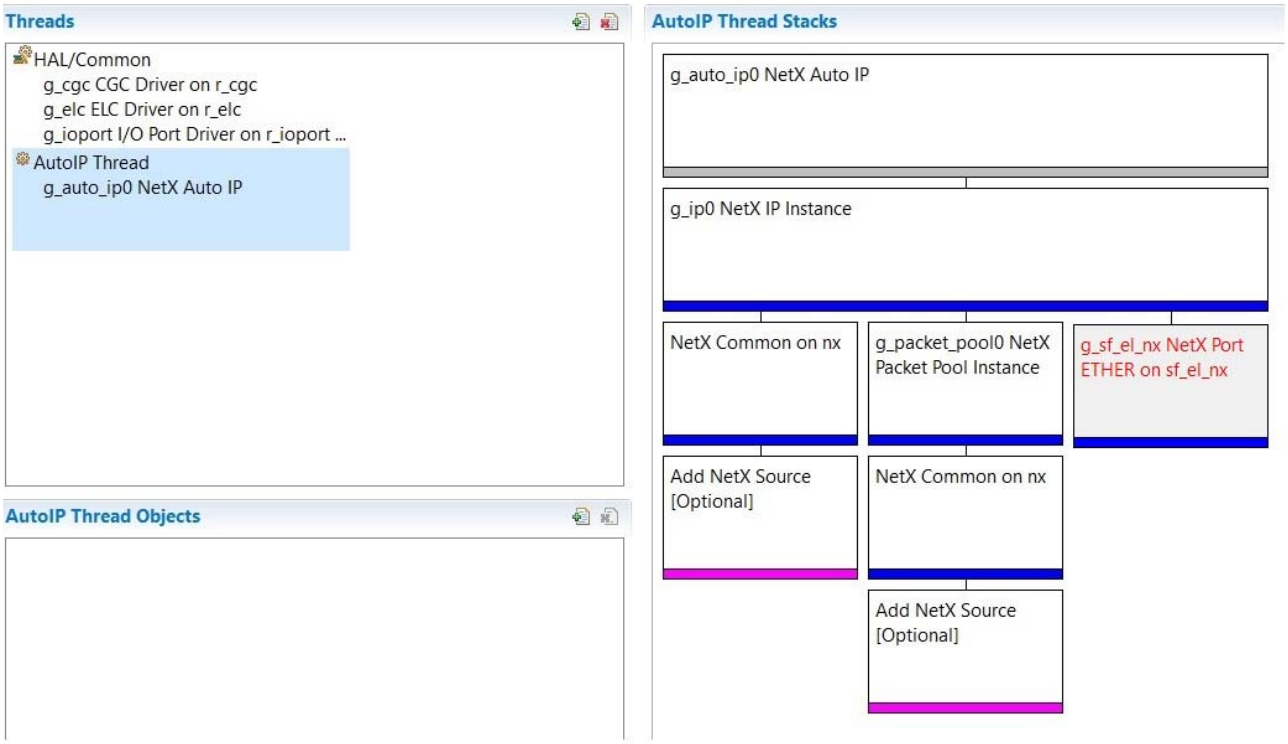


図 275: NetX の AutoIP を使用する NetX ポート ETHER アプリケーション

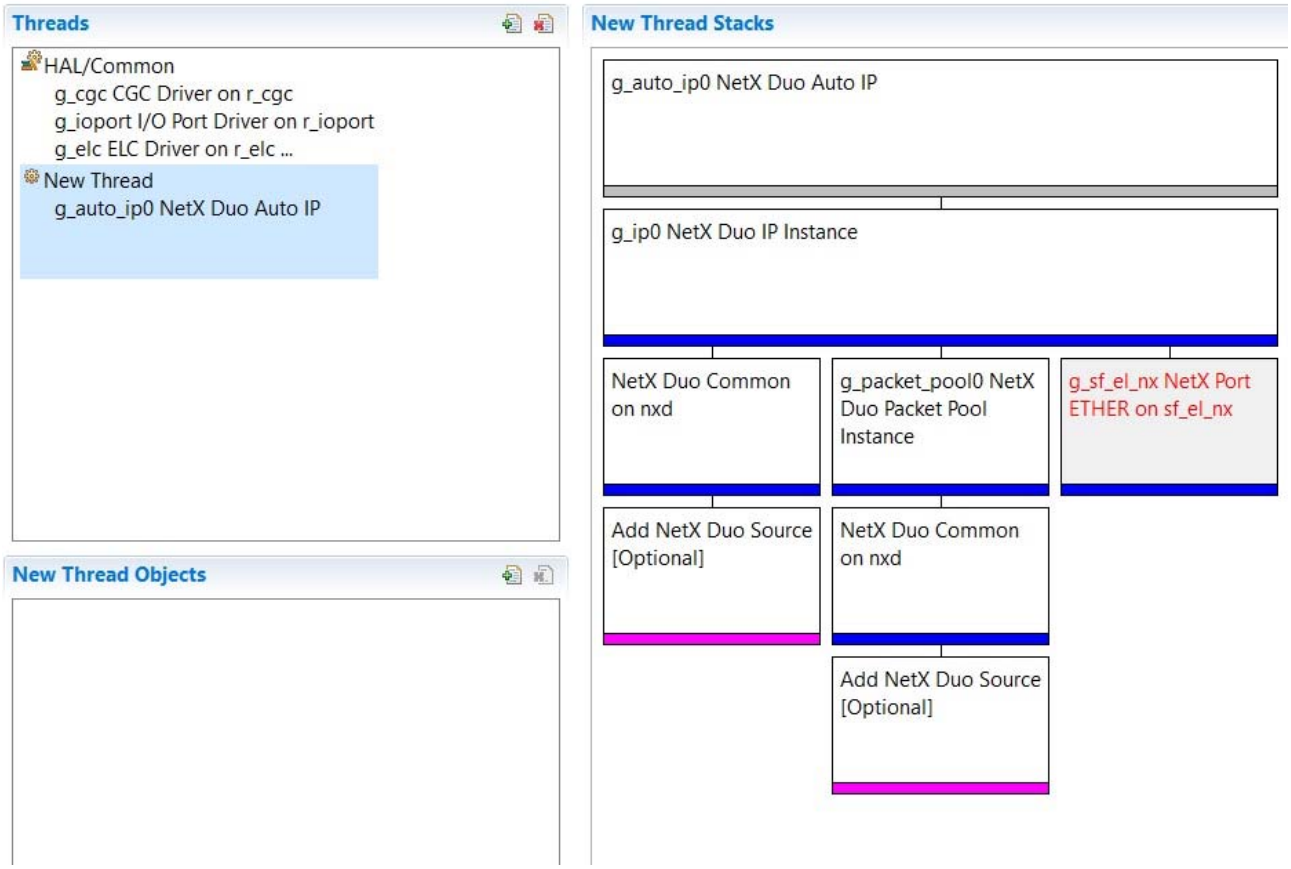


図 276: NetX Duo の AutoIP を使用する NetX ポート ETHER アプリケーション

5.1.27.4 NetX ポート ETHER フレームワークモジュールの構成

ユーザーは必要な動作に合わせて NetX ポート ETHER フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、低レベルモジュールが正常に作動するために構成する必要のあるコンポーネントが、自動的に識別されます（ブロックが赤で強調表示）。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。こ

のレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_el_nx での NetX ポート ETHER モジュールのデフォルト構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Channel 0 Phy Reset Pin	IOPORT_PORT_09_PIN_03	チャネル 0 Phy リセットピンの選択
Channel 0 MAC Address High Bits	0x00002E09	チャネル 0 MAC アドレス高ビットの選択
Channel 0 MAC Address Low Bits	0x0A0076C7	チャネル 0 MAC アドレス低ビットの選択
Channel 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	チャネル 1 Phy リセットピンの選択
Channel 1 MAC Address High Bits	0x00002E09	チャネル 1 MAC アドレス高ビットの選択
Channel 1 MAC Address Low Bits	0x0A0076C8	チャネル 1 MAC アドレス低ビットの選択
Number of Receive Buffer Descriptors	8	受信バッファ記述子の数の選択
Number of Transmit Buffer Descriptors	32	送信バッファ記述子の数の選択
Ethernet Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Priority 5)	イーサネット割り込み優先順位の選択
Name	g_sf_el_nx	モジュール名
Channel	0	チャネルの選択
Callback	NULL	コールバックの選択

注: *上記の設定例とデフォルトは、Synergy S7G2 ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

5.1.27.5 アプリケーションでの NetX ポート ETHER モジュールの使用

NetX ポート ETHER モジュールは NetX と組み合わせて使用する必要があります。NetX アプリケーションをサポートするために Synergy が自動的に実行する手順は以下のとおりです。

1) システムを nx_system_initialize. で初期化します。
2) nx_packet_pool_create. でパケットプールを作成します。これにより、IP インスタンスとイーサネットドライバーによって使用されるデフォルトの IP パケットプールが作成されます。
3) nx_ip_create. で IP インスタンスを作成します。
4) nx_arp_enable. で ARP を有効にします。
5) nx_auto_ip_create. で AutoIP インスタンスを作成します。

AutoIP スレッドのタスクをセットアップして実行するには、アプリケーションで以下の手順を直接実行します。

1) nx_ip_interface_status_check API を使用して、ネットワークリンクが有効になっていることを確認します。
2) nx_ip_interface_address_get API を呼び出して、デバイスが IP アドレスを持っていないことを確認します。
3) nx_ip_address_change_notify API を呼び出して、IP アドレス通知コールバックを設定します。
4) nx_auto_ip_start API を使用して、AutoIP インスタンスを開始します。
5) IP インスタンスに IP アドレスが割り当てられたことを示すフラグを IP アドレス変更コールバックが設定するのを待ちます。
6) nx_auto_ip_stop API を呼び出して、AutoIP タスクを停止します。
7) nx_ip_interface_address_get API を再び呼び出して、IP インスタンスにゼロではない IP アドレスが設定されていることを確認します。

上記の一般的な手順を、次の図の動作フロー図に示します。

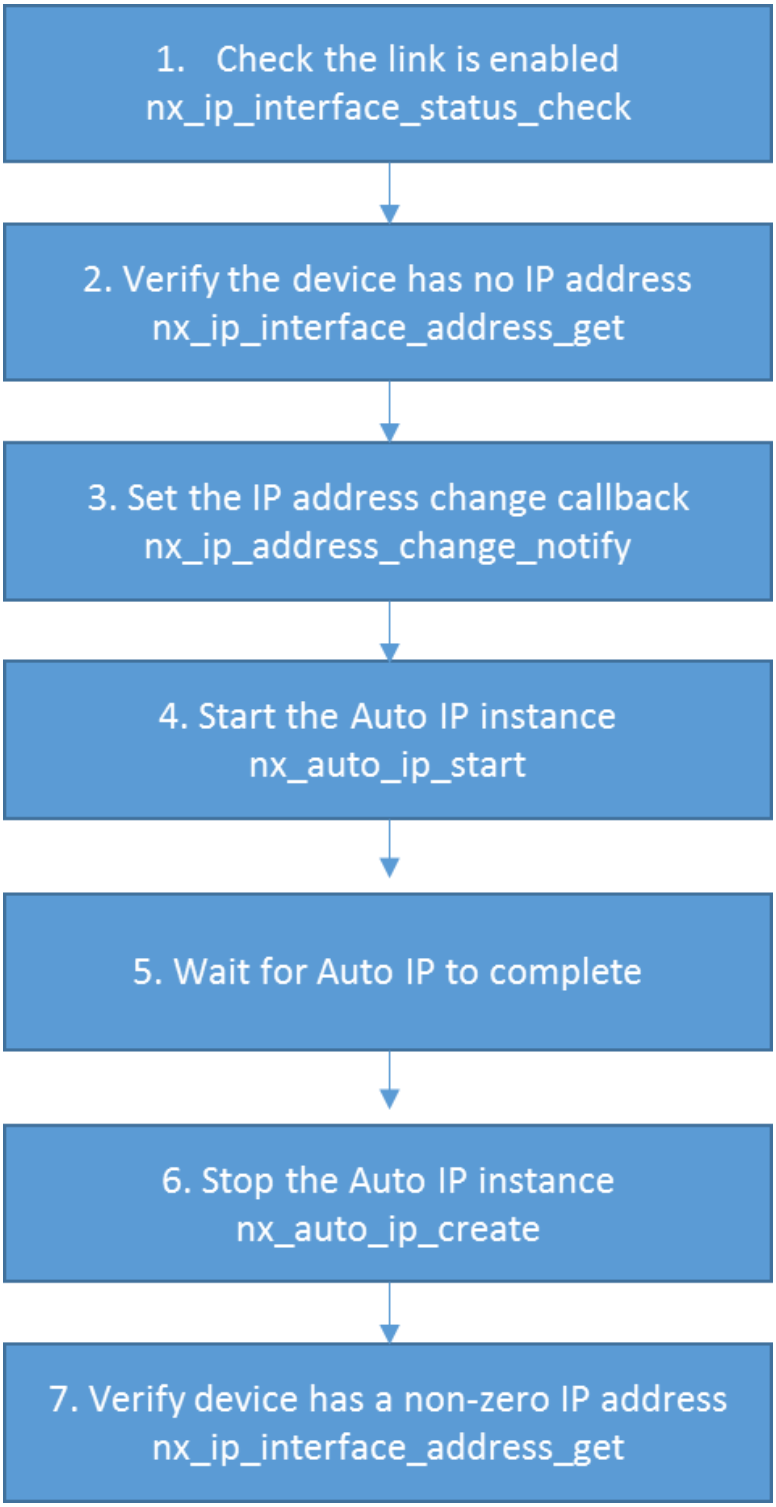
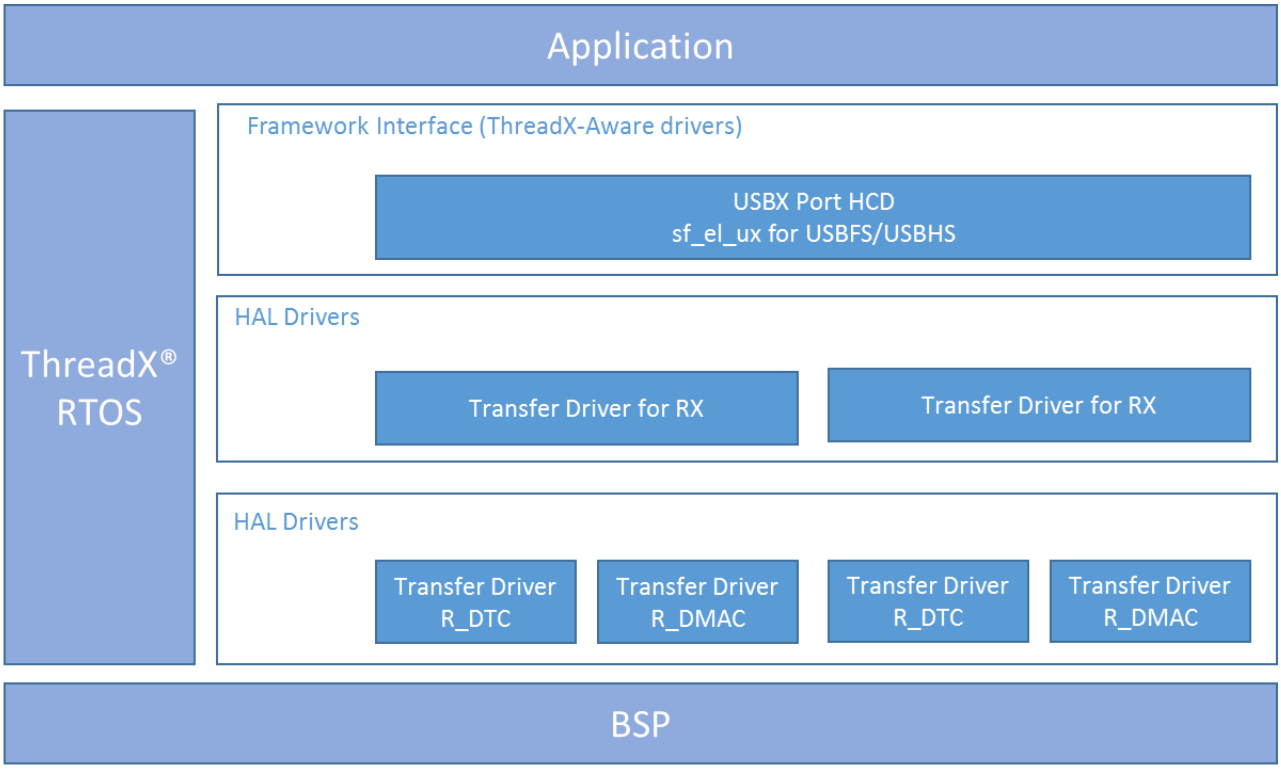


図 277: AutoIP を使用する簡単な NetX アプリケーションのフロー図

5.1.28 Express Logic 社の USBX Synergy ポートフレームワーク

Express Logic 社の USBX Synergy ポートフレームワークモジュールは以下の機能に対応します。

- Express Logic USBX を SSP サポート USBX API に実装します
- USBHS 用のポートデバイスコントローラドライバー（DCD）をサポートします
- USBFS 用のポートデバイスコントローラドライバー（DCD）をサポートします
- USBHS 用のポートホストコントローラドライバー（HCD）をサポートします
- USBFS 用のポートホストコントローラドライバー（HCD）をサポートします
- 転送モジュールの動作をサポートします（オプション）



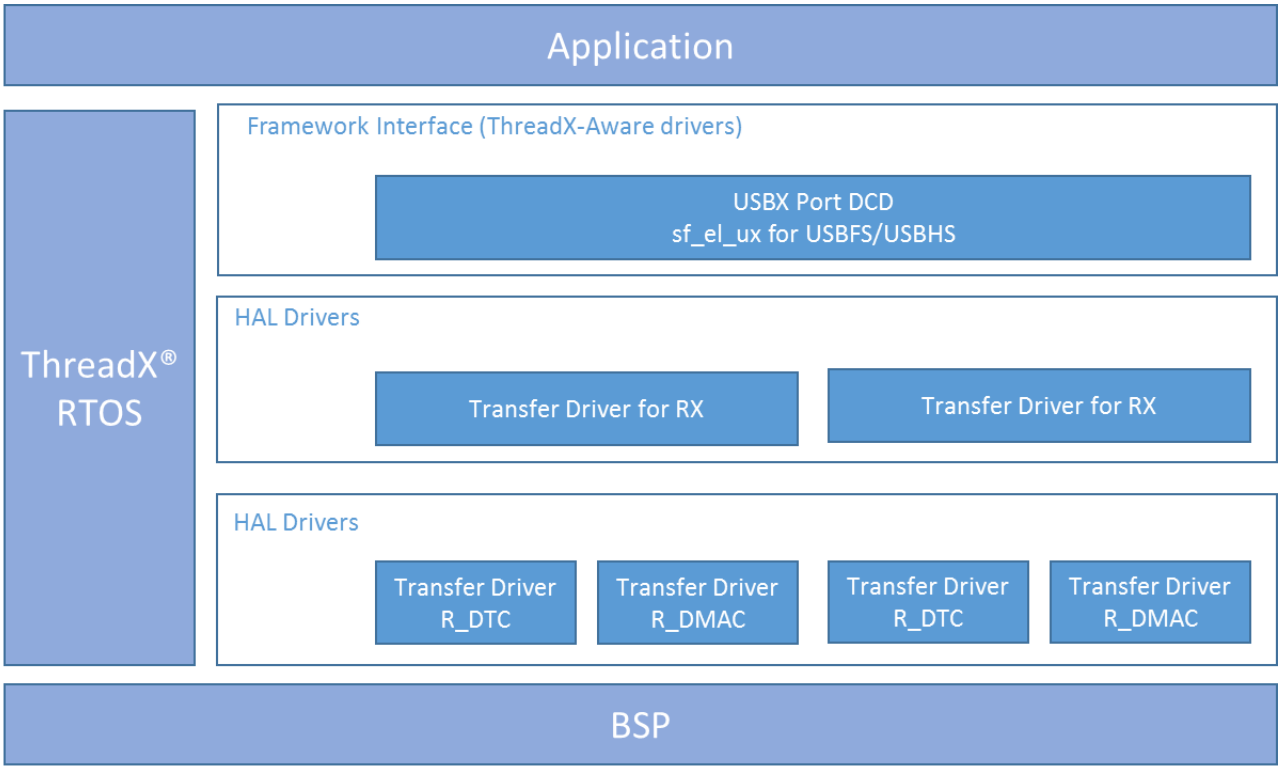


図 278: Express Logic USBX Synergy ポートフレームワークの編成、オプション、スタックの実装

5.1.28.1 Express Logic USBX Synergy ポートフレームワークモジュール API の概要

Express Logic USBX Synergy ポートフレームワークモジュール自体は API コールを持たず、Express Logic USBX API コールの API コールを実装しています。これらの API についてのドキュメントは、このドキュメントの最後の「参考文献」セクションの説明に従って入手できる『Express Logic USBX User Manual』で参照できます。

5.1.28.2 Express Logic USBX Synergy ポートフレームワークモジュールの動作の概要

Express Logic USBX Synergy ポートフレームワークモジュールは、Synergy ハードウェアの USBX スタックを使用するために必要な Synergy USB ハードウェアポート機能を提供します。このモジュールを使用したアプリケーションコードは、USBX API コールの使用を想定したものとなっています。

Express Logic USBX Synergy ポートフレームワークモジュールの動作に関する重要な注意事項と制限事項

Express Logic USBX Synergy ポートフレームワークモジュールには、SSP における Express Logic USBX API のサポートが含まれます。使用可能な API の詳細な説明については、『Express Logic USBX User Manual』を参照してください。

Express Logic USBX Synergy ポートフレームワークモジュールは、USBHS および USBFS のポートデバイスコントローラドライバー（DCD）と、USBHS および USBFS のポートホストコントローラドライバー（HCD）をサポートします。

ユーザーは、USBX Synergy ポートフレームワークモジュールの転送モジュールを使用して、ブロック転送モードでデータを転送することにより、USB のデータスループットを向上させることができます。転送モジュールを有効にするには、 Synergy 構成ツールで 2 つのコンポーネントインスタンスを USBX クラススタックに追加し、プロパティで割り込みを有効にするだけです。Synergy 構成ツールは、ドライバーセットアップコードを自動生成して、 common_data.c. での DMAC 転送または DTC 転送を有効にします。

- このモジュールは、USB コントローラの割り込みを使用します。Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。設定しないとモジュールは機能しません。
- このモジュールが使用されると、転送モジュール （DMAC または DTC として実装）の割り込みが使用されます。Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。優先順位レベルは USB コントローラよりも高くします。設定しないとモジュールは機能しません。
- USBX ドライバーの現在のバージョンはアイソクロナス転送をサポートしていません。そのため、アイソクロナス転送を使用する USBX クラスはサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.1.28.3 アプリケーションへの Express Logic USBX Synergy ポートフレームワークモジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションにオーディオ再生フレームワークモジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』の最初のいくつかのセクションを参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

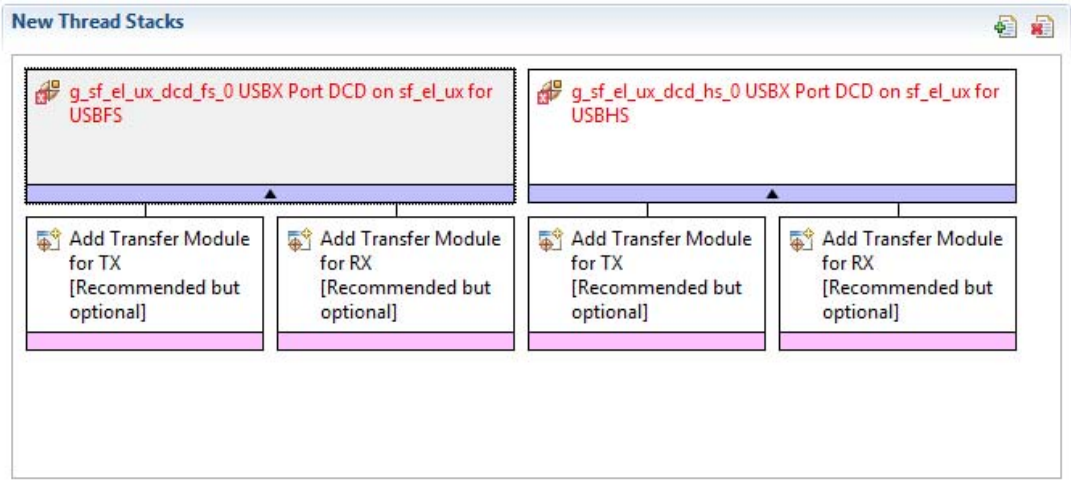
USBX Synergy ポートをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（USBX Synergy ポートのデフォルト名は g_sf_el_ux. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

USBX Synergy ポートの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_ux_hcd_hs_0 USBX Port HCD on sf_el_ux for USBHS	Threads	New Stack> X-Ware> USBX> Host > Synergy Port> USBX Port HCD on sf_el_ux for USBHS
g_sf_el_ux_hcd_fs_0 USBX Port HCD on sf_el_ux for USBFS	Threads	New Stack> X-Ware> USBX> Host > Synergy Port> USBX Port HCD on sf_el_ux for USBFS
g_sf_el_ux_dcd_hs_0 USBX Port HCD on sf_el_ux for USBHS	Threads	New Stack> X-Ware> USBX> Device > Synergy Port> USBX Port HCD on sf_el_ux for USBHS

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_ux_dcd_fs_0 USBX Port HCD on sf_el_ux for USBFS	Threads	New Stack> X-Ware> USBX> Device > Synergy Port> USBX Port HCD on sf_el_ux for USBFS

次の図に示すように、sf_el_ux の USBX Synergy ポートフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、ブロック内でテキストによって示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。



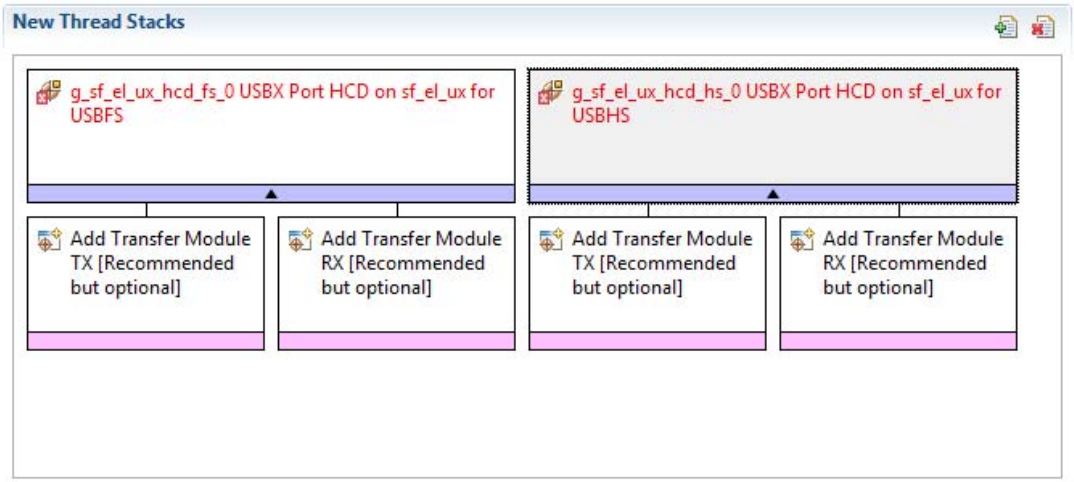


図 279: Express Logic USBX Synergy ポートフレームワークのスタック

5.1.28.4 Express Logic USBX Synergy ポートフレームワークモジュールの構成

ユーザーは必要な動作に合わせて、USBX Synergy ポートフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

sf_el_ux の Express Logic USBX ポート DCD の USBFS 用の構成設定

ISDE Property	Value	説明
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フルスピードの割り込みの優先順位レベルを設定します
LDO Regulator (Only for S3 and S1 MCUs)	Enable, Disable (Default: Disable)	USB LDO レギュレータが使用されている場合、プロパティを [Enable] に設定します。詳細については、S3 または S1 MCU グループのハードウェアマニュアルのUSBFSに関するセクションを参照してください。
Name	g_sf_el_ux_dcd_fs_0	モジュール名。
USB Controller Selection	USBFS	USB コントローラタイプ

sf_el_ux の Express Logic USBX ポート DCD の USBHS 用の構成設定

ISDE Property	Value	説明
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フルスピードの割り込みの優先順位レベルを設定します
Name	g_sf_el_ux_dcd_hs_0	モジュール名。
USB Controller Selection	USBHS	USB コントローラタイプ

sf_el_ux の Express Logic USBX ポート HCD の USBFS 用の構成設定

ISDE Property	Value	説明
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	高速割り込みの優先順位レベルを設定します
VBUSEN pin Signal Logic	Active High, Active Low (Default: Active High)	VBUS ピンレベルの設定
Name	g_sf_el_ux_hcd_fs_0	モジュール名。
USB Controller Selection	USBFS	USB コントローラタイプ

sf_el_ux の Express Logic USBX ポート HCD の USHFS 用の構成設定

ISDE Property	Value	説明
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	高速割り込みの優先順位レベルを設定します
VBUSEN pin Signal Logic	Active High, Active Low (Default: Active High)	VBUS ピンレベルの設定

ISDE Property	Value	説明
Name	g_sf_el_ux_dcd_fs_0	モジュール名。
USB Controller Selection	USBFS	USB コントローラタイプ

注: *例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *低レベルモジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。*

Express Logic USBX Synergy ポートフレームワークの低レベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

注 : 低レベルドライバーは 1 コピーだけ提供されます。転送ドライバーの他の 3 つのセットは提供されているものと非常によく似ており、RX と TX の実装でわずかに異なるだけです。違いを確認するには、SSP 構成ウィンドウのプロパティを見てください。

r_dmac 上の転送ドライバーの構成 （転送ドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer0	モジュール名
Channel	0	チャネルの選択
Mode	Block	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。

ISDE Property	Value	説明
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source	Software Activation	アクティベーションソースの選択
Auto Enable	False	自動有効の選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

r_dtc 上の転送ドライバーの構成設定 （転送ドライバーオプション）

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択

ISDE Property	Value	説明
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Block	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation 1	アクティベーションソースの選択
Auto Enable	False	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

***注:** 例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

Express Logic USBX Synergy ポートフレームワークのクロック構成

USB 周辺モジュールは、UCLK をそのクロックソースとして使用します。クロック周波数は 48MHz に設定する必要があり、これは構成ウィンドウの [Clocks] タブで行うことができます。

Express Logic USBX Synergy ポートフレームワークのピン構成

USB 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では SSP 構成ウィンドウでの（必要な USB 機能が USBFS または USBHS のどちらであるかに応じた）ピンの選択方法を示し、その後の表では各機能での USB ピンの選択例を示します。

注 : 動作モードによって、使用可能なペリフェラル信号が決まり、したがって必要な MCU ピンが決まります。動作モードの選択は、USBX デバイスモジュールによって使用されるモードと整合している必要があります。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBFS0 での USBX デバイスモジュールのピン構成設定

[Properties]	Value	説明
Operation Mode	Device	動作モードとして [Device] を選択します
USBDP	USBDP	USPDP
USBDM	USBDM	USBDM
OVRCURB	None	OVRCURB
OVRCURA	None	OVRCURA
VBUSEN	None	VBUSEN
VBUS	None, P407 (Default: P407)	VBUS

[Properties]	Value	説明
EXICEN	None	EXICEN
ID	None	ID
VCCUSB	VCCUSB	VCCUSB
VSSUSB	VSSUSB	VSSUSB

注: *例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

USBX デバイス USBHS0 のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Device	動作モードとして [Device] を選択します
USBHSDP	USBHSDP	USBHSDP
USBHSDM	USBHSDM	USBHSDM
OVRCURB	None	OVRCURB
OVRCURA	None	OVRCURA
VBUSEN	None	VBUSEN
VBUS	None, PB01 (Default: PB01)	VBUS
EXICEN	None	EXICEN
ID	None	ID
USBHSRREF	USBHSRREF	USBHSRREF
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS
VCCUSBHS	VCCUSBHS	VCCUSBHS

Pin Configuration Property	Value	説明
VSS1USBHS	VSS1USBHS	VSS1USBHS
VSS2USBHS	VSS2USBHS	VSS2USBHS

注 : 例の値は、Synergy S7G2 グループおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.1.28.5 アプリケーションでの Express Logic USBX Synergy ポートフレームワークモジュールの使用

Express Logic Synergy ポート USBX フレームワークモジュールをスレッドに追加すると、Express Logic API を使用できるようになります。

5.1.29 NetX Duo MQTT クライアントモジュールガイド

MQTT（Message Queue Telemetry Transport）は、パブリッシャ / サブスクライバモデルに基づいています。クライアントは、ブローカを介して他のクライアントに情報をパブリッシュできます。クライアントは、トピックに関心がある場合、ブローカを介してトピックをサブスクライブできます。ブローカは、クライアントの認証と承認を行い、パブリッシュされた情報を、トピックをサブスクライブしているクライアントに提供する必要があります。このパブリッシャ / サブスクライバモデルでは、複数のクライアントが同じトピックでデータをパブリッシュできます。

クライアントは、同じトピックをサブスクライブしている場合、メッセージのパブリッシュを受け取ります。

次の図は、 MQTT クライアントのパブリッシュ / サブスクライブモデルの概要です。

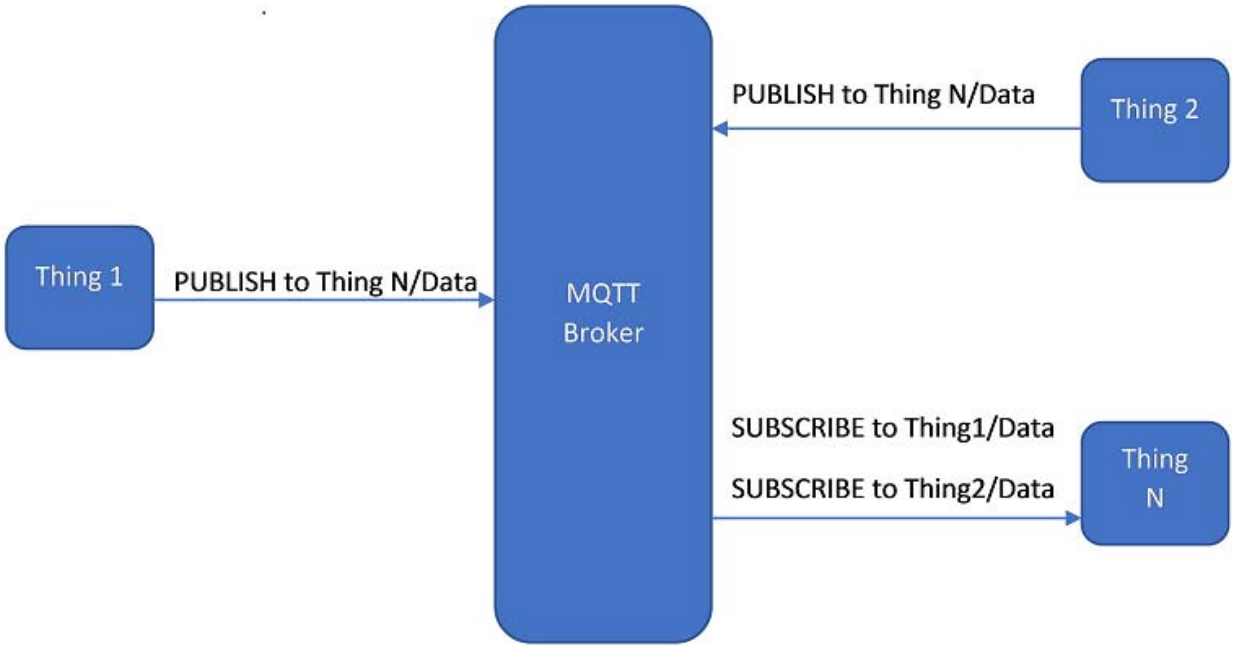


図 280: MQTT クライアントのパブリッシュ / サブスクライブモデル

5.1.29.1 NetX Duo MQTT クライアントの機能

- 2014 年 10 月 29 日の OASIS MQTT バージョン 3.1.1 に準拠します。仕様は次の場所で確認できます。http://mqtt.org/
- SSP で NetX Secure を使用する安全な通信のために TLS を有効または無効にするオプションを提供します
- QoS をサポートし、メッセージをパブリッシュするときにレベルを選択できます
- 内部バッファを備え、受信したメッセージのキューを保持します
- 新しいメッセージを受信したときのコールバックを登録するメカニズムを提供します
- ブローカとの接続が終了されたときのコールバックを登録するメカニズムを提供します。

5.1.29.2 MQTT クライアントの動作の概要

次に示すのは、NetX Duo TLS のサポートが有効な NetX Duo MQTT クライアントがプロジェクトに追加されたスレッドペインビューの例です。

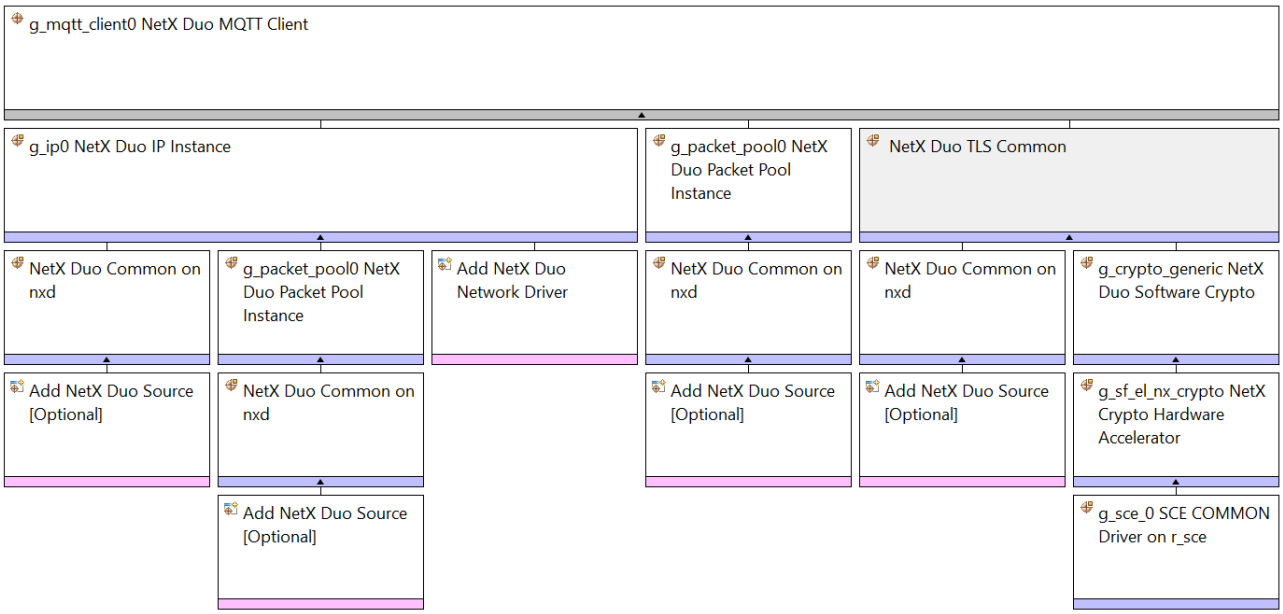


図 281: NetX TLS が有効な NetX Duo MQTT クライアントを含むスレッドペインビューの例

NetX Duo MQTT は、NetX Duo TCP/IP スタックに基づくプロトコルです。NetX Duo MQTT クライアントコンポーネントを追加するためのエントリは、[X-Ware] -> [NetX Duo] -> [Protocols] -> [NetX Duo MQTT Client] にあります。

NetX Duo MQTT クライアントコンポーネントをプロジェクトに追加すると、MQTT クライアントのインスタンスを作成するために必要な MQTT クライアントモジュールが追加されます。ユーザーのコードでこの MQTT クライアントインスタンスを使用して、接続、パブリッシュ、サブスクライブなどの操作をさらに実行できます。

また、次の図で示すようないくつかのプロパティが関連付けられています。

Property	Value
Common	
NX Secure	Enable
Topic Name Max Length	12
Message Max Length	32
Keepalive Timer Rate (s)	1
Ping Timeout Delay (s)	1
Module g_mqtt_client0 NetX Duo MQTT Client	
Name	g_mqtt_client0
Name of generated initialization function	mqtt_client_init0
Auto Initialization	Enable
Client ID Callback	mqtt_client_id_callback
Client ID Max Length	12
Client Thread Stack Size	4096
Number of Messages to be stored in memory	1
Client thread priority	2

図 282: NetX Duo MQTT クライアントブロックの構成可能なプロパティ

以下では構成可能なプロパティについて説明します。

共通プロパティ

- NX Secure: TLS のサポートを有効または無効にします。これを有効にすると、MQTT クライアントは TLS サポートありでビルドされます。これを有効にすると、NX_SECURE_ENABLE マクロが定義されます。
- Topic Name Max Length: アプリケーションがサブスクライブするトピックの最大長です（バイト単位）。デフォルトは 12 バイトです。これは、NXD_MQTT_MAX_TOPIC_NAME_LENGTH マクロの値を定義します。
- Message Max Length: アプリケーションが送信または受信するメッセージの最大長です（バイト単位）。デフォルトは 32 バイトです。これは、NXD_MQTT_MAX_MESSAGE_LENGTH マクロの値を定義します。
- Keepalive Timer Rate: このタイマは、最後に MQTT 制御メッセージが送信されてからの時間を追跡し、キープアライブ時間が経過する前に MQTT PINGREQ メッセージを送信するために使用されます。デフォルト値は 1 秒です。これは、NXD_MQTT_KEEPALIVE_TIMER_RATE マクロの値を定義します。
- Ping Timeout Delay: MQTT クライアントが MQTT PINGREQ を送信した後、ブローカからの PINGRESP を待機する時間です。デフォルト値は 1 秒です。これは、NXD_MQTT_PING_TIMEOUT_DELAY マクロの値を定義します。

MQTT クライアントのコードの動作を制御する共通プロパティ。これらのプロパティを変更すると、synergy_cfg/ssp_cfg/framework/el/nxd_mqtt_client_cfg.h. での MQTT クライアントの構成が変更されます。

モジュールのプロパティ

- Name: MQTT クライアントインスタンス変数の名前

- Name of generated initialization function: MQTT クライアントインスタンスを作成する自動生成関数（つまり初期化関数）の名前。
- Auto Initialization: 初期化関数の呼び出しを有効または無効にします。
- Client ID Callback: 一意のクライアント ID を取得するために使用される、ユーザーによって提供されるコールバック関数。
- Client ID Max Length: クライアント ID の最大長（バイト単位）。
- Client Thread Stack Size: MQTT クライアントのスレッドスタックサイズ（バイト単位）。
- Number of Messages to be stored in Memory: MQTT クライアントは、メモリ領域を使用してメッセージを格納します。 MQTT クライアントの動作に必要なメモリ領域のサイズは、送受信されるデータの量によって異なります。最少のメモリサイズは MQTT_MESSAGE_BLOCK 構造体のサイズであり、デフォルトでは 60 バイトです。
- Client Thread Priority: MQTT クライアントのスレッドのプライオリティ

モジュールのプロパティは、nxd_mqtt_client_create() API を使用して MQTT クライアントのインスタンスを作成するために、自動生成されるコードによって使用されます。

5.1.29.3 NetX Duo MQTT クライアントの動作に関する注意

セキュアな通信の使用

デフォルトでは、MQTT クライアントは TLS を使用しません。その場合、MQTT クライアントとブローカの間の通信はセキュアではありません。

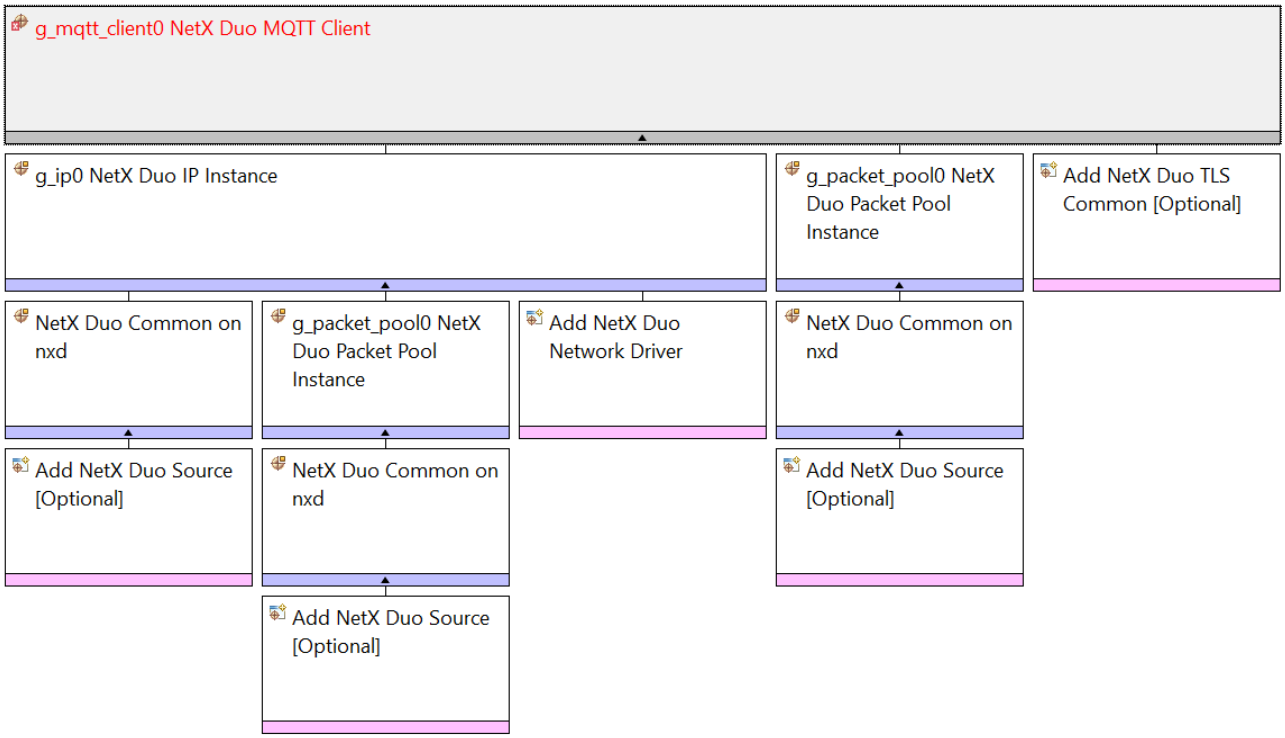


図 283: NetX Duo MQTT クライアントコンポーネントのスレッドペインビュー

MQTT クライアントとブローカの間の通信をセキュアにするには、TLS プロトコルを使用する必要があります。次の図で示すように、スレッドペインでは、TLS プロトコルは [Add NetX Duo TLS common [Optional]] ブロックによって表されます。 NetX Duo TLS 共通ブロックを追加すると TLS のサポートが有効になり、**NX_SECURE_ENABLE** マクロが内部的に定義されます。次の図は、 TLS のサポートが有効になっている MQTT クライアントのスレッドペインビューです。

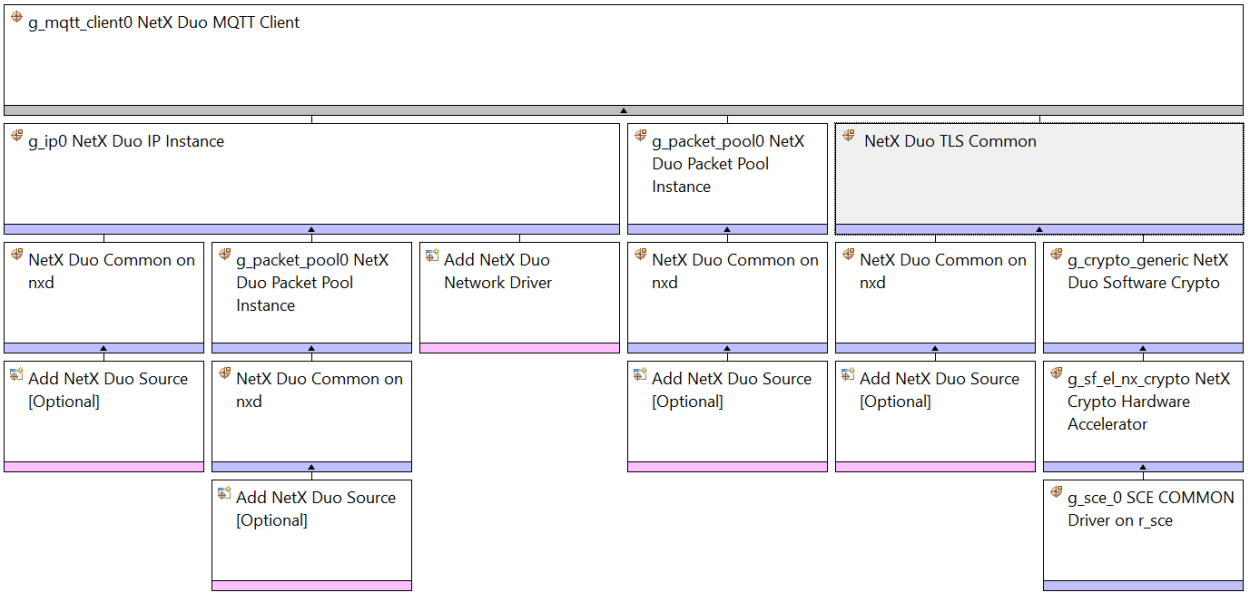


図 284: TLS のサポートが有効になっている NetX Duo MQTT クライアントコンポーネント

MQTT クライアントによって NetX Duo TLS セッションブロックが追加されないことに注意してください。追加されるのは NetX Duo TLS 共通ブロックだけです。このブロックは、NetX Secure の共通プロパティを定義および制御します。これらのプロパティの詳細については、NetX Secure のユーザーノートを参照してください。NetX Duo TLS セッションブロックは追加されないので、TLS セッションを生成するためのコードは自動生成されません。ユーザー / アプリケーションコードで TLS セッションを手動で作成する必要があります。ユーザーコードは、nxd_mqtt_client_secure_connect() API によって提供される TLS セットアップコールバックの中で、TLS セッションの作成を実行し、セキュリティパラメータを構成し、関連する証明書を読み込むことができます。

TLS セッションを作成する前に、メタデータバッファを割り当てる必要があります。メタデータのサイズは、プロパティペインを使用して設定できます。また、ユーザーは、NetX Secure の nx_secure_metadata_size_calculate() API を使用して、メタデータバッファのサイズを計算できます。詳細については、『NetX Duo MQTT Client user guide』を参照してください。次のサンプル参照コードでは、TLS セットアップコールバックで TLS セッションの初期化を行っています。

```
/* Cipher Suite Variables  */
 extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers_synergys7;
 extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers;
UINT tls_setup(NXD_MQTT_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session,
               NX_SECURE_X509_CERT *certificate, NX_SECURE_X509_CERT *trusted_ce
rtificate)
{
    status = nx_secure_tls_session_create(tls_session,
                                          &nx_crypto_tls_ciphers_synergys7,
                                          crypto_metadata,
                                          sizeof(crypto_metadata));
    if (status)
```

```
    {
        sprintf(str,"Error in creating TLS Session: 0x%02x\n", status);
        print_to_console(str);
        return(1);
    }
    else
    {
        sprintf(str,"TLS Session has been created. \r\n");
        print_to_console(str);
    }
    /* Allocate space for packet reassembly. */
    status = nx_secure_tls_session_packet_buffer_set(&(client_ptr->nxd_mqtt_tls_
session), tls_packet_buffer, sizeof(tls_packet_buffer));
    /* Check for error.  */
    if (status)
    {
        sprintf(str,"Error in setting Packet Buffer for TLS : %d \r\n",status);
        print_to_console(str);
        return(1);
    }
    else
    {
        sprintf(str,"Packet Buffer for TLS Has been set \r\n");
        print_to_console(str);
    }

    /* Need to allocate space for the certificate coming in from the remote host
. */
    nx_secure_tls_remote_certificate_allocate(tls_session, &remote_certificate,
remote_cert_buffer, sizeof(remote_cert_buffer));
    nx_secure_tls_remote_certificate_allocate(tls_session, &remote_issuer, remot
e_issuer_buffer, sizeof(remote_issuer_buffer));


    /* Add a CA Certificate to our trusted store for verifying incoming server c
ertificates. */
    nx_secure_x509_certificate_initialize(trusted_certificate, ca_cert_der, ca_c
ert_der_len, NX_NULL, 0, NULL, 0);
    nx_secure_tls_trusted_certificate_add(tls_session, trusted_certificate);

    /* Add the local certificate for client authentication. */
    nx_secure_x509_certificate_initialize(certificate, cert_der, cert_der_len, N
X_NULL, 0, private_key_der, private_key_der_len);
    nx_secure_tls_local_certificate_add(tls_session, certificate);


    /* Add a timestamp function for time checking and timestamps in the TLS hand
shake. */
    nx_secure_tls_timestamp_function_set(tls_session, tls_timestamp_function);
```

```
    /* Setup the callback invoked when TLS has a certificate it wants to verify
so we can
       do additional checks not done automatically by TLS. */
    nx_secure_tls_session_certificate_callback_set(tls_session, certificate_veri
fication_callback);

    return NX_SUCCESS;
}
```

一意のクライアント ID の設定

MQTT クライアントインスタンスは、nxd_mqtt_client_create() API を使用して作成します。この API のパラメータの 1 つは一意のクライアント ID であり、MQTT ブローカはそれを使用してクライアントを一意に識別します。MQTT クライアントコンポーネントではクライアント ID コールバックやクライアント ID 最大長などのプロパティが提供されており、一意のクライアント ID を取得するために使用されます。次の図を参照してください。

Module g_mqtt_client0 NetX Duo MQTT Client	
Name	g_mqtt_client0
Name of generated initialization function	mqtt_client_init0
Auto Initialization	Enable
Client ID Callback	mqtt_client_id_callback
Client ID Max Length	12
Client Thread Stack Size	4096
Number of Messages to be stored in memory	1
Client thread priority	2

図 285: MQTT クライアント ID のプロパティ

ISDE で自動生成されるコードは、nxd_mqtt_client_create() を呼び出す前に、このクライアント ID コールバックを呼び出します。クライアント ID コールバックのプロトタイプは次のとおりです。

```
void mqtt_client_id_callback(char * p_client_id, uint32_t * p_client_id_length);
```

client_id はこのコールバック関数によって設定される出力パラメータであり、client_id_length は入力 / 出力パラメータです。

次に示すクライアント ID コールバック関数のサンプル参照実装では、MAC アドレスをクライアント ID にコピーしています。

5.1.29.4 アプリケーションでの NetX Duo MQTT クライアントの使用

1) [New Stack] > [X-Ware] > [NetX Duo] > [Protocols] > [NetX Duo MQTT Client] を使用して新しいスレッドスタックを追加することで、スレッドに NetX Duo MQTT クライアントコンポーネントを追加します。次の図を参照してください。

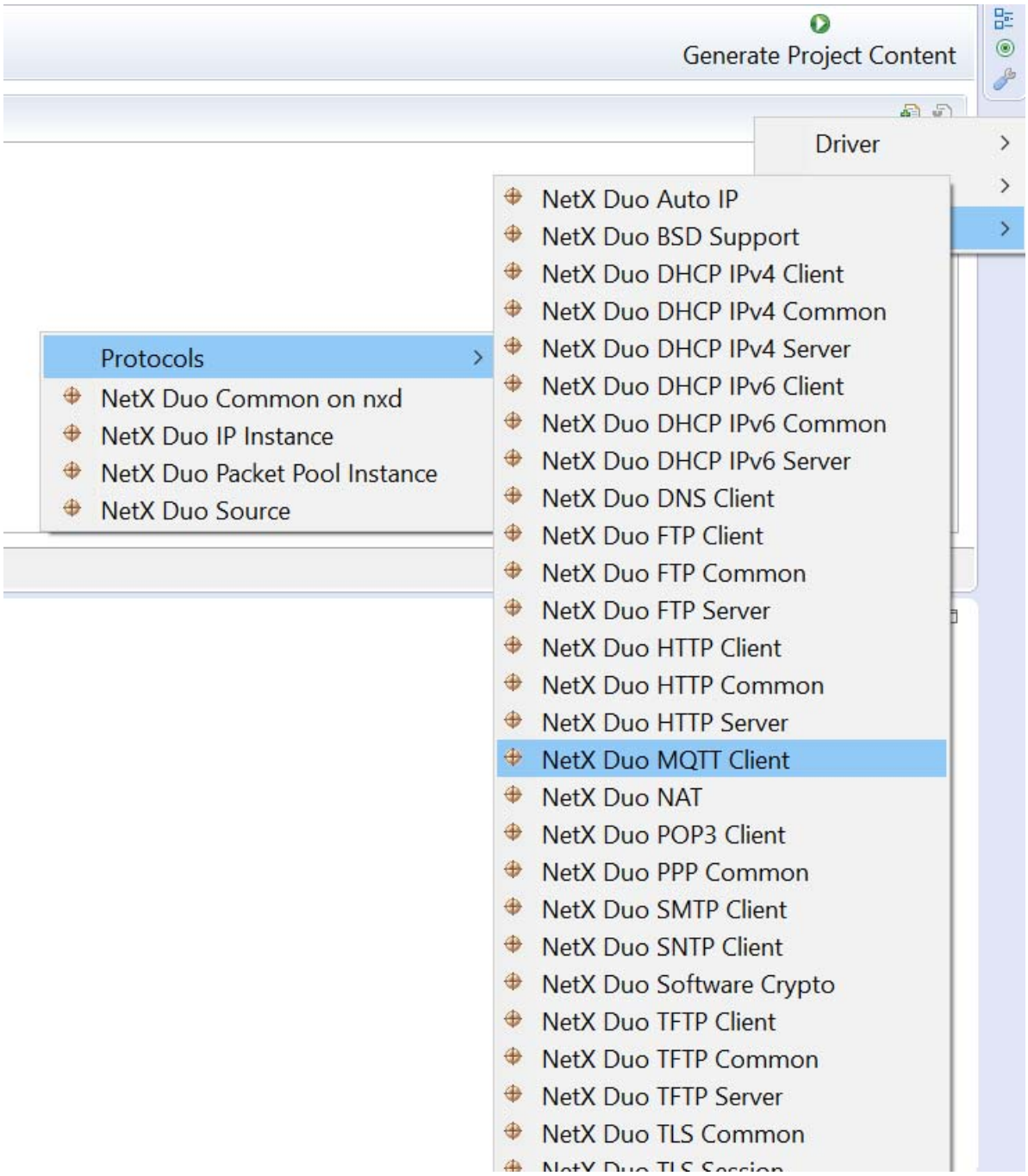


図 286: NetX Duo MQTT クライアントのメニュー

MQTT クライアントコンポーネントと NetX Duo IP インスタンスをスレッドに追加すると、次のようになります。

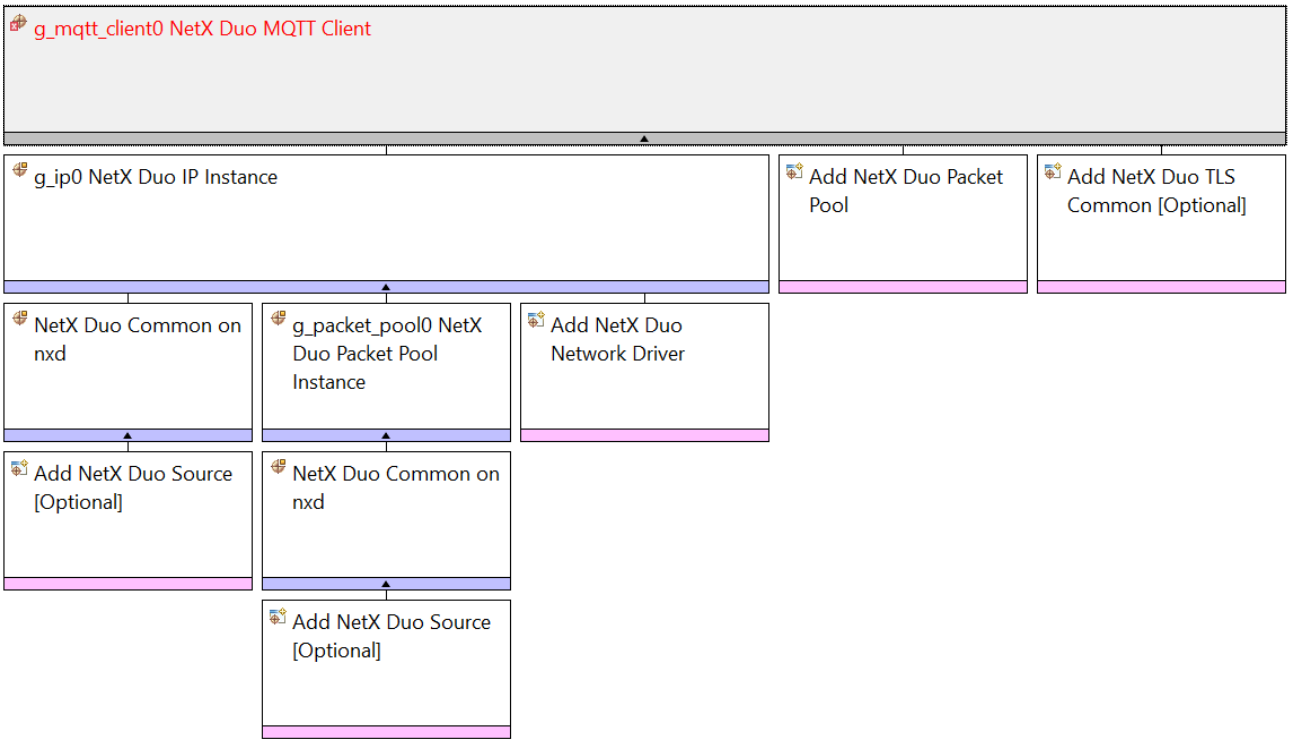


図 287: MQTT クライアントブロックのスレッドペインビュー

2) 既存の IP インスタンスがない場合、MQTT クライアントブロックを追加すると、IP インスタンスがその下に追加されます。既存の IP インスタンスがある場合は、それを使用することも、新しいインスタンスを追加することもできます。次の図を参照してください。

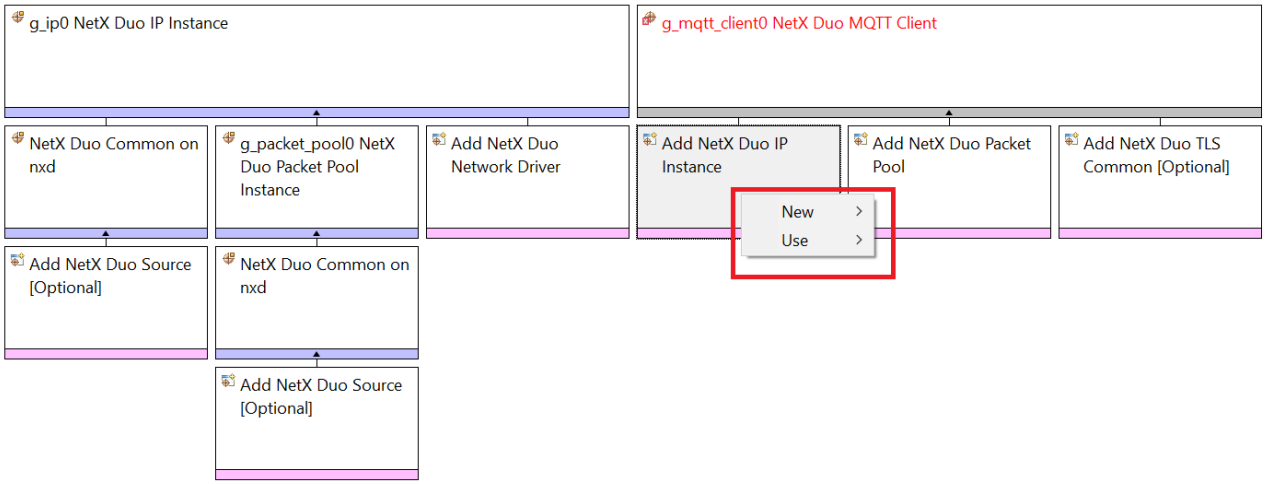


図 288: IP インスタンスブロックの新規追加または既存使用

3) IP インスタンスブロックを追加した後は、MQTT クライアントに必要なパケットプールを新しく追加するか、または既存のプールを使用できます。次の図を参照してください。

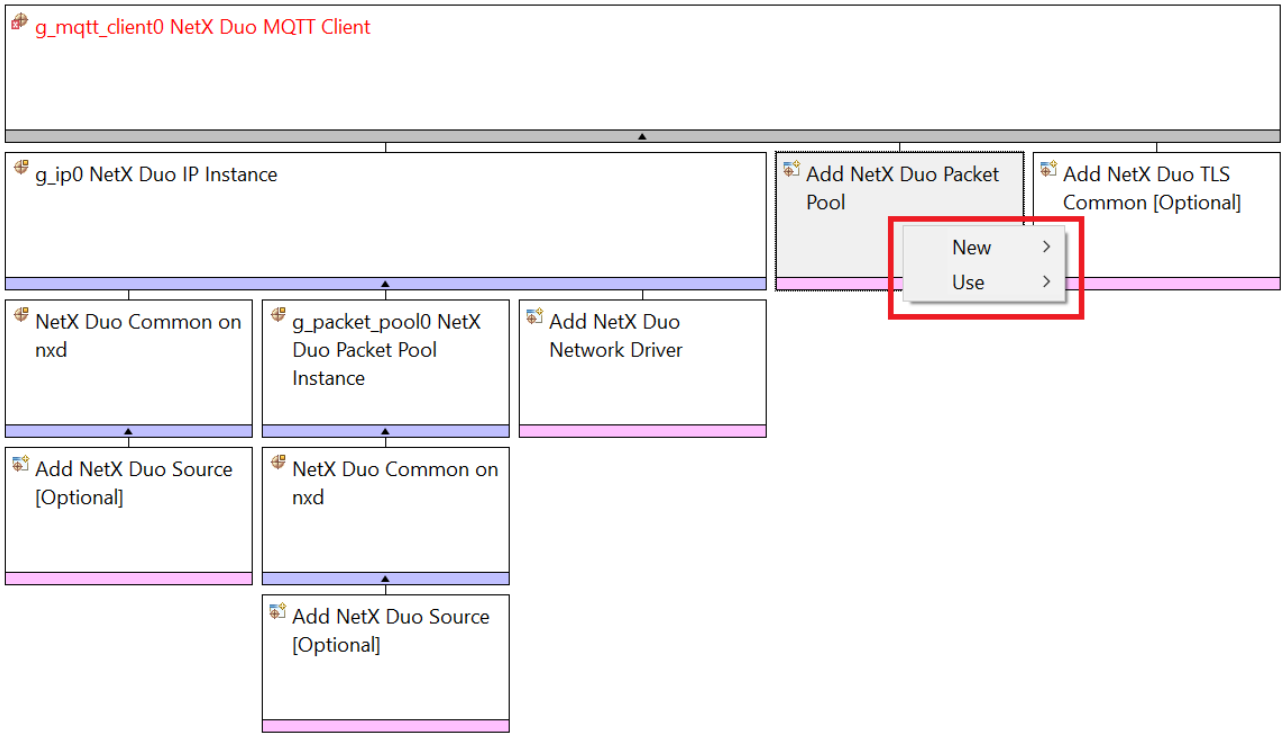


図 289: パケットプールブロックの新規追加または使用

4) パケットプールを追加した後のスレッドペインビューは次のようになります。

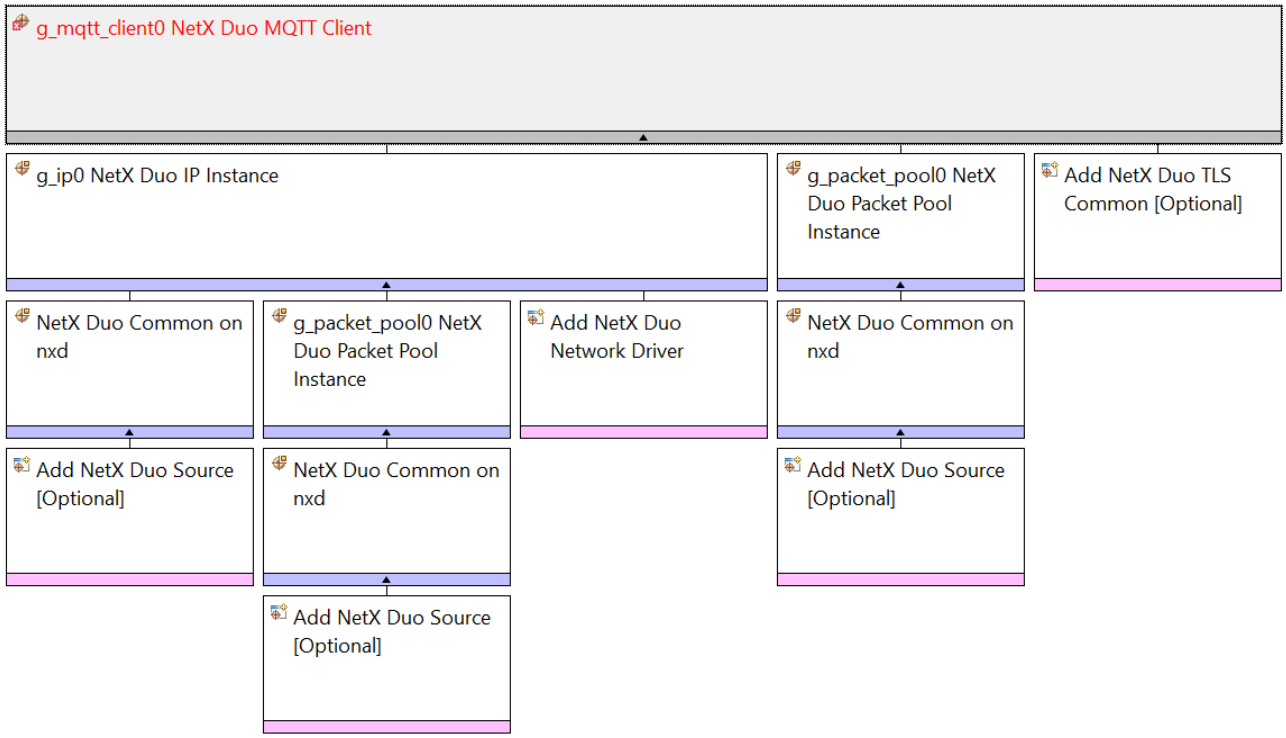


図 290: IP インスタンスとパケットプールが追加された状態の MQTT クライアントブロック

5) NetX Duo TLS 共通ブロックはオプションです。 MQTT クライアントブロックの **[NX Secure]** プロパティが [Enable] に設定されている場合は、このブロックを追加する必要があります。無効になっている場合は、 TLS 共通ブロックを追加する必要はありません。デフォルトでは、 **[NX Secure]** プロパティは [Enable] に設定されています。[Properties] ウィンドウで、MQTT クライアントブロックのこのプロパティを構成できます。

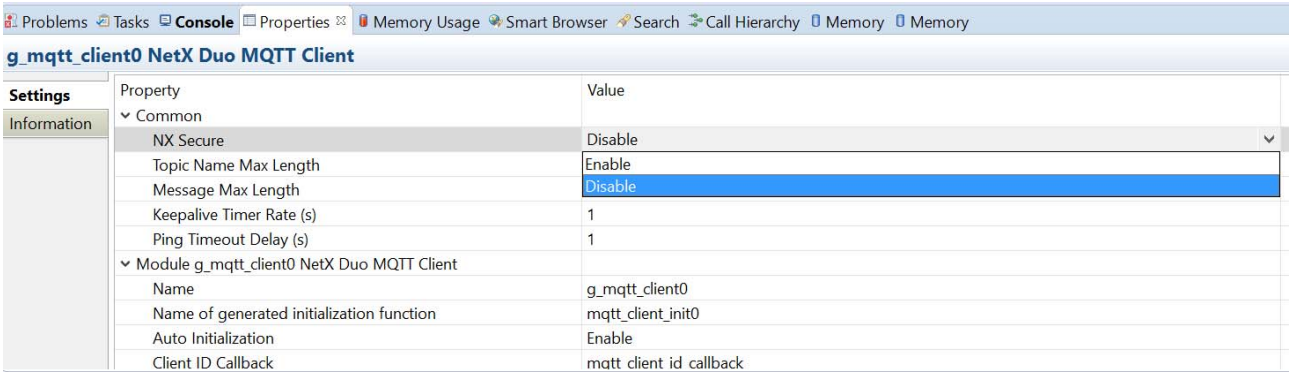


図 291: NX Secure の有効化 / 無効化プロパティ

6) **[NX Secure]** プロパティを [Disable] に設定すると、MQTT クライアントとブローカの間の接続はセキュアではありません。このプロパティを [Enable] に設定する必要があります。そうすれば、 MQTT クライアントとブローカの間の接続はセキュアになります。このプロパティを [Enable] に設定するには、NetX Duo TLS 共通ブロックを追加する必要があります。[NX Secure] プロパティを [Enable] に設定して TLS 共通ブロックを追加した MQTT クライアントブロックの図を次に示します。

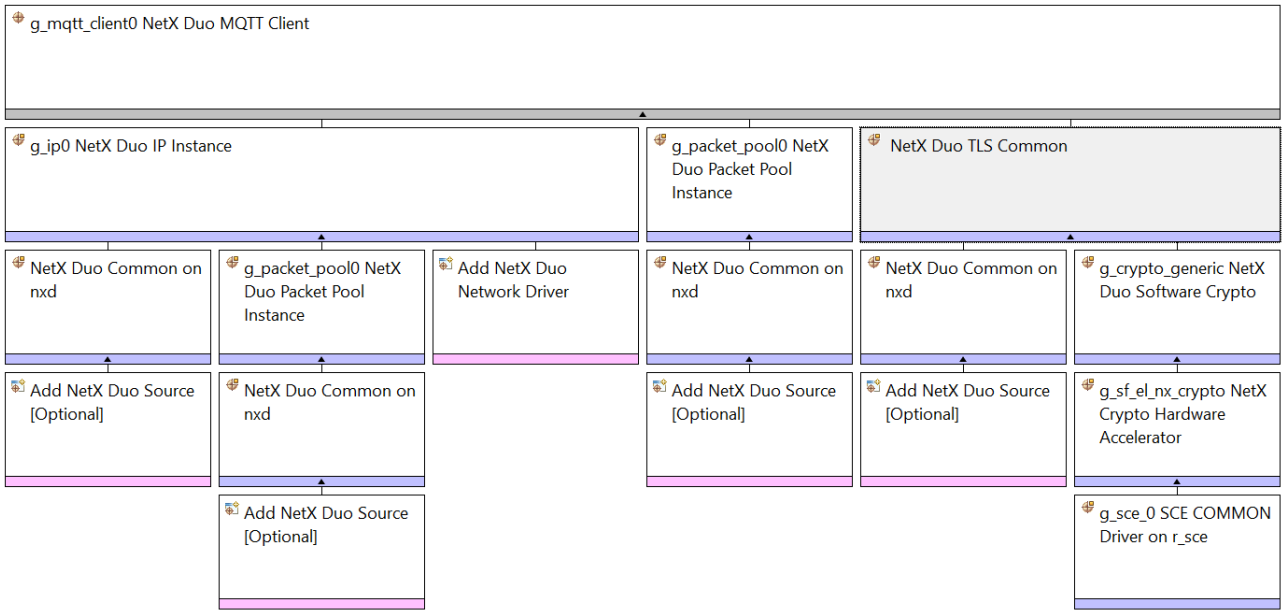


図 292: TLS 共通ブロックが追加された MQTT クライアントブロック

7) [Properties] ウィンドウで、 MQTT クライアントの他のプロパティを構成します。

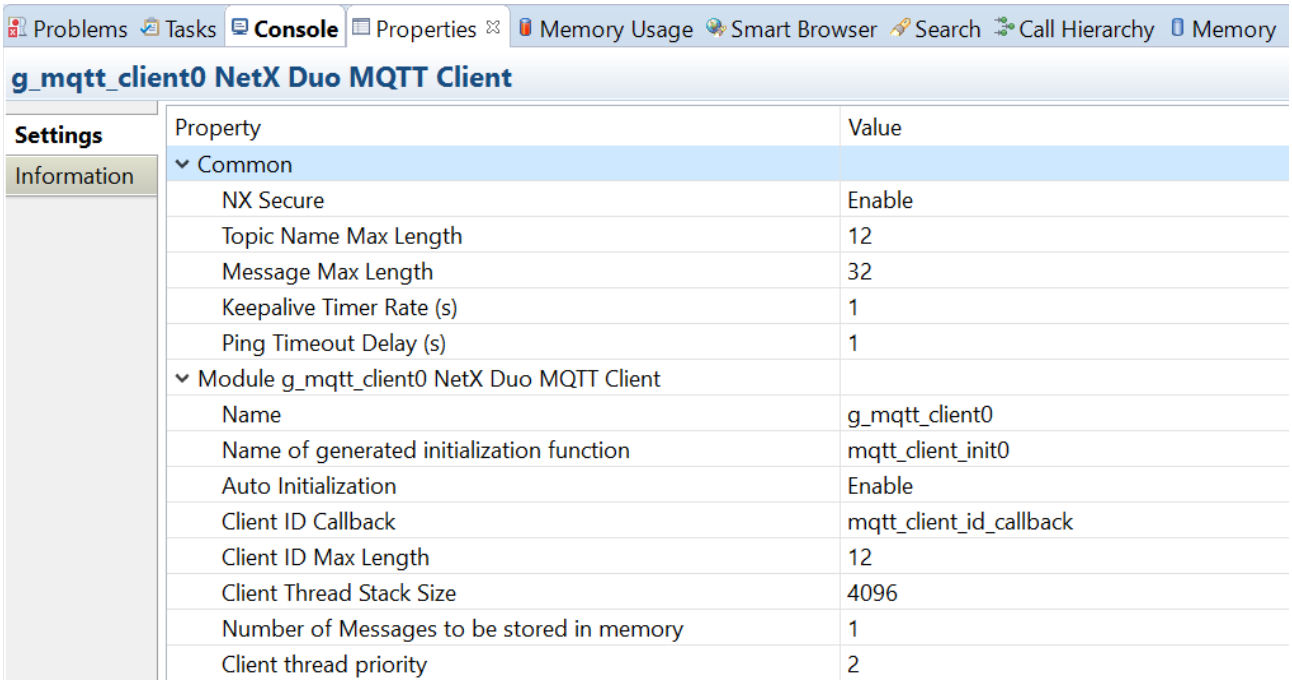
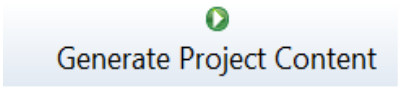


図 293: MQTT クライアントのプロパティ

詳細については、「MQTT クライアントの動作の概要」の構成可能なプロパティを参照してください。

8) プロパティを構成した後は、スレッドペインの [Generate Project Content] 機能を使用してコードを生成します。



9) これにより、構成したスレッドの下にコードが自動生成されます。この自動生成されるコードには、**[Name of generated initialization function]** プロパティで指定した名前の初期化関数が含まれます。この関数は nxd_mqtt_client_create()API を内部的に呼び出して、[Name**] プロパティで指定されている名前の MQTT クライアントインスタンスを作成します。この初期化関数の呼び出しは、**[Auto Initialization]** プロパティの値に応じて有効または無効になります。

10) ユーザーコードでは、nxd_mqtt_client_create()APIを呼び出す前に初期化関数によって呼び出されるクライアント ID コールバックを実装する必要があります。このクライアント ID コールバックは、一意のクライアント ID（MAC アドレスなど）を返す必要があります。「NetX Duo MQTT クライアントの動作に関する注意」の 「一意のクライアント ID の設定」セクションを参照してください。

11) MQTT クライアントがセキュア通信を使用するように構成されている場合、つまり [NX Secure**] プロパティが [Enable] に設定されている場合は、TLS の初期化とセットアップに必要なコードを追加する必要があります。「NetX Duo MQTT クライアントの動作に関する注意」の 「セキュアな通信の使用」セクションを参照してください。

12) MQTT エンドポイントつまりブローカとの接続、メッセージのパブリッシュ、トピックのサブスクライブ / サブスクライブ解除などを行うコードの追加については、 Synergy Gallery で入手できる 『NetX Duo MQTT client user guide』を参照してください。

5.1.30 NetX Duo TLS セッションモジュールガイド

NetX Duo TLS セッションモジュールは、 Secure Socket Layer （SSL）プロトコルおよびそれを置き換える Transport Layer Security（TLS）プロトコル（RFC 2246（バージョン 1.0）および RFC 5246（バージョン 1.2）を参照）を実装している Express Logic 社の NetX Secure が基になっています。 NetX Secure には、基本的な X.509（RFC 5280）用のルーチンも含まれます。 NetX Secure は、プロジェクト内の ThreadX RTOS を使用するアプリケーション用です。

TLS/SSL プロトコルは、 2 つのアプリケーション間の通信にプライバシと信頼性を提供します。次のような基本特性を備えています。

- 暗号化 : 通信するアプリケーション間で交換されるメッセージは暗号化され、接続のプライベート性が保証されます。データの暗号化には、AES（Advanced Encryption Standard）などの対称暗号メカニズムが使用されます。
- 認証 : 証明書を使用してピアのアイデンティティを確認するメカニズムです。
- 整合性 : メッセージの改ざんや偽造を検出して接続が信頼できることを保証するメカニズムです。メッセージの整合性を保証するには、セキュアハッシュアルゴリズム （SHA）などのメッセージ認証コード （MAC）が使用されます。

TLS/SSL は TCP を使用し、HTTP や MQTT などのアプリケーションレイヤープロトコルにセキュアな通信を提供します。

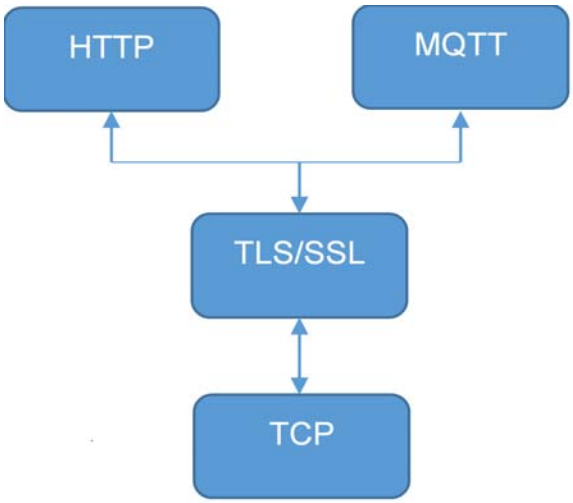


図 294: TLS/SSL の階層

TLS/SSL にはよく知られたポート番号はありません。代わりに、上位レイヤープロトコルのセキュア版の指定ポート番号を使用します。たとえば、セキュア HTTP の場合はポート番号 443、 MQTT の場合はポート番号 8883 などです。

TLS/SSL を使用してセキュアな接続が確立されると、クライアント（常に、接続を開始します）とサーバーの間でたとえば HTTPS メッセージが交換されます。次の図で示すように、最初のメッセージセットによってハンドシェイクプロトコルが実行された後、クライアントとサーバーは双方向にデータをセキュアに送受信できます。

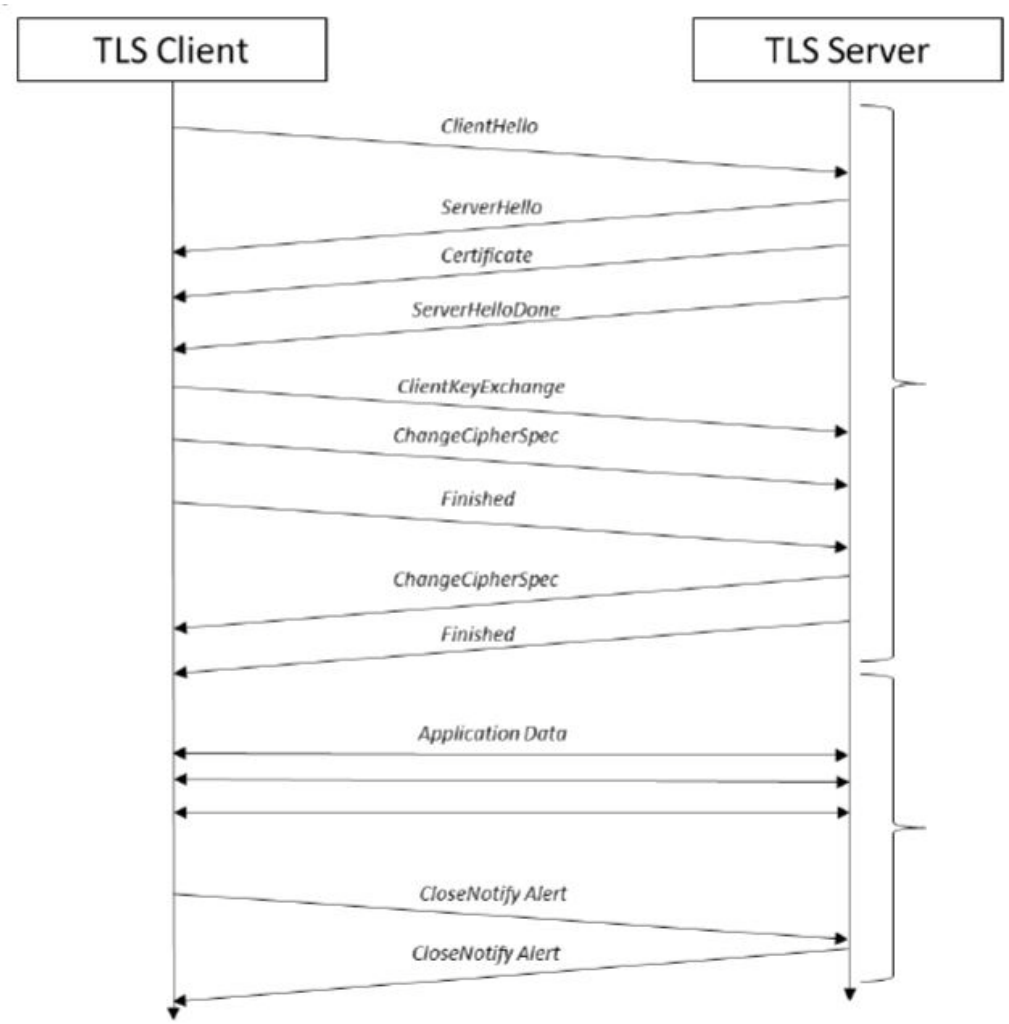


図 295: TLS/SSL プロトコルのシーケンス

5.1.30.1 NetX Duo TLS セッションの機能

- TLS プロトコルバージョン 1.0 （RFC2246）のサポート
- Transport Layer Security （TLS）プロトコルバージョン 1.2 （RFC5246）のサポート
- X.509 PKI 証明書バージョン 3 （RFC5280）のサポート
- Transport Layer Security （TLS）向け Advanced Encryption Standard （AES）暗号スイート （RFC3268）のサポート

- 公開キー暗号化標準 （PKCS） #1: RSA 暗号仕様バージョン 2.1 （RFC3447）
- HMAC: メッセージ認証用キー付きハッシュ （RFC2104）
- US セキュアハッシュアルゴリズム （SHA および SHA ベースの HMAC と HKDF）（RFC6234）
- TLS 向け事前共有キー暗号スイート （RFC4279）

注: *1: 詳細については、『Express Logic NetX Secure User Guide』を参照してください。*

注: *2: NetX Secure モジュールの既知の問題と制限事項については、SSP のリリースノートを参照してください。*

5.1.30.2 NetX Duo TLS セッションの動作の概要

次の図では、プロジェクトに NetX Duo TLS セッションモジュールが含まれるスレッドペインビューの例を示します。

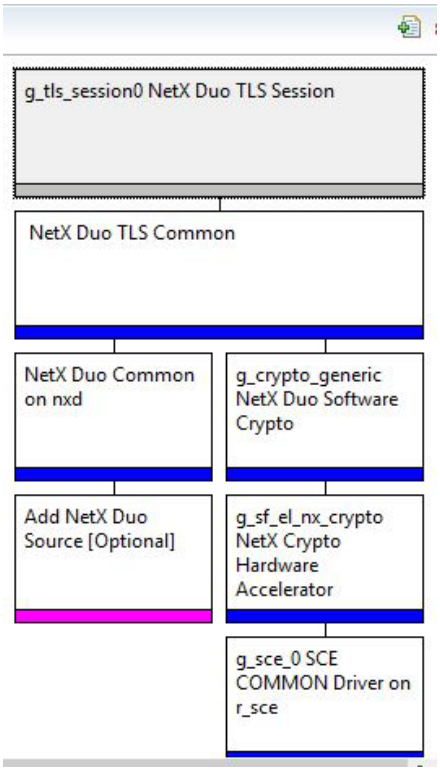


図 296: NetX Duo TLS セッションが含まれるスレッドペインビューの例

NetX Duo TLS セッションを追加するためのエントリは、[X-Ware] -> [NetX Duo] -> [Protocols] -> [NetX Duo TLS Session] にあります。 NetX Duo TLS セッションコンポーネントをプロジェクトに追加すると、Express Logic から提供されている NetX Secure のコードおよび TLS セッションインスタンスを生成するためのコードが追加されます。MQTT クライアントなどのアプリケーションは、TLS セッションのインスタンスを使用して、ピアとのセキュアな接続を確立できます。

次の図では、 NetX Duo TLS セッションに関連付けられている構成可能なプロパティを示します。

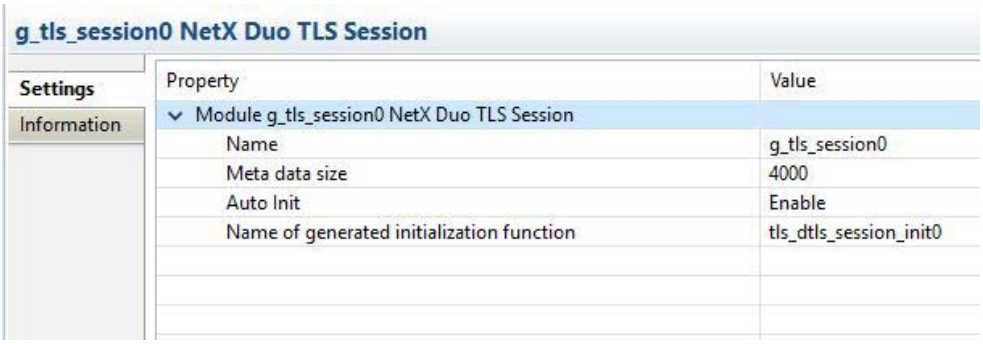


図 297: NetX Duo TLS セッションの構成可能なプロパティ

- Name: TLS セッションインスタンス変数の名前
- Metadata size: 暗号ルーチンメタデータのバッファ空間のサイズ
- Name of generated initialization function: TLS セッションインスタンスを作成する自動生成関数（つまり初期化関数）の名前。
- Auto Initialization: 初期化関数の呼び出しを有効または無効にします。

NetX Duo TLS セッションコンポーネントには、 NetX Duo TLS セッション共通という名前の子モジュールが 1 つあります。次の図のように、このモジュールは TLS のさまざまなグローバル構成プロパティの制御を担当します。

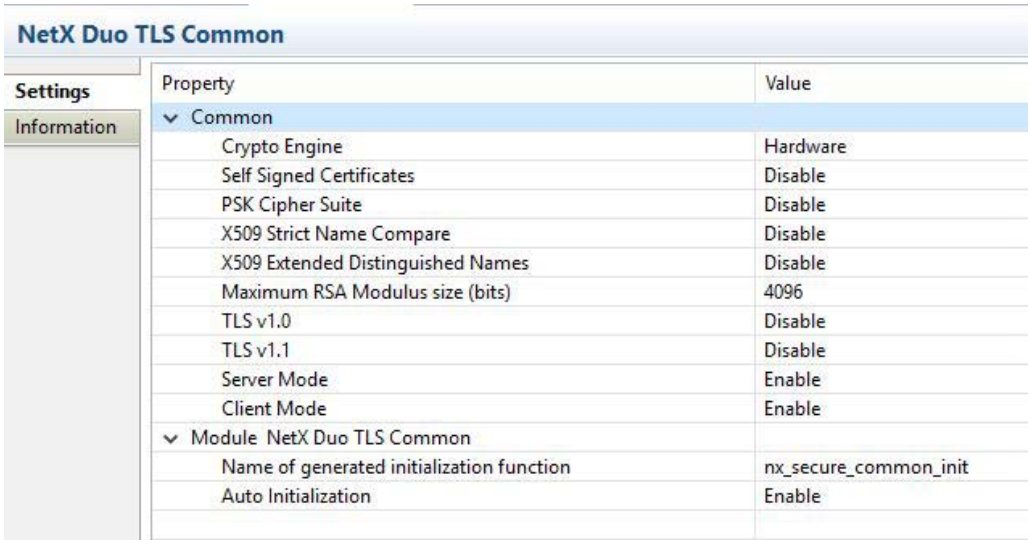


図 298: NetX Duo TLS 共通の構成可能なプロパティ

共通プロパティ

- Crypto Engine: ユーザーは、ソフトウェアまたはハードウェアのどちらを使用して暗号化を行うかを指定できます。

***注:** 現在の実装では、ハードウェアを利用した暗号化のみがサポートされています。*

- Self-Signed Certificates: TLS がリモートホストからの自己署名証明書を受け付けることを許可します。許可しない場合、証明書は信頼できる証明書ストアによって発行されている必要があります。
- PSK Cipher Suite: 事前共有キー（PSK）機能を有効にします。これは、通信パーティ間での証明書ではなく PSK を使用した認証を有効にするオプションの TLS 機能です。PSK は、（通常は、物理的な伝達または他の安全な方法を使用して）通信パーティ間で事前に共有されます。
- X509 Strict Name compare: 証明書の検索と検証のために、X.509 証明書の識別名の厳密な比較を有効にします。
- X509 Extended Distinguished Names: オプションの [X.509 Distinguished Name] フィールドを有効にします。
- Maximum RSA modulus size: 想定される最大 RSA モジュールサイズを指定します（ビット単位）。
- TLVS v1.1 mode: このプロパティは TLS v1.1 を有効または無効にしますが、SSP 1.3.0 は TLS v1.1 をサポートしていません。
- Server mode: 有効 / 無効。このプロパティを無効にすると、プロジェクトからの TLS サーバーモードに関連するすべての TLS スタックコードが除外されます。
- Client mode: 有効 / 無効。このプロパティを無効にすると、プロジェクトからの TLS クライアントモードに関連するすべての TLS スタックコードが除外されます。

モジュールのプロパティ

- Name of generated initialization function: TLS スタックを初期化するには、この関数を呼び出す必要があります。
- Auto Initialization: 自動生成されるスタートアップコードからの初期化関数の呼び出しを有効または無効にします。

スタック内の他のモジュール

- NetX Duo TLS 共通モジュールのスタックは、nxd の NetX Duo 共通と最上部の NetX Duo ソフトウェア暗号化、昼間部の NetX 暗号化ハードウェアアクセラレータ、最下部の SCE 共通ドライバーで構成されています（前の図を参照）。
- NetX Duo ソフトウェア暗号化。アプリケーションでソフトウェアベースの暗号化が必要な場合に使用されます。
- Netx ハードウェアアクセラレータ暗号化。TLS レイヤーでの性能向上のためにアプリケーションで暗号化にハードウェアアクセラレータを使用する必要がある場合に使用されます。
- セキュア暗号化エンジン（SCE）共通。ハードウェアアクセラレータによって使用される AES、HASH などのアルゴリズムのためのドライバーを表します。

5.1.30.3 NetX Duo TLS セッションの動作に関する注意事項

- TLS セッションを作成する前に、メタデータバッファを割り当てる必要があります。メタデータのサイズは、プロパティペインを使用して設定できます。また、ユーザーは NetX Secure を使用して、メタデータバッファの必要なサイズを計算できます。
- nx_secure_metadata_size_calculate() API に入力されるすべての証明書について、NetX Secure TLS は、基本的な X.509 パス検証を実行するだけでなく、検証プロセスの各ステージにおいて、各証明書の有効期限の日付を、アプリケーションのタイムスタンプ関数によって提供される日時と比較してチェックします。TLS が現在の日時を取得する必要があるときに呼び出すアプリケーションタイムスタンプ関数

に対する関数ポインタを設定するには、API nx_secure_tls_timestamp_function_set() を使用します。現在の日時は、さまざまな TLS ハンドシェイクメッセージおよび証明書の検証に使用されます。タイムスタンプ関数が割り当てられていない場合、TLS ハンドシェイクのタイムスタンプには値 0 が使用され、証明書の有効期限のチェックは行われません。

- ユーザーは、nx_secure_tls_session_packet_buffer_set() API を使用してパケット再アセンブリバッファを TLS セッションに関連付けることが必要な場合があります。再アセンブリバッファは、複数の TCP パケットにまたがる可能性のある受信 TLS レコードを格納するために使用されます。受信 TLS レコードが提供されているバッファより大きい場合、TLS セッションはエラーで終了します。

5.1.30.4 NetX Duo TLS セッションに関する制限事項と注意事項

- 内蔵デバイスの特性のため、システムによっては、最大 TLS レコードサイズである 16KB をサポートするのに十分なメモリを用意できない場合があります。NetX Secure は、十分なリソースのあるデバイスでは 16KB のレコードを処理できます。詳細については、『Express Logic NetX Secure User Guide』を参照してください。
- 証明書の最低限の検証。NetX Secure は、証明書に対して基本的な X.509 チェーン検証を実行して、証明書が有効であり、信頼できる証明機関によって署名されていることを確認します。また、リモートホストの最上位のドメイン名と比較するための証明書共通名をアプリケーションに提供できます。ただし、証明書の拡張機能および他のデータの検証は、アプリケーションの実装者が行う必要があります。

5.1.30.5 アプリケーションでの NetX Duo TLS セッションの使用

1) [New Stack] > [X-Ware] > [NetX Duo] > [Protocols] > [NetX Duo TLS Session] を使用して新しいスレッドスタックを追加することで、スレッドに NetX Duo TLS セッションコンポーネントを追加します（次の図を参照）。

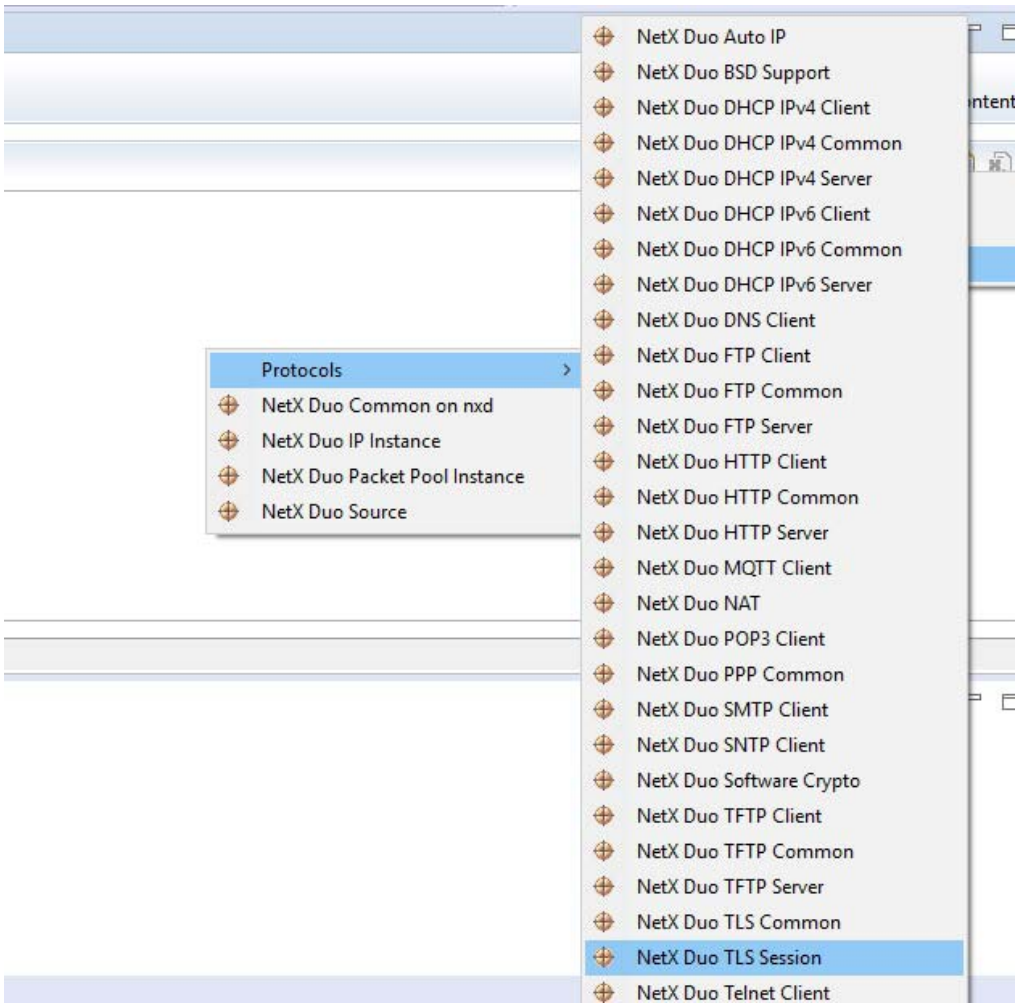


図 299: 図 6 NetX Duo TLS セッションのメニュー

NetX Duo TLS セッションコンポーネントを追加した後、そのスタックのスレッドペインでの表示は次の図のようになります。

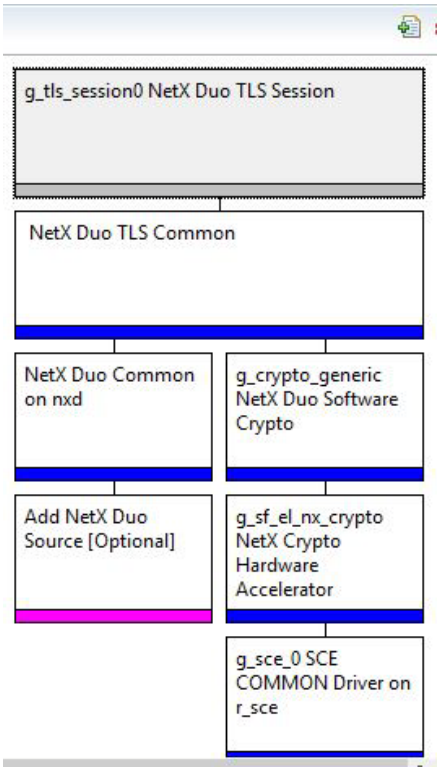
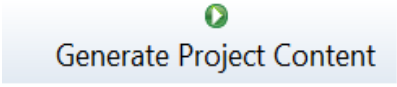


図 300: NetX Duo TLS セッションのスレッドペインビュー

2) プロパティを構成した後は、スレッドペインの [Generate Project Content] ツールを使用してコードを生成します。



3) これにより、構成したスレッドの下にコードが自動生成されます。この自動生成されるコードには、**[Name of generated initialization function]** プロパティで指定した名前の初期化関数が含まれます（tls_dlts_session_init0(). など）。 TLS セッションモジュールの **[Name]** プロパティを使用して指定した TLS セッションインスタンス変数が、初期化関数に入力として渡されます。このインスタンス変数は、TLS レイヤーの複数の API に入力として渡すこともできます。この関数は、内部で nx_secure_tls_session_create()API を呼び出して、TLS セッションを作成します。この初期化関数の呼び出しは、**[Auto Initialization]** プロパティの値に応じて有効または無効になります。自動生成されるコードは、common_data.c と new_thread0.c. の 2 つのファイルに分かれています。new_thread0.c. に関する次の 2 つの図では、自動生成されて common_data.c に格納されるコードを示します。最初の図で示されているスニペットは、[New Stack] > [X-Ware] > [NetX Duo] > [Protocols] > [NetX Duo TLS Common] でモジュールを追加した場合にも自動生成されることに注意してください。

```
void nx_secure_common_init(void)
{
#ifndef     NX_CRYPTO_ENGINE_SW
    /* Initialise secure crypto engine driver */
    sce_initialize ();
#endif

    /* Initialises the various control data structures for the TLS component
*/
    nx_secure_tls_initialize ();
}


void g_common_init(void)
{

    /** Call initialization function if user has selected to do so. */
#if (1)
    nx_common_init0 ();
#endif
    /**  Call initialization function if user has selected to do so. */
#if (1)
    nx_secure_common_init ();
#endif
}
```

```
#ifdef NX_CRYPTO_ENGINE_SW
extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers;
#else
extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers_synergys7;
#endif

void tls_dtls_session_init0(NX_SECURE_TLS_SESSION *p_tls_session)
{
    UINT g_tls_session0_err;
    /* Create TLS client. */

    g_tls_session0_err = nx_secure_tls_session_create (p_tls_session,
#ifdef NX_CRYPTO_ENGINE_SW
                                                       &
nx_crypto_tls_ciphers,
#else

&nx_crypto_tls_ciphers_synergys7,
#endif

g_tls_session0_meta_data,

sizeof(g_tls_session0_meta_data));

    if (NX_SUCCESS != g_tls_session0_err)
    {
        g_tls_session0_err_callback ((void *) p_tls_session,
&g_tls_session0_err);
    }
}



static void new_thread0_func(ULONG thread_input)
{
    /* Initialize each module instance. */
    /** Call initialization function if user has selected to do so. */
#if (1)
    tls_dtls_session_init0 (&g_tls_session0);
#endif

    /* Enter user code for this thread. */
    new_thread0_entry ();
}
```

図 301:　new_thread0.c の自動生成されたコードスニペット

NetX Secure API の詳細については、 Synergy Gallery にある 『NetX Secure User guide』を参照してください。

5.2 HAL レイヤー

5.2.1 ADC ドライバー

- 14 ビット A/D コンバータ （S3A6、 S3A7、 S124、 S128）または 12 ビット A/D コンバータ （S7G2、 S5D5、 S5D9）
- 複数の動作モード
  - シングルスキャン
  - グループスキャン
  - 連続スキャン
- 複数チャネル
  - S7G2 の場合、 13 チャネル （ユニット 0）または 12 チャネル （ユニット 1）
  - S124 用の 18 チャネル
  - S3A7 用の 28 チャネル

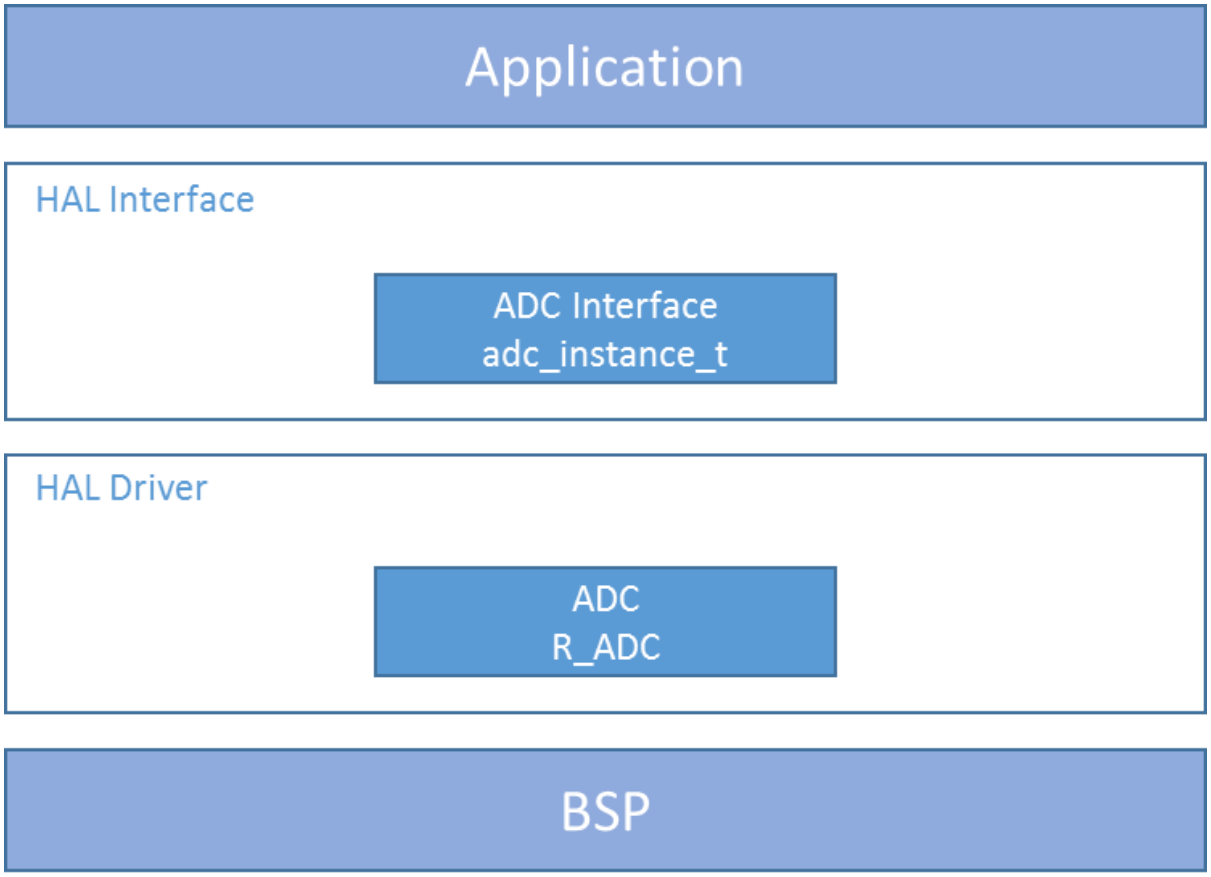


図 302: ADC HAL モジュールのブロック図

5.2.1.1 ADC HAL モジュールの API の概要

ADC HAL モジュールでは、ADC ユニットを開き、スキャンの構成、開始、停止を行い、ADC スキャンの変換結果を読み取り、ADC ユニットを閉じるための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

Function Name	API の呼び出し例と説明
open	g_adc.p_api->open(g_adc.p_ctrl, g_adc.p_cfg); 電源を適用し、すべてのチャネルとセンターに共通の動作モード、トリガーソース、割り込みの優先度、および設定値を指定します。

Function Name	API の呼び出し例と説明
scanCfg	g_adc.p_api->scanCfg(g_adc.p_ctrl, g_adc.*p*_channel_cfg); open 呼び出しで初期化されたユニットに使用されるチャネル、グループ、およびスキャントリガーを含むスキャンを設定します。
scanStart	g_adc.p_api->scanStart(g_adc.p_ctrl); スキャンを開始する（ソフトウェアトリガーの場合）か、ハードウェアトリガーを有効にします。
scanStop	g_adc.p_api->scanStop(g_adc.*p_ctrl*); ADC スキャンを停止する（ソフトウェアトリガーの場合）か、ハードウェアトリガーを無効にします。
scanStatusGet	g_adc.p_api->scanStatusGet(g_adc.p_ctrl); スキャンステータスをチェックします。
read	g_adc.p_api->read(g_adc.p_ctrl, ADC_REG_CHANNEL_13, &adc_data); ADC 変換結果を読み取ります。
sampleStateCountSet	g_adc.p_api-> sampleStateCountSet(g_adc.p_ctrl,&adc_sample); 指定されたチャネルのサンプル状態カウントを設定します。
close	g_adc.p_api->close(g_adc.p_ctrl); 進行中のスキャンを停止して、割り込みを無効にし、指定された A/D ユニットへの電源を遮断することによって、指定された ADC ユニットを閉じます。

Function Name	API の呼び出し例と説明
infoGet	g_adc.p_api->infoGet(g_adc.p_ctrl, &adc_info); 構成されたすべてのチャネルの変換結果を DTC/DMAC で読み取るために、先頭（最も小さい番号の）チャネルの ADC データのレジスタアドレスと読み取る合計バイト数を返します。
versionGet	g_adc.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

***注：** 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、関連モジュールの 『Renesas Synergy Software Package Reference』 を参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です。
SSP_ERR_NOT_OPEN	ユニットが開いていません。
SSP_ERR_ASSERTION	パラメータ p_ctrl または p_sample が NULL です。
SSP_ERR_IN_USE	ペリフェラルが別のモードで実行しています。先に R_ADC_Close を実行してください。
SSP_ERR_INVALID_POINTER	パラメータ p_data が NULL です。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、関連モジュールの 『Renesas Synergy Software Package Reference』 を参照してください。*

5.2.1.2 ADC HAL モジュールの動作の概要

ADC HAL モジュールは、ユーザーの構成に従って、 Synergy マイクロコントローラ上の ADC ユニットを制御します。RTOS 要素を使用しないで ADC ハードウェアを直接制御し、開発が簡単になる便利な API を提供します。このモジュールは、シングルスキャン、連続スキャン、グループスキャンの 3 つの動作モードをサポートしています。

- シングルスキャンモードは、チャネル番号の昇順にチャネルを順番に選択して、トリガごとに 1 回だけアナログ入力を変換します。
- 連続スキャンモードは、チャネル番号の昇順にチャネルを順番に連続して選択し、アナログ入力を変換します。スキャンを開始するために必要なトリガは 1 回だけです。
- グループスキャンモードでは、チャネルを 2 つのグループ（A と B）のどちらかに追加でき、グループごとに指定されている開始トリガを受け取ると、そのグループのチャネルがチャネル番号の昇順に選択されてアナログ入力が変換されます。

グループモードでは、グループ A とグループ B の 2 つのグループのどちらかに、動作チャネルを割り当てることができます。スキャンを開始するトリガはグループごとに割り当てられます。グループモードではハードウェアトリガのみを使用できるのに対し、通常モードではソフトウェアトリガまたは外部トリガを使用できます。プライオリティ構成パラメータを使用すると、以下のことを指定できます。

- あるグループのトリガが、他のグループの進行中のスキャンを中断できるかどうか。
- 中断されたスキャンを、再開するか、最初から開始し直すか、単に現在のスキャンを中止して次のトリガを待つかどうか。

割り込みとコールバック

ユーザーから（割り込みが有効な）コールバックが提供されている場合、スキャンが完了すると、ADC HAL モジュールは、コールバック (p_callback) を呼び出して、ユニットとイベント adc_cb_event_t. を示す引数 adc_callback_args_t を渡します。割り込みが有効になっていない場合は、スキャンが完了したかどうかをポーリングするためのスキャンステータスの確認が API でサポートされ（scanStatusGet）、変換後の ADC の結果を読み取るための関数が用意されています。

ADC HAL モジュールは 2 つの割り込みをサポートしています。

- 通常 / グループ A 割り込みは、シングルスキャンモードでスキャンが完了したとき、またはグループモードでグループ A のスキャンが完了したときに発生します。
- グループ B 割り込みは、グループモードでグループ B のスキャンが完了すると発生します。

割り込みの機能はモードによって異なります。

- シングルスキャンモードでは、スキャンが完了すると通常の割り込みがトリガされます。
- 連続スキャンモードでは、選択されたチャネルをハードウェアが常にスキャンします。このモードでは、割り込みが有効になっていると、ドライバーがエラーを返します（したがって、割り込みを無効にする必要があります）。
- グループモードでは、ADC ユニットは通常割り込み（グループ A 割り込みとも呼ばれます）とグループ B 割り込みの 2 つの割り込みをサポートしています。グループ A 割り込みは、グループ A スキャンが完了するとトリガされます。グループ B 割り込みは、グループ B スキャンが完了するとトリガされます。グループモードでは、通常割り込みは、同じベクトルであってもグループ A 割り込みと呼ばれます。

以下の状況では、BSP で選択されているユニットに対して、ADC スキャン完了割り込みを有効にする必要があります。

- 通常のスキャンが完了したときに割り込みを取得する。
- グループスキャンが完了したときに割り込みを取得する。

ADC HAL モジュールの動作に関する重要な注意事項と制限事項

サンプル状態カウントの設定

ユーザーは、sampleStateCountSet API を呼び出すことにより、サンプル状態カウントの設定を変更できます。サンプル状態カウントの設定をデフォルト値から変更する必要があるのは、入力信号のインピーダンスが高すぎて十分なサンプリング時間が確保できないか、 ADCLK が低すぎるために、サンプリング時間を長くする必要がある場合のみです。指定したチャネルのサンプル状態カウントを変更するには、チャネル番号 adc_sample_state_reg_t と状態の数 num_states. を設定します。有効なサンプル状態カウントは 5 ～ 255 です。複数のチャネルのサンプル状態カウントを変更するには、チャネルごとに引数を変えて sampleStateCountSet() API を繰り返し呼び出します。

ADC でのデータ転送のトリガ

ADC スキャンが完了したときにデータの転送をトリガするには、 activation_source を ELC_EVENT_ADCn_SCAN_END または ELC_EVENT_ADCn_SCAN_END_B（n は ADC チャネル番号）に設定してデータ転送を構成します。 ELC イベントは elc_event_t. に列記されています。 ADC ユニット固有の情報を取得して転送 API で使用するには、転送 API で infoGet 関数を呼び出します。

ADC を使用した ELC イベントのトリガ

ADC ユニットは、elc_peripheral_t. にリストされている他のペリフェラルの起動をトリガーできます。詳細については、『SSP ユーザーズマニュアル』の 「ELC インタフェース」を参照してください。

ADC での温度センサーの使用

ADC HAL モジュールは、オンチップ温度センサーからのデータの読み取りをサポートしています。センサーから返された値は、T = (Vs – V1)/ スロープ + T1 という式を使用して、ユーザーアプリケーションでセ氏またはカ氏に変換できます。変数の意味は次のとおりです。

- T: 測定温度 (°C)
- Vs: 温度センサーによる温度測定時の電圧出力 （ボルト）
- T1: 1 地点で実験的に測定された温度 (°C)
- V1: 温度センサーによる T1 測定時の電圧出力 （ボルト）
- T2: 別の地点で実験的に測定された温度 (°C)
- V2: 温度センサーによる T2 測定時の電圧出力 （ボルト）
- スロープ : 温度センサーの温度勾配 (V/°C)、スロープ = (V2 – V1)/(T2 – T1)

スロープの値は、各デバイスのハードウェアマニュアルの電気特性、TSN 特性から取得できます。スロープは、 S7/S5 デバイスでは正、 S3/S1 デバイスでは負です。

ADC HAL モジュールで使用する ADC チャネルを構成するときに、他の使用可能なチャネルも選択する場合は、温度センサーと電圧センサーを選択しないでください。温度センサーだけ、電圧センサーだけ、または任意の数の通常 ADC チャネルを使用できます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.1.3 アプリケーションへの ADC HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに ADC HAL モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ADC HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/ 共通スレッドに単純に追加します。（ADC HAL モジュールのデフォルト名は g_adc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

ADC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_adc0 ADC Driver on r_adc	Threads	New Stack> Driver> Analog> ADC Driver on r_adc

次の図に示すように、r_adc の ADC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

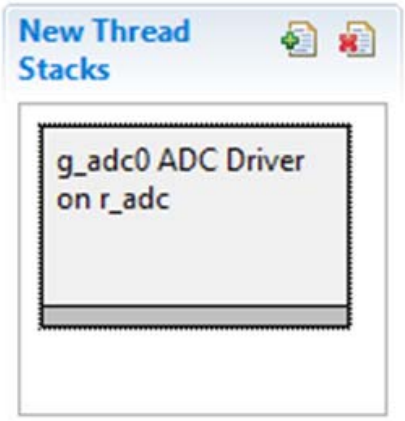


図 303: ADC HAL モジュールのスタック

5.2.1.4 ADC HAL モジュールの構成

ユーザーは必要な動作に合わせて ADC HAL モジュールを構成する必要があります。 SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、 ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定

とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注 : ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

r_adc の ADC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	選択するとパラメータチェック用のコードがビルドに含まれます。
Name	g_adc0	モジュール名
Unit	0, 1 (S7G2 Only) Default: 0	使用する ADC ユニットを指定します。S7G2 には、0 と 1 の 2 つのユニットがあります。
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2 only), 8-bit (S7G2 only) Default: 8-Bit (S7G2 Only)	このユニットの変換解像度を指定します。
Alignment	Right, Left Default: Right	変換結果のアラインメントを指定します。

ISDE Property	Value	説明
Clear after read	Off, On Default: On	変換結果を読み取った後で、結果レジスタを自動的にクリアするかどうかを指定します。 注: これが有効になっている場合、デバッガを使用して結果レジスタをウォッチすると、常に 0 になります。
Mode	Single Scan, Continuous Scan, Group Scan Default: Single Scan	この ADC ユニットを使用するモードを指定します。
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。

ISDE Property	Value	説明
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B (Default: Unused)	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	通常の動作モードでは、このビットマスクフィールドを使用して、その ADC ユニットで有効なチャネルを指定します。たとえば、0x101 に設定されている場合、チャネル 0 と 2 が有効になります。グループモードでは、このフィールドはグループ A に属するチャネルを指定するために使用されます。
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	温度センサーはチャネルスキャンマスクの選択肢を使用します。
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	電圧センサーはチャネルスキャンマスクの選択肢を使用します。

ISDE Property	Value	説明
Scan Mask Group B	Use #define ADC_MASK_xxx which are defined in r_adc.h.Use (ADC_MASK_xxxADC_MASK_xxx) for multiple channels.	このフィールドは、グループモードでのみ有効です。このフィールドは、グループ B に属するチャネルを指定するために使用されます。 警告：同じチャネルをグループ A と B で指定しないようにしてください。
Normal/Group A Trigger	None, Asynchronous External Trigger 0, ELC Event, Software Default: Software	このユニットに使用するトリガタイプを指定します。グループモードを mode で使用する場合、このフィールドは、グループ A トリガーを設定するために使用されます。 注：グループモードで有効な唯一のオプションは ELC トリガです。
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	グループ B トリガを指定します。このオプションは、mode でグループモードが選択されている場合のみ有効です。
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously (Default: Group A cannot interrupt Group B)	進行中のグループ B のスキャンをグループ A のトリガで割り込むことができるかどうか、グループ A のトリガで中断するかどうか、または一時停止してグループ A のスキャンを許可し、グループ A のスキャンが完了したらすぐに再開するかを決定します。 注: このフィールドは、グループモードでのみ有効です。
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	このユニット内のいずれかのチャネルに対して、加算または平均化を行う必要があるかどうかを指定します。実際のチャネルはチャネルマスク add_mask を使用して指定します。

ISDE Property	Value	説明
Channels 0-27	Disabled, Enabled Default: Disabled	このフィールドは、add_average_count が有効になっている場合にのみ有効です。平均化または合計するチャネル結果を決定するために使用します。
Temperature Sensor	Disabled, Enabled Default: Disabled	温度センサーは追加マスクまたは平均マスクの選択肢を使用します。
Voltage Sensor	Disabled, Enabled Default: Disabled	電圧センサーは追加マスクまたは平均マスクの選択肢を使用します。
Channels 0-2	Disabled, Enabled Default: Disabled	チャネル 0、1、2 のどれが、sample_hold_states で指定されたサンプルアンドホールド状態の更新値を使用するかを決定します。このフィールドは、チャネル 0、1、および 2 のデフォルトのサンプルアンドホールドカウント値を変更する場合にのみ設定する必要があります。
Sample Hold States (Applies only to the 3 channels selected above)	24	チャネル専用のサンプルアンドホールド回路用の、更新されたサンプルアンドホールドカウントを指定します。このフィールドは、sample_hold_mask が 0 でない場合にのみ有効です。チャネル 0、1、2 のみに専用のサンプルアンドホールド回路があります。 注 : 値をサンプリングするデフォルトの状態数 (24) を変更するにはこのフィールドを使用します。各状態は、1/ADCLK 時間に等しくなります。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 ADC スキャンが完了するたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	スキャン終了割り込み優先順位の選択
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	スキャン終了グループ B 割り込み優先順位の選択

***注:** 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ADC HAL モジュールのクロック構成

ADC HAL モジュールは PCLKC をクロックソースとして使用します （ADCLK）。このクロックを構成するときの唯一の制限事項は、最大 ADC クロックより小さい値に設定することです。また、PCLKC クロックと PCLKB クロックの比率には、ハードウェアマニュアルで指定されている制限があります。

ADC の変換時間は、 PCLKC の設定に依存します。

PCLKB と PCLKC の周波数を設定するには、 ISDE のクロックコンフィギュレータを使用します。

実行時にクロック周波数を変更するには、 CGC インタフェースを使用します。

ADC HAL モジュールのピン構成

ADC HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ISDE のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、 ISDE[Configuration] ウィンドウでのピンの選択方法を示します。

ADC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog Pins>ADC0\1 > AN_XX

5.2.1.5 アプリケーションでの ADC HAL モジュールの使用

モジュールを構成してファイルの生成が済むと、 ADC はアプリケーションで使用できる状態になります。

アプリケーションで ADC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して ADC を初期化します。
2) scanCfg API を使用してチャネルを構成します
3) scanStart API で必要なトリガを使用して変換を開始します
    a) ハードウェアトリガを使用する場合、この呼び出しにより、ADC ユニットをハードウェアトリガでトリガできるようになります。ソフトウェアトリガを使用する場合、この呼び出しにより ADC スキャンが開始されます。
4) 割り込みが無効になっている場合は、scanStatusGet を呼び出して、スキャンが完了したかどうかを特定します。
5) 割り込みが有効になっている場合は、スキャンが完了するとコールバックが呼び出されます。
6) read API を使用して変換結果を読み取ります。
7) scanStop API を呼び出して ADC のスキャンを停止します
    a) これにより、 ADC は外部トリガまたはハードウェアトリガによってトリガされなくなります。また、進行中のソフトウェアトリガスキャンがある場合は強制的に停止されます。
8) 受信したデータをアプリケーションの必要に応じて操作します。
9) close を呼び出して、ペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フローに示します。

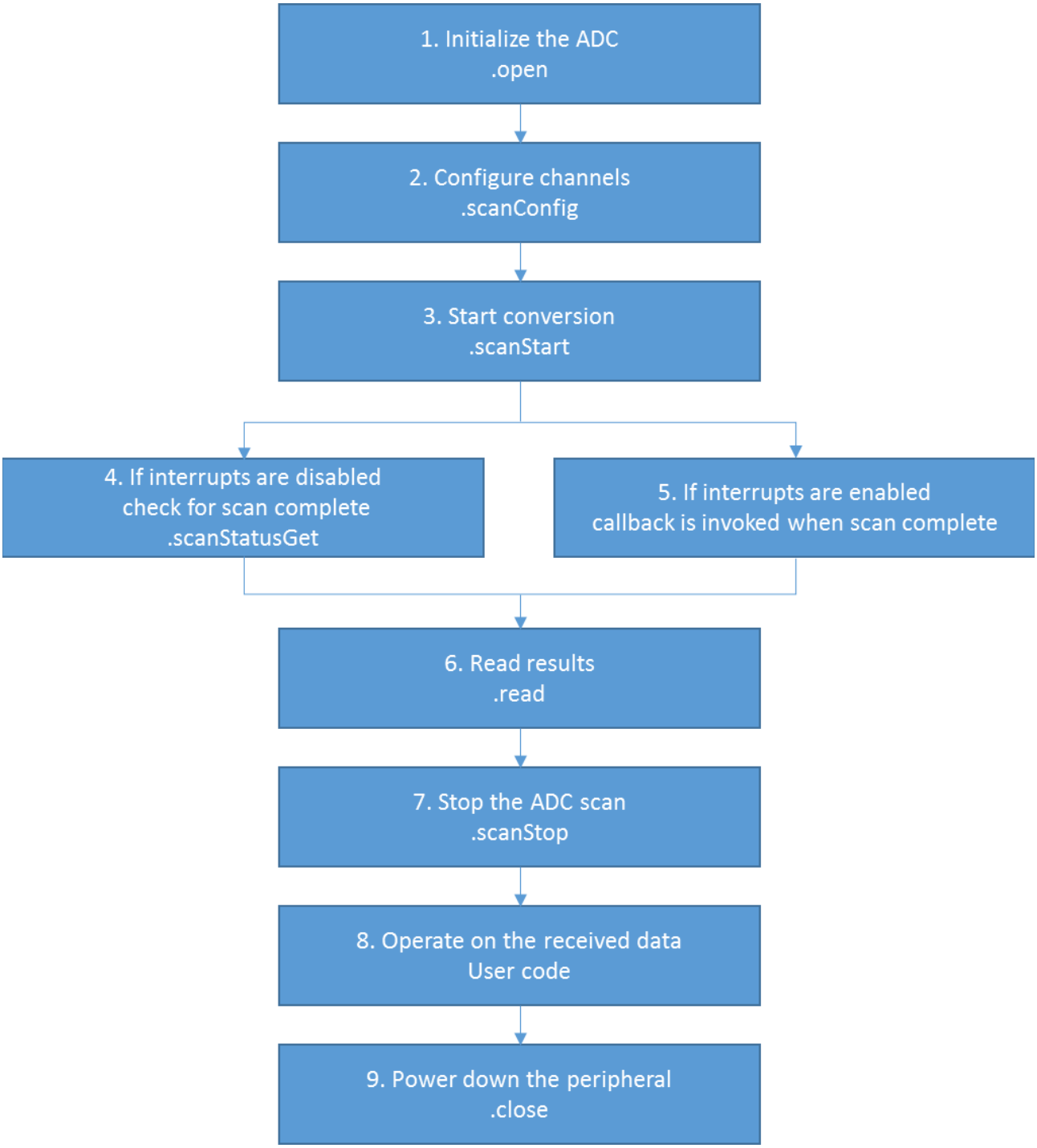


図 304:　一般的な ADC HAL モジュールアプリケーションのフロー図

5.2.2 AGT ドライバー

- 複数チャネル : 16 ビットのチャネルが 2 個
  - チャネル 1 はチャネル 0 のアンダーフローによってクロック供給することができ、カスケードされた 32 ビットタイマが作成されます。
- コアクロック : PCLKB、LOCO、または Fsub を使用してクロックを供給できます。LOCO または Fsub でクロックが供給された場合、それを使用して MCU をスリープモードから起動できます。

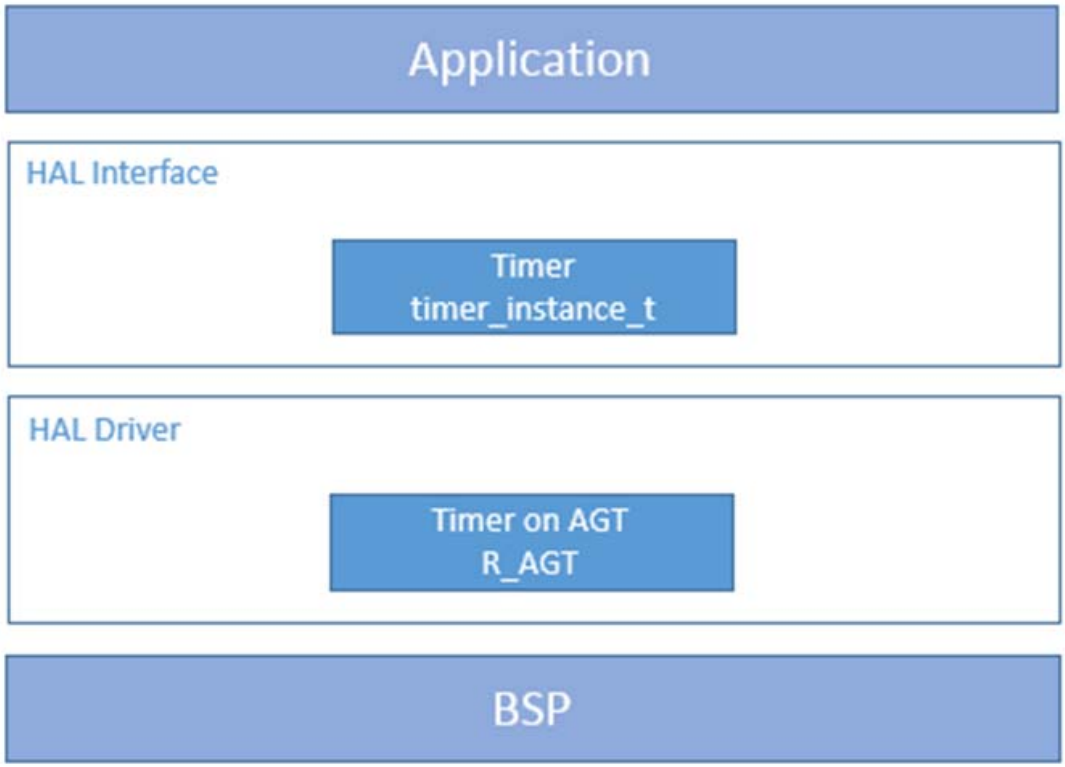


図 305: AGT HAL モジュールのブロック図

5.2.2.1 AGT HAL モジュールの API の概要

AGT HAL モジュールでは、タイマをオープン、クローズ、開始、停止するための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

AGT HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg) 初期設定。
stop	g_timer0.p_api->stop(g_timer0.p_ctrl) カウンタを停止します。
start	g_timer0.p_api->start(g_timer0.p_ctrl) カウンタを開始します。
reset	g_timer0.p_api->reset(g_timer0.p_ctrl) カウンタを初期値にリセットします。
counterGet	g_timer0.p_api->counterGet(g_timer0.p_ctrl, &value) 現在のカウンタ値を取得し、指定されたポインタ value に格納します。
periodSet	g_timer0.p_api->periodSet(g_timer0.p_ctrl, period, unit) タイマが期限切れになるまでの時間を設定します。
dutyCycleSet	g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, period, unit, pin) 定義されたピンでデューティサイクルが切れるまでの時間を設定します。
infoGet	g_timer0.p_api->infoGet(g_timer0.p_ctrl, &info) タイマが期限切れになるまでの時間をクロックカウント単位で取得し、指定されたポインタ info に格納します。

Function Name	API の呼び出し例と説明
close	g_timer0.p_api->close(g_timer0.p_ctrl) ドライバーを再設定できるようになります。
versionGet	g_timer0.p_api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。

注: 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	操作が正常に行われました
SSP_ERR_ASSERTION	パラメータが NULL であるか、構成設定が許可されていません
SSP_ERR_IN_USE	指定したチャネルはすでに開かれています
SSP_ERR_IRQ_BSP_DISABLED	要求した割り込みは BSP で有効になっていません
SSP_ERROR_NOT_OPEN	チャネルが開かれていません
SSP_ERR_INVALID_ARG	無効な引数が提供されました
SSP_ERR_INVALID_HW_CONDITION	無効なハードウェア設定が検出されました
SSP_ERR_INVALID_PTR	ポインタパラメータが NULL でしたが、NULL ではない値が必要です

注: ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.2.2 AGT HAL モジュールの動作の概要

AGT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。時間が経過した時点で、CPU の割り込み、ポートピンの切り替え、DMAC または DTC を使用したデータ転送の開始、別のペリフェラルの動作開始のトリガなどを行うことができます。

次の図では、指定した時間後にポートピンの切り替えまたは CPU 割り込みの生成を行うためのフローチャートを示します。このフローチャートは、AGT カウンタと GPT カウンタの両方に適用されます。（AGT 動作の場合は GPT の参照を AGT の参照に置き換え、AGT はダウンカウンタです）。

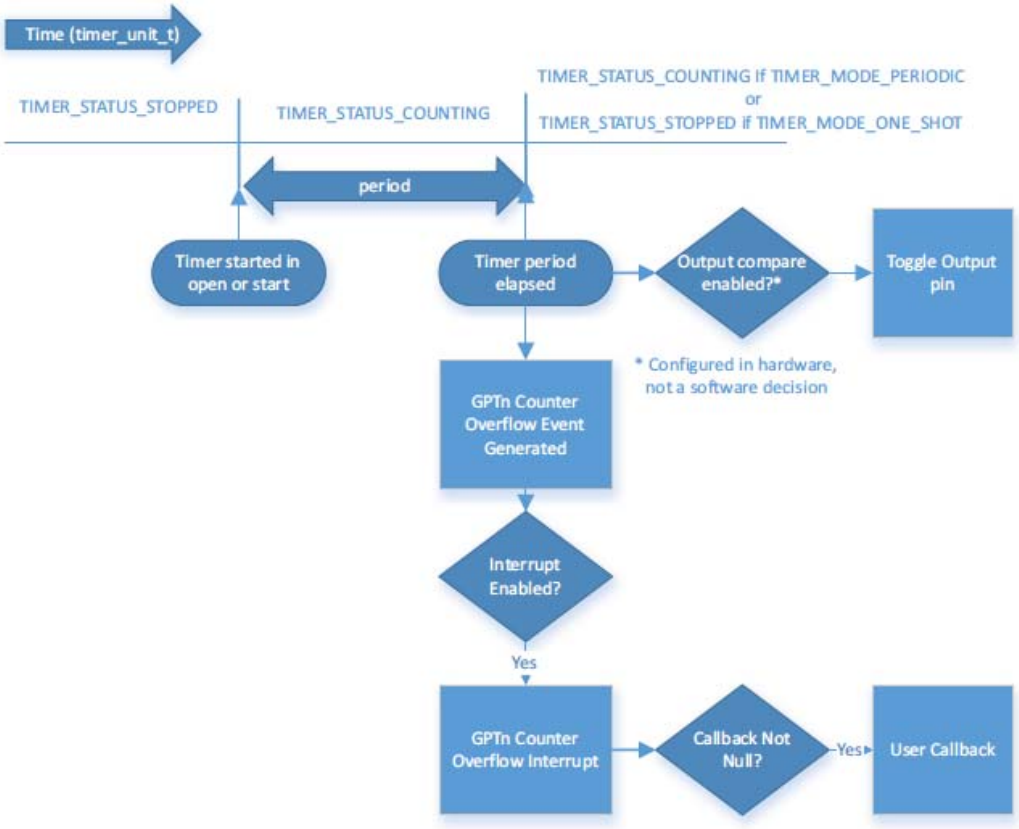


図 306: GPT タイマ - 周期またはワンショットモード

SSP では、GPT と AGT の 2 種類のタイマモジュールがサポートされます。以下のセクションでは、開発者が特定のアプリケーションに対する各モジュールの機能を比較して対照できるように、両方のモジュールについての情報を提供します。GPT の追加情報については、『GPT ユーザーズガイド』を参照してください。

GPT モジュールは、ほとんどの汎用タイマアプリケーションに推奨されますが、基本的なタイマ機能にはどちらのモジュールでも使用できます。一方のタイマモジュールが他方より推奨されるユースケースを以下で説明します。

GPT タイマモジュールの選択

GPT モジュールは、PCLKA のみでクロック供給可能な、高解像度の 32 ビットカウンタを使用します。Synergy デバイスでは AGT チャネルより GPT チャネルの方が多いので、GPT を使用することでリソースが競合する可能性が低くなります。

AGT タイマモジュールの選択

AGT モジュールは、PCLKB、LOCO、または Fsub でクロック供給可能な 16 ビットカウンタを使用します。LOCO または Fsub でクロック供給された場合、AGT 割り込みを使用して MCU をスリープモードから起動で

きます。2 つのチャネルがあり、チャネル 1 はチャネル 0 のアンダーフローによってクロック供給できるため、実質的に 32 ビットのカスケードされたタイマが作成されます。

AGT HAL モジュールの動作に関する重要な注意事項と制限事項

最大の時間間隔は、タイマの種類と入力クロック周波数に依存します。

- 120MHz で動作する PCLKA を使用した 32 ビット解像度の GPT では、最大期間は約 36650 秒であり、10 時間を少し超えます。
- 32MHz で動作する PCLKA を使用した 16 ビット解像度の GPT では、最大期間は約 2.09 秒です。
- 60MHz で動作する PCLKB を使用した 16 ビット解像度の AGT では、最大期間は約 8.7ms です。
- 32kHz で動作する Fsub を使用した 16 ビット解像度の AGT では、最大周期は約 2.0 秒です。

以下の状況では、選択した使用チャネルに対する AGT カウンタアンダーフロー割り込みを BSP で有効にする必要があります。

1）タイマ時間が経過したらソフトウェア割り込みを取得する。

2）ワンショットモードを使用する。

BSP で AGTn AGTI 割り込みが有効になっている場合、対応する ISR がタイマドライバーで定義されます。ISR は、ユーザーコールバック関数が open で登録されていれば、それを呼び出します。

***注:** IRQ が TRANSFER_IRQ_END. に設定された DTC で使用した場合、割り込みはスキップされることがあります。*

AGT 出力タイマ信号

タイマ出力が構成されている場合（[AGTO Output Enabled] が true に設定されている）、出力ピンは、出力反転が True に構成されている場合は高レベルで開始し、False に構成されている場合は低レベルで開始します。出力ピンは、タイマ開始後最初の時間間隔が経過して以降、時間間隔が経過するたびに切り替わります。

ワンショットモードでは、タイマのカウントが開始されるときにも切り替わるように出力が設定されます。これによりパルスが生成されます。タイマは、カウントが始まるときに停止レベルから切り替わり、カウントが終了するときに停止レベルに戻ります。

タイマ期間の計算

タイマ期間は、タイマが期限切れになるまでの時間として定義されます。出力比較を使用する場合、出力ピンが時間間隔ごとに 1 回切り替わるため、従来の時間間隔（立ち上がりエッジから立ち上がりエッジ）はソフトウェアで指定された時間間隔の 2 倍になります。

現在のクロック設定に基づく実行時の時間間隔の計算は、 open および periodSet から使用できます。

指定されたタイマ時間間隔が raw カウントと異なる場合は、現在のタイマクロック周波数を使用して時間間隔が計算されます（「GPT クロックの設定」または 「AGT クロックの設定」を参照してください）。現在のタイマクロック周波数を確認するには、 systemClockFreqGet を使用します。この周波数は、次の表の適切な式で、 clk_freq_hz. として使用されます。

タイマ期間の計算

Timer Units	説明
TIMER_UNIT_PERIOD_NSEC	カウント = （時間間隔 * clk_freq_hz） / 1000000000
TIMER_UNIT_PERIOD_USEC	カウント = （時間間隔 * clk_freq_hz） / 1000000
TIMER_UNIT_PERIOD_MSEC	カウント = （時間間隔 * clk_freq_hz） / 1000
TIMER_UNIT_PERIOD_SEC	カウント = （時間間隔 * clk_freq_hz）
TIMER_UNIT_FREQUENCY_HZ	カウント = （clk_freq_hz） / 時間間隔
TIMER_UNIT_FREQUENCY_KHZ	カウント = （clk_freq_hz） /（1000 * 時間間隔）

必要な時間間隔がカウンタのサイズ（32 ビットまたは 16 ビット）より長い場合、ドライバーは結果がカウンタのサイズに収まる最も小さい除数を選択します。カウンタ値が、最大の除数 (1024) を持つカウンタサイズよりも大きい場合は、エラーコード (SSP_ERR_INVALID_ARGUMENT) が返されます。

GPT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_GPTn_COUNTER_OVERFLOW（n は GPT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、DMAC または DTC のガイドを参照してください。

***注:** DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマの時間間隔が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH に設定するか、DMAC を使用して転送します。*

GPT を使用した ELC イベントのトリガー

GPT タイマは、他のペリフェラルの起動をトリガできます。ELC のガイドには、すべての使用可能なペリフェラルのリストがあります。

AGT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_AGTn_AGTI（n は AGT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、「転送インタフェース」を参照してください。

***注:** DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマの時間間隔が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH, に設定するか、DMAC を使用して転送します。*

AGT を使用した ELC イベントのトリガー

AGT タイマは、他のペリフェラルの起動をトリガできます。ELC のガイドには、elc_peripheral_t. に列記されているすべての使用可能なペリフェラルのリストがあります。（詳細については、イベントとペリフェラルの定義を参照してください）。

32 ビットタイマを作成するカスケードされた AGT タイマ

AGT チャネル 1 は AGT チャネル 0 のアンダーフローによってクロック供給することができ、カスケードされた 32 ビットタイマが作成されます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.2.3 アプリケーションへの AGT HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに AGT HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

AGT ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（AGT ドライバーのデフォルト名は g_timer0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

AGT ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_agt0 AGT Driver on r_agt	**Threads**	**New Stack> Driver> Timers> AGT Driver on r_agt**

次の図に示すように、r_agt の AGT HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。

図 307: AGT HAL モジュールのスタック

5.2.2.4 AGT HAL モジュールの構成

ユーザーは必要な動作に合わせて AGT HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_timer0	モジュール名。
Channel	0	物理ハードウェア チャネル。
Mode	Periodic, One Shot Default: Periodic	警告 : ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。
Period Value	10	「タイマ期間の計算」を参照

ISDE Property	Value	説明
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Microseconds	「タイマ期間の計算」を参照
Auto Start	True, False Default: True	設定後にタイマを起動するには、true に設定します。start を呼び出すまで測定を無効状態にするには false に設定します。
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	AGT カウンタのクロックソース。
AGTO Output Enabled	True, False Default: False	AGT 用に設定されたポートピン（AGTO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
AGTIO Output Enabled	True, False Default: False	AGT 用に設定されたポートピン（AGTIO ピン）上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
Output Inverted	True, False Default: False	出力信号 low を開始するには false を設定します。出力信号 high を開始するには true を設定します。

ISDE Property	Value	説明
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	タイマ割り込み優先順位。0が最高のプライオリティです。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

デフォルト以外の設定が望ましい場合もあります。たとえば、異なる時間間隔やデューティサイクルの値を選択する場合に役立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

AGT HAL モジュールのクロック構成

AGT タイマは、PCLKB、LOCO、Fsub、または AGT アンダーフロー周波数に基づいてクロック供給されます。 AGT クロックは、 e^2 studio の [Properties] ウィンドウで選択できます。クロック周波数は、 e^2 studio の [Configuring Clocks] で、または実行時に CGC インタフェースで、クロックコンフィギュレータを使用して設定できます。

AGT HAL モジュールのピン構成

AGT 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は関連するピンの選択例を示しています。

注: *動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。*

AGT HAL ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
AGT	Pins	Select **Peripherals > Timer: AGT>AGT0**

***注：** 選択シーケンスでは、AGT0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

AGT HAL ドライバーのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Custom, Timer Output, Compare Match, Count Measurement, Gated Count (Default: Disabled)	タイマ動作モードを選択します
AGTIO	None (Default: None)	AGTIO ピン
AGTO	None, P102 (Default: P102)	AGTO ピン
AGTOA	None (Default: None)	AGTOA ピン
AGTOB	None (Default: None)	AGTOB ピン
AGTEE	None, P101 (Default: P101)	AGTEE ピン

***注：** 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.2.5 アプリケーションでの AGT HAL モジュールの使用

アプリケーションで AGT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 AGT HAL モジュールを初期化します。
2) start API を呼び出して、 AGT HAL モジュールを開始します。
3) counterGet API を呼び出して、カウンタ値を読み取ります。
4) periodSet API を呼び出して、時間間隔の値を読み取ります。
5) dutyCycleSet API を使用して、デューティサイクル (duty_cycle_counts) を設定します。
6) infoGet API を使用して、タイマの情報を取得します。
7) 必要に応じて、 AGT HAL モジュールのコールバックに応答します。
8) reset API を使用して、カウンタの値をリセットします。
9) stop API を使用して、 AGT チャネルを停止します。
10) close を呼び出して、ペリフェラルの電源をオフにします。

注: *アプリケーションのニーズに基づいて、 timer-period および duty-cycle パラメータを再構成できます。*

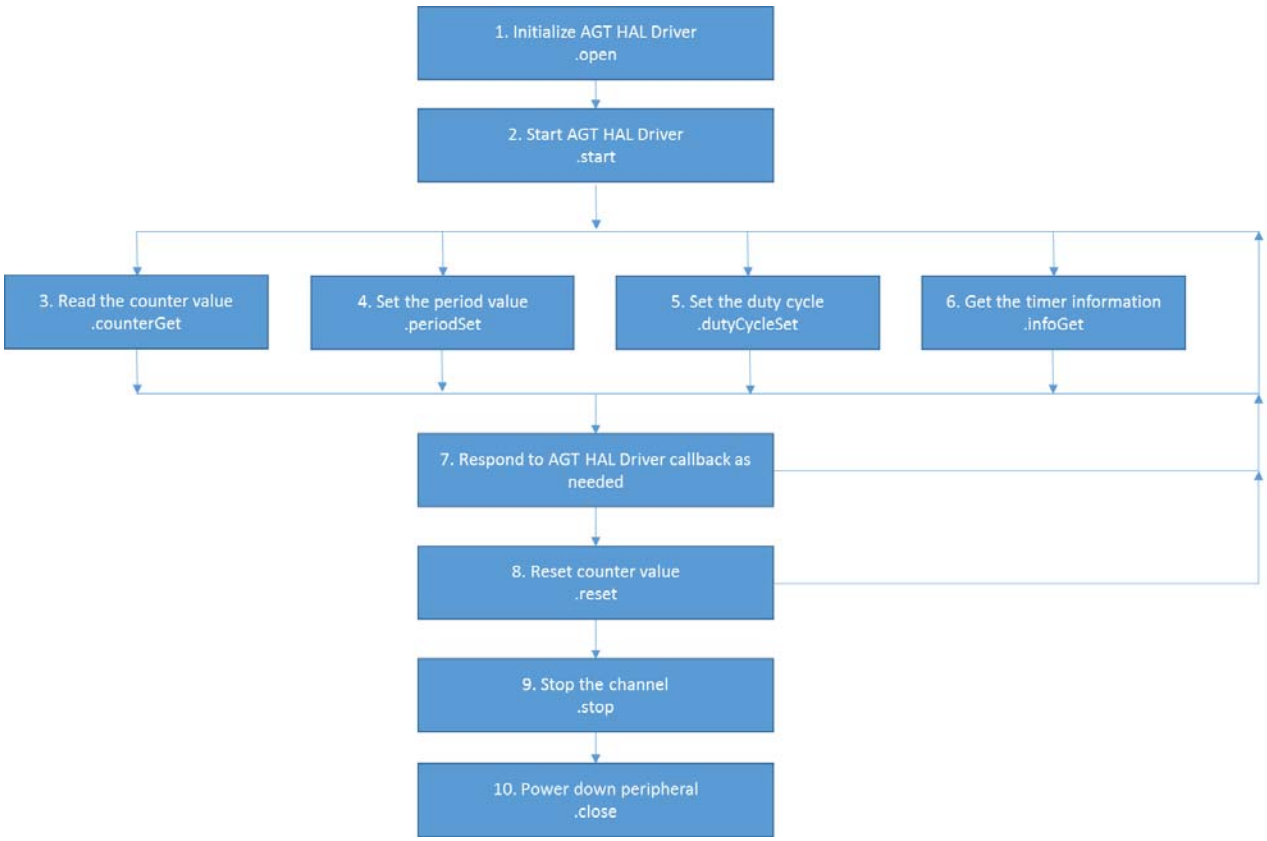


図 308: 一般的な AGT HAL モジュールアプリケーションのフローチャート

5.2.3 CAC ドライバー

CAC HAL モジュールの API には、参照信号入力に基づいてクロック周波数を監視できるクロック周波数測定回路との間にインタフェースがあります。参照信号は、外部から供給されるクロックソース、またはいずれかの内部クロックソースの可能性があります。割り込みリクエストは、完了した測定、検出された周波数エラー、またはカウンタオーバーフローによってオプションで生成されることがあります。外部から供給される参照クロックではデジタルフィルターが利用でき、除算値は内部提供の測定と参照クロックの両方で利用できます。参照クロックのエッジ検出オプションは、上昇、下降、またはその両方で構成できます。

次のクロックの周波数を測定できます。

- メインクロック発振器からのクロック出力（メインクロック）
- サブクロック発振器からのクロック出力（サブクロック）
- 高速オンチップ発振器のクロック出力（HOCO クロック）
- 中速オンチップ発振器のクロック出力（MOCO クロック）
- 低速オンチップ発振器のクロック出力（LOCO クロック）
- IWDT 専用オンチップ発振器のクロック出力（IWDTCLK クロック）

- 周辺モジュールクロック (PCLKB)

測定クロックは参照クロックを使って監視されます。参照クロックとしては、外部クロック（CACREF 入力ピンで提供）または次のいずれかの内部クロックを使用できます。

- メインクロック発振器からのクロック出力（メインクロック）
- サブクロック発振器からのクロック出力（サブクロック）
- 高速オンチップ発振器のクロック出力（HOCO クロック）
- 中速オンチップ発振器のクロック出力（MOCO クロック）
- 低速オンチップ発振器のクロック出力（LOCO クロック）
- IWDT 専用オンチップ発振器のクロック出力（IWDTCLK クロック）
- 周辺モジュールクロック (PCLKB)

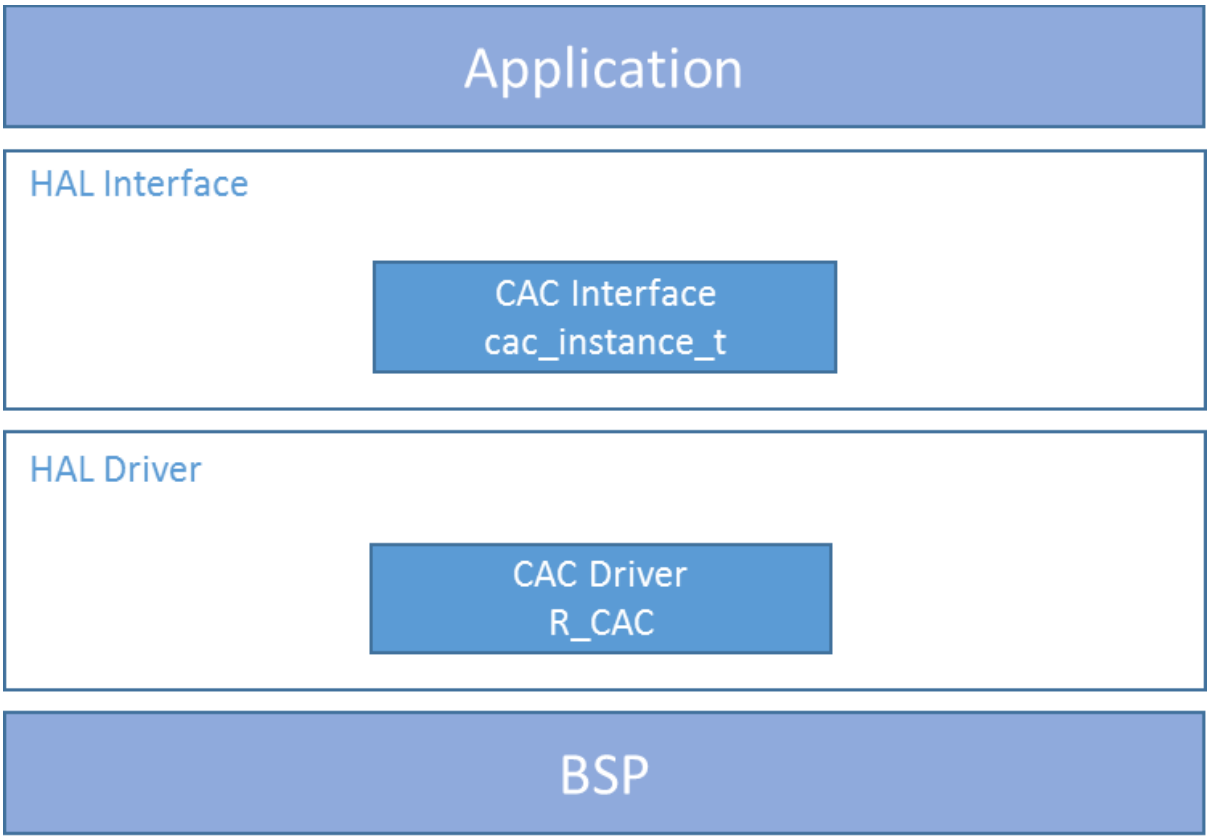


図 309: CAC HAL モジュールのブロック図

5.2.3.1 CAC HAL モジュールの API の概要

CAC HAL モジュールでは、CAC のオープン、クローズ、リード、開始、停止、リセットのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

CAC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_cac0.p_api->open(g_cac0.p_ctrl, g_cac0.p_cfg) CAC デバイス用の関数を開きます。
read	g_cac0.p_api->read(g_cac0.p_ctrl, &cac0_status, &cac0_counter) CAC ペリフェラル用の関数を読み取ります。
close	g_cac0.p_api->close(g_cac0.p_ctrl) CAC デバイス用の関数を閉じます。
stopMeasurement	g_cac0.p_api->stopMeasurement(g_cac0.p_ctrl) CAC の測定を終了します。
startMeasurement	g_cac0.p_api->startMeasurement(g_cac0.p_ctrl) CAC ペリフェラルの測定を開始します。
reset	g_cac0.p_api->reset(g_cac0.p_ctrl) CAC デバイス用の関数をリセットします。
versionGet	g_cac0.p_api->versionGet(&cac0_version) CAC API とコード バージョンに関する情報を取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	1 つまたは複数の構成オプションが無効です
SSP_ERR_NOT_OPEN	オープンが正常に呼び出されませんでした
SSP_ERR_ASSERTION	p_ctrl, p_cfg、その他に NULL が提供されました
SSP_ERR_INVALID_POINTER	コールバックが NULL の割り込みが指定されています
SSP_ERR_HW_LOCKED	CAC のハードウェアロックが既に取得されています
SSP_ERR_INVALID_CAC_REF_CLOCK	測定クロックレートが参照クロックレートより小さくなっています

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.3.2 CAC HAL モジュールの動作の概要

CAC HAL モジュールの主要な機能は、選択されたクロックの動作と精度を測定することです。測定が要求されると、参照クロックで最初に検出された有効なエッジでカウントが始まり、次の有効なエッジで終わります。有効なエッジは、上昇、下降、またはその両方に設定できます。カウントは、クロックを 1、4、8、または 32 で分割可能な分周器回路を経由した後に、測定クロックの各サイクルでインクリメントされます。内部で提供される参照クロックも、クロックを 32、128、1024、または 8192 で分割可能な分周器回路を経由します。外部から提供される参照クロックは分周器回路を経由しませんが、設定されている場合はデジタルフィルタを経由することがあります。

たとえば、サブクロックが測定クロック（32kHz）として指定されており、除数が 1 に指定されている場合、カウンタは 32kHz のレートでインクリメントします。1kHz の参照クロックが提供されている場合、参照クロックの 1 サイクル後には、カウンタが 32 になることが予想されます。ここで、 CAC の上限および下限の設定が確認されます。CAC 測定のセットアップの一部には、測定の上限と下限の仕様が含まれます。測定が完了すると、CAC はカウンタの内容と測定の制限を比較します。カウンタが上限以下かつ下限以上の場合、測定はエラーなしで完了し、測定された周波数は定義された制限内で運用されていたことになります。カウンタがこれらの条件を満たさなかった場合、周波数エラーが示されます。完了した測定は、 API を呼び出してドライバーをポーリングするか、コールバック関数を設定することによって、識別できます。

CAC HAL モジュールの動作に関する重要な注意事項と制限事項

連続モード

CAC モジュールは、単一測定モードまたは連続測定モードで動作できます。連続モードでは、各測定の完了後に測定プロセスが再度開始されます。非連続モードつまり単一測定モードでは、最初の測定が完了した後で測定は停止します。この機能は、 continuous_mode 構成メンバーによって制御されます。

割り込みとコールバック

ユーザーから（1 つ以上の割り込みが有効な）コールバックが提供されている場合、測定が完了すると、CAC HAL モジュールは、コールバック (p_callback) を呼び出して、イベント cac_event_t. を示す引数 cac_callback_args_t を渡します。割り込みが有効になっていない場合は、API により測定が完了した（読み取られた）かどうかをポーリングして測定の状態を確認でき、測定の状態と CAC カウンタレジスタの現在値の両方が提供されます。

CAC ドライバーは 3 種類の割り込みをサポートしています。

- 周波数エラー割り込み。測定が完了し、CAC カウンタレジスタの値が CAC ドライバーオープンの一部として指定された範囲内ではない場合に発生します。この割り込みが生成されるためには、 open の呼び出しで提供される構成の ferr_interrupt_enabled メンバーが有効になっている必要があります。
- オーバーフローエラー割り込み。CAC カウンタレジスタが最大値（0xFFFF）をオーバーフローすると発生します。この割り込みが生成されるためには、 open の呼び出しで提供される構成の ovf_interrupt_enabled メンバーが有効になっている必要があります。
- 測定完了割り込み。測定が完了し、CAC カウンタレジスタの値が CAC ドライバーオープンの一部として指定された範囲内である場合に発生します。この割り込みが生成されるためには、 open の呼び出しで提供される構成の mei_interrupt_enabled メンバーが有効になっている必要があります。

リセット

測定が停止された後、 reset API を使用して、オーバーフロー、測定終了、周波数エラーの各割り込みフラグをリセットし、誤ったトリガを除去できます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.3.3 アプリケーションへの CAC HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに CAC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

CAC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（CAC のデフォルト名は g_cac0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

CAC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_cac0 CAC Driver on r_cac	Threads > HAL/Common	New Stack > Driver> Monitoring > Clock Accuracy Circuit Driver on r_cac

次の図に示すように、r_cac の CAC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

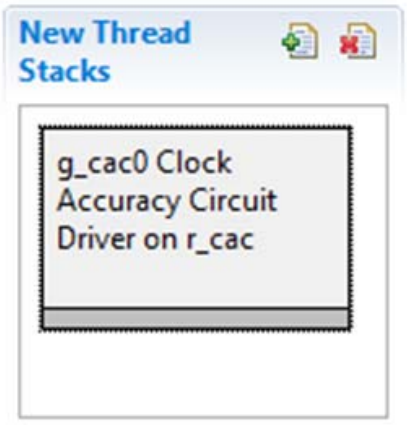


図 310: CAC HAL モジュールのスタック

5.2.3.4 CAC HAL モジュールの構成

ユーザーは必要な動作に合わせて CAC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_cac での CAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータ チェック用のコードを含めるかどうかを制御します。
Name	g_cac0	このインスタンスを特定します
Continuous Measurement Operation	Enabled, Disabled Default: Enabled	有効な場合、完了後に測定が継続して再開されます
Measurement Complete Interrupt	Enabled, Disabled Default: Enabled	有効にすると、測定の完了後に、 ICU で CAC MEASUREMENT END 割り込みが有効な場合に、CAC ドライバーは割り込みを生成できます。
Overflow Interrupt	Enabled, Disabled Default: Enabled	有効にすると、CAC オーバーフローが生じ、ICU で CAC OVERFLOW 割り込みが有効な場合に、CAC ドライバーは割り込みを生成できます。
Frequency Error Interrupt	Enabled, Disabled Default: Enabled	有効にすると、周波数エラーが生じ、ICU で CAC FREQUENCY ERROR 割り込みが有効な場合に、CAC ドライバーは割り込みを生成できます。
Upper Limit Threshold	0 – 0xFFFF Default: 0	測定完了で可能な範囲の上限
Lower Limit Threshold	0 – 0xFFFF, must be < Upper Limit threshold Default: 0	測定完了で可能な範囲の最低値
Reference Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT Default: Main Oscillator	参照クロック ソース

ISDE Property	Value	説明
Reference Clock Divider	32,128,1024,8192 Default: 32	参照クロック分周器
Reference Clock Edge Detect	Rising, Falling, Both Default: Rising	参照クロック エッジ検出
Reference Clock Digital Filter	Disabled, Sampling clock = Measuring freq, Sampling clock = Measuring freq/4, Sampling clock = Measuring freq/16 Default: Disabled	参照クロックデジタルフィルター
Measurement Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT Default: HOCO	測定クロックソース
Measurement Clock Divider	1,4,8,32 Default: 1	測定クロック分周器
Callback	NULL	コールバック用の関数名
Frequency Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled Default: Disabled	CAC の周波数エラー割り込み優先順位

ISDE Property	Value	説明
Measurement End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled Default: Disabled	CAC の測定終了割り込み優先順位
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled Default: Disabled	CAC のオーバーフロー割り込み優先順位

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なるクロックソースおよび分周器を選択する場合に役立つことがあります。

CAC HAL モジュールのクロック構成

アプリケーションの必要に応じて、[Clock] タブでクロックを選択します。

CAC HAL モジュールのピン構成

CAC HAL モジュールのピンは、次の表に示すように選択します。ピンの設定は後の表で示します。参照クロックの入力ピンに CACREF が使用されている場合は、このピンを構成する必要があります。

CAC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
CAC HAL Module	Pins	Peripherals > Monitoring: CAC > CAC0

注: *選択シーケンスでは、CAC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

CAC HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, External Reference (Default: Disabled)	CAC の動作モードとして Enable を選択します
CACREF:	None, P204 (Default: None)	CACREF のピン

注: 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.2.3.5 アプリケーションでの CAC HAL モジュールの使用

アプリケーションで CAC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 CAC HAL モジュールを初期化します
2) startMeasurement API を使用して、測定を開始します
3) CGC read 関数を使用してポーリングを行い測定結果を調べるか、 ISR で呼び出されるコールバック関数を使用して測定の状態と結果を取得します
4) CGC stopMeasurement API を使用して、測定を停止します
5) CGC reset API を使用して、オーバーフロー、測定終了、周波数エラーの各割り込みフラグをリセットします
6) それ以上測定が必要ない場合は、 CGC close API を使用して、 CAC HAL モジュールを閉じます

CAC HAL モジュールの使用に関するこれらの一般的な手順を、次の図の通常の動作フロー図に示します。

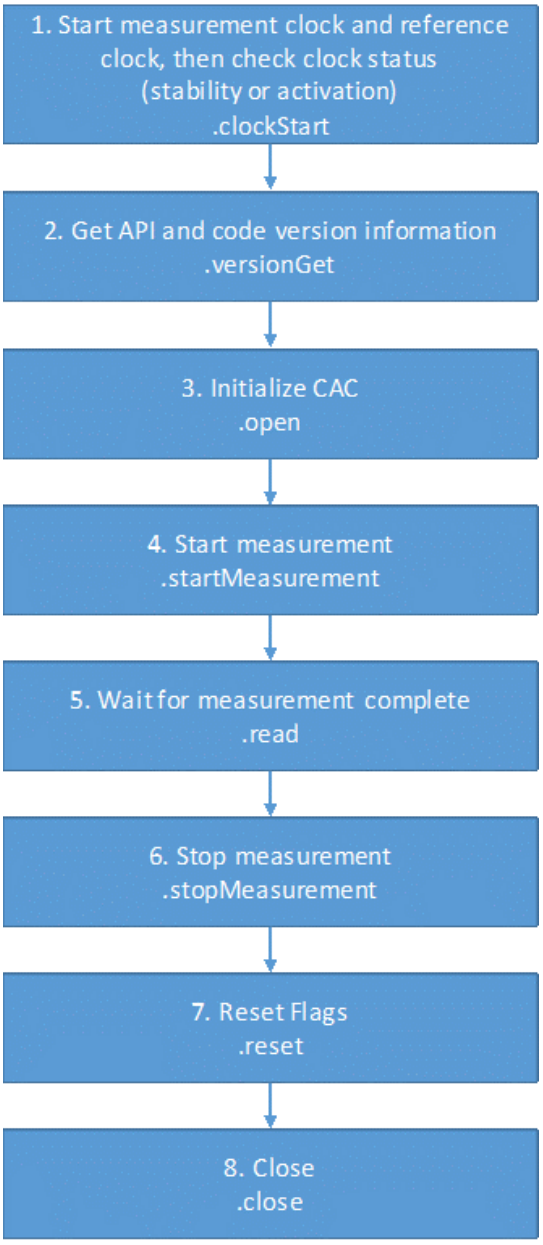


図 311: 一般的な CAC HAL モジュールアプリケーションのフロー図

5.2.4 CAN ドライバー

- 標準（11 ビット）と拡張（29 ビット）両方のメッセージ形式をサポートします
- CAN の仕様で定義されているビットタイミングの構成をサポートします
- 標準または拡張の ID フレームで最大 32 個の送信または受信メールボックスをサポートします

- データまたはリモートの CAN フレームをキャプチャするように、受信メールボックスを構成できます
- 送信、受信、またはエラーの割り込みを受け取ったときのユーザーコールバック関数をサポートします

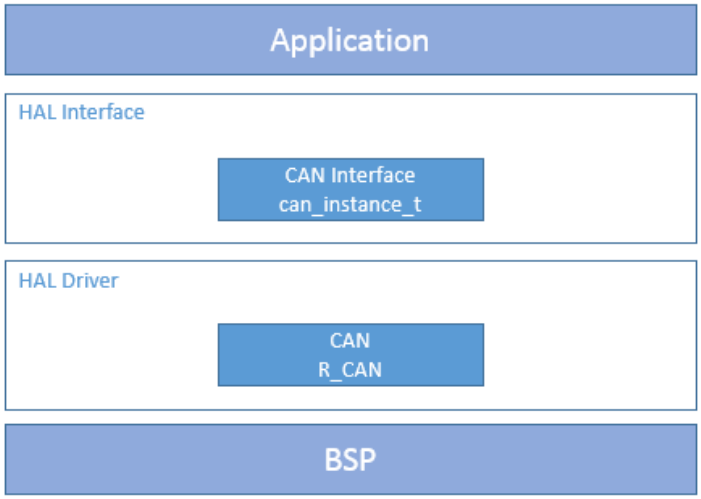


図 312: CAN HAL モジュールの編成、オプション、スタックの実装

5.2.4.1 CAN HAL モジュールの API の概要

CAN HAL では、CAN データのオープン、クローズ、ライト（送信）、リード（受信）のための API が定義されています。また、さらに複雑なコマンドの処理を補助するための control、Infoget、VersionGet などの追加関数も提供されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

CAN HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_can0.p_api->open(g_can0.p_ctrl, g_can0.p_cfg) open API は CAN チャネル 0 を構成します。この関数は他の CAN 関数の前に呼び出す必要があります。 注：この呼び出しは、ユーザースレッドに入る前、システムの初期化の間に自動的に行われます。ユーザーがモジュールを閉じる場合を除き、open を呼び出す必要はありません。

Function Name	API の呼び出し例と説明
close	g_can0.p_api->close(g_can0.p_ctrl) close API は、内部ドライバデータのクリーンアップを処理します。
read	g_can0.p_api->read (g_can0.p_ctrl, p_args->mailbox, &receiveFrame) read API は、受信した CAN データを読み取ります。
write	g_can0.p_api->write (g_can0.p_ctrl, 0, &transmitFrame) write API は、データを CAN 送信フレームバッファに書き込んで送出します。

control	g_can0.p_api->control(g_can0.p_ctrl, CAN_COMMAND_MODE_SWITCH, &mode);
	
	with can_mode_t mode = CAN_MODE_LOOPBACK_INTERNAL;
	
	control API は、動作の CAN モードを変更できます。

infoGet	g_can0.p_api->infoGet(g_can0.p_ctrl, p_info) infoGet API は、動作の CAN モードを取得します。
versionGet	g_can0.p_api->versionGet(version) versionGet API は、モジュールのバージョン情報を取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です。
SSP_ERR_HW_LOCKED	ロックは既に別のユーザーが所有しています。
SSP_ERR_CAN_MODE_SWITCH_FAILED	チャネルがモードの切り替えに失敗しました。
SSP_ERR_CAN_INIT_FAILED	チャネルが初期化に失敗しました。
SSP_ERR_ASSERTION	NULL ポインタが示されました。
SSP_ERR_NOT_OPEN	ポートが開いていません。
SSP_ERR_CAN_DATA_UNAVAILABLE	利用可能なデータがありません。
SSP_ERR_CAN_TRANSMIT_MAILBOX	メールボックスが受信用に設定されていません。
SSP_ERR_CAN_TRANSMIT_NOT_READY	送信を実行中です。現時点ではデータを書き込むことができません。
SSP_ERR_CAN_RECEIVE_MAILBOX	メールボックスが受信用に設定されておらず、送信できません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.4.2 CAN HAL モジュールの動作の概要

CAN HAL モジュールは、ユーザーの構成に従って Synergy マイクロコントローラ上の CAN を制御します。API は open、close、read、write、control、information の各関数を提供します。このドライバーでは、CAN の仕様で定義されているとおり、ビットタイミングの構成が可能です。これは、標準または拡張 ID フレームで最大 32 の送信または受信メールボックスで構成できます。受信メールボックスは、データまたはリモート CAN フレームをキャプチャするように構成できます。ユーザーコールバックは、送信、受信、またはエラーの割り込みが発生したときに、チャネル、メールボックス、イベントの情報と共に呼び出されます。

CAN の ID とマスクの使用

メッセージを受信するように構成された各 CAN メールボックスは、ID とマスクセットを持っています。受信メッセージは、以下の条件を満たす最も低いメールボックスに格納されます。

受信 ID& メールボックスマスク == メールボックス ID& メールボックスマスク

例 1: ID が 0x25 でマスクが 0x1FFFFFFF のメールボックスは、ID が 0x25 のメッセージを受信できます。

例 2: ID が 0x25 でマスクが 0x1FFFFFF0 のメールボックスは、ID が 0x20～0x2F のメッセージを受信できます。

CAN HAL モジュールのテストモードの使用

CAN モジュールには、受信待ちのみ、外部ループバック、内部ループバックの 3 つのテストモードがあります。

- 受信待ちのみモードでは、有効なデータフレームとリモートフレームを受信できます。ただし、CAN バスでは後退ビットのみを送信できます。ACK ビット、オーバーロードフラグ、アクティブエラーフラグは送信できません。受信待ちのみモードは、ボーレートの検出に使用できます。
- 外部ループバックモードでは、プロトコルモジュールは自分が送信したメッセージを CAN トランシーバによって受信されたメッセージ同じように扱い、受信メールボックスに格納します。外部シミュレーションと区別するため、プロトコルモジュールは ACK ビットを生成します。CTX ピンと CRX ピンをトランシーバに接続します。
- 内部ループバックモードは外部ループバックモードと似ていますが、プロトコルコントローラが内部 CTX ピンから内部 CRX ピンへの内部ループバックモードを実行することが異なります。外部 CRX ピンの入力値は無視されます。外部 CTX ピンは後退ビットのみを出力します。CTX ピンと CRX ピンを、CAN バスまたは何らかの外部デバイスに接続する必要はありません。

CAN HAL モジュールの動作モードの変更

CAN モジュールのモードは、control API を使用して切り替えることができます。CAN_COMMAND_MODE_SWITCH と、変更後のモードを設定された can_mode_t 変数へのポインタを、control API に渡します。

Mode	can_mode_t value	使用する理由
Normal	CAN_MODE_NORMAL	通常の動作モード
Internal Loopback	CAN_MODE_LOOPBACK_INTERNAL	内部ループバックテスト
External Loopback	CAN_MODE_LOOPBACK_EXTERNAL	外部ループバックテスト
Listen Only	CAN_MODE_LISTEN	ボーレートの検出
Halt	CAN_MODE_HALT	メールボックスの構成とテストモードの設定
Sleep	CAN_MODE_SLEEP	CAN モジュールへのクロック供給を停止して、電力消費を減らします
Exit Sleep	CAN_MODE_EXIT_SLEEP	内部使用のみ
Reset	CAN_MODE_RESET	通信の構成

***注:** control API を使用してループバックモードを設定する例については、前に示した「CAN HAL モジュールの API の要約」の表を参照してください。*

CAN HAL モジュールの動作に関する重要な注意事項と制限事項

- 実行時にメインのクロック発振器（CANMCLK または XTAL）がまだ開始されていない場合は（たとえば、MCU クロックソースとして使用されていない場合）、ユーザーアプリケーションで CGC インタフェースを使用して開始する必要があります。
- S7、S5、S3、S1 の各 MCU では、クロックソースがメインのクロック発振器（CANMCLK）である場合、CAN HAL モジュールに対して次のクロック制限事項が満たされている必要があります。fPCLKB >= fCANCLK（XTAL/ ボーレートプリスケーラー）
- S7、S5、S3 の各 MCU では、クロックソースが PCLKB の場合、周辺モジュールクロックのソースは CAN HAL モジュールの PLL である必要があります。
- S3 MCU では、CAN HAL モジュールを使用する場合、PCLKA と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- S1 MCU では、CAN HAL モジュールを使用する場合、ICLK と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- SJW（同期ジャンプ幅）は多くの場合、バス管理者から供給されます。1 <= SJW <= 4 を選択します。
  - タイムセグメントと SJW の設定は、次の制限に従う必要があります。TS1 > TS2 >= SJW。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.4.3 アプリケーションへの CAN HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CAN HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

CAN HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/ 共通スタックまたは任意のスレッドに単純に追加します。（CAN HAL モジュールのデフォルト名は g_can0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

CAN HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_can0 CAN Driver on r_can	Threads	New Stack> Driver> Connectivity> CAN Driver on r_can

次の図に示すように、r_can の CAN HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。

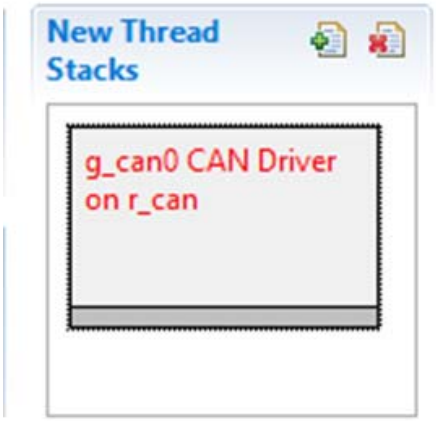


図 313: CAN HAL モジュールのスタック

5.2.4.4 CAN HAL モジュールの構成

ユーザーは必要な動作に合わせて CAN HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

***注:** ISDE を開き、次に示す構成テーブルの設定を見ながら、CAN HAL モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_can での CAN HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択します。
Error Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	エラー割り込みの優先度 0 ～ 15 を指定します（必須）。

ISDE Property	Value	説明
Receive Mailbox Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	エラー割り込みの優先度 0 ～ 15 を指定します（必須）。
Transmit Mailbox Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	エラー割り込みの優先度 0 ～ 15 を指定します（必須）。
Name	Default: g_can0	CAN ドライバーモジュールの名前。
Channel	0, 1	使用する CAN チャネルを指定します。0 または 1（S7G2 のみ）。
Baud Rate Prescaler	Default 5	ボーレートプリスケーラー (0 ～ 1023) を指定します。
Time Segment 1	15 Time Quanta	タイムセグメント 1 の値を指定します (4-16)。
Time Segment 2	8 Time Quanta	タイムセグメント 2 の値を指定します (2 ～ 8)。
Synchronization Jump Width	2 Time Quanta	同期ジャンプ幅の値を指定します (1 ～ 4)。
Clock Source	PCLKB (S7G2, S5D9, S5D5, S3A7, and S3A7 only), CANMCLK Default: CANMCLK	CAN クロックソース。CANMCLK または PCLKB（S7G2、S5D9、S5D5、S3A7、S3A7 のみ）。
Callback	NULL	ユーザー定義コールバック関数を can_api_t::open. で登録できます。このコールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービスルーチン(ISR)から呼び出されます。
Overwrite/Overrun mode	Overwrite Mode, Overrun mode Default: Overwrite mode	データが時間内に読み取られなかった場合に、受信メールボックスが上書きされるかオーバーランされるかを選択します。

ISDE Property	Value	説明
Standard or Extended ID Mode	Standard ID Mode, Extended ID Mode Default: Standard ID Mode	ドライバーが CAN 標準 ID、拡張 ID のどちらを使用するかを選択します。
Number of Mailboxes	32 Mailboxes	4、8、16、32 の中からメールボックス数を選択します。
Mailbox [0] ID … Mailbox [31] ID	Default n	メールボックス 0 の受信 ID を選択します（標準 ID を使用する場合は 0 ～ 0x7ff、拡張 ID を使用する場合は 0 ～ 0x1FFFFFFF)。メールボックスが送信型に設定されている場合、値は使用されません。
Mailbox [0] Type … Mailbox [31] Type	N = 0: Transmit Mailbox; N = 1 …31: Receive Mailbox	メールボックスが受信に使用されるか、送信に使用されるかを選択します。
Mailbox [0] Frame Type … Mailbox [31] Frame Type	N = 0: Remote Mailbox; N = 1 …31: Data Mailbox	メールボックスがデータフレームのキャプチャに使用されるか、リモートフレームのキャプチャに使用されるかを選択します（受信のみ）。
Mailbox 0-3 Group Mask	Default: 0x1FFF FFFF	メールボックス 0 ～ 3 のマスクを選択します。
Mailbox 4 - 7 Group Mask	Default: 0x1FFF FFFF	メールボックス 4 ～ 7 のマスクを選択します。
Mailbox 8 - 11 Group Mask	Default: 0x1FFF FFFF	メールボックス 8 ～ 11 のマスクを選択します。
Mailbox 12 - 15 Group Mask	Default: 0x1FFF FFFF	メールボックス 12 ～ 15 のマスクを選択します。
Mailbox 16 - 19 Group Mask	Default: 0x1FFF FFFF	メールボックス 16 ～ 19 のマスクを選択します。
Mailbox 20 - 23 Group Mask	Default: 0x1FFF FFFF	メールボックス 20 ～ 23 のマスクを選択します。
Mailbox 24 - 27 Group Mask	Default: 0x1FFF FFFF	メールボックス 24 ～ 27 のマスクを選択します。
Mailbox 28 - 31 Group Mask	Default: 0x1FFF FFFF	メールボックス 28 ～ 31 のマスクを選択します。

注: 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

CAN HAL モジュールのクロック構成

CAN は、CANMCLK（メインクロック発振器）または PCLKB（S7G2、S5D9、S5D5、S3A7、S3A7 のみ）をクロックソース（fCAN、CAN システムクロック）として使用します。PCLKB をデフォルトの 60MHz で、Synergy をデフォルト （S7G2 DK）の CAN 構成で使用すると、 CAN ビットレートは 500Kbit になります。

PCLKB の周波数を設定するには、e^2 studio のクロックコンフィギュレータを使用します（コンフィギュレータの [Clocks] タブ）。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。クロックの構成の詳細については、 CGC モジュールのガイドを参照してください。

CAN HAL モジュールのピン構成

CAN 周辺モジュールは、MCU のピンを使用して、CAN バスに接続されている外部デバイスと通信します。[Peripherals] で [CAN] を選択し、続いて [CAN0] でチャネル 0、または [CAN1]（S7G2 と S3A7 のみ）でチャネル 1 を選択します。チャネルの動作モードを有効にし、PC ボードのレイアウトと一致するように CRXn ピンと CTXn ピンを選択する必要があります。ピンコンフィギュレーターは、pin_cfg フィールドで関連するピンの CAN ピン設定を適切に構成します。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では CAN ピンの選択例を示します。

注: 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。

r_can での CAN ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
CAN	Pins	Select **Peripherals > Connectivity CAN>CAN0**

注: 選択シーケンスでは、 CAN0 がドライバーに必要なハードウェアターゲットであることを想定しています。

CAN0 のピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Enabled	CAN0 を使用するモードを有効にします
CRX	None, P202, P402 (Default: P402)	CAN0_CRX0
CTX	None, P203 P401 (Default: P401)	CAN0_CTX0

注: 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

CAN ビットレートの計算

タイムクォンタム（Tq）は、CAN 通信クロック fCANCLK の 1 ビットタイムです。これは CAN ビットタイムではなく、CAN の内部クロック周期です。周波数はボーレートプリスケーラの値と CAN ソースクロック fCAN（CANMCLK または PCLKB）によって決定されます。1 ビットタイムは複数のタイムクォンタム Tqtot に分割されます。1 タイムクォンタムは fCANCLK の周期と同じです。各ビットレートレジスタには、1 CAN ビット周期を構成する合計 Tq のうち、一定の数の Tq が付与されます。デフォルトの ISDE ビットレート設定 （S7G2 DK テンプレート）は、 fCAN が 60MHz の場合、 500Kbps です （PCLKB を使用）。

ビットレートレジスタ設定を計算するための式は次のとおりです。

fCAN = (fPCLKB または fCANMCLK)

ボーレートプリスケーラは CAN 周辺クロックをスケールダウンします。

fCANCLK = fCAN/ ボーレートプリスケーラー = 60 MHz （デフォルト） / 5 （デフォルト） = 12 MHz

1 タイムクォンタムは CAN クロックの 1 クロック周期です。

Tqtot =1/fCANCLK

Tqtot は 1 CAN ビットタイムあたりの CAN 周辺クロックサイクルの合計数で、「タイムセグメント」と「SS」（常に 1）の合計です。

Tqtot = TSEG1 + TSEG2 + SS (TSEG1 > TSEG2) = 15 + 8 + 1 = 24 （デフォルト）

この場合、ビットレートは次のようにして計算します。

ビットレート = fCANCLK/Tqtot = 12 MHz / 24 = 500 Kbps

重要な注意 :

- 実行時にメインのクロック発振器 （CANMCLK または XTAL）がまだ開始されていない場合は （たとえば、MCU クロックソースとして使用されていない場合）、ユーザーアプリケーションで CGC インタフェースを使用して開始する必要があります。
- S7G2、S3A7、S124 の各 MCU では、クロックソースがメインのクロック発振器（CANMCLK）である場合、CAN モジュールに対して次のクロック制限事項が満たされている必要があります。 fPCLKB >= fCANCLK （fCAN/ ボーレートプリスケーラー）
- S7G2 および S3A7 MCU では、クロックソースが PCLKB の場合、周辺モジュールクロックのソースは CAN HAL モジュールの PLL である必要があります。
- S3A7 MCU では、 CAN HAL モジュールを使用する場合、 PCLKA と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- S124 MCU では、CAN HAL モジュールを使用する場合、ICLK と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- SJW （同期ジャンプ幅）は多くの場合、バス管理者から供給されます。 1 <= SJW <= 4 を選択します。

異なる CAN ビットレートに対するコンフィギュレータのサンプル値

fCAN（PCLKB または CAN MCLK）	ボーレートプリスケーラ	fCANCLK = fCAN/ ボーレートプリスケーラ	タイムセグメント 1（TSEG1）	タイムセグメント 2（TSEG2）	同期ジャンプ幅（SS）	Tqtot = TSEG1 + TSEG2 +SS	ビットレート = fCANCLK/Tqtot
240	10	24	15	8	1	24	1Mbps
60	5	12	15	8	1	24	500kbps
240	48	5	16	2	2	20	250kbps
240	96	2.5	16	2	2	20	125kbps

メールボックスグループマスクの設定

4 つのメールボックスのグループごとに、合計 8 つのメールボックスグループマスクがあります。これらのマスクでは、複数の ID を受信できるようにメールボックスを設定できます。マスクがすべて 1 に設定されている場合（標準 ID では 0x7ff、拡張 ID では 0x1FFFFFFF）、そのグループ内のメールボックスは ID のビットをマスクせず、メッセージがキャプチャされる前にメールボックス ID のすべてのビットがそのメールボックス ID に一致しなければなりません。マスクのビットの中に 0 に設定されているものがある場合、これらのビットはメールボックス ID の同じビットと一致する必要はありません。たとえば、メールボックス ID 1 が 0x7ff に設定され、メールボックス 0 ～ 3 のグループマスクが 0x7ff に設定されている場合、メールボックス 1 は ID が 0x7ff であるメッセージのみをキャプチャします。メールボックス 0 ～ 3 のグループマスクが 0x7fe に設定されている場合、メールボックス 1 は ID が 0x7f のメッセージに加え、 ID が 0x7fe のメッセージもキャプチャします。

5.2.4.5 アプリケーションでの CAN HAL モジュールの使用

CAN アプリケーションで CAN 通信を行うには、少なくとも 2 つのノードが必要です。一方のノードはトランスミッタで、他のノードはレシーバです（または、両方がトランスミッタかつレシーバとして動作することもできます）。

アプリケーションで CAN モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 CAN HAL モジュールを初期化します。
2) （オプション）control API を使用して、内部ループバックまたは外部ループバックのテストモードに入ります。
3) （オプション）モジュールの状態に関する情報（ビットレートなど）を、 infoGet API を使用して取得できます。
4) メッセージを送信するには ： a.CAN フレームを作成し、正しい ID とフレームタイプに構成します。b.write API を使用して、送信モードに構成されているメールボックスに CAN フレームを書き込みます。
5) メッセージを受信するには : a.read API を使用して、フレームを受信したメールボックスから読み取ります。
6) close API を使用して、 CAN HAL モジュールを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

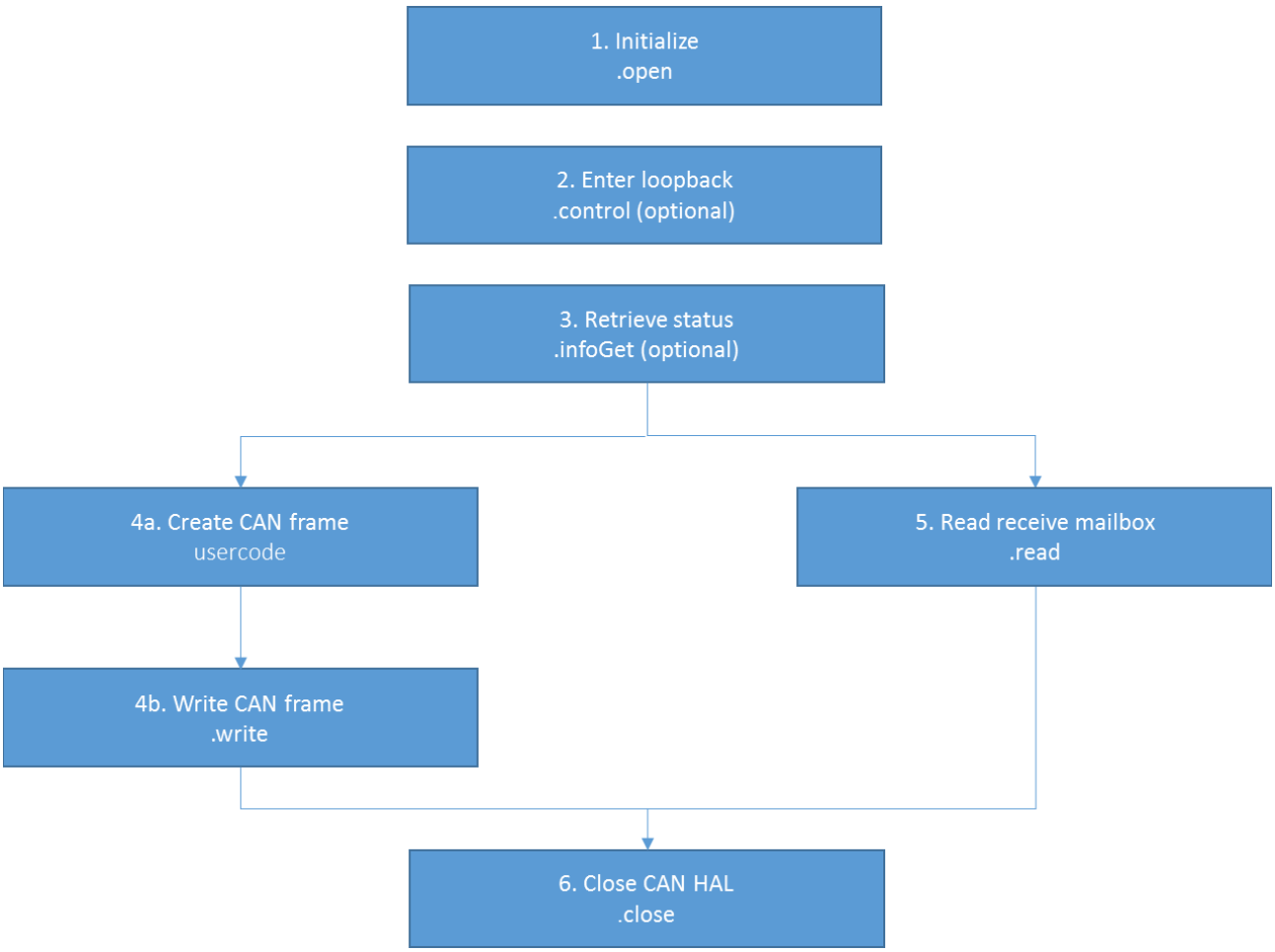


図 314: 一般的な CAN HAL モジュールアプリケーションのフロー図

5.2.5 CGC ドライバー

CGC HAL モジュールは、 Synergy MCU のさまざまなクロック機能の構成と制御をサポートします。主な特長を以下に示します。

- システムクロックソースを選択します
  - HOCO（高速オンチップ発振器）、 MOCO（中速オンチップ発振器）、 LOCO（低速オンチップ発振器）、メインクロック、 PLL、またはサブ発振器
- 内部クロックを構成し、オンまたはオフにします
- 出力クロックを構成します
- 発振停止検出機能を設定します

- 最大 6 つの各クロックドメインにクロック除数を設定します
- 一部の Synergy MCU は、制御可能な外部クロック出力もサポートしており、独立した除数を持つことができます。

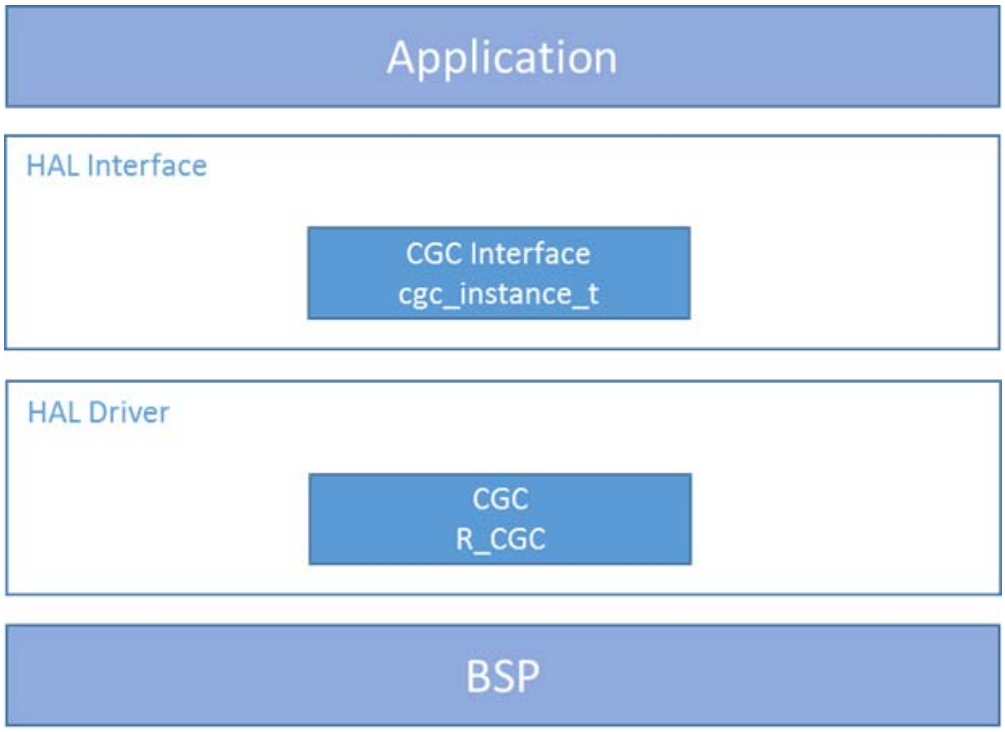


図 315: CGC HAL モジュールのブロック図

5.2.5.1 CGC HAL モジュールの API の概要

CGC HAL モジュールでは、 MCU クロックの初期化、開始、制御、停止のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の API の要約の表に含まれます。ステータス戻り値の表は API の要約の後にあります。

CGC HAL モジュールの API の要約の表

Function Name	API の呼び出し例と説明
init	g_cgc.p_api->init(); BSP によって自動的に呼び出される初期クロック構成。

Function Name	API の呼び出し例と説明
clocksCfg	g_cgc.p_api->clocksCfg(&p_clock_cfg); この関数は、開始時に BSP によって呼び出されますが、実行時にアプリケーションから呼び出してクロックを変更することもできます。
clockStart	g_cgc.p_api->clockStart(clock_source, &p_clock_cfg); クロックを開始します。
clockStop	g_cgc.p_api->clockStop(clock_source); クロックを停止します。
systemClockSet	g_cgc.p_api->systemClockSet(clock_source, &p_clock_cfg); システムクロックを設定します。
systemClockGet	g_cgc.p_api->systemClockGet(&clock_source, &clock_config); システムクロック情報を取得します。
systemClockFreqGet	g_cgc.p_api->systemClockFreqGet(&clock_source, &frequency_hz); 選択されたクロックの周波数を返します。
clockCheck	g_cgc.p_api->clockCheck(clock_source); 選択されたクロックの安定性をチェックします。
oscStopDetect	g_cgc.p_api->oscStopDetect(callback, enable); メインオシレーター停止検出を設定します。

Function Name	API の呼び出し例と説明
oscStopStatusClear	g_cgc.p_api->oscStopStatusClear(); オシレーター停止検出フラグをクリアします。
busClockOutCfg	g_cgc.p_api->busClockOutCfg (divider); バスクロック出力セカンダリ除算値。プライマリ分周器は、BSP クロック構成と systemClockSet 関数を使用して設定されます（S7G2 および S3A7 MCU のみ）。
busClockOutEnable	g_cgc.p_api->busClockOutEnable (); バスクロック出力を有効にします（S7G2 および S3A7 MCU のみ）。
busClockOutDisable	g_cgc.p_api->busClockOutDisable (); バスクロック出力を無効にします（S7G2 および S3A7 MCU のみ）。
clockOutCfg	g_cgc.p_api->clockOutCfg(clock_source, clock_dividers); clockOut を設定します。
clockOutEnable	g_cgc.p_api->clockOutEnable(); CLKOUT ピンでのクロック出力を有効にします。クロックのソースは clockOutCfg によって制御されます。
clockOutDisable	g_cgc.p_api->clockOutDisable(); CLKOUT ピンでのクロック出力を無効にします。クロックのソースは clockOutCfg によって制御されます。
lcdClockCfg	g_cgc.p_api->lcdClockCfg(clock); セグメント LCD クロックを構成します（S3A7 および S124 MCU のみ）。

Function Name	API の呼び出し例と説明
lcdClockEnable	g_cgc.p_api->lcdClockEnable(); LCD クロックを有効にします（S3A7 および S124 MCU のみ）。
lcdClockDisable	g_cgc.p_api->lcdClockDisable(); LCD クロックを無効にします（S3A7 および S124 MCU のみ）。
sdramClockOutEnable	g_cgc.p_api->sdramClockOutEnable(); SDRAM クロック出力を有効にします（S7G2 MCU のみ）。
sdramClockOutDisable	g_cgc.p_api->sdramClockOutDisable(); SDRAM クロックを無効にします（S7G2 のみ）。
usbClockCfg	g_cgc.p_api->usbClockCfg(divider); USB クロックを設定します（S7G2 のみ）。
systickUpdate	g_cgc.p_api->ssystickUpdate(period_count, units); Systick タイマを更新します。
versionGet	g_cgc.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値の表

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ABORTED	Systick タイマの更新に失敗しました。
SSP_ERR_HARDWARE_TIMEOUT	ハードウェアがタイムアウトしました。
SSP_ERR_STABILIZED	クロックが安定しました。
SSP_ERR_CLOCK_INACTIVE	クロックがオンになっていません。
SSP_ERR_MAIN_OCO_INACTIVE	メイン OCO がオフ / 不安定です。
SSP_ERR_CLOCK_ACTIVE	クロックはアクティブです。
SSP_ERR_NOT_STABILIZED	クロックソースは安定していません。
SSP_ERR_CLKOUT_EXCEEDED	クロック出力が超過しました。
SSP_ERR_NULL_PTR	ポインタが NULL です。
SSP_ERR_OSC_DET_ENABLED	発振停止検出機能が有効になっています。
SSP_ERR_OSC_STOP_DETECTED	オシレーション停止検出ステータスフラグが設定されます。この状態では、オシレーション停止検出関数を無効にすることはできません。
SSP_ERR_OSC_STOP_CLOCK_ACTIVE	メイン Osc または PLL がシステムクロックとして設定されている場合、オシレーション検出ステータスフラグはクリアできません。システムクロックを変更してから、このビットのクリアを試みてください。
SSP_ERR_INVALID_ARGUMENT	無効な引数。
SSP_ERR_INVALID_MODE	制限された動作電力制御モードでクロックを開始することを試みます。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.5.2 CGC HAL モジュールの動作の概要

CGC HAL モジュールインタフェースでは、CGC HAL モジュールのすべての機能を構成し、使用することができます。たとえば、複数のクロックソースをシステムクロックソースとして選択できます。また、システムクロックを分周することで、システムやペリフェラルのニーズに適した多様な周波数を提供できます。

クロックの安定性を確認し、クロックが不要なときは停止して電力を節減することができます。 API には、実行時にシステムおよびシステム周辺クロックの周波数を返す関数があります。メイン発振器の停止を検出する機能があり、それにはユーザー定義のコールバック関数を呼び出すオプションも用意されています。

クロック機能を構成および制御するための CGC HAL モジュール：

- 使用可能な任意のクロック（HOCO、MOCO、LOCO、メインクロック、PLL、サブ発振器）をシステムクロックソースとして設定します
- 内部クロックを構成します （ICLK、 PCLK など）。
- クロックのオンとオフを切り替えます
- 出力クロックを構成します
- 発振停止検出機能を設定します

クロック生成回路の特長は、次の発振器とクロック発振器です。

- メイン発振器入力 （最大 24 MHz）
- 32.768kHz サブクロック発振器
- 64MHz で動作する HOCO （デバイスのバージョンに依存）
- 8MHz で動作する MOCO
- 32.768kHz で動作する LOCO
- 24MHz ～ 240MHz で動作する PLL 回路出力 （デバイスに依存）

Synergy マイクロコントローラには、6 つの内部クロックドメインがあります。各ドメインには独立した除数がありますが、システムクロック制御レジスタで選択したクロック入力に依存します。以下の除数があります。

- ICLK – コアクロック。 CPU、 DMAC、 ROM、 RAM 用 （最大 32/48/240MHz）
- PCLKA – 周辺クロック。 EtherC、EDMAC、USB2.0 HS、QSPI、および SCIF を含むモジュール用（最大 32/48/120MHz）
- PCLKB – 周辺クロック。IIC、CAN、DAC12、RTC、USBFS、I/O ポート、WDT、IWDT などのモジュール用 （最大 32/60MHz）
- PCLKC – 周辺クロック。 ADC12 変換クロック用 （最大 64/60MHz）
- PCLKD – 周辺クロック。 GPT カウントクロック用 （最大 64/120MHz）
- FCLK – フラッシュメモリ用クロックソース （最大 32/60MHz）

また、一部の Synergy マイクロコントローラでは、制御可能な外部クロック出力がサポートされており、そのいくつかには独立した除数があります。以下の除数があります。

- CLKOUT – CLOCKOUT/BUZZER クロック （最大 24 MHz）（独立したクロックセレクターと除数）
- BCLK – 外部バスコントローラへの外部バスクロック （最大 16/120MHz）
- SDCLK – SDRAM クロック （最大 120 MHz）
- UCLK – USB クロック （最大 120 MHz）（Synergy バージョン 2 用の独立した除数 / セレクター）
- LCD_CLK – LCD クロック （独立したクロックセレクタ。ただし除数なし）

実行時のシステムクロック周辺クロック除数の変更

CGC HAL モジュールには、実行時に clocksCfg API 関数を使用して、システムクロックおよびクロックツリーの設定を変更するオプションもあります。 clocksCfg API 関数で使用するための構成構造体を作成するには、 [New Stack] > [System] > [CGC Configuration Instance] の順に選択します。

clocksCfg 関数により、システムクロック、周辺クロック分周器、PLL 乗算値および分周器、システムクロックの状態（停止 / 開始）（HOCO、主発振器、サブクロック発振器など）を変更できます。次の図のオプションは、システムクロックを HOCO から MOCO に変更し、周辺クロック分周器を更新している例です。各クロックに対するオプションは、開始、停止、なし （変更なしという意味）です。すべての MCU で、すべてのクロックが利用可能とは限りません。すべての MCU で、すべての周辺クロックが利用可能とは限りません。

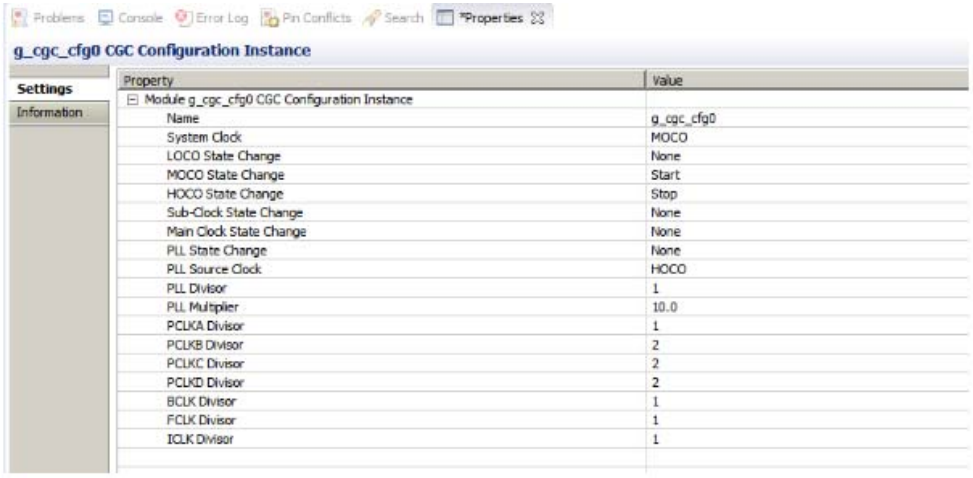


図 316: CGC の構成プロパティ

上の例の関数呼び出しは次のようになります。

```
g_cgc.p_api > clocksCfg(&g_cgc_cfg0);
```

オプション設定メモリ

すべての Synergy マイクロコントローラにはオプション設定メモリが含まれます。このメモリを使用して、リセット後のペリフェラルの動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。次の表では、 OFS レジスタで構成できる CGC HOCO パラメータの一覧を示します。

OFS レジスタで設定できるものの表

Control	説明
HOCO oscillation enable	有効にすると、リセット後に HOCO を自動的に開始します
HOCO Frequency	S7 および S5 シリーズ
	16MHz

Control	説明
	18MHz
	20MHz
	S3 および S1 シリーズ
	24 MHz
	32 MHz
	48 MHz
	64 MHz

OFS レジスタの値は、プロパティダイアログで設定できます。プロパティダイアログは、Synergy 構成エディタで [BSP] タブを選択して使用できます。

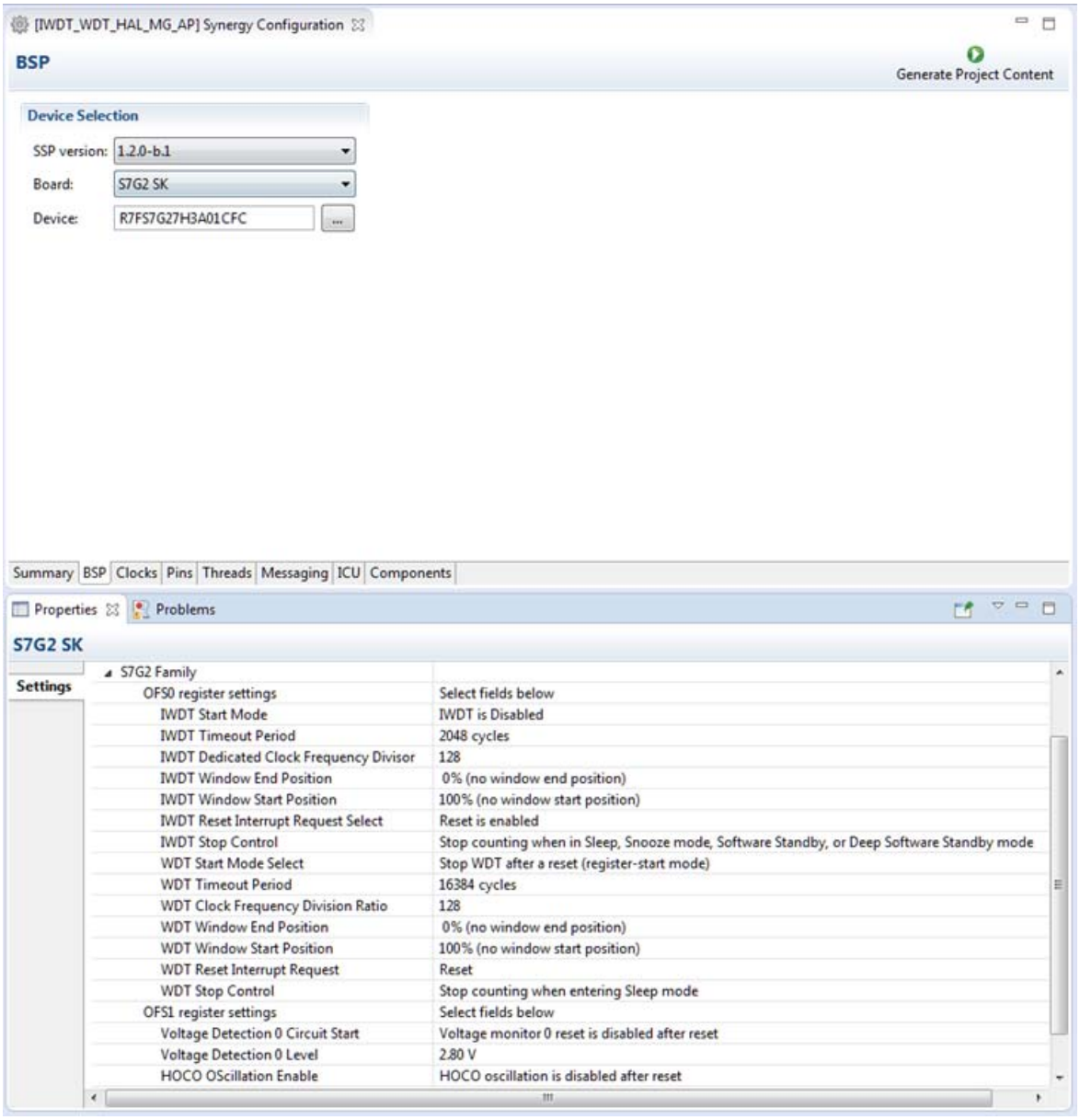


図 317: OFS のレジスタ設定

ローパワーモードバージョン 2 の HAL モジュールは MCU の動作電力制御モードを処理しなくなるので、CGC HAL モジュールが MCU の動作電力制御モードも処理します。

CGC HAL モジュールの動作に関する重要な注意事項と制限事項

- CGC HAL モジュールは、 ThreadX RTOS に依存しません。
- CGC HAL モジュールは MCU のコア機能であり、BSP 初期化プロセスの後で設定されます。多くの場合、CGC を変更する必要はありません。ただし、CGC HAL モジュールにはクロック構成変更関数が用

意されており、アプリケーションの要件に応じて動作速度の要件と電力消費のバランスを取ることができます。

- Synergy マイクロコントローラの CGC は、発振器の停止検出をサポートします。アプリケーションで有効にすると、発振器停止検出機能は、主 /PLL クロックが停止したかどうかを自動的に検出して、MOCO に動作を切り替えます。 SSP でこの機能を有効にするときは、ユーザーが手動でコールバック関数を作成する必要があります。この手順について以下で詳しく説明します。

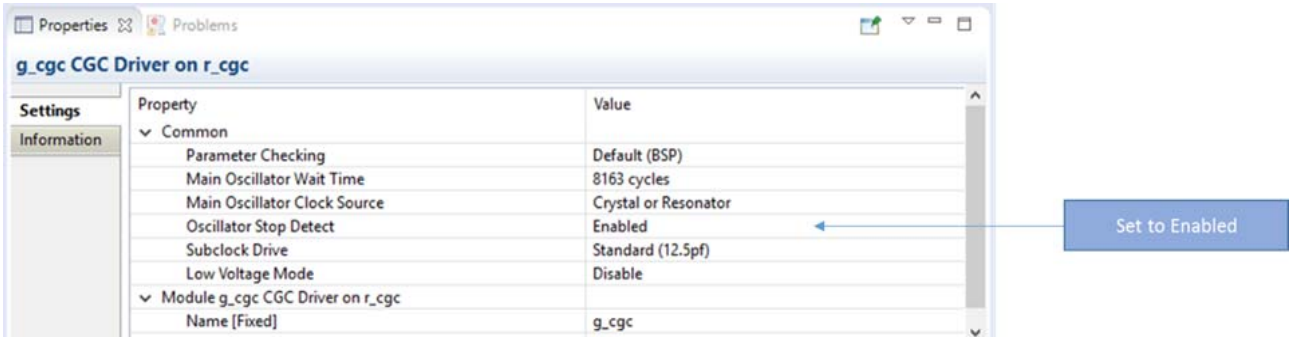


図 318: 発振器停止検出の有効化 / 無効化

アプリケーションのコードで、コールバック関数を作成します。この例では、 osc_stop_callback. という名前にします。

```
__void osc_stop_callback__(cgc_callback_args_t * p_args)

{

/* perform Oscillator Stop Detection processing */

}
```

前に宣言したコールバックで API を呼び出すことにより、発振器停止検出を有効にします。

```
/* Enable the Osc Stop Detect functionality */

g_cgc.p_api->oscStopDetect( osc_stop_callback, true );
```

ICU 内で割り込みを有効にします。

```
/* *Osc* Stop Detect is an NMI interrupt. Enable the NMI in ICU */

R_ICU->NMIER_b.OSTEN = 1;
```

このモジュールの使用に関する制限事項については、 SSP の最新のリリースノートを参照してください。

5.2.5.3 アプリケーションへの CGC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CGC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

CGC ドライバーは HAL/ 共通スレッドに自動的に追加されるので、削除された場合に新しいスレッドに追加することだけが必要です。（CGC のデフォルト名は g_cgc. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

CGC HAL モジュールの選択シーケンスの表

Resource	ISDE Tab	Stacks Selection Sequence
g_cgc CGC HAL on r_cgc	Threads	New Stack> Driver> System> CGC Driver on r_cgc

次の図に示すように、g_cgc の CGC ドライバーは HAL/ 共通スタックに自動的に追加されます。

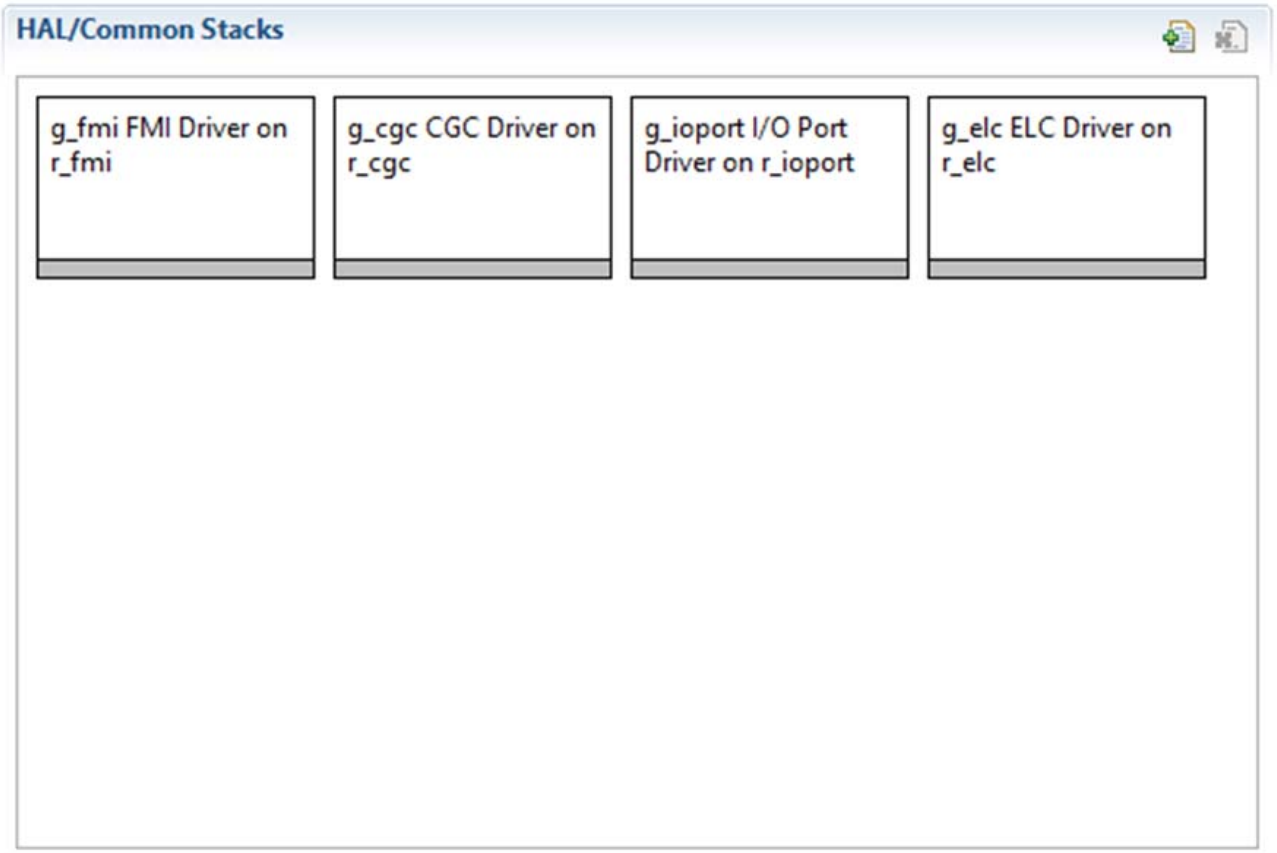


図 319: CGC HAL モジュールのスタック

5.2.5.4 CGC HAL モジュールの構成

ユーザーは必要な動作に合わせて CGC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_cgc での CGC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。

ISDE Property	Value	説明
Main Oscillator Wait Time	3,35,67,131,259,547,1059,2147,4291, 8163 cycles (Default: 8163 cycles)	これらいずれかの値を設定します。この遅延は CGC_CFG_MAIN_OSC_CLOCK_SOURCE が 0 に設定されてレゾネータ / クリスタルが使用されていることを示した場合にのみ構成されます。この遅延は、 #define CGC_CFG_MAIN_OSC_CLOCK_SOURCE が 0 に設定されて発振子 / 水晶が使用されていることを示す場合にのみ構成されます。メインクロック発振安定化時間は、オシレーターの製造元が推奨する安定化時間よりも長いかそれと等しい長さに設定します。
Main Oscillator Clock Source	External Oscillator, Crystal or Resonator (Default: Crystal or Resonator)	共振器または水晶を使用する場合は 0 を設定します。外部オシレーター入力を使用する場合は 1 を設定します。
Oscillator Stop Detect	Enabled, Disabled (Default: Enabled)	有効にすることで、 R_CGC_OscStopDetect 関数のコードが生成されます。この機能を使用するには、コールバックポインタとともにこの関数を呼び出す必要があります。
Subclock Drive	Standard (12.5pf), Middle (4.4pf) (Default: Standard)	この設定は、サブクロック発振器ドライブキャパシタンスを水晶パラメータ #define CGC_CFG_SUBCLOCK_DRIVE に基づいて一致させるためのものです。

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

CGC HAL モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、クロックを選択的にオンまたはオフにしたり、電力と性能の特性を最適化するために周波数を変更したりすることが役に立つ場合があります。

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

CGC HAL モジュールのクロック構成

BSP 初期化プロセスによって設定されるデフォルトの CGC HAL モジュールクロック周波数は、ISDE でコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。

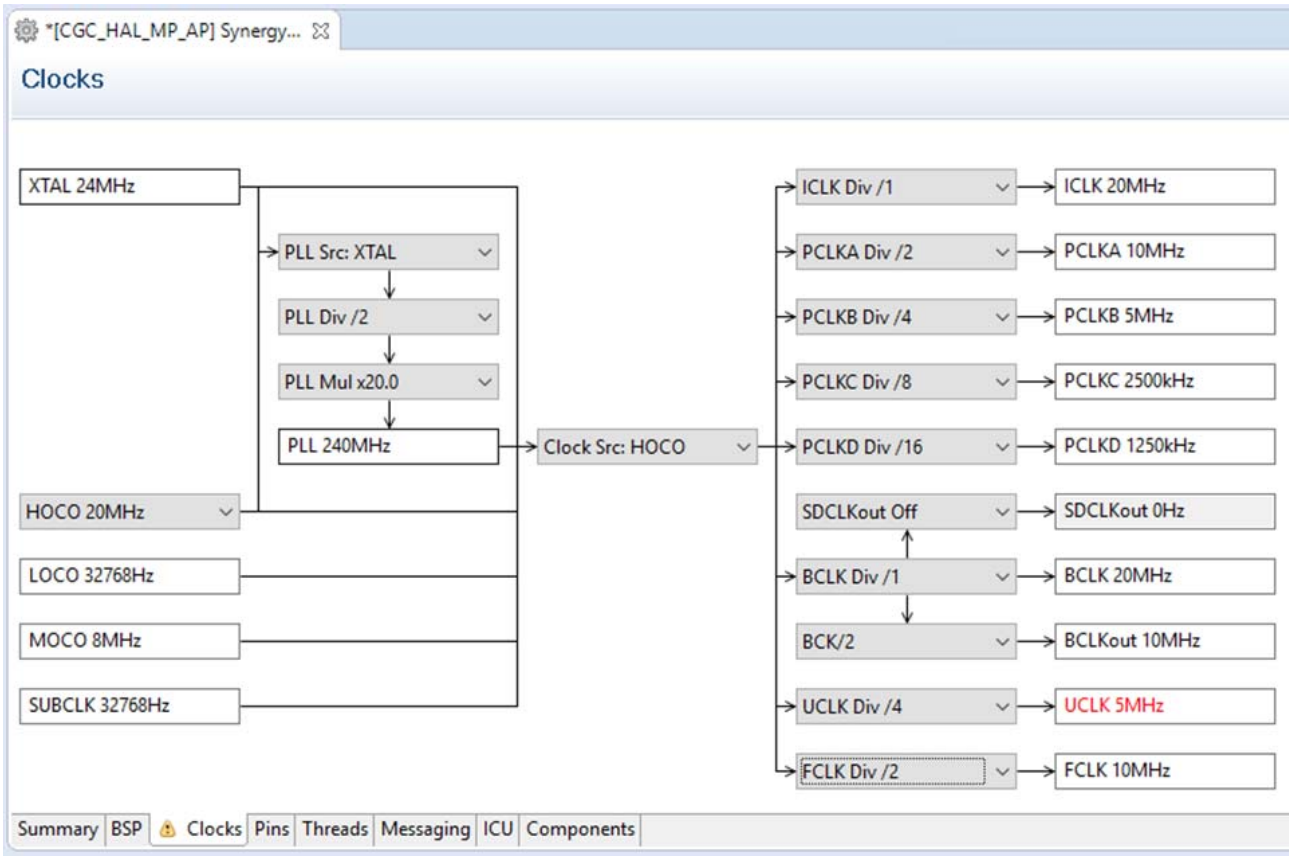


図 320: [Clocks] タブでのデフォルトのクロック設定

この例では、クロックソースは HOCO であり、周辺クロックに対してさまざまなクロック分周器が選択されています。有効な USB クロック（UCLK）を実現できない場合は、赤で強調表示されます。これはあくまでもアドバイスであり、このようなクロック周波数が要求されても、プロジェクトはビルドされます。

CGC HAL モジュールのピン構成

CGC 周辺モジュールは、BCLK および SDCLK 信号の出力を制御します。この機能を有効または無効にするには、[Clocks] タブを使用します。必要に応じて、[Pins] タブで BCLK_SDCLK I/O ピンを選択して構成する必要があります。

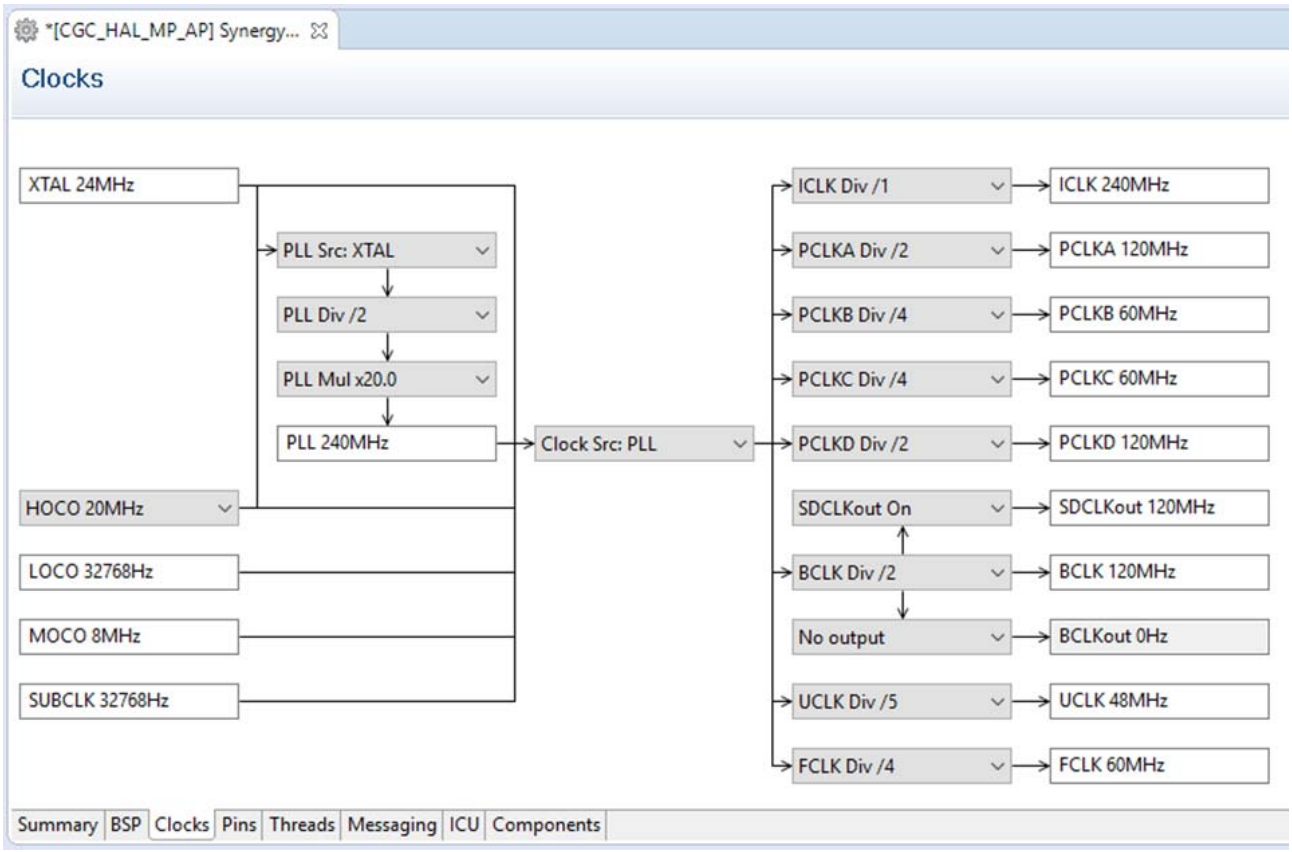


図 321: [Clock] タブでの SDCLK および BCLK の有効化 / 無効化

この例では、SDRAM クロックが有効にされ、BUS クロックが無効にされています。

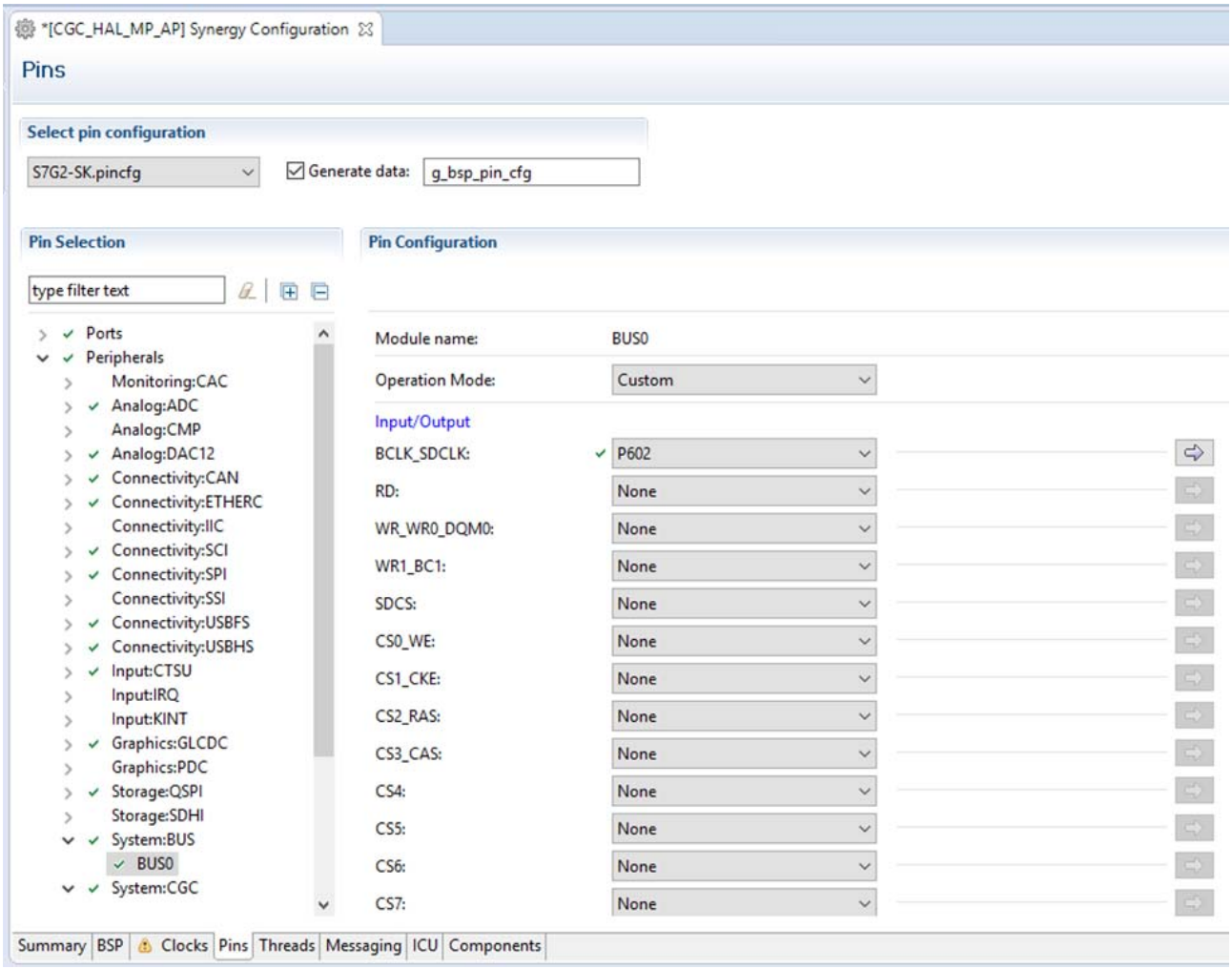


図 322: [Pins] タブでの SDCLK および BCLK の有効化 / 無効化

この例では、SDRAM クロックと BUS クロックが P602 で有効にされています。

CGC に関連付けられている他のピン設定により、外部発振器のピンを有効または無効にしたり、システムクロックの出力ピンを設定したりできます。

Synergy デバイスはオンチップ発振器を使用して動作することができ、その場合は、メインクロックの外部発振器のピン XTAL および EXTAL に対する要件はありません。アプリケーションでは、これらを入力ピンとして使用できます。サブクロック外部発振器のピン XCIN および XCOUT の機能は固定です。

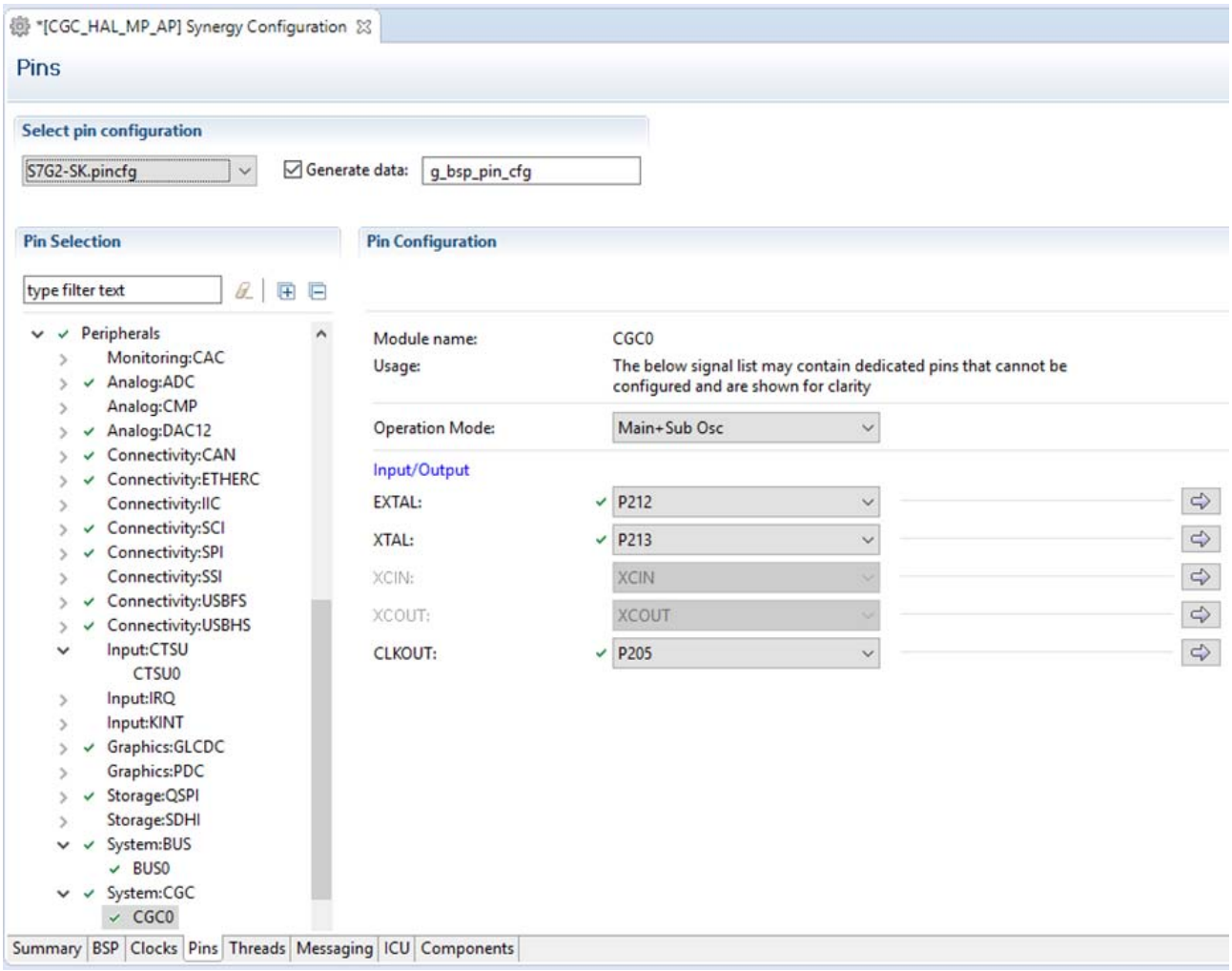


図 323: [Pins] タブでの EXTAL、 XTAL、 CLKOUT の有効化 / 無効化

この例では、外部主発振器がピン P212 と P213 を介して使用され、CLKOUT が P205 で有効にされています。

注: *設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと Synergy MCU では、使用可能なピン構成設定が異なる場合があります。*

5.2.5.5 アプリケーションでの CGC モジュールの使用

アプリケーションで CGC HAL モジュールを使用する一般的な手順は次のとおりです。

1) CGC はリセット後に自動的に設定されます。
2) 必要に応じて、 clocksCfg API を使用してクロックを構成します。
3) clockStart API を使用してクロックを開始します。

4) clockStop API を使用してクロックを停止します。

5) 必要に応じて、他の CGC 関数を呼び出します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

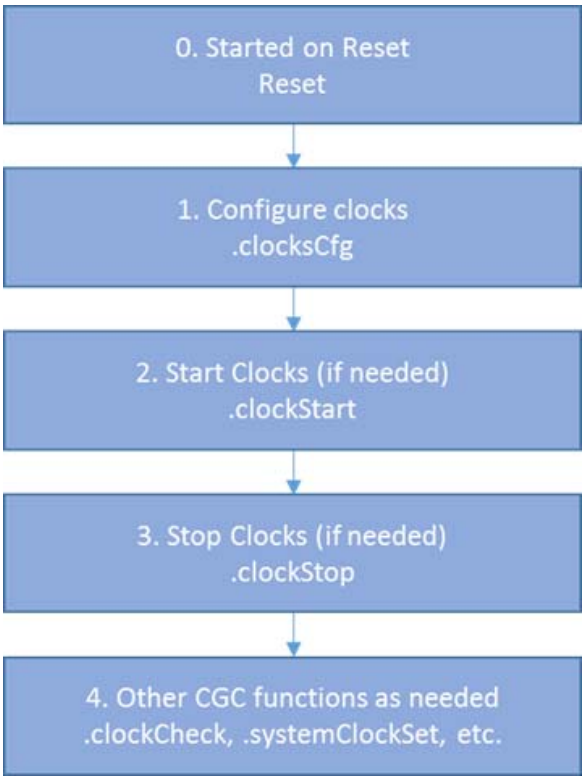


図 324: 一般的な CGC HAL モジュールアプリケーションのフロー図

5.2.6 CRC ドライバー

CRC HAL モジュールでは、**オープン**、**クローズ**、**有効化**、**計算**のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

CRC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_crc.p_api->open(g_crc.p_ctrl, g_crc.p_cfg); CRC ドライバーモジュールを開きます。

Function Name	API の呼び出し例と説明
close	g_crc.p_api->close(g_crc.p_ctrl); CRC モジュールドライバーを閉じます。
crcResultGet	g_crc.p_api->crcResultGet(g_crc.p_ctrl, &result); 現在の計算値を返します。
snoopEnable	g_crc.p_api->snoopEnable(g_crc.p_ctrl, seed); スヌーピングを有効にします。
snoopDisable	g_crc.p_api->snoopDisable(g_crc.p_ctrl); スヌーピングを無効にします。
snoopCfg	g_crc.p_api->snoopCfg(g_crc.p_ctrl, g_crc.p_cfg); スヌープチャネルと方向を設定します。
calculate	g_crc.p_api->calculate(g_crc.p_ctrl, &input_buffer, num_bytes, crc_seed, &crc_result); データブロックに対して CRC 計算を実行します。
versionGet	g_crc.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	構成が正常に行われました。

Name	説明
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_INVALID_ARGUMENT	無効な引数のエラーです。
SSP_ERR_NOT_OPEN	ドライバーが開いていません。
SSP_ERR_IN_USE	ドライバーが既に開いている場合。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.6.1 CRC HAL モジュールの動作の概要

メモリ内のデータブロックの CRC 値の計算に CRC HAL モジュールを使用するときは、 calculate API を使用できます。この API は、入力バッファポインタ、長さ、 CRC シード値を入力として受け取り、計算された CRC 値を出力します。

シリアル通信インタフェース （SCI）チャネルで送受信されるデータストリームの CRC を計算するために CRC HAL モジュールを使用するときは （スヌープモード）、最初に、 snoopCfg API を呼び出してから snoopEnable API を呼び出して、モジュールをスヌープモードで構成する必要があります。要求した数のデータが SCI チャネルで送受信された後、 crcResultGet API を使用して、モジュールから計算された CRC 値を取得できます。

CRC HAL モジュールの動作に関する重要な注意事項と制限事項

- CRC ブロックは割り込みを使用しません。
- CRC モジュールにはクロック構成はありません。
- CRC モジュールにはコールバックはありません。
- メモリ内のデータブロックの CRC 値の計算に 32 ビットの CRC 多項式を使用すると、データブロックはリトルエンディアンのバイト順序を使用して解釈されます。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

5.2.6.2 アプリケーションへの CRC HAL モジュールの組み込み

このセクションでは、*SSP コンフィギュレータ*を使用してアプリケーションに CRC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

CRC HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（CRC ドライバーのデフォルト名は g_crc0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

CRC ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_crc0 CRC Driver on r_crc	**Threads**	**New Stack> Driver> Monitoring> CRC Driver on r_crc**

次の図に示すように、r_crc の CRC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。

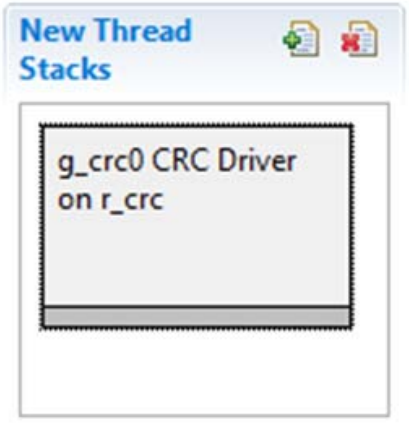


図 325: CRC HAL モジュールのスタック

5.2.6.3 CRC HAL モジュールの構成

ユーザーは必要な動作に合わせて CRC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の **[Properties]** ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることに注意してくだ

さい。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_crc での CRC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g-crc0	モジュール名。
CRC Polynomial	CRC-8, CRC-16, CRC-CCITT, CRC-32, CRC-32C Default: CRC-32C	計算に使用する多項式を 指定します。
Bit Order	LSB, MSB Default: MSB	計算のビット順序を指定します。

***注:** 設定例とデフォルトは、S7G2 Synergy MSU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、デフォルトとは異なるクロック分周器を選択すると役に立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

CRC HAL モジュールのクロック構成

CRC HAL モジュールは周辺クロック A（PCLKA）からクロックを供給されます。

CRC HAL モジュールでは、 API を使用して動作周波数を設定することはできません。

CRC HAL モジュールのピン構成

CRC HAL モジュールには構成可能なピンはありません。

5.2.6.4 アプリケーションでの CRC HAL モジュールの使用

メモリ内のデータブロックの CRC を計算するために CRC HAL モジュールを使用する場合の一般的な手順は次のとおりです。

1) open API を使用して、 CRC HAL モジュールを初期化します
2) calculate API を使用して、 CRC HAL モジュールを計算します
3) close API を使用して、 CRC HAL モジュールを閉じます

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

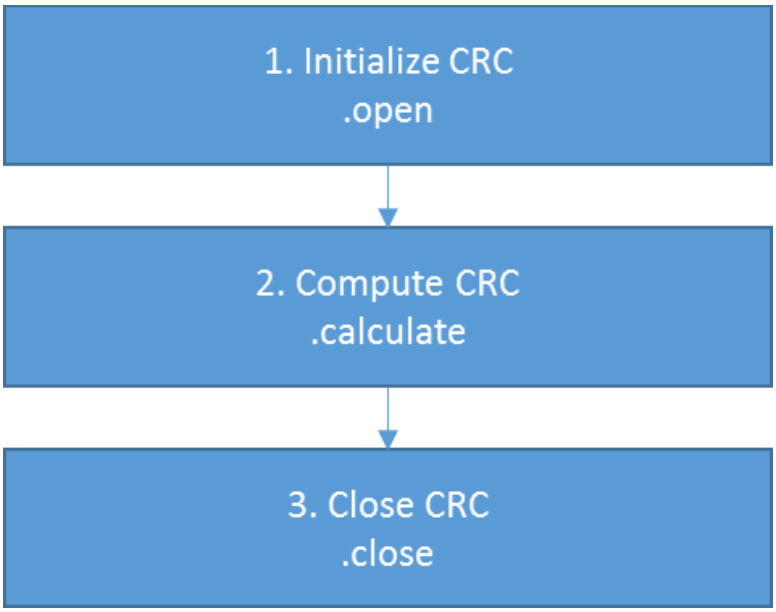


図 326: 一般的な CRC HAL モジュールアプリケーションのフロー図

スヌープモードで CRC を計算するために CRC HAL モジュールを使用する場合の一般的な手順は次のとおりです。

1) open API を使用して、 CRC HAL モジュールを初期化します。
2) snoopCfg API を使用して、SCI チャネル（およびその方向）をスヌープするように CRC モジュールを構成します。
3) snoopEnable API を使用して、 SCI チャネルのスヌーピングを有効にします。
4) SCI チャネルで必要なバイト数が送信または受信された後、crcResultGet API を使用して計算された CRC を取得します。
5) snoopDisable API を使用して、スヌーピング動作を無効にします。
6) close API を使用して、 CRC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

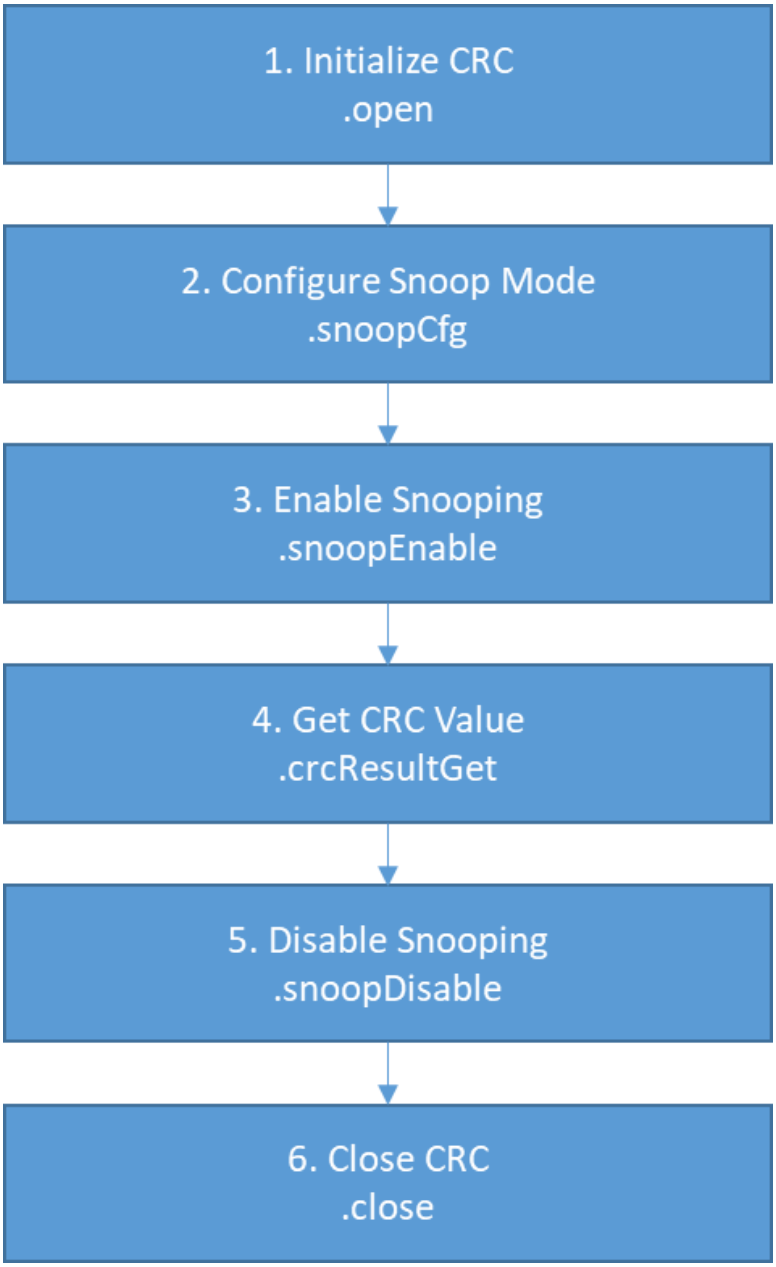


図 327: 通常の CRC HAL モジュールのスヌープモードアプリケーションのフロー図

5.2.6.5 CRC HAL モジュールのアプリケーションプロジェクト

このモジュールガイドに関連付けられているアプリケーションプロジェクトを見ると、設計の詳細な手順がわかります。プロジェクトについては、このドキュメントの最後にある「参考文献」セクションのリンクを参照してください。ISDE でアプリケーションプロジェクトをインポートして開き、CRC HAL モジュールの

構成設定を見ることができます。また、完全な設計での CRC HAL モジュール API を示すために使用されているコード (crc_hal.c) を読むこともできます。

アプリケーションプロジェクトでは、CRC HAL モジュール API の一般的な使用方法が示されています。アプリケーションプロジェクトのメインスレッドエントリは、 CRC HAL モジュールを初期化し、周期的に CRC を計算します。最初に、生成されたバイトが設定されているデータバッファに対して CRC が計算された後、データがいわゆる受信バッファにコピーされて（アプリケーションがデータの送信をシミュレートします）、 CRC が再び計算されます。両方の CRC が比較され、その結果を基にして、**緑**（CRC が等しい）または赤（CRC が異なる）の LED が点灯されます。データ送信のシミュレーションではエラーが発生する可能性があり、発生すると赤い LED が点灯します。次の表では、アプリケーションプロジェクトで使用されている関連するソフトウェアとハードウェアのターゲットバージョンを示します。

アプリケーションプロジェクトで使用されているソフトウェアリソースとハードウェアリソース

Resource	Revision	説明
e2 Studio	5.3.1	統合ソリューション開発環境（ISDE）
SSP	1.2.0	Synergy Software Platform
IAR EW for Synergy	7.71.1	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.2.1.016	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	スターターキット

アプリケーションプロジェクトの簡単なフロー図を次に示します。

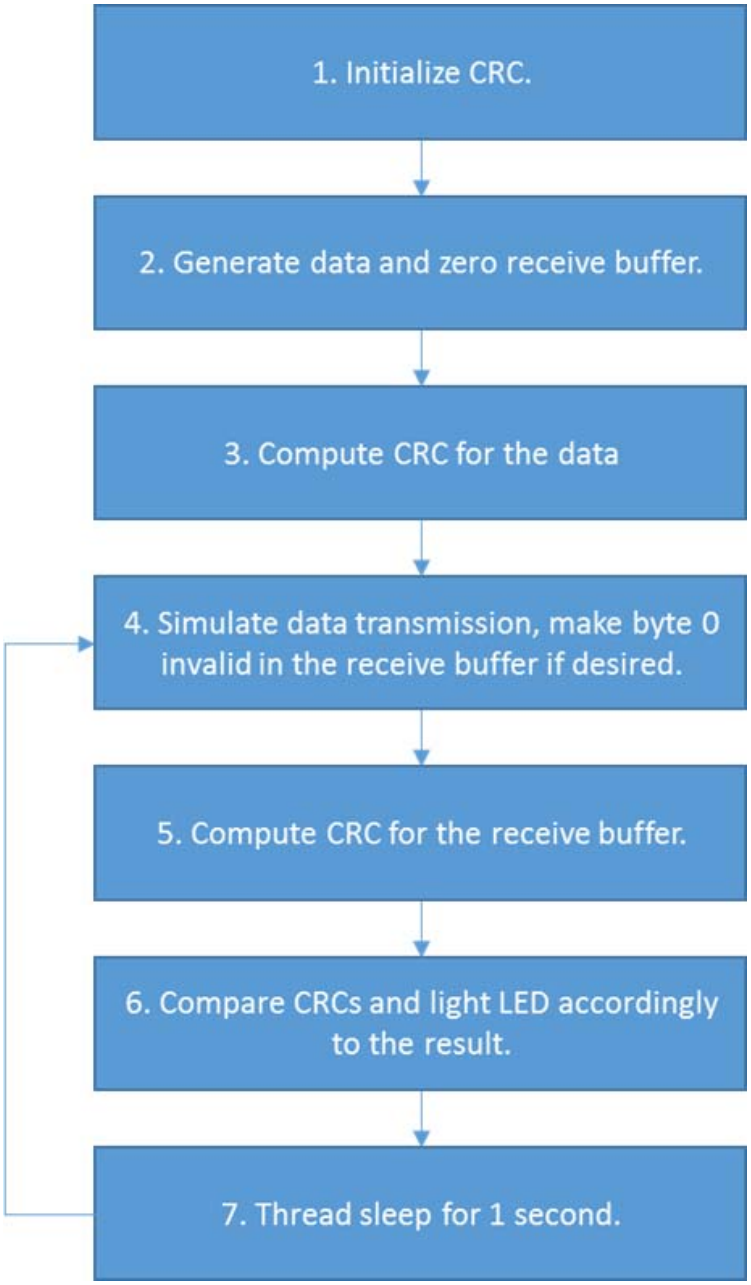


図 328: CRC アプリケーションプロジェクトのフロー図

完全なアプリケーションプロジェクトについては、このドキュメントの最後にある「参考文献」セクションのリンクを参照してください。プロジェクトを ISDE にインポートすると、crc_hal.c ファイルがプロジェクトに配置されます。このファイルを ISDE で開き、以下の説明を読みながら見ることで、API の重要な使用方法がわかります。

アプリケーションプロジェクトのメインスレッドエントリでは、データ配列のすぐ後にある open API を使用して CRC HAL モジュールが初期化され、受信バッファが設定されます。データ配列には 0 から始まって増加する値がバイトとして格納され、受信バッファはゼロで初期化されます。データの準備ができると、CRCHAL モジュールが初期化され、データの CRC が計算されます。その後、アプリケーションフローはループに入り、データ送信をシミュレートして、受信バッファを設定します。それが済むと、バッファの CRC を計算し、CRC の比較結果に従って緑（CRC が等しい場合）または赤（CRC が異なる場合）の LED を点灯します。最初、アプリケーションはデータ送信エラーを生成しないように構成されます。これは、makeError 変数を true に設定することで変更できます。このように設定すると、データ配列から受信バッファにバイトをコピーした後で、バッファのインデックス 0 のバイトが変更されます。その結果、異なる CRC になり、赤い LED が点灯されます。

CRC アプリケーションプロジェクトは、CRC HAL モジュールのデフォルトのプロパティ値を使用します。ユーザーがさらに実行する必要があることはありません。

5.2.7 CTSU ドライバー

CTSU ドライバーは、CTSU 周辺デバイスを初期化して任意の構成済み（かつ有効化済み）チャネルのキャパシタンスの変化を検出し、必要なフィルタリングを実行し、ボタンのホイールやスライダーなど上位のウィジェットレイヤーで使用できるさまざまなデータを生成するために使用されます。こうしたレイヤーで要求されるさまざまな種類のデータをサポートするため、この実装では上位レベルのレイヤーで多様な種類の処理済みデータを必要性に基づいて読み取れるよう、Read() 関数が提供されています。またドライバーには、各スキャンが完了したときおよび新しい処理データが使用可能になったときに呼び出されるコールバックも用意されています。このコールバックを使用して、上位レイヤーでデータを読み取ることができます。

CTSU ドライバーを利用すると、[Mutual Capacitance] や [Self Capacitance] を含む、サポートされているすべての動作モードで CTSU チャネルを設定できます。ドライバーは設定済みチャネルのスキャン、DTC を使用したデータの移動、フィルタリングの実行、ドリフト補償、自動調整などを実行し、各繰り返し処理が完了するたびにコールバック関数によってユーザーに通知を行います。またこのドライバーにより、ユーザーは独自のフィルタリングや自動調整アルゴリズムを設定し、それをプロセスに統合することも可能になります。ドライバーでは一度に 1 つの構成しかサポートされませんが、アプリケーションの必要に応じて複数のチャネル設定でドライバーを再開することができます。

5.2.7.1 e² studio による CTSU ドライバーを使用するアプリケーションの作成

ドライバーは、e² studio の Synergy Software Package に組み込まれています（e² studio ISDE ユーザーガイドを参照）。

e² studio でプロジェクトの作成と設定を行い、ドライバーを追加します。

1) プロジェクトを作成します（Synergy プロジェクトの作成を参照）。
2) プロジェクトを設定します（プロジェクトの設定を参照）。
3) ドライバーを追加します（スレッドとドライバーの追加を参照）。

CTSU ドライバーを使用するアプリケーションでは、以下のリソースが必要です。

Resource	ISDE Tab	Selection
CTSU Driver	Threads	**Driver > Input > CTSU Driver on r_ctsu**
Interrupts	Threads	**CTSU WRITE, CTSU READ, CTSU END**
CTSU Clock	Clocks	CTSU クロックの設定 を参照
CTSU Pins	Pins	CTSU ピンの設定 を参照

CTSU クロックの設定

CTSU クロック速度を、静電容量タッチ調整ツール **Workbench 6** で選択したクロック速度と同じ値に設定します。

CTSU ピンの設定

ピンをタッチセンサーピン TSxx として設定し、 TSCAP 機能ピンを有効にします。

CTSU 割り込みの設定

[ICU] タブで 3 つの CTSU 割り込みをすべて同じプライオリティに設定して、これらの割り込みを有効にします。どのような値を設定しても構いません。

CTSU パラメータの設定

e^2 studio ISDE を使用して、 CTSU ドライバーパラメータを設定します。

RTOS を使用しないアプリケーションの場合 : HAL ドライバーの追加と設定

ThreadX アプリケーションの場合 : ドライバーのスレッドへの追加とドライバーの設定。

Workbench によって生成された **CTSU 構成**を含む**ファイルまたはフォルダー**が ${PROJECT_ROOT}\src フォルダーに存在し、そのファイル / フォルダーがビルドに含まれていることを確認します。

SSP を使用して CTSU パラメータを設定するには、以下の手順を実行します。

1) r_ctsu_api.h で、構造体型 ctsu_cfg_t の詳細を参照します。
2) ctsu_cfg_t 型のローカル変数を作成し、以下のように構成します。

CTSU の共通 / ビルド時構成パラメータ

ISDE Property	Configuration	Setting	説明
Parameter Checking	-	Enabled (Default), Disabled	パラメータ エラー チェックを有効または無効にします。
Offset Adjustment	-	Enabled (Default), Disabled	オフセット調整を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。(レートは、*センサー値調整のランタイム レートにより決定されます*)
Drift Compensation	-	Enabled (Default), Disabled	ドリフト補正を有効または無効にするのに使用します。ボードの調整を行う場合以外は、有効にしておいてください。ドリフト補正により、ボードやサーフェスの老化あるいは温度や環境の変化に応じて、タッチの存在の有無を決定するベースライン値が経時的に変化するよう設定できます。ドリフト補正レートは、*定常状態のドリフト補正レート*および*スタートアップ ドリフト補正レート*、*チャネル解放補正レート*により決定されます。
Drift Compensation Methods	-	Alternate 1	ドリフト補正のアルゴリズムを提供します。現在サポートされているのは Alternate 1 のみです。その他のオプションは、今後のリリースでサポートされる可能性があります。ドリフト補正方法 Alternate 1 を使うと、定常状態、スタートアップ、チャネル解放の各イベントに対し、異なるドリフト補正レートを用いることができます。

ISDE Property	Configuration	Setting	説明
Dynamic Memory Usage	-	Enabled, Disabled (Default)	チャネルが動的に有効あるいは無効であるアプリケーションのメモリの保存に使用されます。動的メモリの使用は現在サポートされていません。
Perform auto-tune and drift compensation only when all channels are untouched	-	Enabled (Default), Disabled	すべてのチャネルがタッチされていない場合にのみ、自動調整とドリフト補正を実行します。
DTC Usage	-	Enabled (Default), Disabled	CTSU を最小化する CPU オーバーヘッドからデータを出し入れするために DTC を使用することを可能にします。このオプションは現在無効にすることができません。
Maximum Active Channels	-	-	アプリケーションによって使用されるチャネルの最大数を指定します。Self Capacitance Mode では、使用されるチャネル数を指します。Mutual Capacitance Mode では、Rx チャネルと Tx チャネルの乗算の結果です。たとえば、Rx チャネル 4 つと Tx チャネル 3 つの場合、最大アクティブチャネル数は 12 です。
Steady state drift compensation rate	-	Defaults to 500	チャネルが定常状態にある場合のドリフト補正レート（N スキャンごとに適用）を決定します。（スキャンレートに依存します）
Startup drift compensation rate	-	Defaults to 5	ドライバーの初期化実行時におけるドリフト補正レート（N スキャンごとに適用）を決定します。(*定常状態ドリフト補正レート*よりも低く設定する必要があります）

ISDE Property	Configuration	Setting	説明
Channel release compensation rate	-	Defaults to 500	チャネルが押された状態から開放された状態に移行する際のドリフト補正レート（N スキャンごとに適用）を決定します。（*定常状態ドリフト補正レート*以下に設定する必要があります）
Default filter depth	-	Defaults to 14	ドライバーが提供するソフトウェアセンサーカウントフィルターで用いられます。このデフォルトフィルターは、比較的一般的な「漏れ積分回路」ソフトウェアフィルターを改良したもので、深さ 4 は約 16 のフィルター定数と同等となります。入力が 16 以上の定数値である場合、フィルター出力はフィルターの 16 サイクル / 繰り返し後に定数入力値の 68 ～ 69% に達します。
Runtime rate of tuning of sensor values	-	Defaults to 800	オフセット調整のレート。（ドリフト補正を使用する場合、この値は定常状態ドリフト補正レートの2倍に設定されます）
DTC Usage for CTSU	-	Defaults to *true*	今回のリリースでは、r_ctsu を使用するには、DTC が必要となります。今後のリリースでは、DTC なしでの r-ctsu データ転送がサポートされる可能性があります。

CTSU モジュール構成パラメータ

ISDE Property	Configuration	Setting	説明
Transfer Instance for DTC Read	p_lower_lvl_transfer_read	-	データ読み取りのための CTSU ドライバーを使用した転送インスタンスの名前

ISDE Property	Configuration	Setting	説明
Transfer Instance for DTC Write	p_lower_lvl_transfer_write	-	構成データの書き込みのための CTSU ドライバーを使用した転送インスタンスの名前
CTSU Hardware Configuration	p_ctsu_hw_cfg	-	Workbench によって生成された構成構造体の名前
Processing Functions	p_ctsu_functions	-	ユーザーがデフォルトの内部処理機能に置き換えて使用するあらゆる処理機能。使用しない場合はNULLに設定できます。
Callback	p_callback	-	スキャンが完了するたびに呼び出されるユーザーコールバック関数。新しい処理済みデータが使用可能になったときにも呼び出されます。使用しない場合は NULL に設定できます。
Processing Option	ctsu_soft_option	-	完了した前回のデータ後にスキャンが開始されるかどうかや、自動較正がスキャンの間に実行されるかどうかなどの指定に使用することはできません。通常はデフォルト設定を使用してください。
Closing Option	ctsu_close_option	-	クローズ時にレジスタステータスをリセットするか保持するか（電力 / メモリは多く消費しますが、短時間で再有効化できます）を指定するのに使用します。

5.2.7.2 CTSU アプリケーションの作成

1) Synergy プロジェクトを生成します。
2) Synergy 設定を開きます。
3) [Pins] タブの [CTSU] セクションに移動し、タッチセンサーピン TSxx として使用する GPIO ピンを有効化します。
4) TSCAP ピンも有効化されていることを確認します。

5) MCU のクロック速度が、外部の CapTouch 調整ツール Workbench 6 で選択されているものと一致することを確認します。

6) [Driver] > [Transfer] > [Transfer Driver on r_dtc] から DTC 転送モジュールを追加します。構成は不要です。

7) CTSU HAL ドライバーを HAL モジュールへ、または [Driver] > [Input] > [CTSU driver on r_ctsu] からスレッドへ追加します。

8) Workbench によって生成された CTSU 構成を含むファイルまたはフォルダーが ${PROJECT_ROOT}\src フォルダーに存在し、そのファイル / フォルダーがビルドに含まれていることを確認します。

9) [ICU] タブで 3 つの CTSU 割り込みをすべて同じプライオリティに設定して、これらの割り込みを有効にします。どのような値を設定しても構いません。

10) r_ctsu 上の CTSU ドライバーの [Properties] タブを使用して、任意の設定で使用されるチャネルの最大数（相互モードの場合はチャネルの組み合わせ）を調整します。

11) ユーザー定義関数にコールバックフィールドを設定します。

12) DTC 転送名フィールドを DTC モジュールの名前に設定します。

13) [CTSU Configuration Used field] フィールドを、 WorkBench によって生成された構成の名前に設定します。

14) [Generate Project Content] をクリックして、必要な構造体を生成します。

15) hal_data.h （または thread_name.c) を開いて、このドライバーを使用するように生成された構造体を確認します。 SSP モジュールの使用方法の概要については、 SSP ユーザーマニュアルを参照してください。

16) Name.api->open を呼び出します。この際、適切に初期化された引数が ctsu_cfg_t 構造体インスタンスを通して提供されていることを確認してください。これはメインの while ループに入る前に行う必要があります。

17) メインの while ループから周期的に Name.api->scan を呼び出し続け、新しいスキャンを開始します。このタスクはタイマを使用して定期的に実行すると便利です。それにより、スキャンレートが決定されます。

18) 各スキャンの終了時には、ユーザーコールバックが呼び出されます。コールバック内で、コールバックのイベントを確認して Name.api->update を呼び出し、データの処理を実行します。

19) 処理が完了すると、ユーザーコールバックが再度、データが処理されたことを示す異なる引数で呼び出されます。

5.2.7.3 CTSU ドライバーの制限事項

ドライバーは Mutual Capacitance Mode および Self Capacitance Multi-Scan Mode のみをサポートします。ドライバーは、いくつかの使用するチャネルが動的に変更されるアプリケーションでの効率的なメモリ使用が可能な動的メモリ割り当てをサポートしません。 SSP のリリースノートも参照してください。

5.2.7.4 CTSU ドライバーファイル

プロジェクト設定中、ISDE により、次の表に記載されているファイルが /ssp ディレクトリに抽出されます。

Module	Directory
CTSU HAL Interface	synergy/ssp/inc/driver/spi/r_ctsu_api.h
CTSU HAL Instance	synergy/ssp/inc/driver/instances/r_ctsu.h
CTSU HAL Driver	synergy/ssp/src/driver/r_ctsu

5.2.7.5 CTSU ドライバーでサポートされるデバイス

このドライバーは、 API への変更なしに、以下のファミリをサポートするように設計されています。

- S7G2
- S3A7
- S128
- S124

CTSU フレームワークは、API への変更なしに、CTSU 周辺デバイスを使用するすべての Synergy デバイスをサポートするように設計されています。

5.2.8 SCE 暗号化ドライバー

SCE HAL モジュールによって構成される暗号モジュールを使用すると、以下の暗号プリミティブを備えたセキュリティ用の暗号プロトコルを作成できます。

- 乱数生成
- AES または Triple DES （3DES）アルゴリズムを使用したデータ暗号化および復号
- RSA または DSA アルゴリズムを使用した署名の生成と検証
- HASH アルゴリズム SHA1、 SHA224、または SHA256 を使用したメッセージダイジェストの計算
- キー生成 - AES ラップキー、 RSA プレーンテキストおよびラップキー

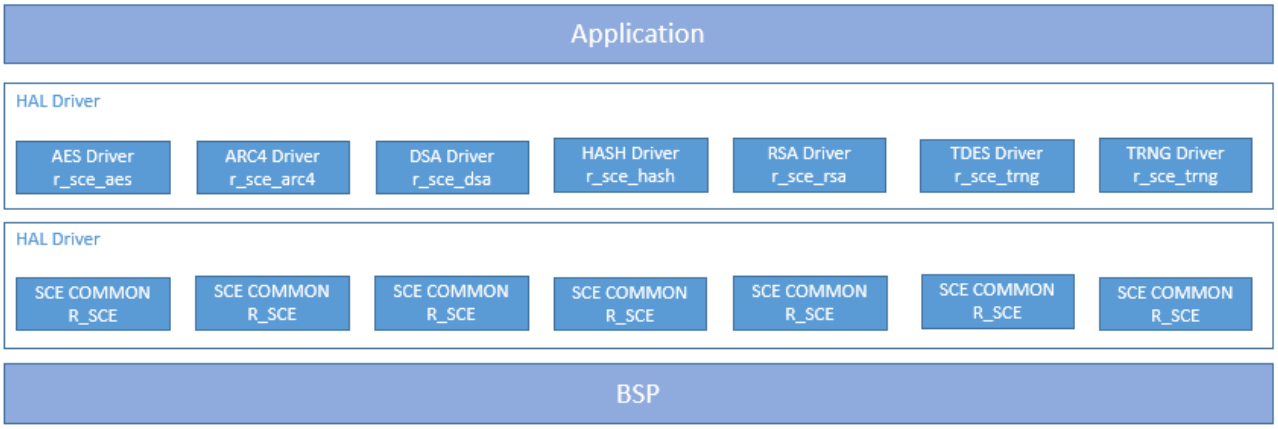


図 329: SCE HAL モジュールのブロック図

5.2.8.1 SCE API の概要

SCE インタフェースは、SCE HAL モジュール用の共通 API を提供します。SCE インタフェースは、選択されたモジュール（AES、ARC4、RSA、DSA、HASH、TDES、TRNG）に応じて、複数の動作をサポートします。

AES インタフェースでは、AES アルゴリズムの使用に関するオープン、クローズ、ラップキーの生成、データの暗号化、データの復号のための API が定義されています。128 ビット、192 ビット、256 ビットのキーと、ECB、CBC、CTR、GCM、または XTS のチェーンモードオプションを使用します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。戻りステータスの値については、『SSP ユーザーズマニュアル』の SCE API リファレンスセクションを参照してください。

AES HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_aes.p_api->open(g_sce_aes.p_ctrl, g_sce_aes.p_cfg); AES モジュールのオープン関数です。暗号化 / 復号操作を実行する前に呼び出す必要があります。
.createKey	g_sce_aes.p_api->close(g_sce_aes.p_ctrl, num_words, p_key); 暗号化 / 復号操作の AES キーを生成します。

Function Name	API の呼び出し例と説明
.encrypt	g_sce_aes.p_api->encrypt(g_sce_aes.p_ctrl, p_key, p_vi, num_words, p_source, p_dest); aes.open() 関数呼び出しで指定されたチェーンモードとパディングモードを使用した AES 暗号化。
.addAdditionalAuthenticationData	g_sce_aes.p_api->addAdditionalAuthenticationData (g_sce_aes.p_ctrl, p_key, p_vi, num_words, p_source); 新しい認証データ（暗号化または復号操作を開始する前に呼び出される）を追加します。
.encryptFinal	g_sce_aes.p_api->encryptFinal(g_sce_aes.p_ctrl, p_key, p_iv, input_num_words, p_source, output_num_words, p_dest); aes.open() 関数呼び出しで指定されたチェーンモードとパディングモードを使用した AES 最終暗号化。
.decrypt	g_sce_aes.p_api->decrypt(g_sce_aes.p_ctrl, p_key, p_iv, num_words, p_source, p_dest); AES の復号。
.setGcmTag	g_sce_aes.p_api->setGcmTag(g_sce_aes.p_ctrl,num_words, p_source); 認証タグデータを設定します。
.getGcmTag	g_sce_aes.p_api->getGcmTag(g_sce_aes.p_ctrl,num_words, p_dest); 認証タグデータを取得します。

Function Name	API の呼び出し例と説明
.zeroPaddingEncrypt	g_sce_aes.p_api->zeroPaddingEncrypt(g_sce_aes.p_ctrl, p_key, p_iv, num_bytes, p_source, p_dest) 指定されたチェーンモードとパディングモードを使用した AES ゼロパディング暗号化。 GCM モード専用の実装 API の使用方法 1. 追加認証データ（AAD）を提供するには : p_dest = NULL に設定します 2. 暗号化 : p_source に入力データを設定し、 p_dest は暗号化されたデータを返します 3. タグの取得 / 計算 : p_source = NULL に設定します

Function Name	API の呼び出し例と説明
.zeroPaddingDecrypt	g_sce_aes.p_api->zeroPaddingDecrypt(g_sce_aes.p_ctrl, p_key, p_iv, num_words, p_source, p_dest); 指定されたチェーンモードとパディングモードを使用した AES ゼロパディング *復号*。 GCM モード専用の実装 API の使用方法 1.setGcmTag() 関数を使用して、予想されるタグ値を設定します 2. 追加認証データ（AAD）を提供するには、p_dest = NULL に設定してこの API を呼び出します 3. *復号*: p_source に暗号化された入力データを設定すると、*復号されたデータ*が p_dest で返されます 4. タグを検証するには、 p_source = NULL および p_dest = NULL を使用してこの API を呼び出すと、 戻り値で認証タグの検証の状態が示されます。
.versionGet	g_sce_aes.p_api->versionGet(p_version); モジュールのコードと API のバージョンを取得して、指定されたポインタ p_version に格納します
.close	g_sce_aes.p_api->close(g_sce_aes.p_ctrl); AES モジュールを閉じます。

ARC4 インタフェースでは、オープン、クローズ、キーの設定、データの処理のための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。

ARC4 HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_arc4.p_api->open(g_sce_arc4.p_ctrl, g_sce_trng.p_cfg); ARC4 モジュールを開きます。
.keySet	g_sce_arc4.p_api->keySet(g_sce_arc4.p_ctrl, &rngbuf, nbytes); ARC4 モジュールによって使用されるキーを設定します。
.arc4Process	g_sce_arc4.p_api-> arc4Process(g_sce_arc4.p_ctrl, nbytes, &source, &destination); ARC4 モジュールを使用して、データを暗号化または復号します。
.close	g_sce_arc4.p_api->close(g_sce_arc4.p_ctrl); ARC4 モジュールを閉じます。
.versionGet	g_sce_arc4.p_api->versionGet (&version); バージョンポインタを使用してバージョンを取得します。

DSA インタフェースでは、オープン、クローズ、デジタル署名、検証のための API が定義されています。使用できるオプションは、1024 ビット公開キーと 160 ビット秘密キー、2048 ビット公開キーと 224 ビット秘密キー、2048 ビット公開キーと 256 ビット秘密キーです。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

DSA HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_dsa.p_api->open(g_sce_dsa.p_ctrl, g_sce_dsa.p_cfg); DSA モジュールのオープン関数です。署名 / 検証操作を実行する前に呼び出す必要があります。
.verify	g_sce_dsa.p_api->verify(p_key, p_domain, num_words, p_signature, p_paddedHash); DSA 公開キーを使用した DSA 署名の検証。この関数は非推奨です。代わりに関数 hashVerify を使用する必要があります。
.hashVerify	g_sce_dsa.p_api->hashVerify(g_sce_dsa.p_ctrl, p_key, p_domain, num_words, p_signature, p_paddedHash); DSA 公開キーを使用した DSA 署名の検証。
.sign	g_sce_dsa.p_api->sign(p_key, p_domain, num_words, p_padded_hash, p_dest); DSA 秘密キーを使用した DSA 署名の生成。この関数は非推奨です。代わりに関数hashSignを使用する必要があります。
.hashSign	g_sce_dsa.p_api->hashSign(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); DSA 秘密キーを使用した DSA 署名の生成。
.close	g_sce_dsa.p_api->close(g_sce_dsa.p_ctrl); DSA モジュールを閉じます。
.versionGet	g_sce_dsa.p_api->versionGet(p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

HASH インタフェースでは、特定のデータセットのハッシュ値を計算するための API が定義されています。使用できるオプションは、SHA1 アルゴリズムと SHA256 アルゴリズムです。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

HASH HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_hash.p_api->open(g_sce_hash.p_ctrl, g_sce_hash.p_cfg); HASH モジュールのオープン関数です。署名 / 検証操作を実行する前に呼び出す必要があります。
.updateHash	g_sce_hash.p_api->updateHash(p_source, num_words, p_dest); ソースバッファ p_source. から num_words 個のワードのハッシュを更新します。この関数は非推奨です。代わりに関数 hashUpdate を使用する必要があります。
.hashUpdate	g_sce_hash.p_api->hashUpdate(g_sce_hash.p_ctrl, p_source, num_words, p_dest); ソースバッファ p_source. から num_words 個のワードのハッシュを更新します。
.close	g_sce_hash.p_api->close(g_sce_hash.p_ctrl); HASH モジュールのクローズ関数です。
.versionGet	g_sce_hash.p_api->versionGet(p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

RSA インタフェースでは、オープン、クローズ、RSA アルゴリズムを使用したデータの暗号化と復号、デジタル署名、アルゴリズムの検証のための API が定義されています。RSA インタフェースでは、1024 ビットまたは 2048 ビットのキーが使用されます。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

RSA HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_rsa.p_api->open(g_sce_rsa.p_ctrl, g_sce_rsa.p_cfg); RSA モジュールのオープン関数。暗号化 / 復号操作または署名 / 検証操作を実行する前に、呼び出す必要があります。
.encrypt	g_sce_rsa.p_api->encrypt(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_source, p_dest); p_key の RSA 公開キーを使用して、p_source のソースデータを暗号化し、結果を宛先バッファ p_dest に書き込みます。
.decrypt	g_sce_rsa.p_api->decrypt (g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_source, p_dest); p_key の RSA 秘密キーを使用して、p_source のソースデータを復号し、結果を宛先バッファ p_dest に書き込みます。
.decryptCrt	g_sce_rsa.p_api->decryptCrt(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_source, p_dest); p_key の RSA 秘密キーを使用して、p_source のソースデータを復号し、結果を宛先バッファ p_dest に書き込みます。RSA 秘密キーデータは、CRT 形式で指定されます。
.verify	g_sce_rsa.p_api->verify(g_sce_rsa.p_ctrl, num_words, p_source); バッファ p_padded_hash 内のパディングされたメッセージハッシュに関するバッファ p_signature 内の署名を、RSA 公開キー p_key を使用して検証します。
.sign	g_sce_rsa.p_api->sign(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); RSA 秘密キー p_key を使用して、パディングされたハッシュバッファ p_padded_hash に対する署名を生成します。結果をバッファ p_dest に書き込みます。

Function Name	API の呼び出し例と説明
.signCrt	g_sce_rsa.p_api->signCrt(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); RSA 秘密キー p_key を使用して、パディングされたハッシュバッファ p_padded_hash に対する署名を生成します。RSA 秘密キー p_key は、CRT 形式と想定されます。結果をバッファ p_dest に書き込みます。
.close	g_sce_rsa.p_api->close(g_sce_rsa.p_ctrl); RSA モジュールを閉じます。
.keyCreate	g_sce_rsa.p_api->keyCreate(g_sce_rsa.p_ctrl, p_private_key, p_public_key); RSA キーを生成します
.versionGet	g_sce_rsa.p_api->versionGet(p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

TDES インタフェースでは、TDES 標準に従ってデータを暗号化および復号するための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。

TDES HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_tdes.p_api->open(g_sce_tdes.p_ctrl, g_sce_tdes.p_cfg); TDES モジュールを開きます。

Function Name	API の呼び出し例と説明
.encrypt	g_sce_tdes.p_api->read(g_sce_tdes.p_ctrl, &key, &iv, nwords, &source, &destination); データを暗号化します。
.decrypt	g_sce_trng.p_api->close(g_sce_tdes.p_ctrl, &key, &iv, nwords, &source, &destination); データを復号します。
.close	g_sce_tdes.p_api->close(g_sce_tdes.p_ctrl); TDES モジュールを閉じます。
.versionGet	g_sce_tdes.p_api->versionGet(p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

TRNG インタフェースでは、乱数ジェネレータを計算するための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。

TRNG HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_sce_trng.p_api->open(g_sce_trng.p_ctrl, g_sce_trng.p_cfg); ハードウェア TRNG モジュールから乱数データを読み取るために、 TRNG ドライバーを開きます。
.read	g_sce_trng.p_api->read(g_sce_trng.p_ctrl, p_rngbuf, nbytes); nbytes の乱数バイトを生成して、 p_rngbuf バッファに格納します。

Function Name	API の呼び出し例と説明
.close	g_sce_trng.p_api->close(g_sce_trng.p_ctrl); TRNG インタフェースドライバーを閉じます。
.versionGet	g_sce_trng.p_api->versionGet(p_version); バージョンを取得し、指定されたポインタ p_version に格納します。

5.2.8.2 SCE HAL モジュールの動作の概要

異なるターゲット MCU ごとに、異なる暗号関数を使用できます。次の表では、各 MCU シリーズで利用できる機能を示します。

Function	S7G2、S5D9、S5D5	S3A3、S3A7、S3A6	S124、S128	注記
TRNG	乱数の生成と読み取り	乱数の生成と読み取り	乱数の生成と読み取り	乱数の生成と読み取り
AES	暗号化、復号、ラップキーの生成	暗号化、復号、ラップキーの生成	暗号化、復号	AES 標準に基づく共通キー暗号化
AES Key Size	128 ビット、192 ビット、256 ビット	128 ビット、256 ビット	128 ビット、256 ビット	
AES Key Type	プレーンテキスト / 未加工キー、ラップキー	プレーンテキスト / 未加工キー、ラップキー	プレーンテキスト / 未加工キー	
AES Chaining Modes	ECB、CBC、CTR、GCM、XTS	ECB、CBC、CTR、GCM	ECB、CBC、CTR	
ARC4	暗号化、復号	任意の記号	任意の記号	
TDES	暗号化、復号	任意の記号	任意の記号	
TDES Key Size	192-bit	任意の記号	任意の記号	
TDES Chaining Modes	ECB、CBC、CTR	任意の記号	任意の記号	

Function	S7G2、S5D9、S5D5	S3A3、S3A7、S3A6	S124、S128	注記
RSA	署名生成、署名検証、公開キー暗号化、秘密キー復号、ラップキー生成	任意の記号	任意の記号	秘密キーの操作に CRT キーと標準キーをサポート
RSA Key Size	1024 ビット、2048 ビット	任意の記号	任意の記号	
RSA Key Type	プレーンテキスト / 未加工キー、ラップキー	任意の記号	任意の記号	
DSA	署名生成、署名検証	任意の記号	任意の記号	
DSA Key Size	(1024, 128) ビット、(2048, 224) ビット、(2048, 256) ビット	任意の記号	任意の記号	
HASH	SHA1、SHA224、SHA256	任意の記号	任意の記号	メッセージダイジェストアルゴリズム

R_SCE モジュールの構成設定

SCE のエンディアンは、デフォルトでビッグエンディアンに設定されます。これをリトルエンディアンモードに設定することができます。

エンディアン構成パラメータの使用方法については、動作に関する注意事項を参照してください。

AES モジュールの構成設定

AES モジュールは、ユーザーが指定したキーの長さ、キーの種類（プレーンテキストまたはラップキー）、チェーンモードを使用するように構成できます。

ARC4 モジュールの構成設定

ARC4 モジュールは、ユーザーが指定したキーの長さとキーの種類（プレーンテキストまたはラップキー）を使用するように構成できます。

DSA モジュールの構成設定

DSA モジュールは、ユーザーが指定したキーの長さを使用するように構成できます。

HASH モジュールの構成設定

HASH モジュールは、ユーザーが指定した HASH アルゴリズム（対象の MCU に依存）を使用するように構成できます。

RSA モジュールの構成設定

RSA モジュールは、ユーザーが指定したキーの長さを使用するように構成できます。

TDES モジュールの構成設定

TDES モジュールは、ユーザーが指定したチェーンモードの種類を使用するように構成できます。

TRNG モジュールの構成設定

乱数生成では、基礎となるハードウェアが、前回生成した乱数とは異なる固有の 16 バイトの乱数を生成する最大試行回数を設定できます。最大試行回数に達した場合、読み取り API は呼び出し側に対しエラーを返します。それ以外の場合は正常終了コードが返され、呼び出し側が供給するデータバッファに生成された乱数が転送されます。

SCE HAL モジュールの動作に関する注意事項と制限事項

- Synergy S7 および S5 デバイスは SCE7 を備えているので、AES、TRNG、RSA、HASH、DSA をサポートします。
- Synergy S3 デバイスは SCE5 を備えているので、AES、TRNG、GHASH をサポートします。GHASH は、AES GCM モードの一部としてサポートされます。 Synergy S3 デバイスは、 SHA1/SHA256 HASH の機能をサポートしません。
- Synergy S1 デバイスは、 AES および TRNG のみをサポートします。
- サポートされていないモジュールがプロジェクトに追加された場合、そのことを示すコンパイラの警告が生成されます。
- すべてのモジュールは versionGet API をサポートし、この API はモジュールが開かれる前であっても呼び出すことができます。
- R_SCE モジュールの interfaceGet API は、フレームワークレイヤーの SF CRYPTO モジュールが使用するために提供されています。
- すべての crypto API は、入力パラメータが null ポインタまたは無効の場合に、 SSP_ERR_ASSERTION を返す可能性があります。すべての API は、sf_crypto_err_t または ssp_err_t に記載されている、uint32_t. 型に収まるエラーコードを返します。
- 暗号化ハードウェアエンジンは、再入可能性をサポートしていません。暗号化ハードウェアエンジンがタスクの実行でビジー状態の場合、新しい要求は状態エラーコード SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT. を受け取ります。
- エンディアン構成パラメータの使用方法 : デフォルトのモードはビッグエンディアンであり、入力と出力には uint32_t.The を使用する必要があります。リトルエンディアンモードにすると、 uint8_t/ バイト配列を使用でき、その場合は入力と出力の両方で uint32_t にキャストする必要があります。次に例を示します。
  - データが uint32_t およびビッグエンディアンフォーマットの場合は、ビッグエンディアンモードを選択します format:uint32_t test_data[5] = {0x84983E44, 0x1C3BD26E, 0xBAAE4AA1, 0xF95129E5, 0xE54670F1};
  - 同じデータがバイト配列フォーマットになっている場合は、リトルエンディアンモードを選択します formatuint8_t test_data_byte_array[20] ={0x84, 0x98, 0x3E, 0x44, 0x1C, 0x3B, 0xD2, 0x6E, 0xBA, 0xAE, 0x4A, 0xA1, 0xF9, 0x51, 0x29, 0xE5, 0xE5, 0x46, 0x70, 0xF1};
- keyCreate API によって生成される RSA キーのフォーマットは次のとおりです。
  - RSA キーのフォーマット :
  - **RSA 公開キーのフォーマット : バイト 0 ～ 3: 公開キー exponentByte 4: RSA modulus(128 bytes for RSA 1024-bit and 256 bytes for RSA 2048-bit keys)**

- **RSA 秘密キーのプレーンテキスト標準フォーマット : バイト 0: 秘密キー指数（RSA 1024 ビットキーの場合は 128 バイト、 RSA 2048 ビットキーの場合は 256 バイト）、その後に RSA 係数。（RSA 1024 ビットキーの場合は 128 バイト、 RSA 2048 ビットキーの場合は 256 バイト）**
- RSA 秘密キーのプレーンテキスト CRT formatThe: コンポーネントはバイト 0 を exponent2 として次の順序で並べられます。
- **exponent2 // 第 2 係数の CRT 指数、正の整数**
- **prime2 // 第 2 係数、正の整数**
- **exponent1 // 第 2 係数の CRT 指数、正の整数**
- **prime1 // 第 1 係数、正の整数**
- **coefficient // （第 1） CRT 係数、正の整数**

- AES の encrypt() 関数と decrypt() 関数は、データのパディングをサポートしません。これらの関数は、16 バイトの倍数であるデータ長で動作します。（データパディングは、ユーザーアプリケーションで処理する必要があります）。AES GCM モードでは、16 バイトの倍数ではない認証データへのサポートが必要となる場合があります。これをサポートするため、AES GCM モードでのみ zeroPaddingEncrypt() および zeroPaddingDecrypt() 関数 API が提供されています。
- TDES の encrypt() 関数と decrypt() 関数は、データのパディングをサポートしません。これらの関数は、8 バイトの倍数であるデータ長で動作します。（データパディングは、ユーザーアプリケーションで処理する必要があります）。
- このモジュールの最新の制限事項については、 SSP の最新のリリースノートを参照してください。

5.2.8.3 アプリケーションへの SCE HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SCE HAL モジュールを組み込む方法について説明します。

***注 :** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

SCE ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（SCE HAL モジュールのデフォルト名は r_sce. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

SCE ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sce_aes_0 AES Driver on r_sce_aes	Threads	New Stack> Driver> Crypto> AES Driver on r_sce_aes
g_sce_arc4_0 ARC4 Driver on r_sce_arc4	Threads	New Stack> Driver> Crypto> ARC4 Driver on r_sce_arc4

Resource	ISDE Tab	Stacks Selection Sequence
g_sce_dsa_0 DSA Driver on r_sce_dsa	Threads	New Stack> Driver> Crypto> DSA Driver on r_sce_dsa
g_sce_hash_0 HASH Driver on r_sce_hash	Threads	New Stack> Driver> Crypto> HASH Driver on r_sce_hash
g_sce_rsa_0 AES Driver on r_sce_aes	Threads	New Stack> Driver> Crypto> RSA Driver on r_sce_rsa
g_sce_tdes TDES Driver on r_sce_tdes	Threads	New Stack> Driver> Crypto> TDES Driver on r_sce_tdes
g_sce_trng TRNG Driver on r_sce_trng	Threads	New Stack> Driver> Crypto> TRNG Driver on r_sce_trng

次の図に示すように、r_sce の暗号化ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。

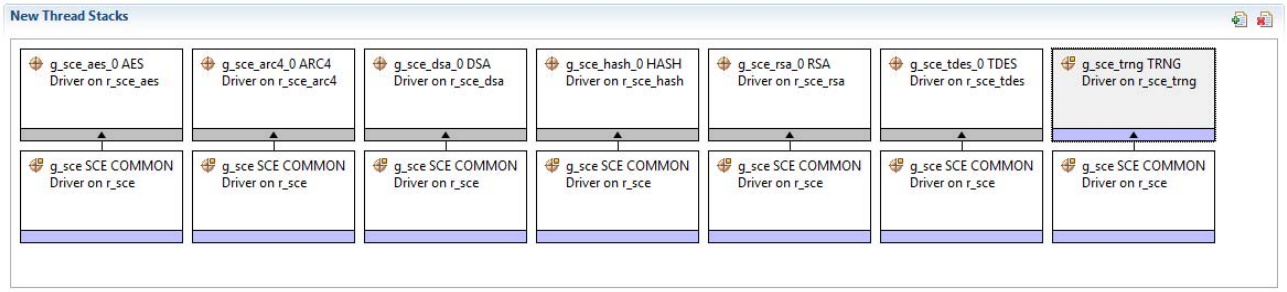


図 330: SCE HAL モジュールのスタック（AES、ARC、DSA、HASH、RSA、TDES、TRNG が含まれます）

5.2.8.4 SCE HAL モジュールの構成

ユーザーは必要な動作に合わせて SCE HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

AES HAL モジュールのパラメータと設定

ISDE Property	Value	説明
Name	g_sce_aes_0	モジュール名
Key Length	User defined, default is 128 bits.Allowed values for S7G2 and S5D9 devices: 128, 192, or 256 bits.Allowed values for S3A7 and S124 devices: 128 or 256 bits	ドライバーのこのインスタンスにより暗号化/復号操作に用いられるキーの長さ
Channel	User defined, default is CBC.Allowed values for S7G2 and S5D9 devices: ECB, CBC, CTR, GCM, XTS.S3A7 デバイスに許容される値 : ECB、 CBC、CTR、GCM。Allowed values for S124 device: ECB, CBC, CTR	ドライバーのこのインスタンスにより暗号化 / 復号操作に用いられる、ブロック暗号チェーンモード

ARC4 HAL モジュールのパラメータと設定

ISDE Property	Value	説明
Name	g_sce_arc4_0	モジュール名
Key Length	Default 0	バイト数でのキーの長さ
Key Name	g_arc4_0_key	キー名は、ユーザーコードで unit8_array 型のデータとして定義されている必要があります

DSA HAL モジュールのパラメータと設定

Parameter	Value	説明
Name	g_sce_dsa_0	モジュール名
Key Length	User defined, default is (2048, 256) bits.Allowed values for S7G2 and S5D9 devices: (1024, 160), (2048, 224) or (2048, 256) bits Allowed values for S3A7 and S124 devices: Not available.	ドライバーのこのインスタンスにより署名 / 検証操作に用いられるキーの長さ。

HASH HAL モジュールのパラメータと設定

Parameter	Value	説明
Name	g_sce_hash_0	モジュール名
Algorithm	User defined, default is SHA256.Allowed values for S7G2 and S5D9 devices: SHA1, or SHA256.Allowed values for S3A7 and S124 devices: Not available.	メッセージデータ上のメッセージダイジェスト/ハッシュの計算に用いられるアルゴリズム。

RSA HAL モジュールのパラメータと設定

Parameter	Value	説明
Name	g_sce_rsa_0	モジュール名
Key Length	User defined, default is 2048 bits.Allowed values for S7G2 and S5D9 devices: 1024 or 2048 bits.Allowed values for S3A7 and S124 devices: Not available.	ドライバーのこのインスタンスにより署名 / 検証 / 暗号化 / 復号操作に用いられるキーの長さ

TDES HAL モジュールのパラメータと設定

Parameter	Value	説明
Name	g_sce_tdes	モジュール名
Chaining Mode	ECB, CBC, CTR	チェーンモードの選択

TRNG HAL モジュールのパラメータと設定

Parameter	Value	説明
Name	g_sce_trng	モジュール名
Max. Attempts	User defined, default is 2	新たに生成された乱数が前回生成された乱数とは異なる場合の、最大試行回数を設定します

5.2.8.5 アプリケーションでの SCE HAL モジュールの使用

アプリケーションで SCE HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、モジュールを初期化します。
2) 対応する API（たとえば、AES addAdditionalAuthenticationData）を使用して、パラメータを指定します。
3) open API を使用して、機能を開始します （たとえば、 AES open）。
4) encrypt API を使用して、データを暗号化します。
5) decrypt API を使用して、データを復号します。
6) SCE close API を使用して、 SCE インスタンスを閉じます。

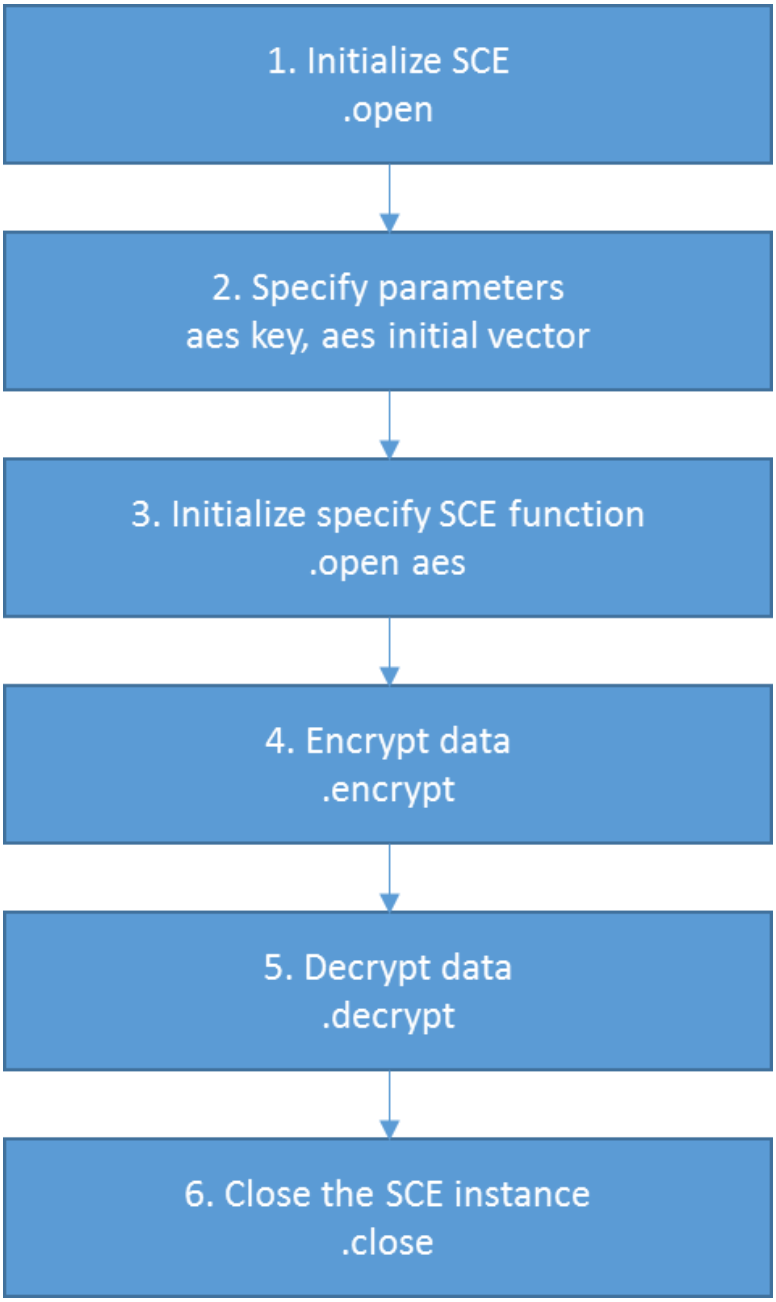


図 331: 一般的な暗号化アプリケーションのフロー図

以下では、具体的な使用例を示します。

1) SCE モジュールを使用するには：

- open API を使用して、SCE 共通ドライバーで SCE および SCE HAL モジュール (R_SCE) を初期化します。

- すべての HAL API の入力 / 出力データのエンディアン（リトルエンディアンまたはビッグエンディアン）を構成します。デフォルトでは、ビッグエンディアンモードが構成されます。エンディアンの構成の詳細については、 HAL モジュールの動作に関する注意事項を参照してください。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。

2) AES の関数を使用するには：

- open API で AES インタフェースのインスタンスを初期化します。
- 関連付けられた API を使用するための、インスタンスの構成パラメータを指定します。使用できる AES キーサイズは、128 ビット、192 ビット、256 ビットです。サポートされているチェーンモードは、 ECB、 CBC、 CTR、 GCM、 XTS です。
- データを暗号化するには、 encrypt API を使用します。
- データを復号するには、 decrypt API を使用します。
- ラップキーを生成するには、createKey API を使用します。（プレーンテキストキーは、TRNG モジュールを使用して生成できます）。
- close API でインタフェースのインスタンスを閉じます。

3) TDES の関数を使用するには：

- open API で TDES インタフェースのインスタンスを初期化します。
- 関連付けられた API を使用するための、インスタンスの構成パラメータを指定します。 TDES のチェーンモード（ECB、 CBC、 CTR）を選択します。
- encrypt API を使用して、データを暗号化します。
- decrypt API を使用して、データを復号します。
- close API でインタフェースのインスタンスを閉じます。

4) ARC4 の関数を使用するには：

- open API で ARC4 インタフェースのインスタンスを初期化します。
- 関連付けられた API を使用するための、インスタンスの構成パラメータを指定します。長さ（64 ビットまたは 2048 ビット）と場所で ARC4 キーを指定できます。
- 使用するキーは、 keySet API で設定できます。
- データを暗号化または復号するには、 arc4Process API を使用します。
- close API でインタフェースのインスタンスを閉じます。

5) RSA の関数を使用するには：

- サポートされているキーの長さは、 1024 ビットと 2048 ビットです。
- 使用するキーの長さを基にして RSA インタフェースのインスタンスを選択し、 open API で初期化します。
- RSA 公開キーを使用してデータを暗号化するには、 encrypt API を使用します。
- RSA 公開キーを使用してデータを復号するには、 decrypt API を使用します。

- CRT フォーマットの RSA 公開キーを使用してデータを復号するには、 decryptCrt API を使用します。
- 標準フォーマットの RSA 公開キーを使用して、特定のパディングされたハッシュに対する署名を生成するには、 sign API を使用します。
- CRT フォーマットの RSA 秘密キーを使用して、特定のパディングされたハッシュに対する署名を生成するには、 signCrt API を使用します。
- 標準フォーマットの RSA 公開キーを使用して、特定のパディングされたハッシュに対する署名を検証するには、 verify API を使用します。
- RSA キーまたはラップキーを生成するには、 keyCreate API を使用します。
- close API でインタフェースのインスタンスを閉じます。

6) DSA の関数を使用するには：

- サポートされているキーの長さは、（1024、 160）ビット、（2048、 224）ビット、（2048、 256）ビットです。
- 使用するキーの長さを基にして DSA インタフェースのインスタンスを選択し、 open API で初期化します。
- DSA 秘密キーを使用して署名を生成するには、 hashSign API を使用します。
- DSA 公開キーを使用して署名を検証するには、 hashVerify API を使用します。
- close API でインタフェースのインスタンスを閉じます。

7) HASH アルゴリズムを使用するには：

- SHA1 ハッシュ方式と SHA256 ハッシュ方式がサポートされています。使用する方式を選択します。
- 使用するハッシュ方式を基にして HASH インタフェースのインスタンスを選択し、 open API で初期化します。
- メッセージのダイジェストを計算するには、 hashUpdate API を使用します。
- close API でインタフェースのインスタンスを閉じます。

8) 真性乱数ジェネレータの関数を使用するには：

- open API で TRNG インタフェースのインスタンスを初期化します。
- read API で乱数を生成します。
- close API でインタフェースのインスタンスを閉じます。

9) close API を使用して、 SCE および SCE HAL モジュールを閉じます。

5.2.9 DAC ドライバー

このモジュールは、デュアルチャネル 12 ビット D/A コンバータ （DAC12）を、正および負の基準電圧の間の 4096 段階の電圧レベルのいずれかを出力するように構成します。このモジュールには以下の構成設定が含まれます。

- 16 ビット入力データレジスタに対する左詰めまたは右詰めの 12 ビット値フォーマットを設定します
- 出力アンプを有効または無効にします
- アナログ / デジタルコンバータ （ADC）モジュールを使用して、同期干渉防止モードで動作します。

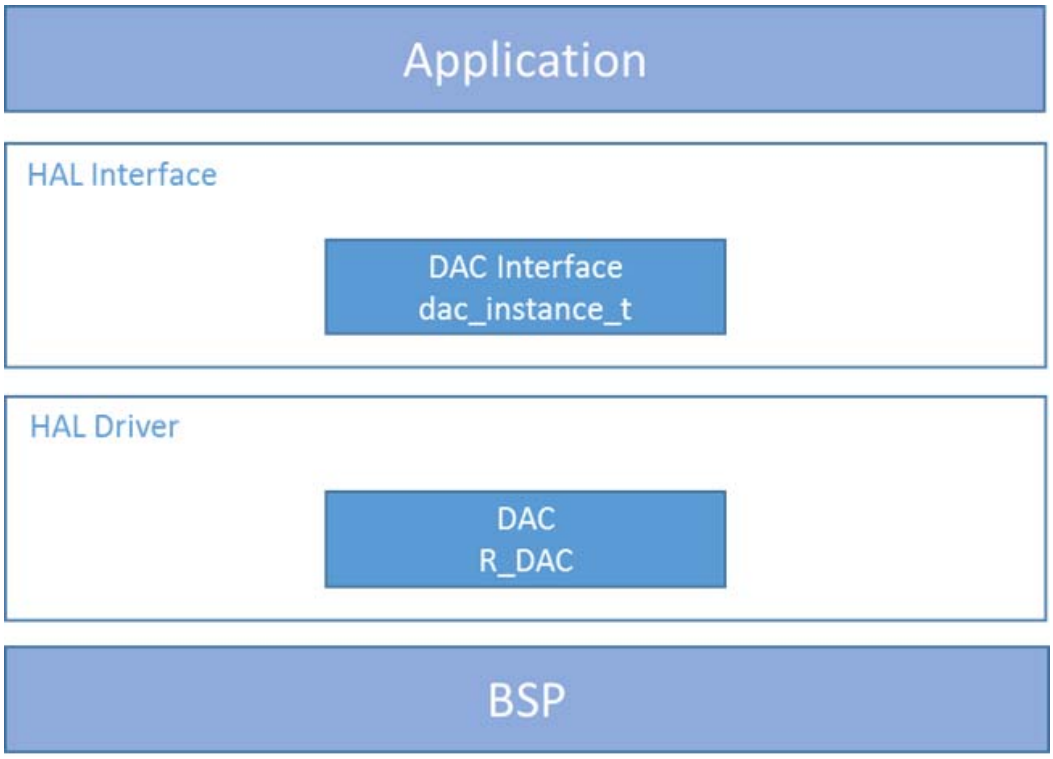


図 332: DAC HAL モジュールのブロック図

5.2.9.1 DAC HAL モジュールの API の概要

DAC HAL モジュールでは、オープン、クローズ、開始、停止、および DAC へのライトを行うための API が定義されています。次の表では、使用可能な各 API のリストと呼び出しの例、および簡単な説明を示します。ステータス戻り値の一覧は API 要約表の後にあります。

DAC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_dac.p_api->open(g_dac.p_ctrl, g_dac.p_cfg) 初期構成。

Function Name	API の呼び出し例と説明
close	g_dac.p_api->close(g_dac.p_ctrl) D/A コンバータを閉じます。
write	g_dac.p_api->write(g_dac.p_ctrl, val) D/A コンバータにサンプル値を書き込みます。
start	g_dac.p_api->start(g_dac.p_ctrl) D/A コンバータが動作していない場合は開始します。
stop	g_dac.p_api->stop(g_dac.p_ctrl) D/A コンバータが動作している場合は停止します。
versionGet	g_dac.p_api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。

***注：** 関連するモジュールについては、『SSP ユーザーズマニュアル』の API リファレンスを参照してください。SSP では、動作、関数データ構造体、型定義、定義、API データ、API 構造体、関数変数が定義されています。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_NOT_OPEN	ユニットが開いていません。
SSP_ERR_ASSERTION	正しくないパラメータ。
SSP_ERR_IN_USE	DAC リソースはロックされています。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。関連するモジュールの関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』の API リファレンスを参照してください。*

5.2.9.2 DAC HAL モジュールの動作の概要

DAC HAL モジュールは、デュアルチャネル 12 ビット D/A コンバータ （DAC12）を、正および負の基準電圧の間の 4096 段階の電圧レベルのいずれかを出力するように構成します。このモジュールを使用して、16 ビット入力データレジスタに対する 12 ビット出力の左詰めフォーマットまたは右詰めフォーマットを構成できます。DAC HAL モジュールでは、出力アンプを有効または無効にしたり、ADC HAL モジュールを使用して同期干渉防止モードで動作したりすることもできます。

DAC HAL モジュールの動作に関する重要な注意事項と制限事項

DAC HAL モジュールでは、以下の初期化手順が必要です。

- DAC モジュールストップビットをゼロにクリアします。
- DAC チャネル出力有効化を 1 に設定します。

DAC モジュールストップビットは、ドライバーのインスタンスカウンタがゼロの場合、open を呼び出した時点でゼロにクリアされます。ドライバーのインスタンスカウンタは、ゼロに初期化された後、チャネルの open が正常に戻るとインクリメントされ、チャネルの close が呼び出されるとデクリメントされます。 DAC モジュールストップビットが 1 に設定されるのは、ドライバーのインスタンスカウンタが close の呼び出しで 0 にデクリメントされたときです。

DAC チャネル出力有効は、 open が正常に呼び出された後で初めてチャネルの write が呼び出された時点で、1 に設定されます。 open の呼び出しでは、 dac_ctrl_t 構造体の要素 channel_started. に 0 が書き込まれます。channel_started が 0 にクリアされた状態で write が呼び出されると、そのチャネルの DAC 出力有効ビットは 1 に設定されます。チャネルの DAC 出力有効は、 close および stop が呼び出されると 0 にクリアされます。

- DAC HAL モジュールには、ピン構成は実装されていません。現在、 DA0 と DA1 の出力は、ピン構成コントロールレジスタの ASEL フィールドのリセット値によって有効になります。
- DAC HAL モジュールの電圧基準の選択は実装されていません。現在、D/A VREF コントロールレジスタ （DAVREFCR）のリセット値によって基準は選択されませんが、これは正しい状態です。
- 変換をトリガする DAC 入力イベントの構成は、現在実装されていません。コントロールレジスタの DAE ビットのデフォルトのリセット値 （0）では、各チャネルを個別にトリガできます。
- DAC HAL モジュール変換の同期のためのイベント信号入力は、現在実装されていません。
- このモジュールの動作に関する制限事項については、最新の SSP リリースノートを参照してください。

5.2.9.3 アプリケーションへの DAC HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに DAC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

DAC ドライバーをアプリケーションに追加するには、次の表のスタック選択シーケンスを使用します。（DAC のデフォルト名は g_dac0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

表 DAC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dac0 DAC Driver on r_dac	Threads	New Stack > Driver > Analog > DAC Driver on r_dac

次の図に示すように、DAC の DAC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

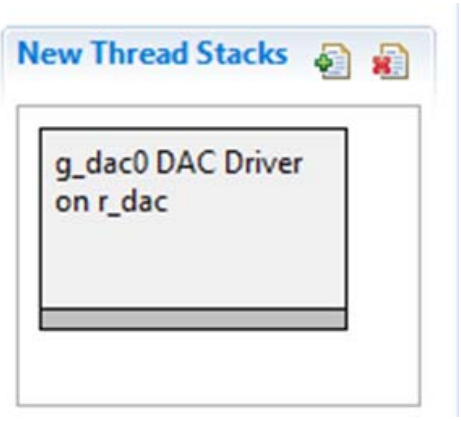


図 333: DAC HAL モジュールのスタック

5.2.9.4 DAC HAL モジュールの構成

ユーザーは必要な動作に合わせて DAC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

r_dac 上の DAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します。
Name	g_dac0	モジュール名。
Channel	0,1	出力 DA0 の場合は 0、出力 DA1 の場合 1 を設定します。
Synchronize with ADC	Enabled, Disabled (Default: Disabled)	A/D コンバータ (ADC) モジュールを使用した干渉防止同期の場合は true を設定します。アナログモジュール間の電源干渉が問題にならない場合、または DAC モジュールによる非同期変換が望ましい場合は、 false に設定します。
Data Format	Right Justified, Left Justified (Default: Right Justified)	12 ビットデータ値をビット 11 から 0 にロードする（右詰め）場合は、 0 に設定します。12 ビットデータ値をビット 15 から 4 にロードする（左詰め）場合は 1 を設定します。
Output Amplifier	Enable, Disable (Default: Disable)	出力アンプハードウェア機能が望ましい場合は true を設定します。出力アンプハードウェア機能をバイパスする場合は false を設定します。

注: 例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、望ましい動作に応じて異なるイベントを選択するときに役立つ場合があります。

DAC HAL モジュールのクロック構成

DAC HAL モジュールには、固有のクロック構成は必要ありません。

DAC HAL モジュールのピン構成

DAC HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ピンコンフィギュレータでアナログピンとして設定する必要があります。次の表では、 SSP 構成ウィンドウでピンを選択する方法を示します。その後の表では、 DAC ピンの構成設定を示します。

r_dac での DAC ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin Selection Sequence
DAC	Pins	Select Peripherals > Analog: DAC12 > DAC120

r_dac での DAC ドライバーのピン構成設定

Property	Value	説明
Module Name	DAC120	DAC 周辺モジュール。
Operation Mode	Enabled, Disabled (Default: Enabled)	DAC 周辺動作モード。
DA0	None, DA0 (Default: None)	DAC 出力ピン。
DA1	None, DA1 (Default: None)	DAC 出力ピン。

注: *設定例では、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトを示します。他の Synergy キットおよび Synergy MCU では、使用可能なピン構成設定が異なる場合があります。*

5.2.9.5 アプリケーションでの DAC HAL モジュールの使用

アプリケーションでの DAC HAL モジュールの一般的な使用手順は次のとおりです。

1) open API を使用して、 DAC HAL モジュールを初期化します。
2) write API を使用して、データ値を書き込みます。
3) start API を使用して、データのライトを開始します。
4) 必要に応じて、 write API を使用してデータ値のライトを続けます。

5) stop API を使用して、データのライトを停止します。

6) close API を使用して、 DAC HAL モジュールを閉じます。

次の図は、通常の動作フローでの一般的な手順です。

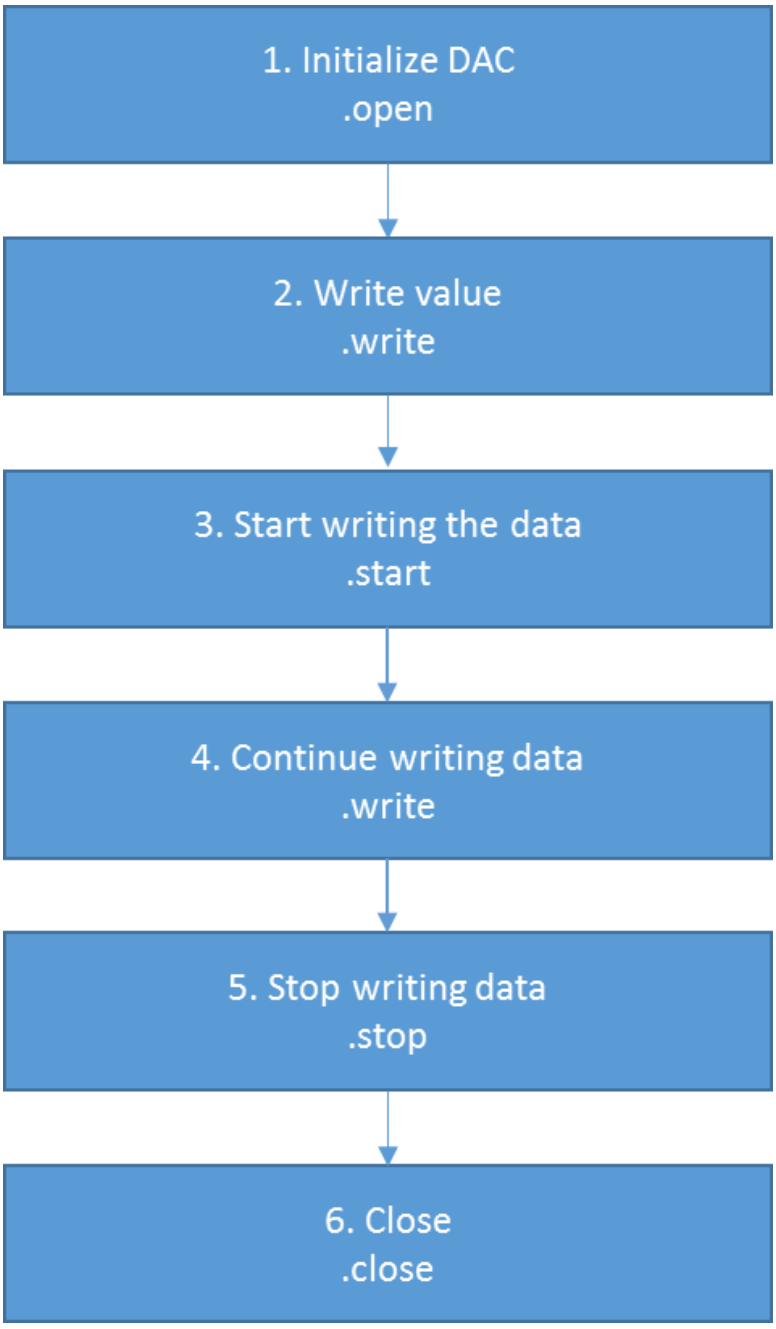


図 334: 一般的な DAC HAL モジュールアプリケーションのフロー図

5.2.10 DAC8 ドライバー

DAC8 HAL モジュールは、r_dac8. に実装されているデジタル / アナログ変換アプリケーションのための高レベル API です。 DAC8 HAL モジュールは、 Synergy S128 MCU の 8 ビット D/A コンバータ （DAC8）をサポートします。

5.2.10.1 DAC8 HAL モジュールの機能

- Synergy S128 MCU で使用できます
- 8 ビット D/A コンバータ - 3 チャネル
- 左詰めまたは右詰めの入力データフォーマット
- アナログ / デジタルコンバータ （ADC）モジュールとの同期
- 複数の動作モード
  - 通常
  - リアルタイム （イベントリンク）
- チャージポンプコントロール

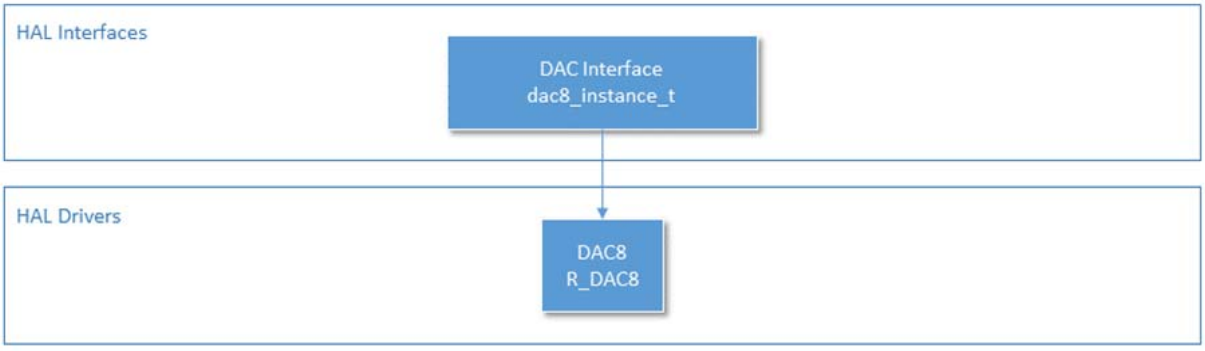


図 335: DAC8 HAL モジュールの編成、オプション、スタックの実装

5.2.10.2 DAC8 HAL モジュールの API の概要

DAC8 HAL モジュールでは、オープン、クローズ、開始、停止、DAC へのライトのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

DAC8 HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_dac8.p_api->open(g_dac8.p_ctrl, g_dac8.p_cfg) 初期構成。
close	g_dac8.p_api->close(g_dac8.p_ctrl) D/A コンバータを閉じます。
write	g_dac8.p_api->write(g_dac8.p_ctrl, val) D/A コンバータにサンプル値を書き込みます。
start	g_dac8.p_api->start(g_dac8.p_ctrl) D/A コンバータが動作していない場合は開始します。
stop	g_dac8.p_api->stop(g_dac8.p_ctrl) D/A コンバータが動作している場合は停止します。
versionGet	g_dac8.p_api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_HW_LOCKED	DAC リソースはロックされています。
SSP_ERR_NOT_OPEN	ユニットが開いていません。

Name	説明
SSP_ERR_ASSERTION	正しくないパラメータ。
SSP_ERR_IP_CHANNEL_NOT_PRESENT	正しくないチャネルが選択されました。

*注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル *』で関連モジュールの API リファレンスを参照してください。*

5.2.10.3 DAC8 HAL モジュールの動作の概要

DAC8 HAL モジュールは、 8 ビット D/A コンバータ （DAC8）を、正および負の基準電圧の間の 256 段階の電圧レベルのいずれかを出力するように構成します。16 ビット入力データの左詰めまたは右詰めフォーマットで 8 ビット出力データを受け付けるように、ドライバーを構成できます。このドライバーは、 2 モードの DAC をサポートします。

- 通常モード –D/A 出力は、データレジスタへのライト時に更新されます。
- リアルタイム（イベントリンク）–D/A 出力は、イベントリンクのイベント時に更新されます。このモードの間は、いつでもデータレジスタに書き込むことができます。イベントリンクイベントは変換の開始をトリガします。詳細については、『*SSP ユーザーズマニュアル*』の 「ELC インタフェース」を参照してください。

ADC のリードでのノイズを減らすには、ドライバーで ADC モジュールを使用して同期干渉防止モードを構成できます。このモードは、 ADC のサンプリング中に DAC チャージを無効にすることによって変換ノイズを減らします。この機能がサポートされているかどうかは、ハードウェアのマニュアルで確認してください。

低 AVCC 電圧での動作では、ドライバーはハードウェアチャージポンプを有効または無効にできます。

5.2.10.4 DAC8 HAL モジュールの動作に関する重要な注意事項と制限事項

5.2.10.5 DAC8 HAL モジュールの動作に関する注意事項

DAC8 チャネルは、 dac8_api_t::start() および dac8_api_t::write() の間は有効になり、 dac8_api_t::stop() および dac8_api_t::close() の間は無効になります。

5.2.10.6 DAC8 HAL モジュールの制限事項

- DAC8 は、 Synergy S128 MCU 上で利用可能な DAC8 を使用します。
- DAC8 ドライバーでは、アナログ出力用の出力ピンは構成されません。
- DAC8 ドライバーでは、リアルタイムモード用の ELC は構成されません。ユーザーは、 DAC8 モジュールでリアルタイムモードを有効にするだけでなく、イベントリンクコントローラを構成する必要があります。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.10.7 アプリケーションへの DAC8 HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DAC8 HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

DAC8 ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（DAC のデフォルト名は g_dac8_0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

DAC 選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dac8_0 DAC Driver on r_dac8	Threads	New Stack > Driver > Analog > DAC Driver on r_dac8

次の図に示すように、DAC の DAC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

図 336: DAC8 HAL モジュールのスタック

5.2.10.8 DAC8 HAL モジュールの構成

必要な動作に合わせて DAC8 HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに

対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

DAC8 HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_dac8_0	モジュール名
Channel	0	チャネルの選択 -3 つのチャネルがサポートされます。
Synchronize with ADC	Enabled, Disabled (Default: Disabled)	ADC モジュールと同期するかどうかを選択します
Data Format	Right Justified, Left Justified (Default: Right Justified)	データフォーマットの選択
DAC Mode	Normal Mode, Real-time (Event Link) Mode (Default: Normal Mode)	DAC モードの選択
Charge Pump Enabled (Requires MOCO active)	Enabled, Disabled (Default: Enabled)	チャージポンプを有効または無効にします

注: *上記の設定例とデフォルトは、Synergy S1 MCU ファミリを使用するプロジェクトに対するものです。*

モジュールのデフォルトのプロパティ以外の設定が望ましい場合もあります。たとえば、アプリケーションによっては出力アンプを有効にすると役に立つことがあります。

5.2.10.9 DAC8 HAL モジュールのクロック構成

DAC8 HAL モジュールでは、特定のクロック構成は必要ありません。

5.2.10.10 DAC8 HAL モジュールのピン構成

DAC8 HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ピンコンフィギュレータでアナログピンとして設定する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では DAC ピンの選択例を示します。

r_dac8 での DAC8 ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin Selection Sequence
DAC8	Pins	Select Peripherals > Analog:DAC8 > DAC80

r_dac8 での DAC8 ドライバーのピン構成設定

Property	Value	説明
Module Name	DAC80	DAC 周辺モジュール。
Operation Mode	Enabled, Disabled (Default: Enabled)	DAC 周辺動作モード。
DA0	None, DA0 (Default: None)	DAC 出力ピン。
DA1	None, DA1 (Default: None)	DAC 出力ピン。

***注:** 前の設定例は、 Synergy S128 MCU および DK-S128 キットを使用するプロジェクトに対するものです。*

5.2.10.11 アプリケーションでの DAC8 HAL モジュールの使用

アプリケーションで DAC8 HAL モジュールを使用する一般的な手順は次のとおりです。

1) dac8_api_t::open API を使用して、 DAC8 HAL モジュールを初期化します。
2) dac8_api_t::write API を使用して、値を書き込みます。
3) dac8_api_t::start API を使用して、変換を開始します。
4) 必要に応じて、 dac8_api_t::write API を使用して値のライトを続けます。
5) dac8_api_t::stop API を使用して、変換を停止します。
6) dac8_api_t::close を呼び出して、ペリフェラルの電源をオフにします。

前記の手順を、次の図の通常の動作フロー図に示します。

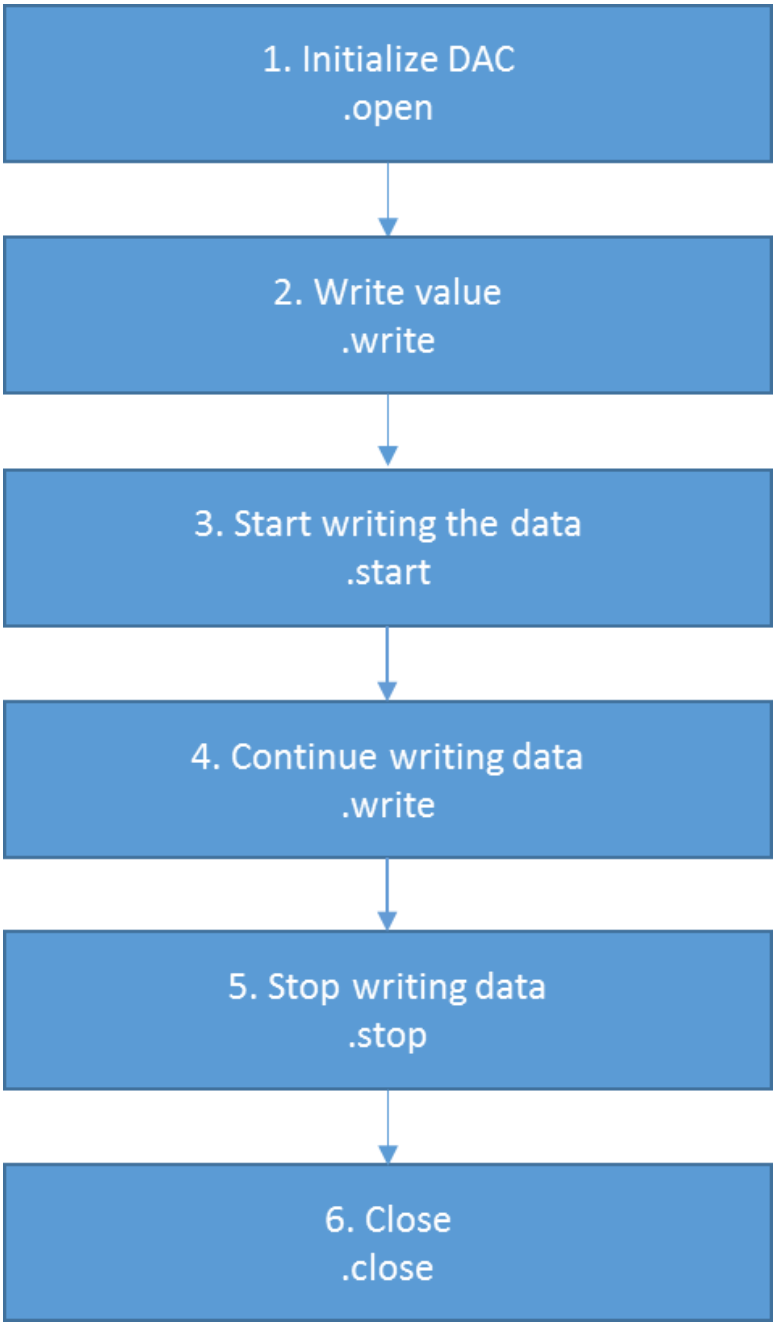


図 337: 一般的な DAC8 HAL モジュールアプリケーションのフロー図

5.2.11 GLCDC ディスプレイドライバー

- RGB インタフェース（最大 24 ビット）と同期信号（HSYNC、VSYNC、およびデータ有効（オプション））で LCD パネルをサポートします
- 入力グラフィックプレーン用の各種の色フォーマットをサポートします（RGB888、ARGB888、RGB565、ARGB1555、ARGB4444、CLUT8、CLUT4、CLUT1）
- 512 ワード（32 ビット / ワード）の入力グラフィックプレーンのためのカラールックアップテーブル（CLUT）の使用をサポートします
- 出力用の各種の色フォーマットをサポートします（RGB888、RGB666、RGB565、シリアル RGB）
- 背景プレーン上に 2 つのグラフィックプレーンを入力し、画面上でブレンドできます
- パネルにドットクロックを生成します。クロックソースは内部または外部 (LCD_EXTCLK) から選択できます
- 明るさ調整、コントラスト調整、ガンマ補正をサポートします
- フレームバッファの切り替えまたはアンダーフロー検出を処理するための、GLCDC 割り込みをサポートします

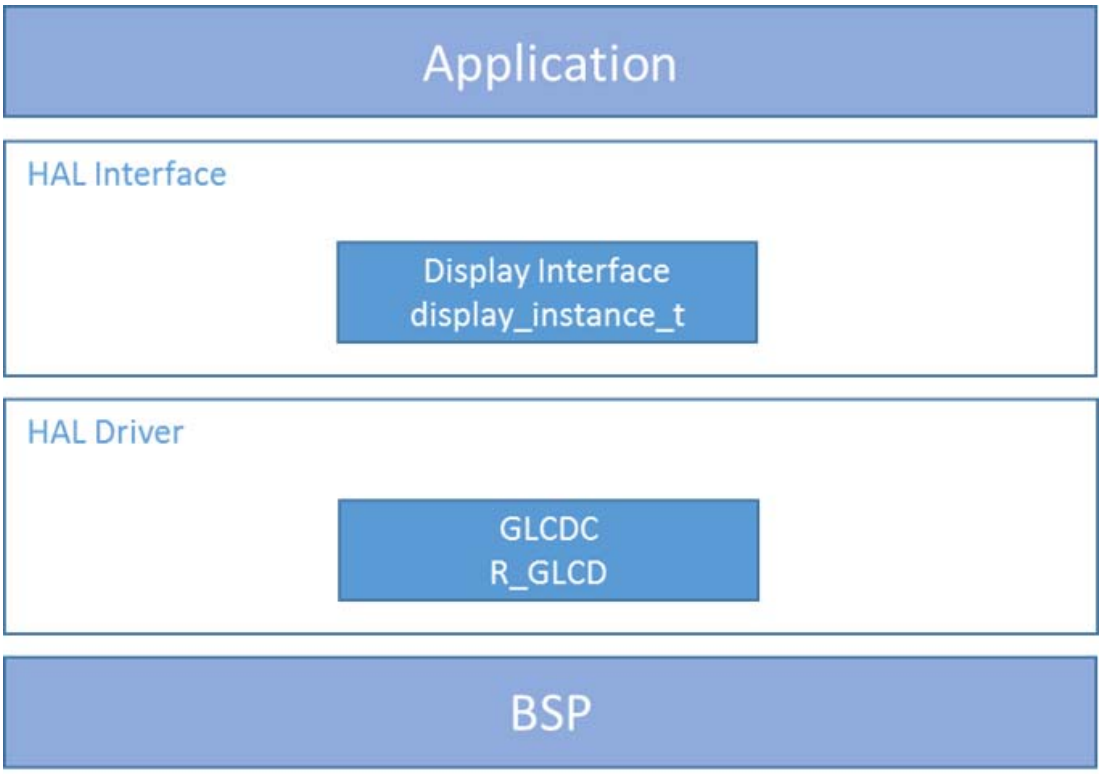


図 338: GLCDC HAL モジュールのブロック図

5.2.11.1 GLCDC HAL モジュールの API の概要

GLCDC HAL モジュールでは、オープン、クローズ、開始、停止、および LCD パネルへの情報表示の制御のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

GLCDC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_display.p_api->open (g_display.p_ctrl, g_display.p_cfg); 表示デバイスを開きます。
close	g_display.p_api->close (g_display.p_ctrl); 表示デバイスを閉じます。
start	g_display.p_api->start(g_display.p_ctrl); 表示の開始。
stop	g_display.p_api->stop(g_display.p_ctrl); 表示の停止。
layerChange	g_display.p_api->layerChange(g_display.p_ctrl, &layercng, frame) ランタイムにレイヤーパラメータを変更します。
correction	g_display.p_api->correction(g_display.p_ctrl, &display_correction) 色補正。
clut	g_display.p_api->clut(g_display.p_ctrl, &clut, frame) 表示デバイスに対して CLUT を設定します。

Function Name	API の呼び出し例と説明
statusGet	g_display.p_api->statusGet(g_display.p_ctrl, &status) 表示デバイスのステータスを取得します。
versionGet	g_display.p_api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。

注: 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。
SSP_ERR_INVALID_ARGUMENT	引数の無効なパラメータ。
SSP_ERR_HW_LOCKED	GLCDCC リソースがロックされています。
SSP_ERR_CLOCK_GENERATION	ドットクロックはクロックソースから生成できません。
SSP_ERR_INVALID_TIMING_SETTING	無効なパネルタイミングパラメータ。
SSP_ERR_INVALID_LAYER_SETTING	無効なレイヤ設定が見つかりました。
SSP_ERR_INVALID_LAYER_FORMAT	無効なフォーマットが指定されました。
SSP_ERR_INVALID_GAMMA_SETTING	無効なガンマ修正設定が見つかりました。
SSP_ERR_NOT_OPEN	ドライバーの状態が DISPLAY_STATE_CLOSED. と等しいときに、関数呼び出しが実行されます。
SSP_ERR_INVALID_UPDATE_TIMING	GLCDC が内部でレジスタ値を更新しているときに、関数呼び出しが実行されます。
SSP_ERR_INVALID_MODE	ドライバー状態が DISPLAY_STATE_OPENED でないときに、関数呼び出しが実行されます。

Name	説明
SSP_ERR_INVALID_CLUT_ACCESS	不正な CLUT エントリまたはサイズが指定されました。
p_version	バージョン番号。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.11.2 GLCDC HAL モジュールの動作の概要

GLCDC HAL モジュールは LCD パネルを制御します。下図に、GLCDC HAL モジュールを使用したときのグラフィックデータフローの概要を示します。このモジュールは、メモリからのグラフィックフレーム画像データのリード（最大 2 フレーム）と、これら画像のモノクロ背景画面上でのブレンドをサポートしています。このドライバーは CLUT メモリをサポートしており、 CLUT 用のグラフィックフレーム形式を指定します。

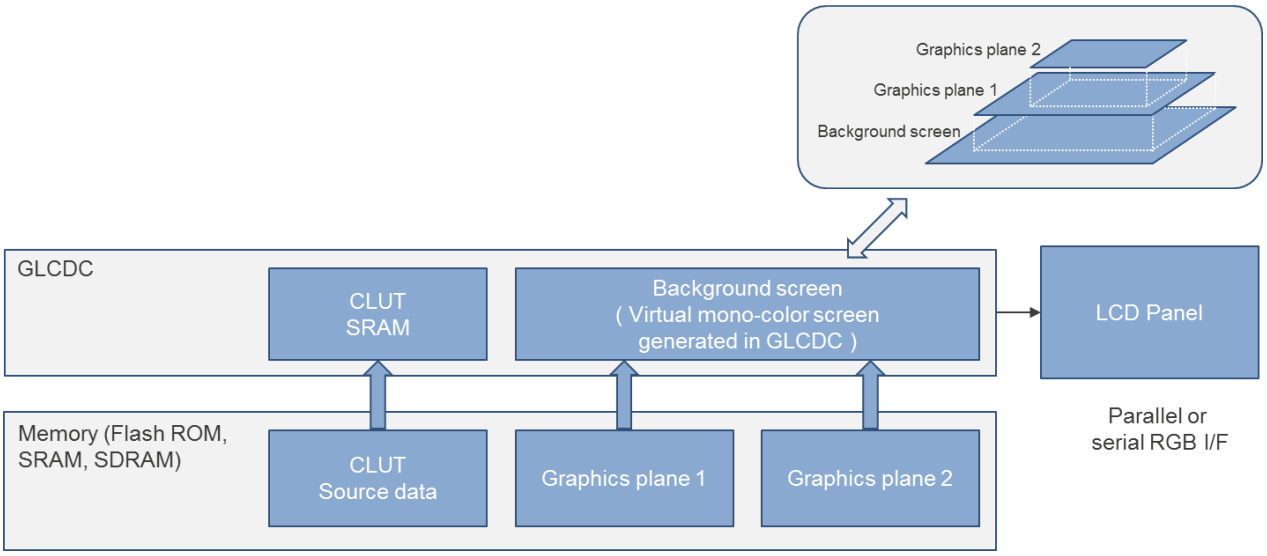


図 339: GLCDC のデータフロー

下の図は、ピンポンフレームバッファを備えた表示システムを示しています。単一フレームバッファ表示システムで発生する画像の切れ目を回避するため、表示システムでは 3 つ以上のフレームバッファを使用することをお勧めします。そのような設計では、GLCDC ハードウェアがグラフィックフレーム画像をいずれかのフレームバッファから読み込んでいる間に、画像描画エンジン （DRW や JPEG など）、CPU、または DMAC/DTC が同時にグラフィックフレーム画像を別のフレームバッファに転送することができます。このモジュールは、 layerChange API による実行時のフレームバッファ切り替えをサポートしています。

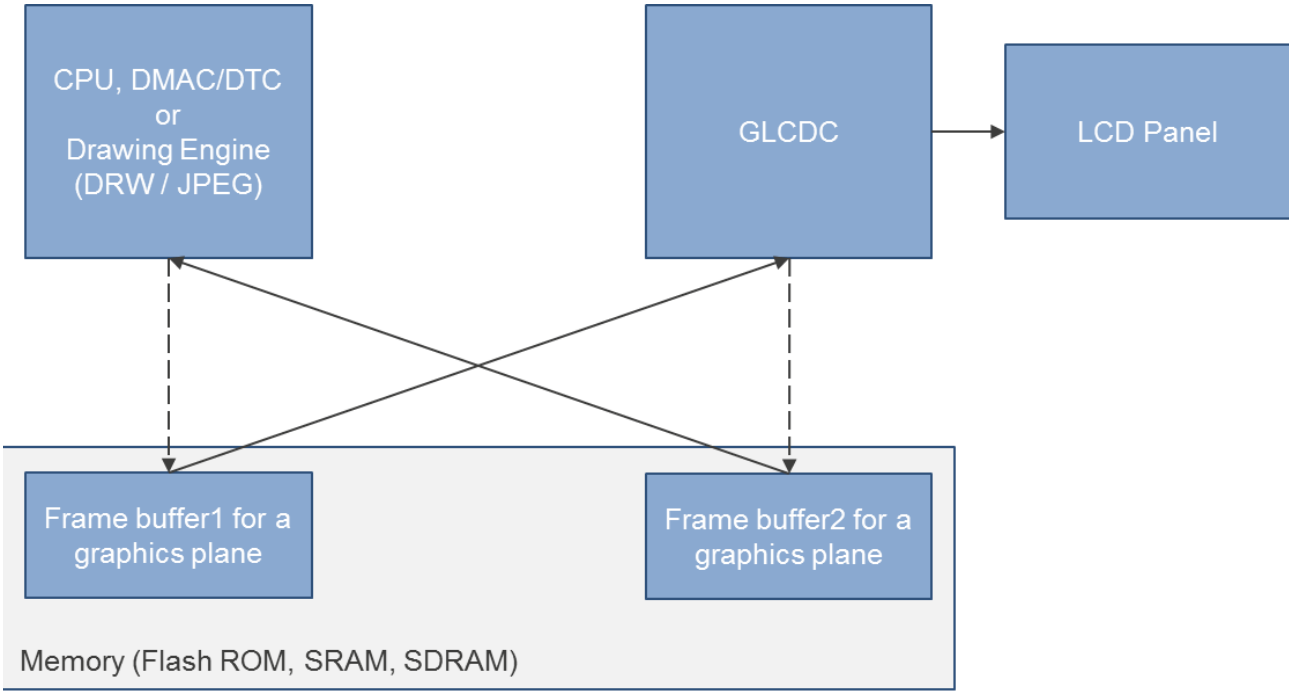


図 340: GLCDC ディスプレイ - GLCDC を使用した一般的なピンポンバッファシステム

画面のフォーマット

次の図は、GLCDC モジュールの LCD 画面フォーマットと LCD タイミングパラメータの関係を示したものです。このモジュールは、さまざまな LCD パネルをサポートする LCD パネル設定用の汎用タイミングパラメータを備えています。

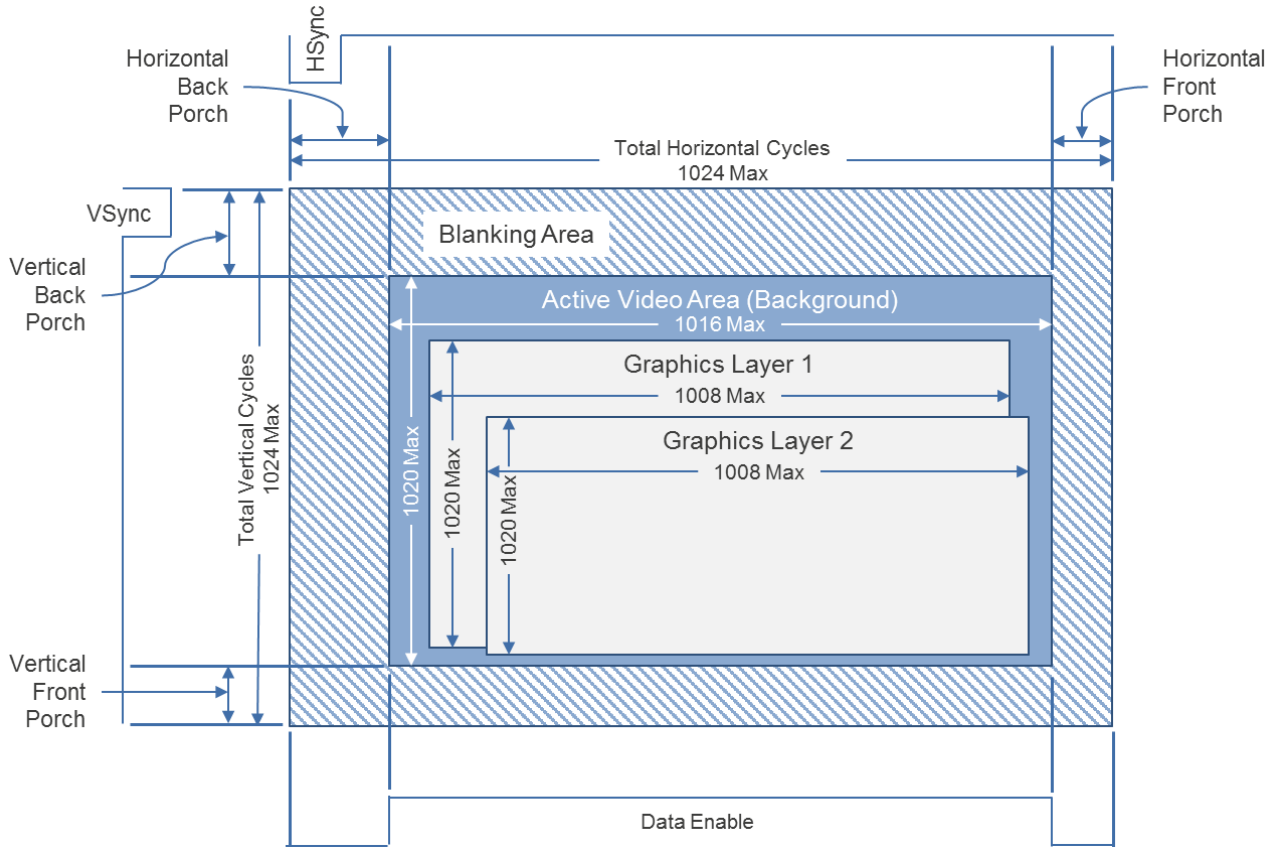


図 341: GLCDC 画面のフォーマット

フロントポーチ期間

GLCDC モジュールは、垂直 / 水平フロントポーチサイクル / ラインに対する設定を備えていません。これらのサイクル / ラインは、合計水平サイクル / 垂直ライン設定に含まれている必要があります。

注: *このモジュールでは、GLCDC のハードウェア仕様に基づいて、バックポーチサイクル / ラインを設定する必要があります。一般的な LCD パネルのバックポーチサイクル / ラインは説明する数より多いため、これはモジュールの真の制限ではありません。*

- 水平バックポーチサイクル数 >= 3
- 垂直バックポーチライン数 >= 2

パラメータ設定例

PE-HMI1 v2.0 ボード (LXD Research & Display, LLC, M7504A):

以下の例では、 60Hz の LCD パネルリフレッシュレートを生成するため、水平合計サイクル、垂直合計ライン、パネルクロック分周比を調整しています。LCD パネルの記号については、M7504A データシートを参照してください。

LCD パネルのパラメータの設定 —PE-HMI1 v2.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen1/2 input horizontal size	800(thd)
Graphics screen1/2 input vertical size	480(tvd)
Graphics screen1/2 input horizontal stride	800(thd)
Horizontal total cycles	976(th)
Horizontal active video cycles	800(thd)
Horizontal back porch cycles	46(thp + thb)
Horizontal sync signal cycles	20(thp)
Horizontal sync signal polarity	Low active
Vertical total lines	512(tv)
Vertical active video lines	480(tvd)
Vertical back porch cycles	23(tvp + tvb)
Vertical sync signal lines	10(tvp)
Vertical sync signal polarity	Low active
Output Format	24bits RGB888
Data Enable Signal Polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON0
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON2
TCON - Panel clock division ratio	1/8

DK-S7G2 v3.0 ボード (LXD Research & Display, LLC, M7190A):

以下の例では、 60Hz の LCD パネルリフレッシュレートを生成するため、水平合計サイクル、垂直合計ライン、パネルクロック分周比を調整しています。LCD パネルの記号については、M7190A データシートを参照してください。

LCD パネルのパラメータの設定 —DK-S7G2 v3.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen N input horizontal size	480(thd)
Graphics screen N input vertical size	272(tvd)
Graphics screen N input horizontal stride	480(thd)
Horizontal total cycles	582(th)
Horizontal active video cycles	480(thd)
Horizontal back porch cycles	43(thp + thb)
Horizontal sync signal cycles	41(thp)
Horizontal sync signal polarity	Low active
Vertical total lines	286(tv)
Vertical active video lines	272(tvd)
Vertical back porch cycles	12(tvp + tvb)
Vertical sync signal lines	10(tvp)
Vertical sync signal polarity	Low active
Output Format	16bits RGB565
Data Enable Signal Polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON1
TCON - Vsync pin select	LCD_TCON2
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/24

SK-S7G2 v2.0 ボード (ILI Technology Corp., IL9341C):

以下の例では、水平合計サイクルと垂直合計ラインを、約 76.8Hz の LCD パネルリフレッシュレートのパネルに許容される最大値に設定しています。 LCD パネルの記号については、 LIL9314V データシートを参照してください。

LCD パネルのパラメータの設定 —SK-S7G2 v2.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen N input horizontal size	256(See note)
Graphics screen N input vertical size	320(VAdr)
Graphics screen N input horizontal stride	256(See note)
Horizontal total cycles	290(HAdr + Hsync(16) + HBP(24) + HFP(16))
Horizontal active video cycles	240(HAdr)
Horizontal back porch cycles	40(Hsync(16) + HBP(24))
Horizontal sync signal cycles	16(Hsync)
Horizontal sync signal polarity	Low active
Vertical total lines	330(VAdr + Vsync(4) + VBP(2) + VFP(4))
Vertical active video lines	320(VAdr)
Vertical back porch cycles	6(Vsync + VBP)
Vertical sync signal lines	4(Vsync)
Vertical sync signal polarity	Low active
Output Format	16bits RGB565
Data Enable Signal Polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON2
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/32

***注：** パネルのパラメータは 240 ピクセルに設定する必要がありますが、入力幅および水平ストライドは、意図的に 256 ピクセルに設定されています。これは、GLCDC ハードウェアに合わせ、水平ラインを 64 バイトにする必要があるためです。ライン開始点での 240 ピクセルが有効であれば、ラインの残りのピクセル（16 ピクセル）は考慮されません。*

CLUT

GLCDC モジュールは、色フォーマットが ARGB1555、CLUT8、CLUT4、または CLUT1 の場合に使用されるカラールックアップテーブルをサポートします。CLUT API は、CLUT0/CLUT1 SRAM（GLCDC ハードウェア内部で実装）を、グラフィックの前景画面または背景画面それぞれに対して更新できます。

***注:** CLUT を使用する色フォーマットを選択する場合は、start API を使用する前に、必ず CLUT API を呼び出してください。そうしないと、CLUT0 と CLUT1 が不明な状態となり、グラフィックが正しく表示されません。*

また、実行時に CLUT API を呼び出して、 CLUT SRAM を更新することもできます。

***注:** API は、 CLUT データのソースを、現在使用されていない CLUT SRAM にコピーします（各 CLUT SRAM はピンポンバッファで構成されます）。CLUT データ更新の完了後、API は、画像が切れるのを避けるため、GLCDC ハードウェアが読み取る CLUT SRAM を次のフレームから自動的に切り替えます。*

ライン繰り返しモード

ライン繰り返しは、メモリが十分にないシステムで特に重要なモードです。このモードでは、GLCDC モジュールは LCD パネルの画面サイズより少ないピクセルのラスタ画像を読み込み、ラスタを繰り返し画面に表示します。下の図は、背景グラフィックプレーンに小さいラスタ画像を繰り返し読み込むことで構築された画面の例です。

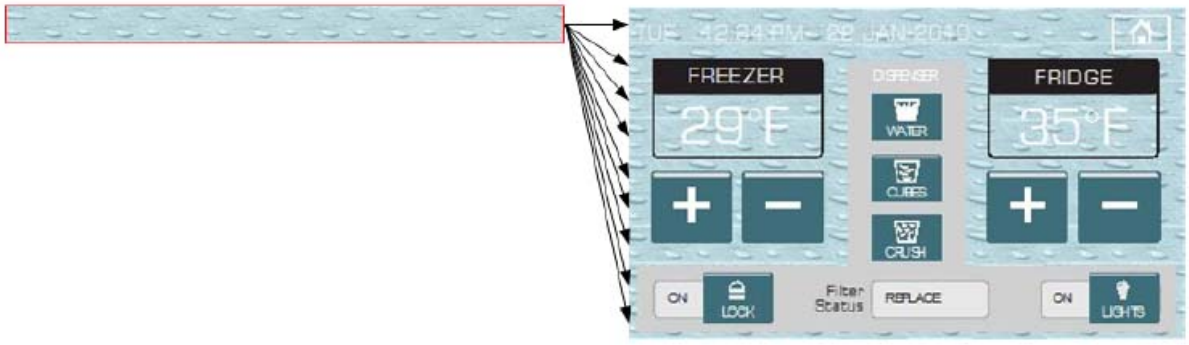


図 342: GLCDC ライン繰り返しモード

***注:** このモードを有効にするには、 Synergy コンフィギュレータで GLCDC モジュールのプロパティ [Input - Graphics screen N input lines repeat]（N = 1 または 2）を [ON] に設定します。また、ラスタ画像を読み込む繰り返し回数を [Input - Graphics screen N input lines repeat times] と指定します。ラスタ画像の水平ピクセルサイズを [Input - Graphics screen N input horizontal size] と [Input - Graphics screen N input horizontal stride] に指定し、ラスタ画像の垂直ピクセルサイズを [Input - Graphics screen N input vertical size] に指定します。*

ガンマ補正

ガンマ補正は、 LCD パネルの色特性をフラットな特性に変更するために使用します。下の図は、GLCDC モジュールで設定可能なガンマ補正カーブです。このモジュールは、（R、G、B）色のそれぞれについて、入力色レベルに対して 16 個のしきい値をサポートしており、 16 のエリアそれぞれに対してしきい値で割ったゲインレベルを定義します。

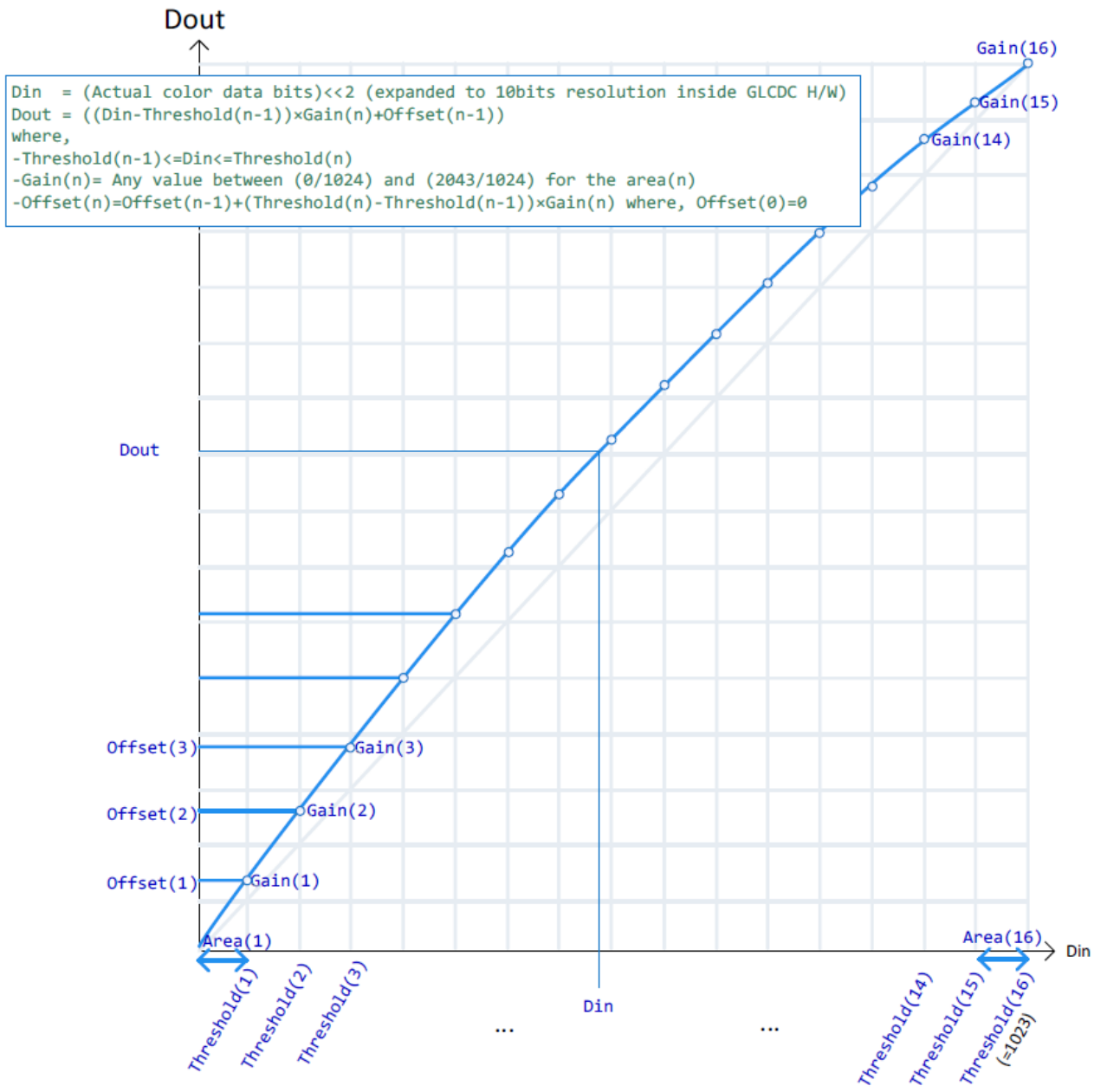


図 343: GLCDC ガンマ補正カーブ

注: *（R、G、B）チャネルのそれぞれについてガンマ補正を有効にするには、Synergy コンフィギュレータを使用して GLCDC モジュールのプロパティ [Color correction – Gamma correction (R, G, B)] を [ON] に設定します。しきい値（合計 16 個）は、[Color correction – Gamma correction threshold (R, G, B) \[n\]]（n=[0..15]）に設定します。各エリアのゲイン値は、[Color correction – Gamma correction gain (R, G, B) \[n\]]（n=[0..15]）に設定します。*

GLCDC HAL モジュールの動作に関する重要な注意事項と制限事項

以降のセクションで説明するように、複数の GLCDC 割り込みを構成することもできます。

ライン検出割り込み

ライン検出割り込みは、GLCDC がすべてのラインの LCD パネルへの出力を完了し、ブランク期間になったことを示すために使用されます。この割り込みを使用してグラフィックシステムでのフレームバッファの切り替えを処理し、3 つ以上のフレームでフレームバッファを使用します。

layer1 または layer2 バッファアンダーフローの割り込み

GLCDC layer1 または layer2 のバッファアンダーフロー割り込みを使用すると、システムのメモリ帯域幅不足を検出できます。バッファアンダーフローが発生するのは、メモリ（SDRAM や SRAM など）から GLCDC の内部ラインバッファへのグラフィックデータ転送が、他のデータ転送によってブロックされ、GLCDC ラインバッファから LCD パネルインタフェースへのデータ転送に対して十分でない場合です。この割り込みが発生しないように、グラフィックシステムを設計する必要があります。

GLCDC コールバック

ユーザーコールバック関数を open で登録できます。ユーザーコールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービスルーチン（ISR）からコールバック関数が呼び出されます。コールバック関数イベントの引数では、グラフィックシステムで発生したイベントをユーザーが識別できるように、表に示す列挙値を受け取ることができます。DISPLAY_EVENT_LINE_DETECTION イベントは、画面を更新するためにフレームバッファを切り替えるために使用でき、DISPLAY_EVENT_GRn_UNDERFLOW イベントは、アンダーフローが発生した場合のエラー処理に使用できます。

イベントと割り込みの要約

Name of Event	Name of Interrupt	イベントの条件
DISPLAY_EVENT_LINE_DETECTION	ライン検出	GLCDCが、アクティブなビデオ領域内の最後のラインを出力した場合
DISPLAY_EVENT_GR1_UNDERFLOW	グラフィックス 1 アンダーフロー	graphics1プレーンのデータのリード中に GLCDC がアンダーフローした場合
DISPLAY_EVENT_GR2_UNDERFLOW	グラフィックス 2 アンダーフロー	graphics2プレーンのデータのリード中に GLCDC がアンダーフローした場合

***注：** コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*

- r_GLCDC 上のディスプレイドライバーは、RGB インデックス彩度キーをサポートしていません。
- r_GLCDC 上のディスプレイドライバーは、イベントリンク機能をサポートしていません。
- このモジュールに適用されるその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.11.3 アプリケーションへの GLCDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに GLCDC HAL モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

GLCDC HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（ディスプレイドライバーのデフォルト名は g_display0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

GLCDC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_display0 Display Driver on r_glcdc	Threads	New Stack> Driver> Graphics> Display Driver on r_glcdc

次の図に示すように、r_glcdc の GLCDC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

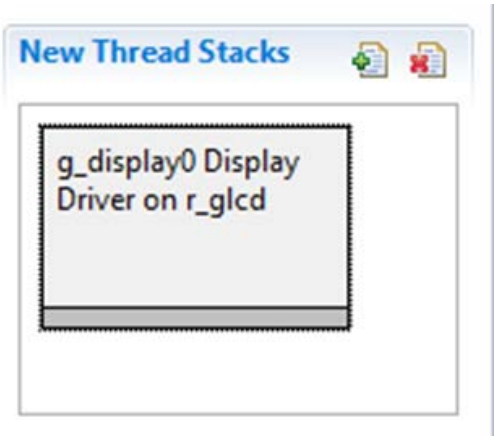


図 344: GLCDC HAL モジュールのスタック

5.2.11.4 GLCDC HAL モジュールの構成

ユーザーは必要な動作に合わせて GLCDC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更

できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_glcdc での GLCDC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効または無効にします。
Name	g_display0	GLCDC モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Name of display callback function to be defined by user	NULL	名前は有効な C シンボルである必要があります。
Input – Panel clock source select	Internal clock(GLCDCLK), External clock(LCD_EXTCLK) (Default: Internal clock)	システムに応じてパネルクロックソースを選択します。

ISDE Property	Value	説明
Input – Graphics screen1	Used, Not used (Default: Used)	グラフィックス画面 N を使用する場合は「使用」を指定します。これにより、グラフィックス画面 1 用の「display_fb_background」と、グラフィックス画面 2 用の「display_fb_foreground」が、ISDE により自動生成されます。いずれのグラフィック画面も使用しない場合は、[Not used] を指定します。その場合、フレームバッファは作成されません。[Not used]を指定した場合、フレームバッファに対するメモリリードアクセスがなく、バス帯域幅の消費量が減ることに注意してください。
Input – Graphics screen1 frame buffer name	fb_background	フレームバッファのカスタム名。
Input – Number of Graphics screen1 frame buffer	2	グラフィック画面 1 のフレームバッファ番号
Input – section where Graphics screen1 frame buffer allocated	sdram	フレームバッファを割り当てるセクション名を指定します。これは、[Input – Graphics screen1] が [Used] に設定されている場合に有効です。
Input – Graphics screen1 input horizontal size	800	幅のピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input – Graphics screen1 vertical size	480	高さのピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input – Graphics screen1 input horizontal stride (not bytes but pixels)	800	水平ラインのメモリストライドを指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。一般に、このパラメータはパラメータ [input horizontal size] と同じ値に設定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。

ISDE Property	Value	説明
Input – Graphics screen1 input format	32 bits ARGB888, 32 bits RGB888, 16 bits RGB565, 16 bits ARGB1555, 16 bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 (Default: 16 bits RGB565)	グラフィックス画面の入力形式を指定します。 CLUT フォーマットを選択する場合は、 clut を使用して CLUT データを書き込んでから start を実行する必要があります。デフォルト設定では、 RGB565 形式の画像がサポートされます。
Input – Graphics screen1 input line descending	Used, Not used (Default: Not used)	画像データが、フレームバッファの一番下のラインから一番上のラインに下降する場合に 「オン」を指定します。通常は [Off] です。
Input – Graphics screen1 input line repeat	On, Off (Default: Off)	LCD パネルのサイズよりも小さいラスター画像を繰り返し読み取ることが期待される場合は「オン」を指定します。通常は「オフ」です。詳細については、ラインリピート機能の説明を参照してください。
Input – Graphics screen1 input line repeat times	0	フレームに繰り返し読み込まれるラスター画像の繰り返し回数を指定します。
Input – Graphics screen1 layer coordinate X	0	グラフィックス画面の背景画面からの水平オフセットをピクセル単位で指定します。
Input – Graphics screen1 layer coordinate Y	0	グラフィックス画面の背景画面からの垂直オフセットをピクセル単位で指定します。
Input – Graphics screen1 layer background color alpha	255	アルファ値に基づいて、グラフィックス画面 2（前景グラフィックス画面）がグラフィックス画面 1（背景グラフィックス画面）にブレンドされるか、グラフィックス画面 1 がモノクロ背景画面にブレンドされます。
Input – Graphics screen1 layer background color Red	255	グラフィックス画面 N の背景色を指定します。
Input – Graphics screen1 layer background color Green	255	グラフィックス画面 N の背景色を指定します。
Input – Graphics screen1 layer background color Blue	255	グラフィックス画面 N の背景色を指定します。

ISDE Property	Value	説明
Input – Graphics screen1 layer fading control	None, Fade-in, Fade-out (Default: None)	グラフィックス画面のフェードインを行うには「オン」を指定します。透明の画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを行うには「オフ」を指定します。不透明の画面が徐々に透明に変化します。この処理は、GLCDC ハードウェアによってアクセラレートされ、いったん開始すると停止できないことに注意してください。遷移ステータスは、statusGet で監視できます。
Input – Graphics screen1 layer fade speed	0	フェード遷移が完了するフレーム数を指定します。
Input – Graphics screen2	Used, Not used (Default: Not used)	グラフィックス画面 N を使用する場合は「使用」を指定します。これにより、グラフィックス画面 1 用の「display_fb_background」と、グラフィックス画面 2 用の「display_fb_foreground」が、ISDE により自動生成されます。いずれのグラフィック画面も使用しない場合は、[Not used] を指定します。その場合、フレームバッファは作成されません。「未使用」を指定した場合、フレームバッファに対するメモリ読み取りアクセスがなく、バス帯域幅の消費量が減ることに注意してください。
Input – Graphics screen2 frame buffer name	fb_foreground	フレームバッファのカスタム名。
Input – Number of Graphics screen2 frame buffer	2	グラフィック画面 2 のフレームバッファ番号
Input – section where Graphics screen2 frame buffer allocated	sdram	フレームバッファを割り当てるセクション名を指定します。これは、[Input – Graphics screen1] が [Used] に設定されている場合に有効です。
Input – Graphics screen2 input horizontal size	800	幅のピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input – Graphics screen2 vertical size	480	高さのピクセル数を指定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。

ISDE Property	Value	説明
Input – Graphics screen2 input horizontal stride (not bytes but pixels)	800	水平ラインのメモリストライドを指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。一般に、このパラメータはパラメータ [input horizontal size] と同じ値に設定します。デフォルト値は、800x480 ピクセルの画像のサイズになります。
Input – Graphics screen2 input format	32 bits ARGB888, 32 bits RGB888, 16 bits RGB565, 16 bits ARGB1555, 16 bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 (Default: 16 bits RGB565)	グラフィックス画面の入力形式を指定します。CLUT フォーマットを選択する場合は、clut を使用して CLUT データを書き込んでから start を実行する必要があります。デフォルト設定では、RGB565 形式の画像がサポートされます。
Input – Graphics screen2 input line descending	On, Off (Default: Off)	画像データが、フレームバッファの一番下のラインから一番上のラインに下降する場合に「オン」を指定します。通常は [Off] です。
Input – Graphics screen2 input line repeat	On, Off (Default: Off)	LCD パネルのサイズより小さいラスタ画像を繰り返し読み取ることが予想される場合は、[On] を指定します。通常は [Off] です。詳細については、ラインリピート機能の説明を参照してください。
Input – Graphics screen2 input line repeat times	0	フレームに繰り返し読み込まれるラスター画像の繰り返し回数を指定します。
Input – Graphics screen2 layer coordinate X	0	グラフィックス画面の背景画面からの水平オフセットをピクセル単位で指定します。
Input – Graphics screen2 layer coordinate Y	0	グラフィックス画面の背景画面からの垂直オフセットをピクセル単位で指定します。
Input – Graphics screen2 layer background color alpha	255	アルファ値に基づいて、グラフィックス画面 2（前景グラフィックス画面）がグラフィックス画面 1（背景グラフィックス画面）にブレンドされるか、グラフィックス画面 1 がモノクロ背景画面にブレンドされます。

ISDE Property	Value	説明
Input – Graphics screen2 layer background color Red	255	グラフィックス画面 N の背景色を指定します。
Input – Graphics screen2 layer background color Green	255	グラフィックス画面 N の背景色を指定します。
Input – Graphics screen2 layer background color Blue	255	グラフィックス画面 N の背景色を指定します。
Input – Graphics screen2 layer fading control	None, Fade-in, Fade-out (Default: None)	グラフィックス画面のフェードインを行うには「オン」を指定します。透明の画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを行うには「オフ」を指定します。不透明の画面が徐々に透明に変化します。この処理は、GLCDC ハードウェアによってアクセラレートされ、いったん開始すると停止できないことに注意してください。遷移ステータスは、statusGet で監視できます。
Input – Graphics screen2 layer fade speed	0	フェード遷移が完了するフレーム数を指定します。
Output – Horizontal total cycles	1024	水平ラインの合計サイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Horizontal active video cycles	800	水平ラインのアクティブビデオサイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Horizontal back porch cycles	46	水平ラインのバックポーチサイクル数を指定します。バックポーチは、Hsync サイクルの始点から開始します。つまり、バックポーチサイクルには Hsync サイクルが含まれます。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。

ISDE Property	Value	説明
Output – Horizontal sync signal cycles	20	Hsync 信号アサーションサイクル数を指定します。システムの LCD パネルシートのデータシートに定義されているサイクル数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Horizontal sync signal polarity	Low active, High active (Default: Low active)	システムに合わせて、Hsync 信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Vertical total lines	525	1 フレームの合計ライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Vertical active video lines	480	1 フレームのアクティブビデオライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Vertical back porch lines	23	1 フレームのバックポーチライン数を指定します。バックポーチは、Vsync ラインの始点から開始します。つまり、バックポーチラインには Vsync ラインが含まれます。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Vertical sync signal lines	10	1 フレームの Vsync 信号アサーションライン数を指定します。システムの LCD パネルシートのデータシートに定義されているライン数を設定します。デフォルト値は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。

ISDE Property	Value	説明
Output – Vertical sync signal polarity	Low active, High active (Default: Low active)	システムに合わせて、Vsync 信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Format	24bits RGB888, 18bits RGB666, 16 bits RGB565, 8bits serial (Default: 24bits RGB888)	LCD パネルに合ったグラフィックス画面の出力形式を指定します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Endian	Little endian, Big endian (Default: Little endian)	LCD パネルへの出力信号のデータエンディアンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Color order	RGB, BGR (Default: RGB)	LCD パネルへの出力信号のデータオーダーを選択します。赤と青の順序は、必要に応じて入れ替えることができます。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Data Enable Signal Polarity	Low active, High active (Default: High active)	システムに合わせて、データ有効信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Sync edge	Rising Edge, Falling Edge (Default: Rising Edge)	システムに合わせて、同期信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネルに一致します。
Output – Background color alpha channel	255	背景画面の背景色を指定します。
Output – Background color R channel	0	背景画面の背景色を指定します。
Output – Background color G channel	0	背景画面の背景色を指定します。
Output – Background color B channel	0	背景画面の背景色を指定します。

ISDE Property	Value	説明
CLUT	Used, Not used (Default: Not used)	グラフィックス画面の入力形式に CLUT 形式を選択する場合は、「使用」を指定します。その場合、CLUT ソースデータ用の「CLUT_buffer」という名前のバッファが、ISDE で自動生成されるソースファイル中に生成されます。
CLUT - CLUT buffer size	256	CLUT ソースデータバッファのエントリ数を指定します。各エントリは 4 バイト（1 ワード）を消費します。このパラメータで指定された CLUT ソースデータのワードは、ISDE で自動生成されるソースファイル中に生成されます。
TCON – Hsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 (Default: LCD_TCON0)	システムに合わせて、Hsync 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON – Vsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 (Default: LCD_TCON1)	システムに合わせて、Vsync 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON – DataEnable pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 (Default: LCD_TCON2)	システムに合わせて、DataEnable 信号に使用する TCON ピンを選択します。デフォルト設定は、S7G2 PE-HMI1 ボード上の LCD パネル用です。
TCON – Panel clock division ratio	8-Jan	クロックソースの除算値を選択します。ピクセルクロックのソースクロックについては、この表の下部にある注記を参照してください。
Color correction – Brightness	Off, On (Default: Off)	明るさ制御を行う場合は「オン」を指定します。「オフ」を指定した場合、以下の設定は出力色に影響を与えません。
Color correction – Brightness R channel	512	出力色レベルは次のように計算されます。出力色レベル = 入力色レベル +/-512。R、G、B チャネルのそれぞれの値を設定します。

ISDE Property	Value	説明
Color correction – Brightness G channel	512	出力色レベルは次のように計算されます。出力色レベル = 入力色レベル +/-512。R、G、B チャネルのそれぞれの値を設定します。
Color correction – Brightness B channel	512	出力色レベルは次のように計算されます。出力色レベル = 入力色レベル +/-512。R、G、B チャネルのそれぞれの値を設定します。
Color correction – Contrast	Off	コントラスト制御を行う場合は 「オン」を指定します。「オフ」を指定した場合、以下の設定は出力色に影響を与えません。
Color correction – Contrast(gain) R channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction – Contrast(gain) G channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction – Contrast(gain) B channel	128	出力色レベルは次のように計算されます：出力色レベル = 入力色レベル x (/128)。R、G、B チャネルのそれぞれの値を設定します。
Color correction – Gamma correction(Red)	Off, On (Default: Off)	各チャネルの R/G/B の制御。赤チャネルのガンマ補正を行う場合は 「オン」を指定します。 [Off] を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。
Color correction – Gamma gain R[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]）<<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます。出力色レベル = 入力色レベル /1024 (/128)。

ISDE Property	Value	説明
Color correction – Gamma threshold R[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます。出力色レベル = 入力色レベル /1024 (/128)。
Color correction – Gamma correction(Green)	Off	各チャネルの R/G/B の制御。赤チャネルのガンマ補正を行う場合は 「オン」を指定します。 [Off] を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。
Color correction – Gamma gain G[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]） <<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます。出力色レベル = 入力色レベル /1024 (/128)。
Color correction – Gamma threshold G[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます。出力色レベル = 入力色レベル /1024 (/128)。
Color correction – Gamma correction(Blue)	Off, On (Default: Off)	各チャネルの R/G/B の制御。赤チャネルのガンマ補正を行う場合は 「オン」を指定します。「オフ」を指定した場合、ゲインとしきい値の設定は出力色に影響を与えません。

ISDE Property	Value	説明
Color correction – Gamma gain B[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのゲイン値を設定します。エリア N のゲイン設定は、色レベルが（（ガンマしきい値 R[N-1]） <<2）と（（ガンマしきい値 R[N]）<<2）の間にある入力データに適用されます。出力値は次のように計算されます：出力色レベル = 入力色レベル / 1024 (/128)。
Color correction – Gamma threshold B[0-15]	0	ガンマ補正カーブ上のエリア N 内の赤チャネルのしきい値を設定します。エリア N のゲイン設定は、色レベルがガンマしきい値 R[N-1] とガンマしきい値 R[N] の間にある入力データに適用されます。出力値は次のように計算されます。出力色レベル = 入力色レベル /1024 (/128)。
Dithering	Off, On (Default: Off)	ディザリング有効。出力形式 RGB666 または RGB565 を選択する場合に、ディザ効果を適用しバンディングを減らす場合は 「オン」を指定します。ディザリングは変換時に適用できます。「オフ」を指定した場合、以下のディザリング設定は出力に影響を与えません。ディザ効果の詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering – Mode	Truncate, Round off, 2x2 Pattern (Default: Truncate)	ディザモードを指定します。詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering – Pattern A	Pattern 00, Pattern 01, Pattern 10, Pattern 11 (Default: Pattern 11)	2X2 パターンモードのディザパターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering – Pattern B	Pattern 00, Pattern 01, Pattern 10, Pattern 11 (Default: Pattern 11)	2X2 パターンモードのディザパターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロックパネルディザ補正レジスタ (OUT_PDTHA) を参照してください。

ISDE Property	Value	説明
Dithering – Pattern C	Pattern 00, Pattern 01, Pattern 10, Pattern 11 (Default: Pattern 11)	2X2 パターンモードのディザパターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロック パネルディザ補正レジスタ (OUT_PDTHA) を参照してください。
Dithering – Pattern D	Pattern 00, Pattern 01, Pattern 10, Pattern 11 (Default: Pattern 11)	2X2 パターンモードのディザパターンを指定します。詳細については、ハードウェアマニュアルの出力制御ブロック パネルディザ補正レジスタ (OUT_PDTHA) を参照してください。
Misc – Correction Process Order	Brightness and Contrast then Gamma, Gamma then Brightness and Contrast (Default: Brightness and Contrast then Gamma)	必要に応じて色補正処理の順序を指定します。
Line Detect Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Disabled)	ライン検出割り込み優先順位の選択。
Underflow 1 Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Disabled)	アンダーフロー 1 割り込み優先順位の選択。
Underflow 2 Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Disabled)	アンダーフロー 2 割り込み優先順位の選択。

***注:** 例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトからのものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

GLCDC HAL モジュールのクロック構成

GLCDC モジュールは、以下のいずれかのクロックソースからピクセルクロックを生成できます。ソースクロックの選択は、 e^2 studio の [Synergy Configuration] で行うことができます。

- 内部クロックソース (PLLOUT; 240 MHz)
- LCD_EXTCLK ピンからの外部クロックソース

***注:** S7G2 WS1 （Working Sample1）チップと WS2 （Working Sample2）チップ以降では、内部クロックが異なります。WS1 チップは PCLKB（最大 60MHz）を使用しますが、WS2 以降のチップは PLLOUT（最大 240MHz）を使用します。*

GLCDC HAL モジュールのピン構成

GLCDC モジュールは、 MCU のピンを使用して外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。ピン選択の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、構成設定の表では GLCDC ピンを選択する例を示します。

GLCDC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
GLCDC	Pins	Select Peripherals > Graphics: GLCDC> GLCDC0

注: *選択シーケンスでは、GLCDC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

GLCDC HAL モジュールのピン構成設定

Property	Value	説明
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	ピングループの選択
Operation Mode	Disabled, Custom, RGB888, RGB666, RGB565 (Default: Disabled)	必要な動作モードを選択します
LCD_CLK	None, P900, P101 (Default: None)	LCD_CLK ピン
LCD_DATA00:15	None, Pn, Pm (Default: None)	LCD_DATA ピン
LCD_TCON0:3	None, Pn, Pm (Default: None)	LCD_TCON ピン
LCD_EXTCLK	None, Pn, Pm (Default: None)	LCD_EXTCLK ピン

注: *表で示されている例の値は、Synergy S7G2 キットおよび SK-S7G2 キットを使用するプロジェクトからのものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

注: *S7G2 PE-HMI1 ボード上で GLCDC モジュールを使用するには、PORT10 ピン 3（PA03）とピン 5（PA05）を、出力レベル HIGH の IOPORT ピンとして設定してください。ピン PA03 は DISP 信号（ディスプレイオン / オフ）を制御し、ピン PA05 は LCD パネルのバックライトを制御します。詳細については、S7G2 PE-HMI1 ボードの回路図を参照してください。*

5.2.11.5 アプリケーションでの GLCD HAL モジュールの使用

アプリケーションで GLCDC HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、GLCDC HAL モジュールを初期化します。
2) アプリケーションのコードで、フレームバッファにプライマリ画像を描画します。
3) start API を使用して、画像の表示を開始します。
4) アプリケーションのコードで、フレームバッファに新しい画像を描画して表示を更新します。通常、ユーザーのシステムはピンポンフレームバッファシステムで構成されているので、描画時に表示に使用されていない別のフレームバッファに画像を描画します。

5) layerChange API を使用して、フレームバッファの切り替えを GLCDC ハードウェアに要求します。
6) GLCDC ハードウェアは、フレームバッファを切り替えて、次のフレームから新しい画像を表示します。

アプリケーションのコードを現在のフレームバッファの描画完了と同期させるには、ライン検出割り込みを使用し、GLCDC コールバックを通してアプリケーションのコードにタイミングを通知します。

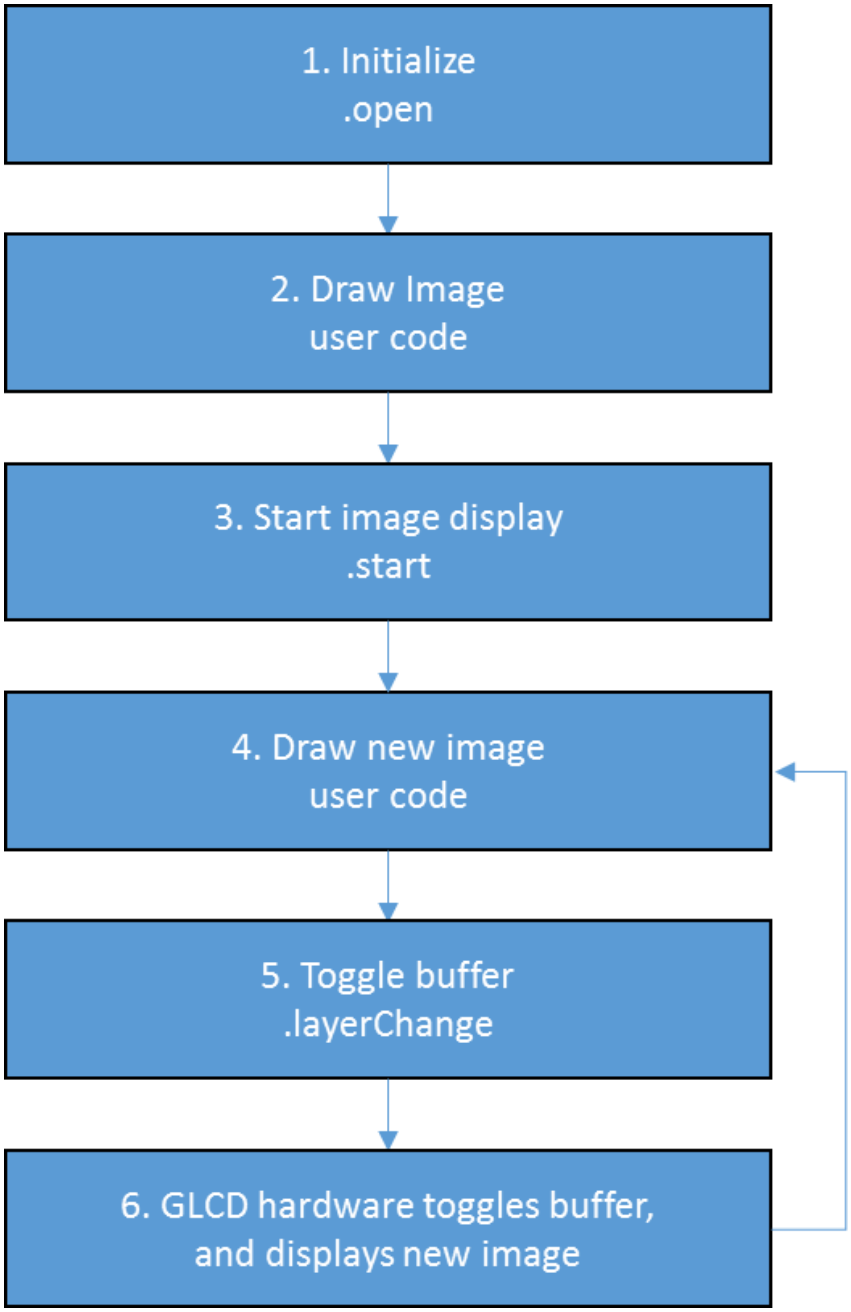


図 345: 通常の GLCDC HAL モジュールアプリケーションのフロー図

5.2.12 データ操作回路ドライバー

DOC HAL モジュールは 16 ビットデータを比較するために使用され、以下のイベントを検出できます。

- データ値の不一致または一致
- 加算演算のオーバフロー
- 減算演算のアンダーフロー

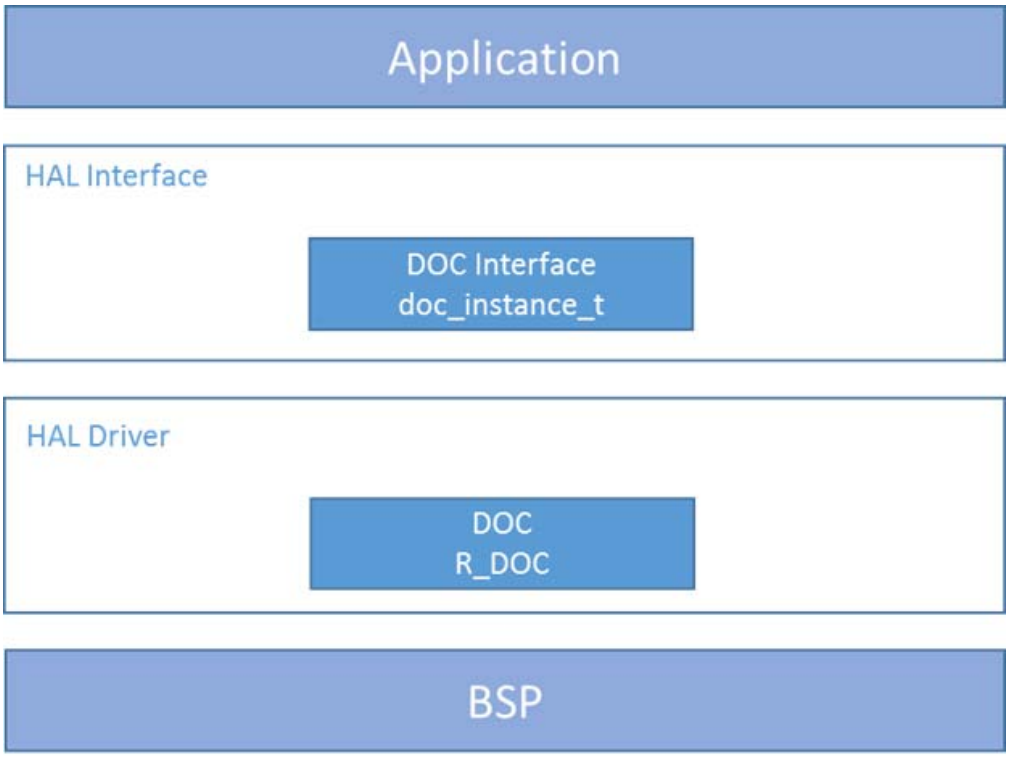


図 346: DOC HAL モジュールのブロック図

5.2.12.1 DOC HAL モジュールの API の概要

DOC HAL モジュールでは、データ演算回路のオープン、クローズ、状態の確認、データのライトのための API が定義されています。DOC HAL モジュールは、Synergy MCU 上の DOC を使用します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

DOC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_doc.p_api->open(g_doc.p_ctrl, g_doc.p_cfg) 初期構成。
close	g_doc.p_api->close(g_doc.p_ctrl) ドライバーを再構成できます。電力消費を低減できます。

Function Name	API の呼び出し例と説明
statusGet	g_doc.p_api->statusGet(g_doc.p_ctrl, &my_Status) DOC の状態を取得し、指定されたポインタp_status.に格納します。
statusClear	g_doc.p_api->statusClear(g_doc.p_ctrl) DOPCF の状態フラグをクリアします。
write	g_doc.p_api->write(g_doc.p_ctrl, &value) DODIRレジスタとDODSR レジスタに書き込みます。
inputRegisterWrite	g_doc.p_api->inputRegisterWrite(g_doc.p_ctrl, doc_values) DODIR レジスタに書き込みます。
versionGet	g_doc.p_api->versionGet(g_doc.p_ctrl, &version) バージョンポインタを使用して API のバージョンを取得します。

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	DOC が正常に構成されました。
SSP_ERR_IN_USE	モジュールはすでに開いています。
SSP_ERR_ASSERTION	1 つまたはそれ以上のポインタが NULL をポイントしています。
SSP_ERR_INVALID_ARGUMENT	ISR が有効ではありません。 bsp_irq_cfg.h で ISR を有効にします。
SSP_ERR_HW_LOCKED	DOC リソースがロックされています。
SSP_ERR_NOT_OPEN	ドライバーが開かれていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.12.2 DOC HAL モジュールの動作の概要

DOC HAL モジュールは、Synergy MCU 上の DOC を制御します。16 ビットデータの比較に使用され、データ値間の不一致、加算値のオーバーフロー、減算演算のアンダーフローを検出できます。コールバックを使用

でき、関連する割り込みが有効になっている場合は、DOC イベントに対応してコールバック関数が呼び出されます。

DOC は、2 つのデータレジスタを使用して演算を実行します。DOC データ入力レジスタ（DOCDIR）は演算対象のデータを保持し、DOC データ設定レジスタ（DOCDSR）は入力データに対する演算に使用される値を保持します。加算モードと減算モードでは、このレジスタはデータ演算の結果を格納します。（これらのレジスタはどちらも 16 ビット幅です）。

DOC HAL モジュールの動作に関する重要な注意事項と制限事項

比較データの初期設定は、write API を呼び出すことで DOC に書き込まれます。write API は、DOC の DODSR レジスタと DODIR レジスタに書き込みます。 write API は、 doc_data_t 型の変数を使用します。

```
doc_data_t g_doc_values;

   g_doc_values.dodir = 0x1000;

   g_doc_values.dodsr = 0x1000;

   g_doc.p_api->write(g_doc.p_ctrl, &g_doc_values)
```

比較対象のデータが変化しない場合は、比較が必要になるたびにそのデータを書き込み直す必要はありません。入力データの値は、inputRegisterWrite API を使用して DOC に書き込むことができます。inputRegisterWrite API は、 DOC のデータ入力レジスタのみに書き込みます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.12.3 アプリケーションへの DOC HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに DOC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

DOC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（DOC ドライバーのデフォルト名は g_doc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

DOC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_doc0 Data Operation Circuit Driver on r_doc	Threads	New Stack > Driver > Monitoring > Data Operation Circuit Driver on r_doc

次の図に示すように、r_doc の DOC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。

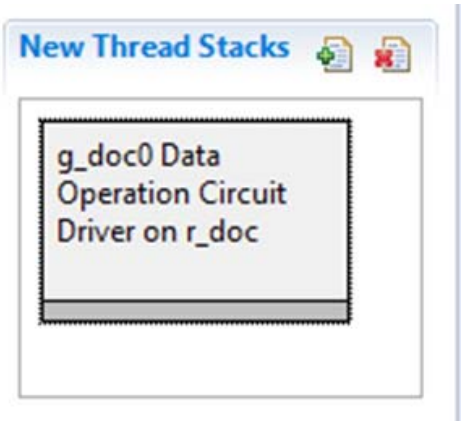


図 347: DOC HAL モジュールのスタック

5.2.12.4 DOC HAL モジュールの構成

ユーザーは必要な動作に合わせて DOC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性の表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_doc での DOC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_doc0	モジュール名。
Event	Comparison mismatch, Comparison match, Addition overflow, Subtraction underflow (Default: Comparison mismatch)	DOC割り込みをトリガーするイベントを指定します。
Callback	NULL	ユーザーのコールバック関数をここで定義できます。このコールバック関数が指定されている場合、設定されたDOCイベントが発生すると、この関数が割り込みサービスルーチン(ISR)から呼び出されます。 ***注:*** *コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
DOC Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	プルダウンを使用してDOC割り込み優先順位を設定します。

注: *設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

デフォルト以外の設定が望ましい場合もあります。たとえば、望ましい動作に応じて異なるイベントを選択するときに役立つ場合があります。

DOC HAL モジュールのクロック構成

DOC HAL モジュールには、特定のクロック構成は必要ありません。

DOC HAL モジュールのピン構成

DOC HAL モジュールには、特定のピン構成は必要ありません。

5.2.12.5 アプリケーションでの DOC HAL モジュールの使用

アプリケーションで DOC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 DOC を初期化します。
2) write API を使用して、 DODIR と DODSR にレジスタ値を設定します。
3) inputRegisterWrite API を使用して、 DOC にデータをストリームします。
4) statusGet API またはコールバック （有効な場合）を使用して、比較の状態を読み取ります。
5) statusClear API を使用して、状態フラグをクリアします。
6) close API を使用して、モジュールを閉じます。

次の図は、以上の一般的な手順を示した通常の動作フロー図です。

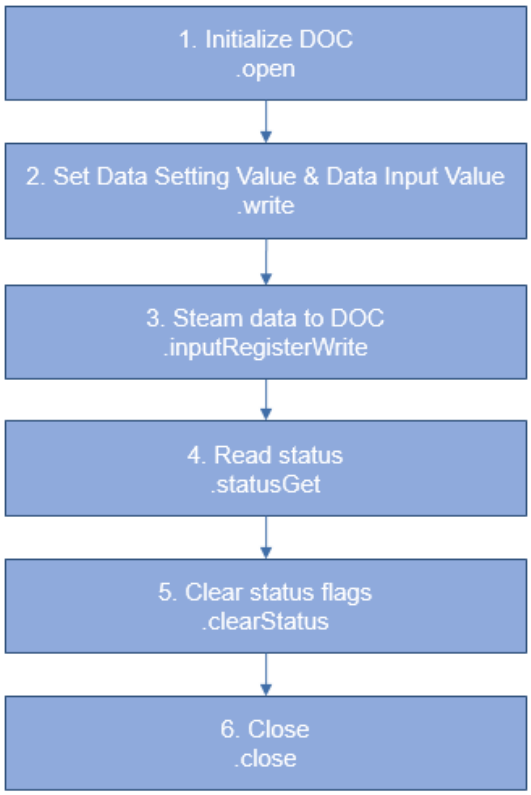


図 348: 一般的な DOC HAL モジュールアプリケーションのフロー図

5.2.13 DMAC ドライバー

ダイレクトメモリアクセスコントローラ（DMAC）HAL モジュールは、データ転送アプリケーションのための高レベル API であり、r_dmac. に実装されています。DMAC HAL モジュールは、Synergy MCU 上の DMAC

ペリフェラルを使用します。ユーザー定義のコールバックを作成し、転送イベント発生時に CPU に通知できます。

5.2.13.1 DMAC HAL モジュールの機能

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。 DMAC HAL モジュールは、以下の機能をサポートしています。

- Synergy MCU での DMAC モジュール
- 割り込み （必要な場合）
- 複数の転送モード
  - 単一転送
  - リピート転送
  - ブロック転送
  - アドレスインクリメントモードまたは固定モード
- 複数チャネル （数は使用されている MCU に依存）

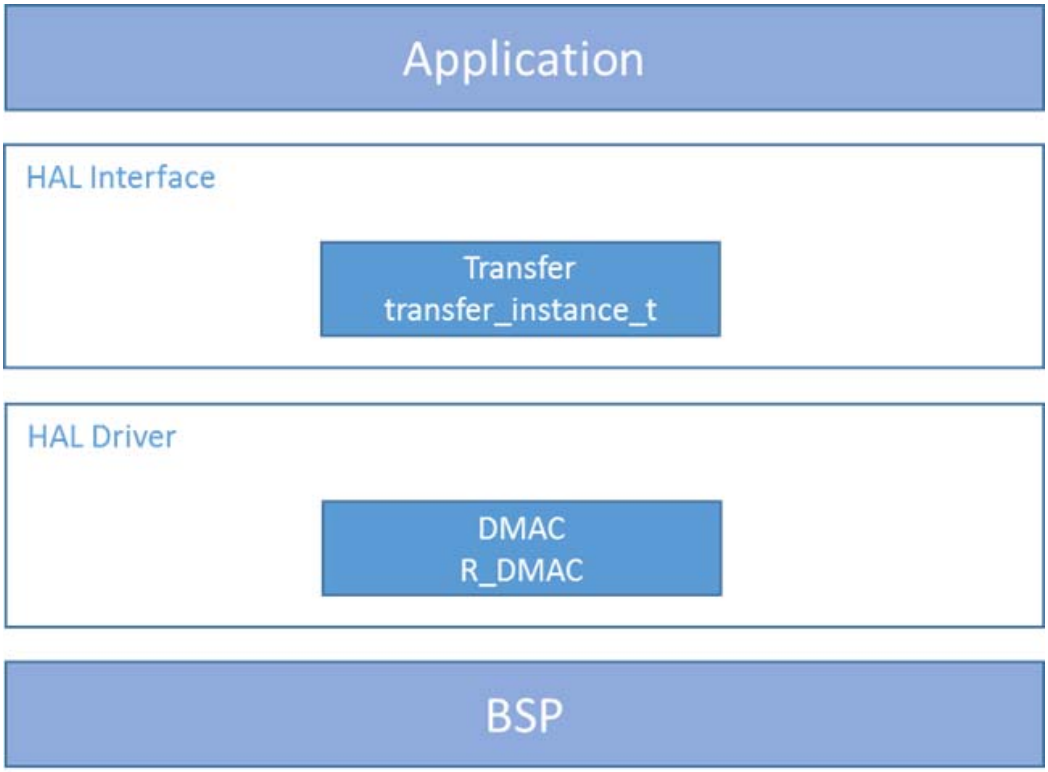


図 349: DMAC HAL モジュールのブロック図

5.2.13.2 DMAC HAL モジュールの API の概要

DMAC HAL モジュールでは、オープン、クローズ、タイマの開始、タイマの停止のための API が定義されています。データトランスファコントローラ（DTC）と DMAC は同じ転送インタフェースを使用することに注意してください。インタフェースを共有することで、DTC と DMA の実装の変更が容易になります。 API の呼び出しは同じであり、下位レベルの実装から独立しています。使用可能なすべての API のリスト、 API コールの例、各関数の簡単な説明が、次の API の要約の表に含まれます。ステータス戻り値の表は API の要約の後にあります。

DMAC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_transfer0.api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg) デバイスチャネルを開きます。最初の呼び出しで、ドライバーとハードウェアを初期化します。
close	g_transfer0.api->close(g_transfer0.p_ctrl) デバイスチャネルを閉じます。開いている最後のチャネルである場合は、ハードウェアをオフにします。
reset	g_transfer0.api->reset(g_transfer0.p_ctrl, &source, &destination, number_of_transfers) チャネルの設定をリセットします。
start	g_transfer0.api->start(g_transfer0.p_ctrl, mode) データの転送を開始します。
stop	g_transfer0.api->stop(g_transfer0.p_ctrl) データの転送を停止します。
enable	g_transfer0.api->enable(g_transfer0.p_ctrl) チャネルを有効にします。

Function Name	API の呼び出し例と説明
disable	g_transfer0.api->disable(g_transfer0.p_ctrl) チャネルを無効にします。
versionGet	g_transfer0.api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。
infoGet	g_transfer0.api->infoGet(g_transfer0.p_ctrl, &info) 転送チャネルの情報を取得します。
blockReset	g_transfer0.api->blockReset(g_transfer0.p_ctrl, &source, &destination, length, size, number_of_transfers) ブロック転送のパラメータをリセットします。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。
SSP_ERR_NOT_OPEN	チャネルは開かれていません。
SSP_ERR_UNSUPPORTED	動作が正しく構成されていません。
SSP_ERR_IN_USE	指定したチャネルはすでに開かれています。構成は変更されていません。関連する Close 関数を呼び出すか、関連する Control コマンドを使用してチャネルを再構成します。
SSP_ERR_IRQ_BSP_DISABLED	IRQ が BSP で有効になっていません。
SSP_ERR_NOT_ENABLED	操作が失敗しました。

Name	説明
SSP_ERR_NOT_OPEN	チャネルは開かれていません。

*注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル *』で関連モジュールの API リファレンスを参照してください。*

5.2.13.3 DMAC HAL モジュールの動作の概要

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。DMAC HAL モジュールは DMAC 周辺レジスタを使用するため、システムでの転送回数はデバイス上の DMAC チャネルの数に制限されます。 DMAC を使用するためにアクティベーションソースを有効にする必要はありません。DMAC 転送が完了すると、DMAC 割り込みが呼び出されます。アクティベーションソース割り込みが有効になっている場合は、転送がトリガーされるのと同時に発生します。DMAC 割り込みが有効にされている場合は、すべての転送が完了した後で発生します。たとえば、長さが 16 である通常モード転送がタイマによってトリガされる場合、タイマ割り込みは各転送が発生するのと同時に発生し、DMAC 割り込みは 16 番目の転送が完了した後で発生します。DMAC ではチェーンされた転送はサポートされません。

DMAC HAL モジュールの動作に関する重要な注意事項と制限事項

DMAC HAL モジュールの動作に関する注意事項

ノーマルモード

通常モードでは、単一転送はアクティベーションソースイベントが発生するたびにトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、転送は完了です。

リピートモード

リピートモードでは、アクティベーションソースイベントが発生するたびに単一転送がトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピートエリアが転送元に設定されている場合、転送が再度開始するときに転送元レジスタにも初期値がリロードされます。また、リピートエリアが転送先に設定されている場合は、転送が再度開始するときに転送先レジスタに初期値がリロードされます。

ブロックモード

ブロックモードでは、アクティベーションソースイベントが発生するたびに転送長全体が転送されます。たとえば、アクティベーションソースがタイマ、バイトサイズが 2、バイト長が 12 である転送がブロックモードで構成されている場合、アクティベーションイベントが発生するたびに 24 バイトが転送されます。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピートエリアがソースに設定されている場合、転送の再開始時にソースレジスタにも初期値がリロードされます。また、リピートエリアが転送先に設定されている場合は、転送が再度開始するときに転送先レジスタに初期値がリロードされます。

アドレスモード

サイズ（1 バイト、2 バイト、または 4 バイト）が転送されるたびに、転送元ポインタは src_addr_mode で、転送先インタは dest_addr_mode, で、それぞれ調整されます。

たとえば、 src_addr_mode が

TRANSFER_ADDR_MODE_INCREMENTED, に設定され、サイズが TRANSFER_SIZE_4_BYTES, に設定されている場合、 p_dest ポインタは転送が終わるたびに 4（転送サイズ）だけインクリメントされます。

TRANSFER_ADDR_MODE_FIXED に設定されている場合、ポインタは変化しません。

DMAC HAL モジュールの制限事項

このモジュールの動作に関するその他の制限事項については、最新の 『*SSP リリースノート*』を参照してください。

5.2.13.4 アプリケーションへの DMAC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して DMAC ドライバーをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の「開発の開始」の章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

DMAC ドライバーをアプリケーションに追加するには、次の表に示す**スタック選択シーケンス**を使用してスレッドに単純に追加します。（DMAC ドライバーのデフォルト名は g_transfer0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

表 3 GPT の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ transfer0 Transfer Driver on r_dmac	Threads	New Stack > Driver > Transfer> Transfer Driver on r_dmac

次の図に示すように、 r_dmac の DMAC ドライバーが**新しいスレッドスタック**に追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

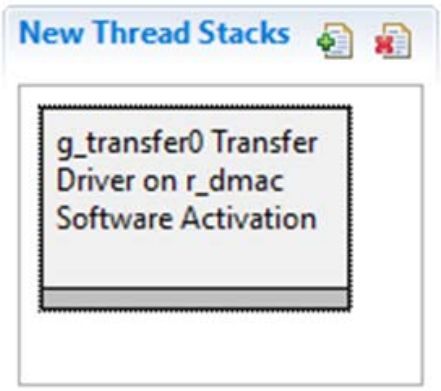


図 350: DMAC HAL モジュールのスタック

5.2.13.5 DMAC HAL モジュールの構成

ユーザーは必要な動作に合わせて DMAC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータの [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位は、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性を示していることに注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位レベルを構成するときに ISDE で簡単に確認できます。

***注：** プロパティ設定と並行して次に示す構成テーブル設定を確認してください。表の構成設定は、正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_dmac での DMAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer0	モジュール名。

ISDE Property	Value	説明
Channel	0	
Mode	Block	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

| Interrupt Priority | Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled) | ||

注: *上記の設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

DMAC HAL モジュールのクロック構成

DMAC 周辺モジュールは、ICLK をそのクロックソースとして使用します。ICLK の周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

DMAC HAL モジュールのピン構成

DMAC HAL モジュールはいずれのピンとも関連付けられていません。

5.2.13.6 アプリケーションでの DMAC HAL モジュールの使用

アプリケーションで DMAC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 DMAC HAL モジュールを初期化します。
2) enable API を使用して、 DMAC HAL モジュールを有効にします （自動的に有効化されない場合）。
3) 必要に応じて、他の API を使用して転送を管理します。
4) 必要な場合は、 DMAC HAL モジュールを閉じます。

次の動作フロー図では、 DMAC ドライバーを使用する一般的な手順を示します。

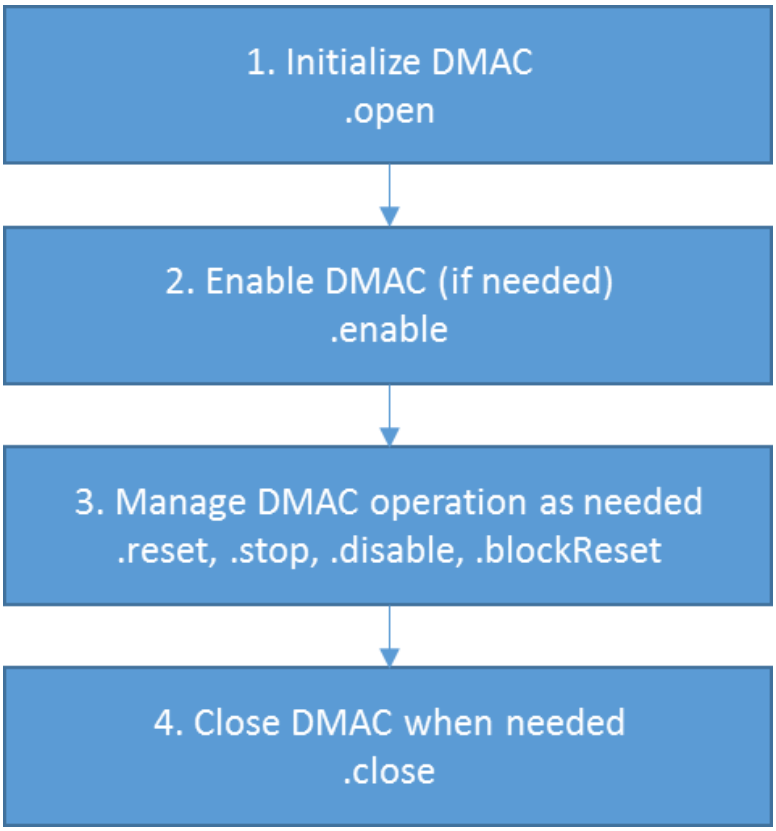


図 351: 一般的な DMAC HAL モジュールアプリケーションのフロー図

5.2.14 DTC ドライバー

データトランスファコントローラ（DTC）HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の転送元からユーザー指定の転送先にデータを移動します。

- Synergy MCU の DTC モジュールをサポートします
- 必要に応じて、割り込みをサポートします

- 複数の転送モードをサポートします
  - 単一転送
  - リピート転送
  - ブロック転送
  - アドレスインクリメントモードまたは固定モード
  - チェーン転送
- 複数のチャネルをサポートします（選択されている実装に依存）
  - チャネルの数は、 RAM ベースのベクタテーブルのサイズによってのみ制限されます

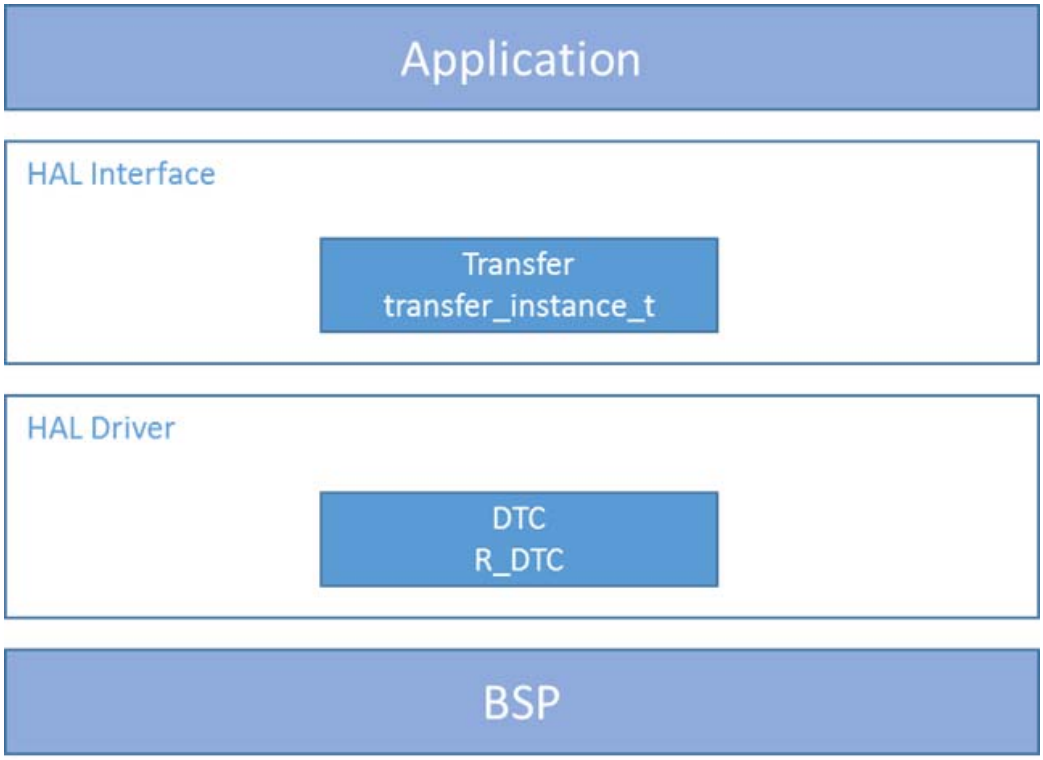


図 352: DTC HAL モジュールのブロック図

5.2.14.1 DTC HAL モジュールの API の概要

DTC HAL モジュールでは、オープン、クローズ、リセット、有効化、無効化、開始、停止のための API が定義されています。 DTC と DMAC では、 DTC と DMA の実装の変更を容易にするため、同じ転送インタフェースが使用されていることに注意してください。 API の呼び出しは同じであり、下位レベルの実装から独立しています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

DTC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_transfer0.api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg) 初期設定。auto_enable が true で、p_src、p_dest、length がすべて有効な場合は、転送が有効になります。 enable または reset を使用して転送を有効にすることもできます。
close	g_transfer0.api->close(g_transfer0.p_ctrl) デバイスチャネルを閉じます。開いている最後のチャネルである場合は、ハードウェアをオフにします。
reset	g_transfer0.api->reset(g_transfer0.p_ctrl, &source, &destination, number_of_transfers) 他のすべての設定値を維持したまま、ソースアドレスポインタ、宛先アドレスポインタ、長さを再設定できます。p_src、p_dest、length がすべて有効な場合は、転送が有効になります。
.start	g_transfer0.api->start(g_transfer0.p_ctrl, mode) 転送をソフトウェア内で開始します。
stop	g_transfer0.api->stop(g_transfer0.p_ctrl) 転送をソフトウェア内で停止します。現在実行中の転送が完了した後、後続の転送が停止されます。
enable	g_transfer0.api->enable(g_transfer0.p_ctrl) 転送を有効にします。転送は、アクティベーションソースイベントの後で （または、アクティベーションソースとして ELC_EVENT_ELC_SOFTWARE_EVENT_0 または ELC_EVENT_ELC_SOFTWARE_EVENT_0 が選択されている場合は、 start が呼び出されたときに）行われます。

Function Name	API の呼び出し例と説明
disable	g_transfer0.api->disable(g_transfer0.p_ctrl) 転送を無効にします。転送は、 transfer_info_t::activation ソースイベントの後では （または、 transfer_info_t::activation_source). として ELC_EVENT_ELC_SOFTWARE_EVENT_0 または ELC_EVENT_ELC_SOFTWARE_EVENT_0 が選択されている場合、 start が呼び出されたときは）行われません。
versionGet	g_transfer0.api->versionGet(&version) バージョンを取得して、指定されたポインタ version に格納します。
infoGet	g_transfer0.api->infoGet(g_transfer0.p_ctrl, &info) この転送に関する情報を提供します。
blockReset	g_transfer0.api->blockReset(g_transfer0.p_ctrl, &source, &destination, length, size, number_of_transfers) ブロック転送で他のすべての設定値を維持したまま、ソースアドレスポインタ、宛先アドレスポインタ、長さを再設定できます。 p_src、 p_dest、 length がすべて有効な場合は、転送が有効になります。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。
SSP_ERR_NOT_OPEN	チャネルは開かれていません。
SSP_ERR_UNSUPPORTED	動作が正しく構成されていません。

Name	説明
SSP_ERR_IN_USE	指定したチャネルはすでに開かれています。構成は変更されていません。関連する Close 関数を呼び出すか、関連する Control コマンドを使用してチャネルを再構成します。
SSP_ERR_HW_LOCKED	DTC ハードウェアリソースがロックされています。
SSP_ERR_IRQ_BSP_DISABLED	IRQ が BSP で有効になっていません。
SSP_ERR_NOT_ENABLED	操作が失敗しました。
SSP_ERR_NOT_OPEN	チャネルは開かれていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.14.2 DTC HAL モジュールの動作の概要

Synergy MCU 内でデータを移動するには、ダイレクトメモリアクセスコントローラ（DMAC）とデータトランスファコントローラ（DTC）を使用できます。どちらの実装を使用するか選択するときは、考慮事項がいくつかあります。以下の動作の概要には、アプリケーションに最適な実装を決定するのに役立つ情報が含まれます。ほとんどの汎用転送アプリケーションでは DTC モジュールが推奨されますが、基本的な転送機能にはどちらのモジュールでも使用できます。各転送モジュールのユースケースは後で示します。

DTC HAL モジュールの選択

DTC HAL モジュールでは、システム内のすべての割り込みに対するスロットを持つ、RAM ベースのベクタテーブルが使用されます。DTC 転送が完了すると、アクティベーションソース割り込みが呼び出されます。DTC を使用するには、アクティベーションソース割り込みを有効にする必要があります。一般に、アクティベーションソース割り込みは、TRANSFER_IRQ_EACH が構成で指定されていないかぎり、転送が完了するまで DTC によってミュートされます。たとえば、長さが 16 である通常モード転送がタイマによってトリガされた場合、転送が有効である間、最初の 15 回のタイマ割り込みは発生しません。タイマ割り込みは 16 番目の転送の後で発生します。DTC ではチェーンされた転送も許可されます。つまり、単一のアクティベーションソース割り込みの後で複数の転送を実行できます。この機能はドライバーによってサポートされていますが、ISDE の外部で構成する必要があります。

DMAC HAL モジュールの選択

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。DMAC HAL モジュールは DMAC 周辺レジスタを使用するため、システムでの転送回数はデバイス上の DMAC チャネルの数に制限されます。DMAC を使用するためにアクティベーションソースを有効にする必要はありません。DMAC 転送が完了すると、DMAC 割り込みが呼び出されます。アクティベーションソース割り込みが有効になっている場合は、転送がトリガーされるのと同時に発生します。DMAC 割り込みが有効にされている場合は、すべての転送が完了した後で発生します。たとえば、長さが 16 である通常モード転送がタイマによってトリガされる場合、タイマ割り込みは各転送が発生するのと同時に発生し、DMAC 割り込みは 16 番目の転送が完了した後で発生します。DMAC HAL モジュールでは、チェーンされた転送はサポートされません。

DTC HAL モジュールの動作に関する重要な注意事項と制限事項

ノーマルモード

通常モードでは、アクティベーションソースイベントが発生するたびに単一の転送がトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送の長さが1だけデクリメントされます。転送の長さが0に達すると、転送は完了です。

リピートモード

リピートモードでは、アクティベーションソースイベントが発生するたびに単一の転送がトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピートエリアがソースに設定されている場合、転送の再開始時にソースレジスタにも初期値がリロードされます。また、リピートエリアが宛先に設定されている場合、転送の再開始時に宛先レジスタに初期値がリロードされます。

ブロックモード

ブロックモードでは、アクティベーションソースイベントが発生するたびに転送長全体が転送されます。たとえば、アクティベーションソースがタイマ、バイトサイズが 2、バイト長が 12 である転送がブロックモードで構成されている場合、アクティベーションイベントが発生するたびに 24 バイトが転送されます。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピートエリアがソースに設定されている場合、転送の再開始時にソースレジスタにも初期値がリロードされます。また、リピートエリアが宛先に設定されている場合、転送の再開始時に宛先レジスタに初期値がリロードされます。

アドレスモード

サイズ（1 バイト、2 バイト、または 4 バイト）が転送されるたびに、転送元ポインタと転送先ポインタはそれぞれ src_addr_mode と dest_addr_mode, によって調整されます。たとえば、src_addr_mode が TRANSFER_ADDR_MODE_INCREMENTED に設定され、サイズが TRANSFER_SIZE_4_BYTES, に設定されている場合、p_dest ポインタは転送が終わるたびに 4（転送サイズ）だけインクリメントされます。TRANSFER_ADDR_MODE_FIXED に設定されている場合、ポインタは変化しません。

チェーンされた転送

チェーンされた転送は DTC でのみサポートされています。チェーンされた転送を使用するには、transfer_info_t 構造体の配列を作成します。最後の転送を除くすべての転送について、chain_mode を TRANSFER_CHAIN_MODE_ENABLED に設定します。p_info を、transfer_info_t 構造体用の配列内にある最初の構造体のベースに設定します。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.14.3 アプリケーションへの DTC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DTC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

DTC HAL ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（転送ドライバーのデフォルト名は g_transfer0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

DTC HAL ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ transfer0 DTC Driver on r_dtc	**Threads > HAL/Common**	**New Stack > Driver > Transfer > Transfer Driver on r_dtc**

次の図に示すように r_dtc の DTC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

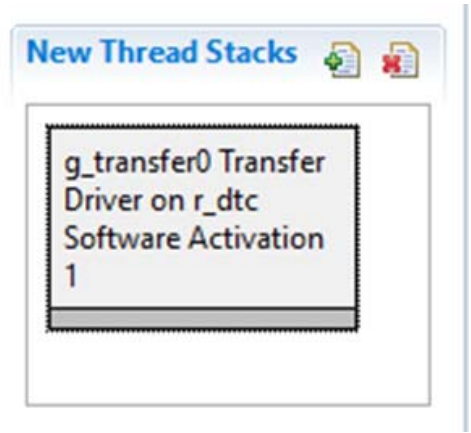


図 353: DTC HAL モジュールのスタック

5.2.14.4 DTC HAL モジュールの構成

ユーザーは必要な動作に合わせて DTC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータの [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまで

スクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_dtc での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択します。
Software Start	Disabled, Enabled (Default: Disabled)	ソフトウェアスタートを有効にするかどうかを選択します
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを格納するセクション
Name	g_transfer0	モジュール名。
Mode	Normal	モードの選択。
Transfer Size	1 Byte, 2 Bytes, 4 Bytes (Default: 2 Bytes)	転送サイズの選択。
Destination Address Mode	Fixed, Incremented, Decremented (Default: Fixed)	宛先アドレスモードの選択。
Source Address Mode	Fixed, Incremented, Decremented (Default: Fixed)	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source, Destination (Default: Source)	リピートエリアの選択。

ISDE Property	Value	説明
Interrupt Frequency	After all transfers have completed, After each transfer (Default: After all transfers have completed)	割り込み発生のタイミングを定義します。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	割り込みアクティベーションソースの詳細なリストについては、MCU ユーザーマニュアルの表 14.4 を参照してください。 (Default: Software Activation 1)	アクティベーションソースの選択。
Auto Enable	True, False (Default: True)	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。
ELC Software Event Interrupt Priority	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	割り込み優先順位。

注: *上記の設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

DTC HAL モジュールのクロック構成

DTC 周辺モジュールは、ICLK をクロックソースとして使用します。ICLK の周波数を設定するには、ビルドの前に SSP コンフィギュレータの **[Clocks]** タブを使用するか、実行時に CGC インタフェースを使用します。

DTC HAL モジュールのピン構成

DTC はいずれのピンとも関連付けられていません。

5.2.14.5 アプリケーションでの DTC HAL モジュールの使用

アプリケーションで DTC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 DTC を初期化します。
2) enable API を使用して、 DTC を有効にします （自動的に有効化されない場合）。
3) 必要に応じて、他の API を使用して転送を管理します。
4) 必要に応じて、 close API を使用して DTC を閉じます。

DTC HAL モジュールの使用に関するこれらの一般的な手順を、次の図の通常の動作フロー図に示します。

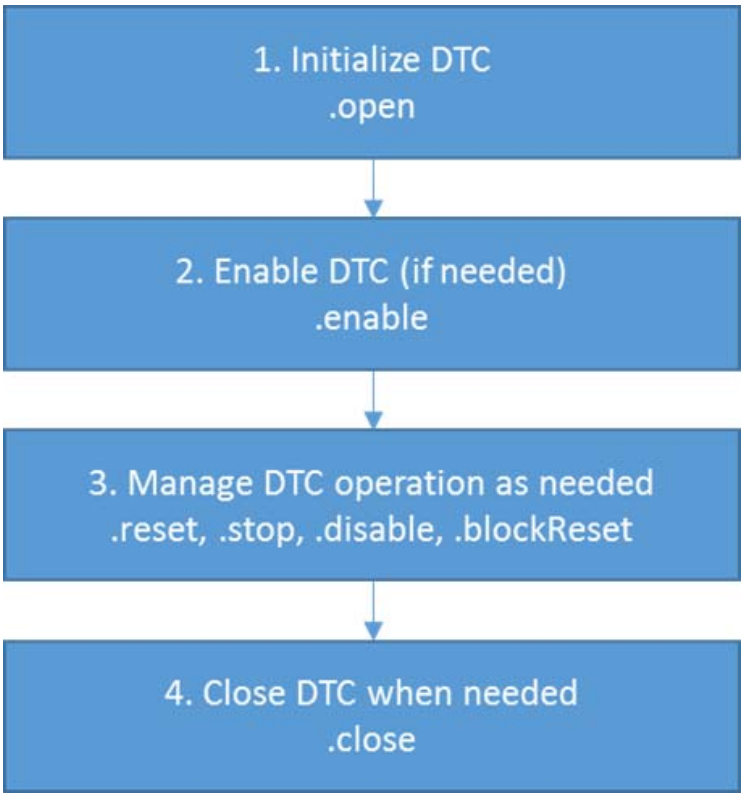


図 354: 一般的な DTC HAL モジュールアプリケーションのフロー図

5.2.15 ELC ドライバー

ELC HAL モジュールは、次の機能をサポートしています。

- 2 つのブロック間にイベントリンクを作成します。
- その 2 つのブロック間のイベントリンクを切断します。

- CPU に割り込む 2 つのソフトウェアイベントのうち 1 つを生成します。

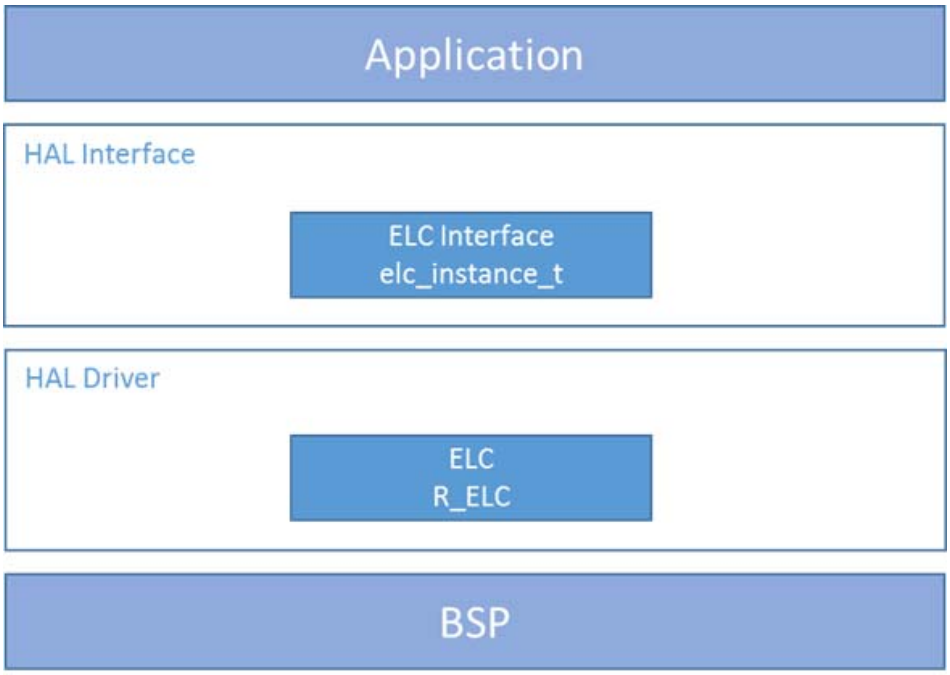


図 355: ELC HAL モジュールのブロック図

5.2.15.1 ELC HAL モジュールの API の概要

ELC HAL モジュールでは、モジュール間のイベントリンクの初期化、有効化、無効化、作成、切断のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ELC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
init	g\elc.p_api->init(g_elc.p_cfg) イベントリンクコントローラ内のすべてのリンクを初期化します。
softwareEventGenerate	g_elc.p_api->softwareEventGenerate(event_num) イベントリンクコントローラでソフトウェアイベントを生成します。

Function Name	API の呼び出し例と説明
linkSet	g_elc.p_api->linkSet(peripheral, signal) 1 つのイベントリンクを作成します。
linkBreak	g_elc.p_api->linkBreak(peripheral) イベントリンクを切断します。
enable	g_elc.p_api->enable() イベントリンクコントローラの動作を有効にします。
disable	g_elc.p_api->disable() イベントリンクコントローラの動作を無効にします。
versionGet	g_elc.p_api->versionGet(&version) バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に完了しました。
SSP_ERR_ASSERTION	p_version が NULL です。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.15.2 ELC HAL モジュールの動作の概要

ELC HAL モジュールを使用すると、開発者は、Synergy MCU 内のあるペリフェラルによって生成されたイベントを使用して別のペリフェラルの動作を開始することで、さまざまなペリフェラルの動作をリンクできま

す。 ELC HAL モジュールの API を使用すると、 2 つのブロック間のリンクを簡単に作成できます （たとえば、周期的なスキャンのインターバルを制御するためのタイマから ADC へのリンク）。このようにしてさまざまなペリフェラルを接続することにより、CPU による介入がほとんど、またはまったく必要のない、インテリジェントな機能を構築できます。

次の図は ELC の簡単なブロック図であり、入力イベントソースと、イベントによってトリガできるペリフェラルが示されています。（入力トリガと出力トリガの数は S7G2 MCU に固有です）。他の Synergy デバイスは、異なる数のイベントをサポートします。

図 356: ELC ハードウェアのブロック図

このモジュールガイドに関連付けられているアプリケーションプロジェクトを見ると、ELC の使用例がわかります。

bsp_elc.h, ファイルの Synergy によって生成されたコードでは、 ELC のマッピングを確認できます。 S7G2 マイクロコントローラユニット（MCU）のイベントリストとペリフェラルは次のとおりです。関連する『MCU ユーザーズマニュアル』では、 ELC の動作についての追加情報もわかります。

```
#ifdef NX_CRYPTO_ENGINE_SW
extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers;
#else
extern const NX_SECURE_TLS_CRYPTO nx_crypto_tls_ciphers_synergys7;
#endif

void tls_dtls_session_init0(NX_SECURE_TLS_SESSION *p_tls_session)
{
    UINT g_tls_session0_err;
    /* Create TLS client. */

    g_tls_session0_err = nx_secure_tls_session_create (p_tls_session,
#ifdef NX_CRYPTO_ENGINE_SW
                                                       &
nx_crypto_tls_ciphers,
#else

&nx_crypto_tls_ciphers_synergys7,
#endif

g_tls_session0_meta_data,

sizeof(g_tls_session0_meta_data));

    if (NX_SUCCESS != g_tls_session0_err)
    {
        g_tls_session0_err_callback ((void *) p_tls_session,
&g_tls_session0_err);
    }
}


static void new_thread0_func(ULONG thread_input)
{
    /* Initialize each module instance. */
    /** Call initialization function if user has selected to do so. */
#if (1)
    tls_dtls_session_init0 (&g_tls_session0);
#endif

    /* Enter user code for this thread. */
    new_thread0_entry ();
}
```

図 357: bsp_elc.h の例での ELC のマッピング

ELC HAL モジュールの動作に関する重要な注意事項と制限事項

ELC HAL モジュールでは、ピン、クロック、または割り込みを構成する必要はありません。 ELC HAL モジュールはペリフェラルの間の「接続」メカニズムに過ぎません。ただし、ELC を介して I/O ポートをリンクする場合は、 I/O ピンを入力または出力として構成する必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.15.3 アプリケーションへの ELC HAL モジュールの組み込み

ISDE は、デフォルトで、必要な ELC HAL モジュールをプロジェクトの HAL/ 共通スレッドに自動的に追加します。そのため、後で説明する手順は必要ないか、禁止されることさえあります（ELC HAL モジュールが HAL/ 共通スレッドに既に構成されている場合、許可されるドライバーインスタンスは 1 つだけです）。以下の説明は、ドキュメントを完全なものにするため、および ELC HAL モジュールを誤って削除したときのために提供されます。

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ELC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらのタスクに精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

ELC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（ELC HAL モジュールのデフォルト名は g_elc. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

ELC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_elc ELC Driver on r_elc	Threads	New Stack> Driver> System> ELC Driver on r_elc

次の図に示すように、 r_elc の ELC HAL ドライバーがスレッドスタックに追加されると、コンフィギュレータはオプションダイアログを自動的に表示します。ELC HAL モジュールに対して行うことができる唯一の変更は、パラメータのチェックを有効または無効にすることです。

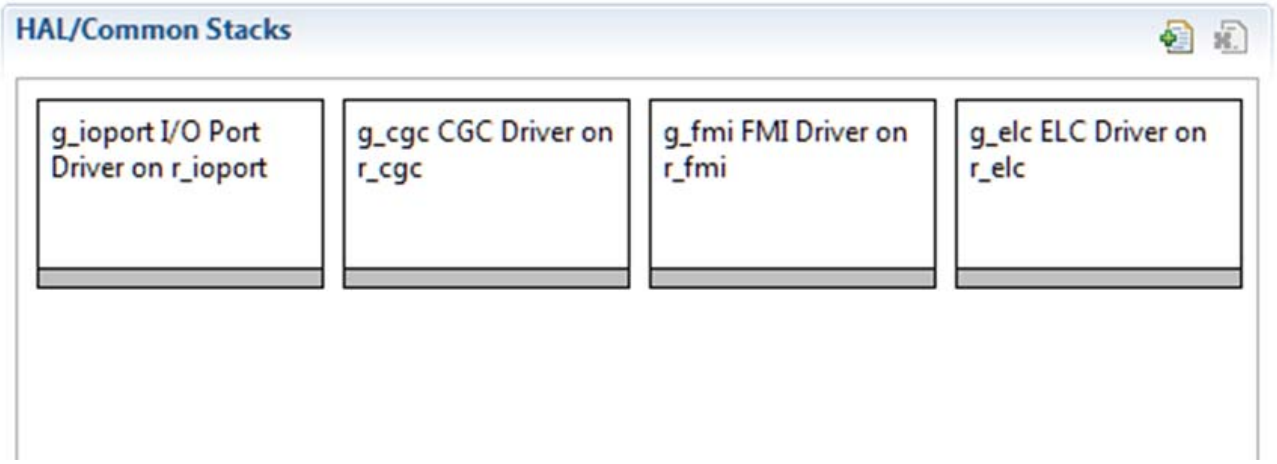


図 358: ELC HAL モジュールのスタック

5.2.15.4 ELC HAL モジュールの構成

ユーザーは必要な動作に合わせて ELC HAL モジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、 ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、ELC HAL モジュールを作成して、プロパティ設定を調べることができます。これは正しい判断に役立ち、 SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_elc での ELC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_elc	モジュール名。

***注:** 設定例とデフォルトは、Synergy S7G2 グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

ELC HAL モジュールのクロック構成

ELC ブロックにはクロック設定はありません。

ELC HAL モジュールのピン構成

ELC HAL モジュールに直接関連付けられている、構成の必要なピンはありません。

5.2.15.5 アプリケーションでの HAL モジュールの使用

アプリケーションで ELC HAL モジュールを使用する一般的な手順は次のとおりです。

1) init および enable API を使用して、 ELC を初期化します （ISDE によって自動的に行われます）。
2) linkSet API を使用して、ペリフェラルとイベントをリンクします。
3) enable API を使用して、リンクを有効にします。

次の図は、以上の一般的な手順を示した通常の動作フロー図です。

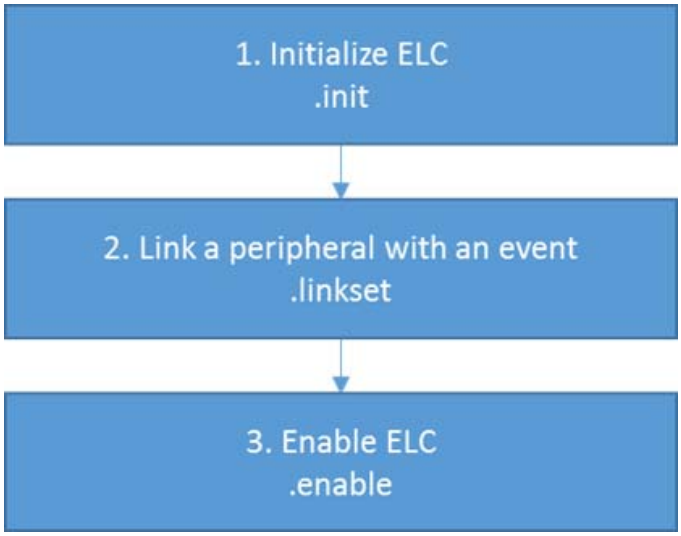


図 359: 一般的な ELC HAL モジュールアプリケーションのフロー図

5.2.16 外部 IRQ ドライバー

- 対象の Synergy MCU で使用可能な外部割り込み要求ピンをサポートします
- 複数の機能オプションをサポートします
  - 割り込みの生成の有効化と無効化。
  - IRQ ノイズフィルタの有効化と無効化
  - 外部ピン IRQ トリガの設定 （IRQ ピンでの立ち上がりエッジ、立ち下がりエッジ、低レベル）
- ユーザーコールバック関数の構成をサポートします。この関数は、外部ピン割り込みが生成されると、HAL モジュールによって呼び出されます。

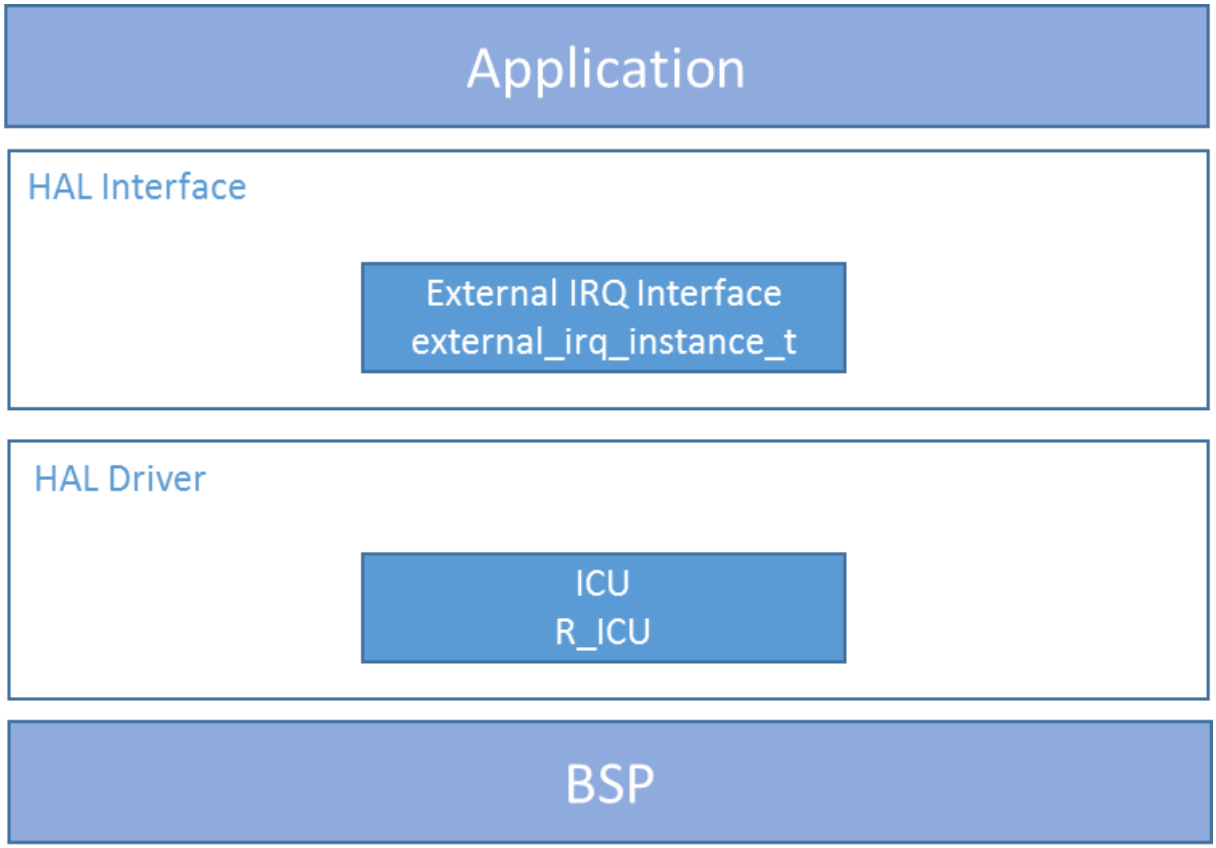


図 360:　外部 IRQ HAL モジュールのブロック図

5.2.16.1 外部 IRQ HAL モジュールの API の概要

外部 IRQ HAL モジュールでは、オープン、クローズ、および外部ピンからの割り込みイベントの待機のための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

外部 IRQ HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_external_irq.p_api->open(g_external_irq.p_ctrl, g_external_irq.p_cfg) インスタンスを開いて初期化します。

Function Name	API の呼び出し例と説明
enable	g_external_irq.p_api->enable(g_external_irq.p_ctrl) IRQ 発生時のコールバックを有効にします。
disable	g_external_irq.p_api->disable(g_external_irq.p_ctrl) IRQ 発生時のコールバックを無効にします。
triggerSet	g_external_irq.p_api->triggerSet(g_external_irq.p_ctrl, trigger) トリガーを設定します。
filterEnable	g_external_irq.p_api->filterEnable(g_external_irq.p_ctrl) ノイズフィルタを有効にします。
filterDisable	g_external_irq.p_api->filterDisable(g_external_irq.p_ctrl) ノイズフィルターを無効にします。
close	g_external_irq.p_api->close(g_external_irq.p_ctrl); インスタンスを閉じます。
versionGet	g_external_irq.p_api->wait(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。

Name	説明
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_INVALID_ARGUMENT	コールバックは NULL ではありませんが、ISR が無効になっています。
SSP_ERR_IN_USE	デバイスが使用中です。
SSP_ERR_NOT_OPEN	デバイスが開かれていません。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.16.2 外部 IRQ HAL モジュールの動作の概要

外部 IRQ HAL モジュールでは、外部割り込み要求を制御するための API のセットが提供されています。割り込みは、外部 IRQ ピンでの入力信号の立ち上がりエッジ、立ち下がりエッジ、両方のエッジ、または低レベルでトリガできます。デジタルフィルタ機能を有効にして、入力信号のノイズをある程度除去できます。ユーザーコールバック関数がサポートされており、IRQ イベントが発生するたびにトリガされます。

構成されている外部 IRQ イベントが発生したときに、DMAC または DTC を使用してデータの転送をトリガするには、アクティベーションソースを ELC_EVENT_PORTn_IRQ（n は IRQ チャネル番号）に設定して、DMAC または DTC 転送を構成します。

イベントリンクコントローラ（ELC）を使用して、外部割り込み要求から他のペリフェラルをトリガして開始できます。ELC HAL モジュールの詳細については、『SSP ユーザーズマニュアル』ユーザーガイドを参照してください。

外部 IRQ HAL モジュールの動作に関する重要な注意事項と制限事項

- 外部割り込み要求機能をサポートするポートピンを調べるときや、特定のポートピンの外部 IRQ 番号が知りたいときは、プログラムする Synergy デバイスのデータシートを参照してください。
- 外部 IRQ 番号は、ISDE の [Properties] ウィンドウでの外部 IRQ HAL モジュールに対するチャネル設定に対応します。
- 予想されるハードウェアイベントが発生したことをモジュールに通知するには、PORTn（n は IRQ 番号）の割り込みを BSP で有効にする必要があります。
- ユーザーコールバック関数を open API で登録できます。このコールバック関数が指定されている場合、IRQn がトリガされるたびに割り込みサービスルーチン (ISR) から呼び出されます。注 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に悪影響を及ぼす可能性があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.16.3 アプリケーションへの外部 IRQ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに外部 IRQ HAL モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

外部 IRQ HAL ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（モジュールのデフォルト名は g_external_irq0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

外部 IRQ HAL ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_icu0 External IRQ Driver on r_icu	Threads	New Stack> Driver> Input> External IRQ Driver on r_icu

次の図に示すように r_icu の IRQ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

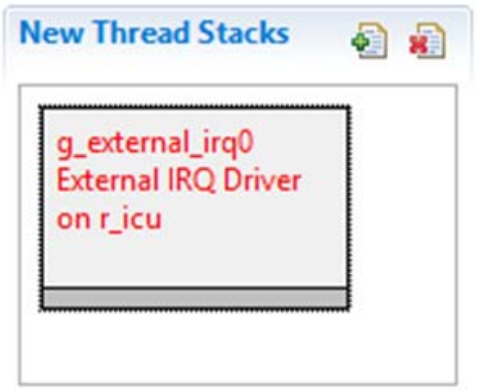


図 361: 外部 IRQ HAL モジュールのスタック

5.2.16.4 外部 IRQ HAL モジュールの構成

ユーザーは必要な動作に合わせて外部 IRQ HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロ

パティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を設定するときに ISDE で簡単に確認できます。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Default, Enabled, Disabled ; (Default: Default)	パラメータチェック
Name	g_external_irq0	ドライバー名。
Channel	0	使用するハードウェア IRQ チャネルを指定します。
Trigger condition	Falling, Rising, Both Edges, Low Level; (Default: Rising)	トリガの選択。
Digital Filtering	Enabled, Disabled ; (Default: Disabled)	デジタルフィルター有効 / 無効。
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64;(Default: PCKL/64)	ノイズフィルターサンプリング期間を設定します。
Interrupt enabled after initialization	True, False ; (Default: True)	割り込み要求イネーブルの選択。

ISDE Property	Value	説明
Callback	NULL	open API を使用して、ユーザーコールバック関数を登録できます。このコールバック関数が指定されている場合、IRQnがトリガーされるたびに割り込みサービスルーチン(ISR)から呼び出されます。注：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Interrupt Priority	Enabled, Disabled ; (Default: Disabled)	割り込み優先順位の設定。

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

外部 IRQ HAL モジュールのクロック構成

IRQ 周辺モジュールには、特定のクロック設定は必要ありません。

外部 IRQ HAL モジュールのピン構成

外部 IRQ 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では IRQ ピンの選択例を示します。

r_icu での外部 IRQ HAL ドライバーの選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
IRQ	Pins	Select Peripherals > Input: IRQ > IRQ0

注: *選択シーケンスでは、IRQ0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

r_icu での外部 IRQ ドライバーのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Enabled; (Default: Disabled)	割り込みを有効にするには [Enabled] を選択します。
NMI	None, P200; (Default: None)	マスク不可能な割り込みピン
IRQ00:14	None, Pnn, Pmm; Default: None	割り込み要求ピン

注: 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.2.16.5 アプリケーションでの外部 IRQ HAL モジュールの使用

アプリケーションで外部 IRQ HAL モジュールを使用するときの一般的な手順は次のとおりです。

アプリケーションで外部 IRQ HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、外部 IRQ HAL モジュールを初期化します
2) enable API を使用して、 IRQ を有効にします （必要な場合）
3) filterEnable API を使用して、ノイズフィルタを有効にします （必要な場合）
4) triggerSet API を使用して、トリガ条件を変更します （誤ったイベントを避けるため、それ以前にモジュールが閉じられている場合のみ）
5) filterDisable API を使用して、ノイズフィルタを無効にします （有効になっている場合）
6) close API を使用して、モジュールを閉じます （必要な場合）

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

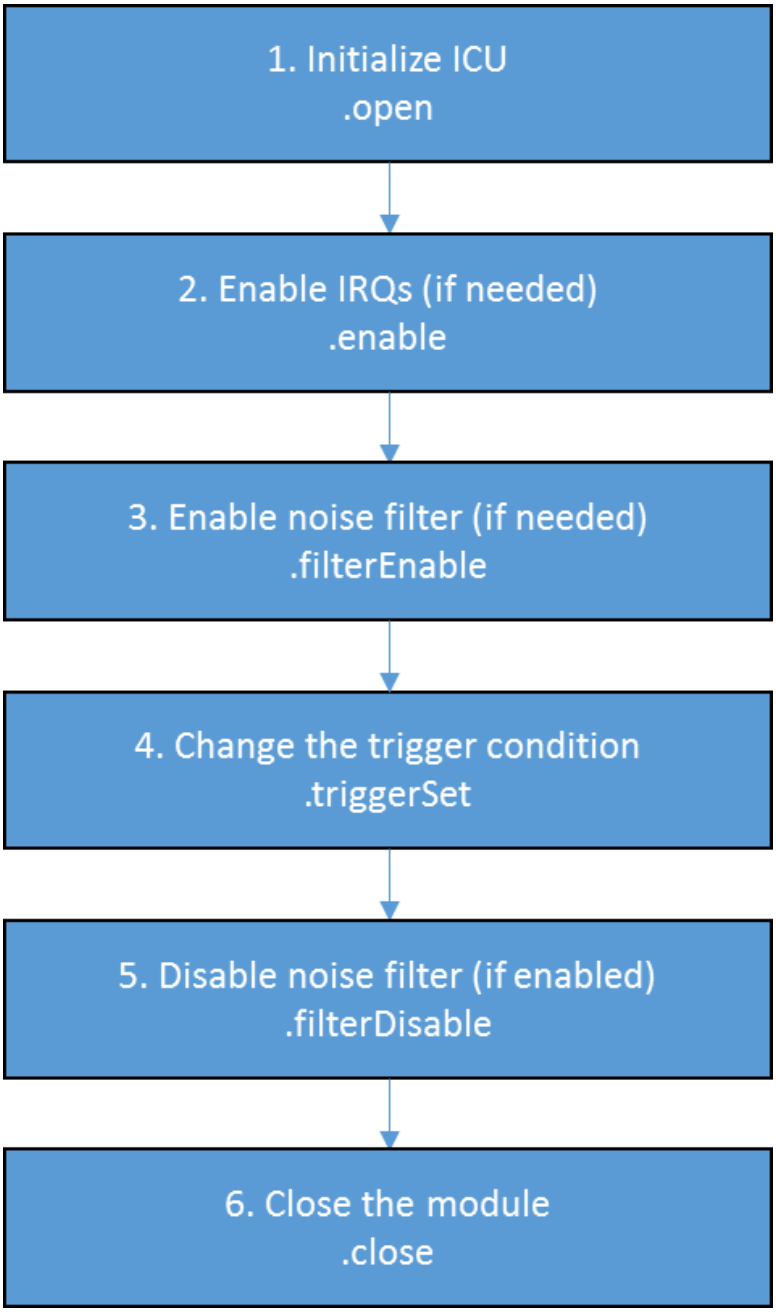


図 362:　一般的な外部 IRQ HAL モジュールアプリケーションのフローチャート

5.2.17 フラッシュドライバー

フラッシュ HAL モジュールの API により、アプリケーションは MCU 内に存在するデータと ROM 両方のフラッシュエリアのリード、ライト、消去を行うことができます。使用可能なフラッシュメモリの量は MCU パ

一ツごとに異なりますが、API 関数はすべてのデバイスに適用されます。フラッシュ HAL モジュールの主な特長を以下に示します。

- コードフラッシュ（ROM）のブロック消去、リード、ライト、およびブランクチェックのサポート。
- データフラッシュのブロッキングおよび非ブロッキング消去、リード、ライト、およびブランクチェックのサポート。
- コードフラッシュのブロッキング消去、リード、ライト、およびブランクチェックのサポート。
- 非ブロッキングデータフラッシュ操作の完了時のコールバック関数のサポート。
- コードフラッシュの指定したエリアだけを消去またはライトできるようにする、ROM フラッシュのアクセスウィンドウ（ライト保護）のサポート。
- スタートアッププログラムを最初に消去することなく安全にリライトできるようにする、ブートブロックスワッピングのサポート。

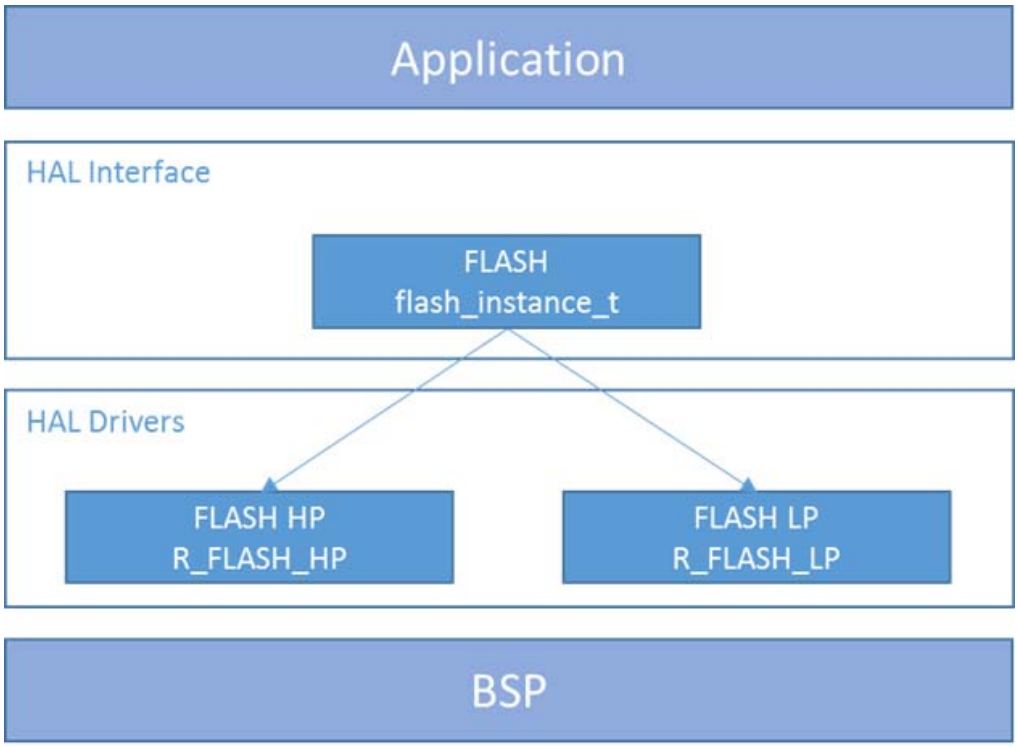


図 363: フラッシュ HAL モジュールのブロック図

5.2.17.1 フラッシュ HAL モジュールの API の概要

フラッシュ HAL モジュールでは、フラッシュメモリのオープン、リード、消去、クローズなどの操作のための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

フラッシュ HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_flash0.p_api->open(g_flash0.p_ctrl, g_flash0.p_cfg); FLASH デバイスを開きます。
write	g_flash0.p_api->write(g_flash0.p_ctrl,(uint32_t) write_buffer, FLASH_CF_32KB_BLOCK55, CODE_BLOCK_SIZE_32KB); FLASH デバイスに書き込みます。
read	g_flash0.p_api->read(g_flash0.p_ctrl, read_buffer, DATA_FLASH_ADDR, num_bytes); FLASH デバイスから読み取ります。
read	g_flash0.p_api->erase(g_flash0.p_ctrl, FLASH_CF_32KB_BLOCK55,num_sectors); FLASH デバイスを消去します。
blankCheck	g_flash0.p_api->blankCheck(g_flash0.p_ctrl, FLASH_CF_32KB_BLOCK55, FLASH_DATA_BLOCK_SIZE, &blankCheck); FLASH デバイスのブランクチェックを実行します。
close	g_flash0.p_api->close(g_flash0.p_ctrl); FLASH デバイスを閉じます。
statusGet	g_flash0.p_api->statusGet(g_flash0.p_ctrl); FLASH デバイスのステータスを取得します。

Function Name	API の呼び出し例と説明
accessWindowSet	g_flash0.p_api->accessWindowSet(g_flash0.p_ctrl, FLASH_CF_32KB_BLOCK1, FLASH_CF_32KB_BLOCK3); FLASH デバイスのアクセスウィンドウを設定します。
accessWindowClear	g_flash0.p_api->accessWindowClear(g_flash0.p_ctrl); FLASH デバイスの既存のコードフラッシュアクセスウィンドウをクリアします。
reset	g_flash0.p_api->reset(g_flash0.p_ctrl); FLASH デバイス用の関数をリセットします。
updateFlashClockFreq	g_flash0.p_api-> updateFlashClockFreq (g_flash0.p_ctrl); フラッシュクロック周波数（FCLK）を更新し、タイムアウト値を再計算します。
startupAreaSelect	g_flash0.p_api->startupAreaSelect(g_flash0.p_ctrl, FLASH_STARTUP_AREA_BLOCK1, true); Default（ブロック 0）と Alternate（ブロック 1）のどちらのブロックをスタートアップ領域ブロックとして使用するかを選択します。 パラメータ 2 に使用可能なすべての値については、後の表を参照してください。
versionGet	g_flash0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの最後にある「参考文献」セクションで説明されている『SSP ユーザーズマニュアル』を参照してください。*

.setupAreaSelect のパラメータ 2 のオプション

Swap Type	Is_temporary	操作
FLASH_STARTUP_AREA_BLOCK0	False	次回リセット時のスタートアップ領域はブロック 0 です。
FLASH_STARTUP_AREA_BLOCK0	False	次回リセット時のスタートアップ領域はブロック 0 です。
FLASH_STARTUP_AREA_BLOCK1	False	次回リセット時のスタートアップ領域はブロック 1 です。
FLASH_STARTUP_AREA_BLOCK1	True	スタートアップ領域は、即座に、ただし一時的に、ブロック 1 に切り替わります。
FLASH_STARTUP_AREA_BTFLG	True	スタートアップ領域は、即座に、ただし一時的に、構成 BTFLG によって決定されるブロックに切り替わります。

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_IN_USE	デバイス使用中エラーです。
SSP_FLASH_ERR_FAILURE	フラッシュ障害エラーです。
SSP_ERR_FCLK	フラッシュ操作では、 FCLK が 4 MHz 以上でなければなりません。
SSP_ERR_TIMEOUT	タイムアウトエラー。
SSP_ERR_INVALID_SIZE	無効サイズエラーです。
SSP_ERR_INVALID_ADDRESS	無効アドレスエラーです。
SSP_ERR_ASSERTION	アサートエラーです。
SSP_ERR_INVALID_BLOCKS	無効なブロック数が指定されました。
SSP_ERR_INVALID_ARGUMENT	無効な引数のエラーです。
SSP_ERR_HW_LOCKED	ペリフェラルは既に使用中です。

Name	説明
SSP_ERR_CMD_LOCKED	FCU がロック状態です。通常、この状況は、アクセスウィンドウで保護されている領域に消去を試みたことが原因で発生します。
SSP_ERR_NOT_OPEN	フラッシュがまだ開かれていません。
SSP_ERR_IRQ_BSP_DISABLED	呼び出し側は BGO（バックグラウンドモード操作）を要求していますが、フラッシュの割り込みが有効にされていません。
SSP_ERR_WRITE_FAILED	書き込み操作が失敗しました。要求されたフラッシュ領域がブランクでない場合に、この値が返されることがあります。
SSP_ERR_PE_FAILURE	P/E モードに移行できませんでした。
false	指定されたアドレスは、この MCU 上の有効なフラッシュアドレスです。
true	指定されたアドレスは有効であり、p_block の情報にはこのアドレスブロックに関する詳細が含まれます。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスセクションを参照してください。*

5.2.17.2 フラッシュ HAL モジュールの動作の概要

フラッシュ API により、オンチップフラッシュエリアのプログラミングと消去のプロセスをより容易に実行できます。コード（ユーザー ROM）フラッシュエリアとデータフラッシュエリアの両方がサポートされています。この API の最も単純な形式を使用して、ブロッキング消去およびプログラム操作を実行できます。「ブロッキング」という用語は、プログラムまたは消去関数が呼び出されたときに、操作が完了するまで関数が戻らないことを意味します。この API はコードとデータフラッシュの両方のブロッキングをサポートしていますが、BGO（バックグラウンドモード操作）はデータフラッシュ操作でのみ使用できます。コードフラッシュ操作が実行中のとき、ユーザーがコードフラッシュエリアにアクセスすることはできません。コードフラッシュ操作が進行中のときにコードフラッシュエリアにアクセスしようとすると、フラッシュコントロールユニットがエラー状態に遷移します。

コードフラッシュ操作はブロッキングですが、一部の状況では操作がブロッキングのときでもコードフラッシュにアクセスできる場合があり、そのようなときにアクセスしてはならないということを憶えておくことが重要です。次のような場合です。

- ベクタテーブルが ROM に存在する場合のベクタテーブルアクセス。
- ベクタテーブル自体は ROM に存在しない場合であっても、ROM アドレスをベクタが示している割り込みによる ROM アクセス。

コードフラッシュ操作がブロッキング中に複数のスレッドが実行継続を許可されているマルチスレッドアプリケーション。

フラッシュ HAL モジュールの動作に関する重要な注意事項と制限事項

startupAreaSelect() はデータをブロック 0 にスワップします。 startupAreaSelect() を使用する場合は、スワップされるデータが有効であることを確認する必要があります。

データフラッシュ BGO の使用上の注意

データフラッシュ BGO を使用しているときは、ユーザー ROM、RAM、および外部メモリに引き続きアクセスできます。データフラッシュ操作の間に、データフラッシュにアクセスしないようにする必要があります。これには、データフラッシュにアクセスする可能性のある割り込みが含まれます。

コードフラッシュの使用上の注意

BGO モードはコードフラッシュでサポートされていないため、操作が完了する前にコードフラッシュ操作から戻ることはありません。デフォルトでは、ベクタテーブルはユーザー ROM（コードフラッシュ）内に存在します。ROM 操作中に割り込みが発生すると、割り込みの開始アドレスをフェッチするために ROM へのアクセスが行われ、エラーが発生します。

最も単純な回避策は、コードフラッシュ操作中の割り込みを無効化することです。別のオプションは、ベクタテーブルを RAM にコピーし、それに応じて VTOR（ベクタテーブルオフセットレジスタ）を更新して、割り込みサービスルーチンが RAM 外で実行されるようにすることです。同様に、マルチスレッド環境では、コードフラッシュ操作の進行中に、ROM から実行するスレッドがアクティブになることができないようにする必要があります。

ブランクチェック

blankCheck API 関数は、コードまたはデータフラッシュの内容がブランクかどうかをチェックします。フラッシュ（コードまたはデータ）には、最初に消去しない限り書き込みができないことに注意してください。blankCheck 関数は、指定されたエリアがブランクで書き込み可能かどうかを判定します。ほとんどすべての場合、フラッシュの内容を 0xFF と比較するだけでは、エリアがブランクかどうかを判定するのに十分ではありません。唯一の例外はフラッシュ HP コードフラッシュです。Flash_HP コードフラッシュの 0xFF は、確実にブランクを示します。すべての場合において、 blankCheck API 関数を使用することを強くお勧めします。

フラッシュステータス

statusGet API 関数を使用すると、アプリケーションがフラッシュの「準備完了」状態をクエリできるようになります。これは、ユーザーがコールバック関数を使用しないことを選択したためにデータフラッシュ操作の完了が非同期的に通知されないデータフラッシュ BGO 操作で有用です。この場合、データフラッシュは BGO モードで動作するように構成されているため、操作（たとえば消去）が開始されると、呼び出しは即座に戻り、操作はバックグラウンドで実行されます。 statusGet API 関数を呼び出すことにより、ユーザーは、操作が安全に完了したかエラーを生成し、別のフラッシュ操作に安全に移行できるようになったことを判定できます。

ブロックのスワップ

startupAreaSelect API 関数を使用すると、スタートアップエリアブロックとして使用されるブロック（デフォルト（ブロック 0）または代替（ブロック 1））を選択できます。提供されているパラメータは、どのブロックがアクティブスタートアップブロックになり、そのアクションが次のリセットの後すぐに（ただし一時的に）実行されるのか、恒常的に実行されるのかを決定します。

一時的な切り替えには限定的なメリットしかないように見えますが、ブロック 0 が書き込み保護となるようなアクセスウィンドウがある場合、アクセスウィンドウに触らずに、一時的な切り替えを行い、ブロックを更新して、元に切り替えることができます。

フラッシュクロック (FCLK)

FCLK は、すべてのフラッシュ操作の実行においてフラッシュにより使用されるクロックです。フラッシュ操作が成功するためには、4MHz 以上でなければなりません。open 関数の中でフラッシュクロックがチェックされて、4MHz 未満の場合、 open は SSP_ERR_FCLK. を返します。フラッシュ API が開かれた後で FCLK の周波数を変更する場合は、updateFlashClockFreq API 関数を呼び出して API に変更を通知する必要があります。そうしないと、フラッシュ操作が失敗し、パーツが損傷する可能性があります。

割り込み

フラッシュ対応割り込みを有効にする必要があるのは、データフラッシュ BGO を使用する予定のある場合だけです。このモードでは、アプリケーションはデータフラッシュ操作を開始した後、ユーザー提供のコールバック関数を使用してその完了またはエラーの通知を非同期的に受けることができます。コールバックには、コールバックイベント（つまり、 is,FLASH_EVENT_ERASE_COMPLETE) のソースを示すイベント情報を含む構造体が渡されます。

FLASH FRDYI 割り込みが有効になっている場合、対応する ISR がフラッシュドライバーで定義されます。ISR は、ユーザーコールバック関数が open API で登録されている場合は、それを呼び出します。

注: *フラッシュ HP は追加のフラッシュエラー割り込みをサポートしており、FLASH HP で BGO モードが有効になっている場合は、 FRDYI および FIFERR 割り込みの両方にプライオリティを与える必要があります。*

アクセスウィンドウ

アクセスウィンドウは、コードフラッシュ内でプログラミング / 消去が有効な連続した領域を定義します。この領域はブロック境界上にあり、開始アドレスと終了アドレスが accessWindowSet に提供されます。開始アドレスを含むブロックが最初のブロックです。エンドアドレスを含むブロックが最後のブロックです。したがって、アクセスウィンドウは、先頭のブロックから最後のブロックまでとなります（先頭のブロックと最後のブロックを含む）。この範囲の外側はすべて、ライト保護されます。無効なアドレス情報が accessWindowSet に提供されると、SSP_ERR_INVALID_ADDRESS. が返されます。accessWindowClear API 関数を呼び出すことにより、アクセスウィンドウを削除できます。

- 高性能フラッシュモジュール (Flash_HP) は、MCU の S7 ファミリおよび S5 ファミリのプログラミングに使用される API です。
- ローパワーフラッシュモジュール (Flash_LP) は、MCU の S3 ファミリおよび S1 ファミリのプログラミングに使用される API です。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.17.3 アプリケーションへのフラッシュ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにフラッシュ HAL モジュールを組み込む方法について説明します。

注: *このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

フラッシュドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（フラッシュドライバーのデフォルト名は g_flash0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

フラッシュドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_flash0 Flash Driver on r_rflash_hp	Threads	New Stack > Driver > Storage > Flash Driver on r_flash_hp
g_flash0 Flash Driver on r_rflash_lp	Threads	New Stack > Driver > Storage > Flash Driver on r_flash_lp

次の図に示すように、r_flash_hp または r_flash_lp のフラッシュ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

***注:** 次の図には両方のフラッシュ HAL モジュールが示されていますが、使用する必要があるのは、選択されている MCU に応じたどちらか一方だけです。図を完全なものにするために両方示してあります。*

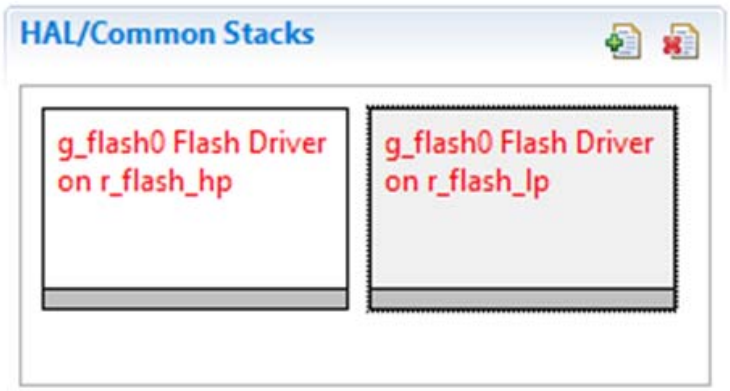


図 364: フラッシュ HAL モジュールのスタック

5.2.17.4 フラッシュ HAL モジュールの構成

ユーザーは必要な動作に合わせてフラッシュ HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties]

ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** 次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

フラッシュ HAL ドライバーは 2 つの異なるモジュール（r_flash_hp と r_flash_lp,）のどちらか一方に実装されており、次の表ではこれらの実装の構成設定を示してあります。

表 5 r_flash_hp でのフラッシュ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
code-flash Programming Enable	Enable, Disabled Default: Disabled	コードフラッシュプログラミングを有効にするかどうかを制御します。無効化すると、API が使用する ROM の量を小さくできます。
Name	g_flash0	モジュール名。
data-flash Background Operation	Enabled, Disabled Default: Enabled	有効にすると、データフラッシュを参照するフラッシュ API コールは即座に戻ることができ、操作はバックグラウンドで続行されます。
Callback	NULL	データフラッシュBGO操作が完了またはエラーになったときに呼び出されるコールバック関数。ユーザーコールバック関数を open で登録できます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

ISDE Property	Value	説明
Flash Ready Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フラッシュ対応の割り込み優先順位の選択。
Flash Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フラッシュエラーの割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_flash_lp でのフラッシュ HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
code-flash Programming Enable	Enable, Disabled Default: Disabled	コードフラッシュプログラミングを有効にするかどうかを制御します。無効化すると、API が使用する ROM の量を小さくできます。
Name	g_flash0	モジュール名。
data-flash Background Operation	Enabled, Disabled Default: Enabled	有効にすると、データフラッシュを参照するフラッシュ API コールは即座に戻ることができ、操作はバックグラウンドで続行されます。

ISDE Property	Value	説明
Callback	NULL	データフラッシュBGO操作が完了またはエラーになったときに呼び出されるコールバック関数。ユーザーコールバック関数を open で登録できます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Flash Ready Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フラッシュ対応の割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_flash_lp でのフラッシュ HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	API パラメータチェック用のコードを含めるかどうかを制御します。
code-flash Programming Enable	Enable, Disabled Default: Disabled	コードフラッシュプログラミングを有効にするかどうかを制御します。無効化すると、API が使用する ROM の量を小さくできます。
Name	g_flash0	モジュール名。
data-flash Background Operation	Enabled, Disabled Default: Enabled	有効にすると、データフラッシュを参照するフラッシュ API コールは即座に戻ることができ、操作はバックグラウンドで続行されます。

ISDE Property	Value	説明
Callback	NULL	データフラッシュBGO操作が完了またはエラーになったときに呼び出されるコールバック関数。ユーザーコールバック関数を open で登録できます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Flash Ready Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	フラッシュ対応の割り込み優先順位の選択。

注: *設定例とデフォルトは、 Synergy S3A7 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

デフォルト以外の設定が望ましい場合もあります。たとえば、コードフラッシュプログラミングを無効にすると、ドライバーのコードサイズを減らすのに役立つことがあります。

フラッシュ HAL モジュールのクロック構成

フラッシュ対応割り込みを有効にする必要があるのは、データフラッシュ BGO（バックグラウンドモード操作）を使用する予定がある場合だけです。このモードでは、アプリケーションはデータフラッシュ操作を開始した後、ユーザー提供のコールバック関数を使用してその完了（またはエラー）の通知を非同期的に受けることができます。コールバック関数には、コールバックイベント (FLASH_EVENT_ERASE_COMPLETE.) のソースを示すイベント情報が含まれる構造体が渡されます。

割り込みを有効にするには、 e^2 studio のプロジェクトコンフィギュレーターの [ICU] タブで、 FCU > FRDYI 割り込みの優先度を設定します。これにより、 synergy_cfg/ssp_cfg/bsp/bsp_irq_cfg.h の BSP_IRQ_CFG_FCU_FRDYI に、選択した優先度が設定されます。

BSP で FLASH FRDYI 割り込みが有効になっている場合、対応する ISR がフラッシュドライバーで定義されます。 ISR は、ユーザーコールバック関数が open で登録されている場合は、それを呼び出します。

注: *フラッシュ HP は追加のフラッシュエラー割り込みをサポートしており、FLASH HP で BGO モードが有効になっている場合は、 FRDYI および FIFERR 割り込みの両方にプライオリティを与える必要があります。*

フラッシュ HAL モジュールのクロック構成

フラッシュ回路は FCLK をそのクロックとして使用します。 FCLK は、 4MHz 以下である必要があります。 flash Open() 関数を呼び出した後でこのクロックレートを変更する場合は、updateFlashClockFreq() を呼び出してフラッシュ API にその変更を通知する必要があります。

フラッシュ HAL モジュールのピン構成

フラッシュ回路はどの MCU ピンも使用しません。

5.2.17.5 アプリケーションでのフラッシュ HAL モジュールの使用

アプリケーションでフラッシュ HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、フラッシュ HAL を初期化します。
2) 割り込みを無効にします。
3) blankCheck API を使用して、コードフラッシュエリアのブランクチェックを行います。
4) erase API を使用して、1 つまたは複数のコードフラッシュブロックを消去します。
5) write API を使用して、コードフラッシュを書き込みます。
6) 割り込みを有効にします。
7) blankCheck API を使用して、データフラッシュエリアのブランクチェックを行います。
8) erase API を使用して、1 つまたは複数のデータフラッシュブロックを消去します。
9) write API を使用して、データフラッシュを書き込みます。
10) すべてのフラッシュ操作が終了したら、close API を使用して閉じます。

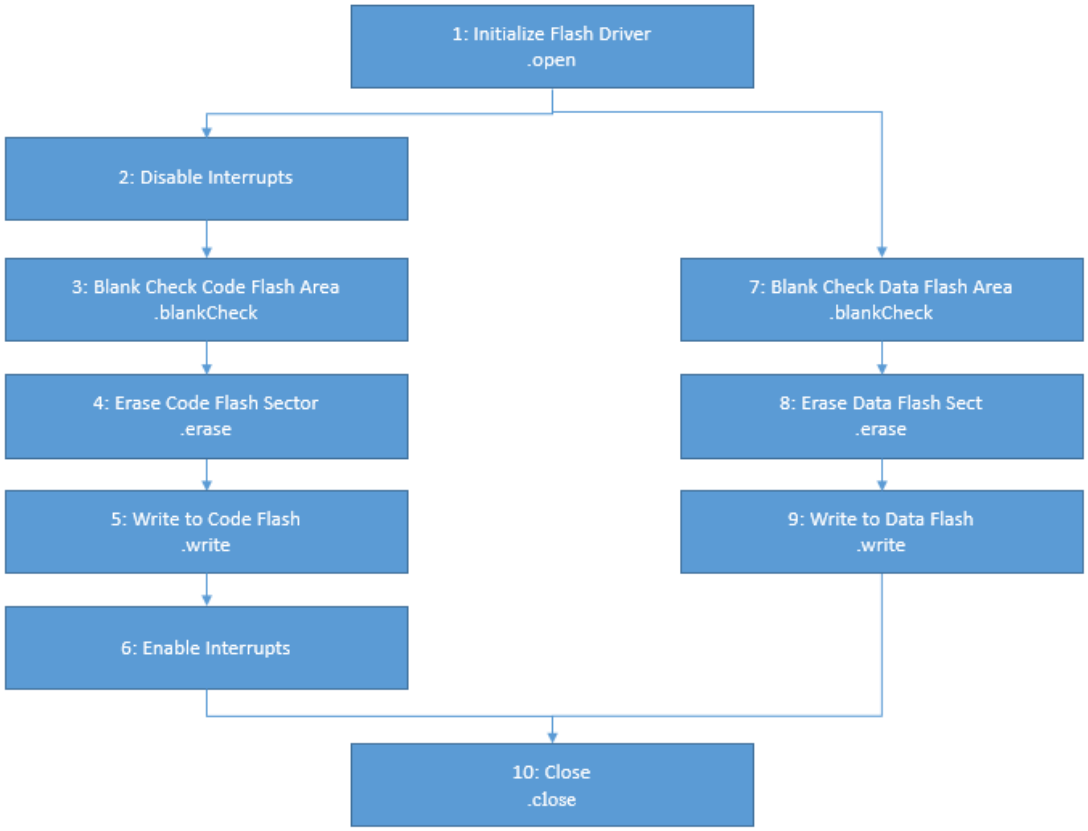


図 365: 通常のフラッシュ HP HAL モジュールアプリケーションのフロー図

5.2.18 FMI ドライバー

FMI HAL モジュールは、Synergy マイクロコントローラの FMIFRT（Factory MCU Information Flash Root Table）を読み取って、フラッシュ内のテーブルの先頭アドレスを取得します。このモジュールは、テーブルの製品情報レコードに呼び出し側のポインタを設定します。この情報を使用して、この MCU パッケージ固有の機能を判定できます。 FMI HAL モジュールからは次の情報を利用できます。

- 製品情報 : 製品名、パッケージ、ピン数、温度範囲
- 製品の機能 : メジャーバージョン、マイナーバージョン、バリエーションデータ
- 割り込みやイベントなどのイベント情報

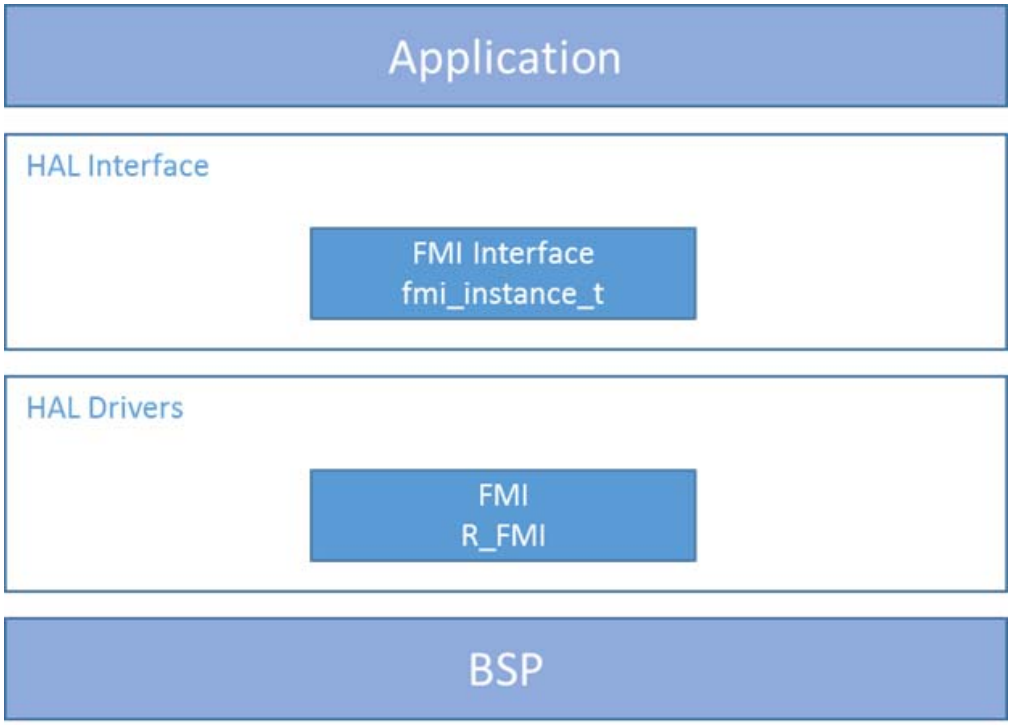


図 366: FMI HAL モジュールのブロック図

5.2.18.1 FMI HAL モジュールの API の概要

FMI HAL モジュールでは、FMIFRT にアクセスするための API が定義されています。次の表では、使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明を示します。API 要約の表の後に、ステータス戻り値の一覧の表があります。

FMI HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
init	g_fmi.p_api->init(); FMI ベースポインタを初期化します。
productInfoGet	g_fmi.p_api->productInfoGet(&g_pp_product_info); 製品情報レコードのアドレスを取得して、g_pp_product_info ポインタに格納します。

Function Name	API の呼び出し例と説明
uniqueIdGet	g_fmi.p_api->uniqueIdGet(&g_p_unique_id); 一意の ID を g_p_unique_id ポインタにコピーします。
productFeatureGet	g_fmi.p_api->productFeatureGet(&g_ssp_feature, &g_feature_info); 機能の情報を取得し、 g_feature_info ポインタに格納します。
eventInfoGet	g_fmi.p_api->peventInfoGet(&g_ssp_feature,SSP_SIGNAL_GPT_COUNTER_OVERFLOW, &g_event_info); イベントの情報を取得し、g_event_info ポインタに格納します。
versionGet	g_fmi.p_api->versionGet(&g_p_version); コンパイル時マクロに基づいて、ドライバーのバージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_FMI_DATA	提供されている FMI データテーブルが無効です
SSP_ERR_IP_CHANNEL_NOT_PRESENT	要求されたチャネルがこの MCU に存在しません。
SSP_ERR_IP_UNIT_NOT_PRESENT	要求されたユニットがこの MCU に存在しません。
SSP_ERR_INTERNAL	要求された機能は、現時点ではサポートされていないフォーマットです。
SSP_ERR_IRQ_BSP_DISABLED	イベント情報が見つかりませんでした

Name	説明
SSP_ERR_ASSERTION	呼び出し側のポインタが null です
SSP_ERR_INVALID_FACTORY_FLASH	ファクトリーフラッシュが無効です。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.18.2 FMI HAL モジュールの動作の概要

FMI HAL モジュールは、productInfoGet API を使用して、製品情報レコードのアドレスを取得し、fmi_product_info_t 構造体に設定します。

FMI HAL モジュールは、uniqueIdGet API を使用して、一意の ID をコピーし、fmi_unique_id_t 構造体に設定します。

FMI HAL モジュールは、productFeatureGet API を使用して、機能の情報を取得し、fmi_feature_info_t 構造体に設定します。

FMI HAL モジュールは、eventInfoGet API を使用して、イベントの情報をフェッチし、fmi_event_info_t 構造体に設定します。

FMI HAL モジュールは、versionGet API を使用して、コードのバージョンと API のバージョンを取得し、ssp_version_t 構造体に設定します。

FMI HAL モジュールの動作に関する重要な注意事項と制限事項

unique_id は非推奨です。ファクトリーMCU情報がアプリケーションコードによってリンクされている場合、一意の ID は含まれません。一意の識別子には uniqueIdGet を使用してください。

- FMI HAL インタフェースとその実装の制限事項については、SSP の最新のリリースノートを参照してください。
- FMI ドライバーは、FMIFRT レジスタを使用して、S7G2（WS2）Synergy マイクロコントローラファミリでテストされています。現在、ファクトリー MCU 情報テーブルのデータでプログラミングされている唯一の Synergy MCU です。
- S5D9 MCU の場合、r_fmi モジュールの uniqueIdGet API はエラーを返します。回避策はありません。

5.2.18.3 アプリケーションへの FMI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに FMI HAL モジュールを組み込む方法について説明します。

注：このプロセスを理解するには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。

FMI をアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。（FMI のデフォルト名は r_fmi0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

フラッシュドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_fmi FMI Driver on r_fmi	**Threads > HAL/Common**	**New Stack> Driver> System> FMI Driver on r_fmi**

次の図では、新しいプロジェクトを作成すると、r_fmi の FMI ドライバーが HAL/ 共通スタックに自動的に追加されることが示されています。これは、すべての設計でメッセージは FMI HAL モジュールによって提供される必要があるためです。

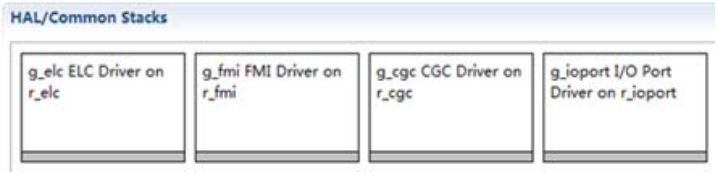


図 367: FMI HAL モジュールのスタック

5.2.18.4 FMI HAL モジュールの構成

ユーザーは必要な動作に合わせて FMI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチを提供します。*

r_fmi での FMI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	API パラメータチェック用のコードを含めるかどうかを制御します。
SSP MCU Information Symbol Name	g_fmi_data (Default)	このシンボルはファクトリフラッシュテーブル情報が格納されているベースアドレスにマッピングされます。変更しないでください。

ISDE Property	Value	説明
Part Number Mask	0xFE00 (Default)	各ビットは、 Synergy の部品番号の 1 文字を表します。 MSB は部品番号（'R'）の最初の文字です。MCU ファクトリーフラッシュの部品番号が SSP MCU情報の部品番号と一致するようにビットを設定してください。デフォルトのマスクは、動作温度、ソフトウェア ID、および品質 ID を除くすべてを確認します。
Name	g_fmi (Default)	モジュールインスタンス名。

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

FMI HAL モジュールのクロック構成

FMI HAL ドライバーには、特定のクロック構成は必要ありません。

FMI HAL モジュールのピン構成

FMI HAL ドライバーには、特定のピン構成は必要ありません。

5.2.18.5 アプリケーションでの FMI HAL モジュールの使用

アプリケーションで FMI を使用する一般的な手順は次のとおりです。

1) init API を使用して、 FMI を初期化します。 FMI はリセットの後で自動的に初期化されます。
2) 製品の情報を取得するには、 productInfoGet API を使用します。
3) 一意の ID を取得するには、 uniqueIdGet API を使用します。
4) 機能の情報を取得するには、 productFeatureGet API を使用します。
5) イベントの情報を取得するには、 eventInfoGet API を使用します。
6) ドライバーのバージョンの情報を取得するには、 versionGet API を使用します。

次の図は、以上の一般的な手順を示した通常の動作フロー図です。

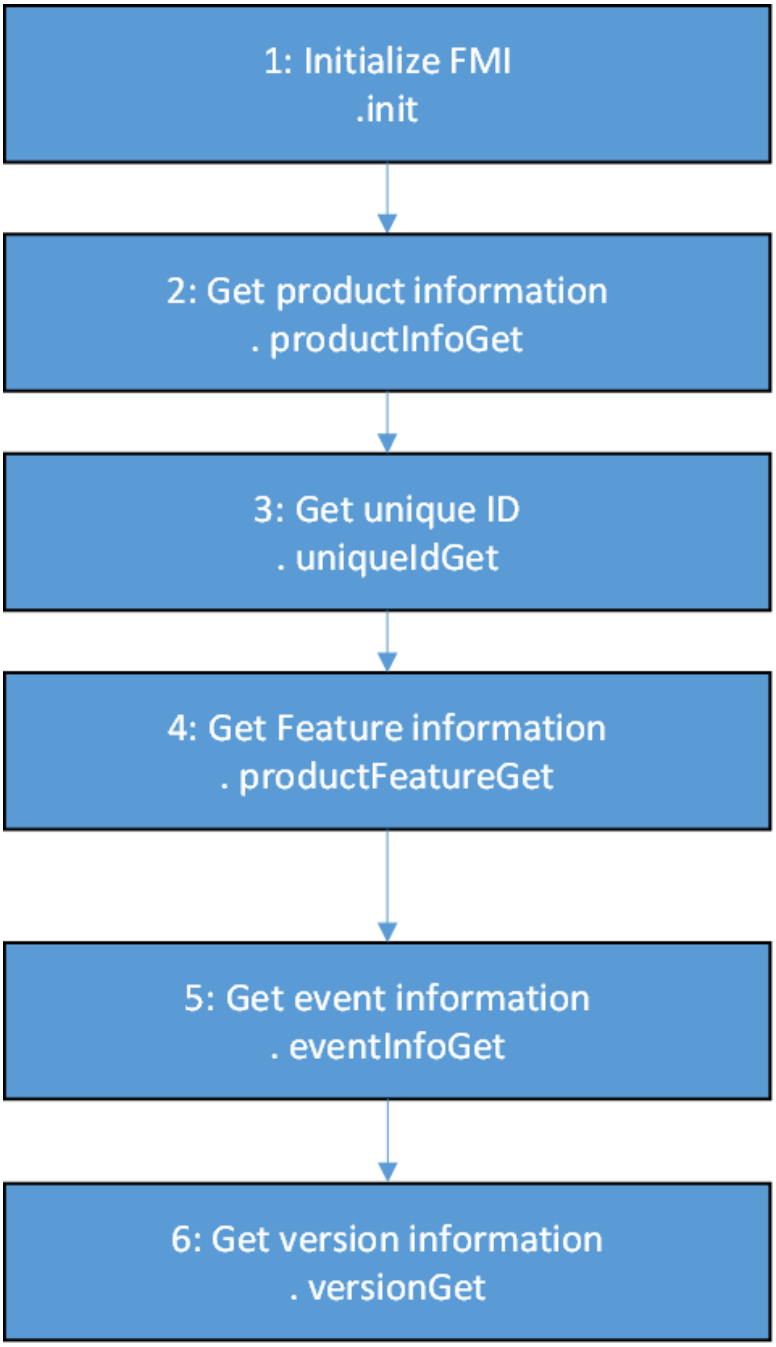


図 368:　一般的な FMI HAL モジュールアプリケーションのフロー図

5.2.19 GPT ドライバー

GPT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。この期間が経過すると、以下のいずれかのイベントが発生します。

- ユーザーコールバック関数が指定されている場合はそれを呼び出す CPU 割り込み
- ポートピンのトグル
- DMAC/DTC が転送インタフェースで構成されている場合はそれを使用したデータ転送
- イベントおよび周辺定義で構成されている別のペリフェラルの開始

汎用 PWM タイマ (GPT)

- 複数チャネル
  - S7G2: 32 ビット x 14 チャネル
  - S5D9: 32 ビット x 14 チャネル
  - S3A7: 32 ビット x 10 チャネル
  - S124: 32 ビット x 1 チャネル、 16 ビット x 6 チャネル
- コアクロックとしての PCLKD
- チャネルごとに 2 個の I/O ピン
- アップ / ダウンをカウントする鋸歯状 / 三角波
- 2 つの出力比較レジスタと入力キャプチャレジスタ
- 8 つのイベントリンクコントローラ （ELC）へのレスポンスおよび ELC 内のイベントとして構成可能

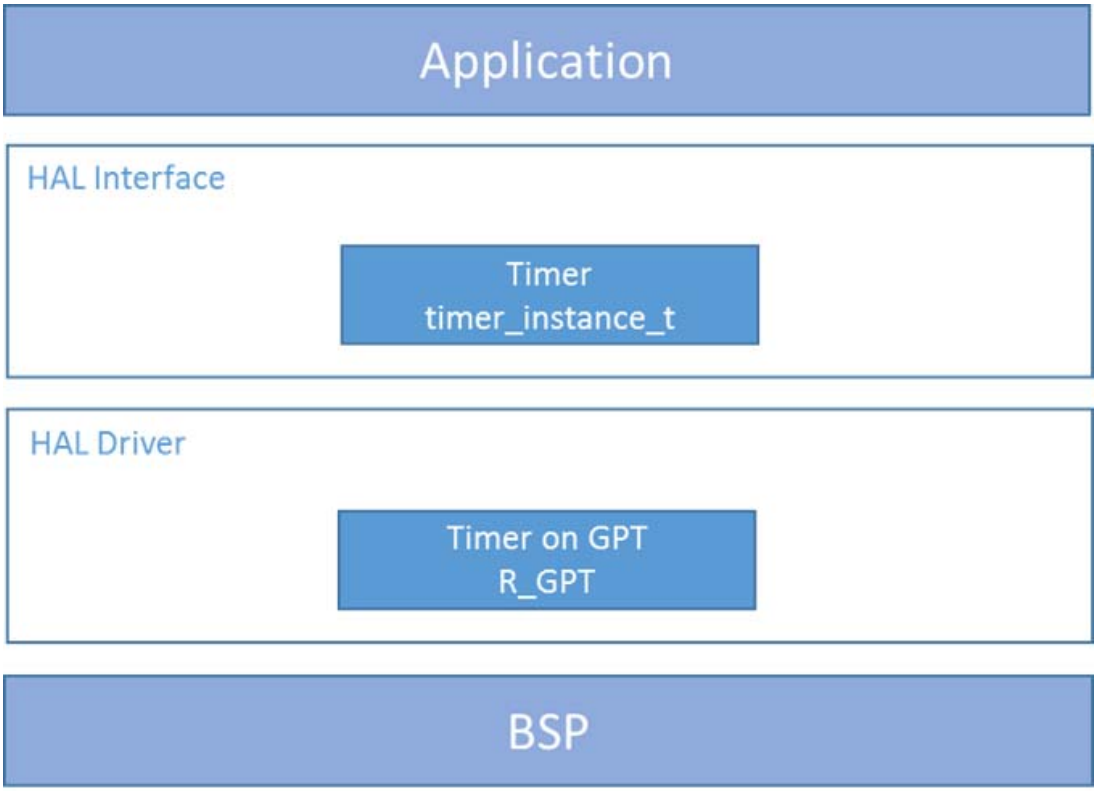


図 369: GPT HAL モジュールのブロック図

5.2.19.1 GPT HAL モジュールの API の概要

GPT HAL モジュールでは、タイマをオープン、クローズ、開始、停止するための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

GPT HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg) 初期設定。
stop	g_timer0.p_api->stop(g_timer0.p_ctrl) カウンタを停止します。

Function Name	API の呼び出し例と説明
start	g_timer0.p_api->start(g_timer0.p_ctrl) カウンタを開始します。
reset	g_timer0.p_api->reset(g_timer0.p_ctrl) カウンタを初期値にリセットします。
counterGet	g_timer0.p_api->counterGet(g_timer0.p_ctrl, &value) 現在のカウンタ値を取得し、指定されたポインタ value に格納します。
periodSet	g_timer0.p_api->periodSet(g_timer0.p_ctrl, period, unit) タイマが期限切れになるまでの時間を設定します。
dutyCycleSet	g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, duty_cycle, duty_cycle_unit, pin) デューティサイクルが期限切れになるまでの時間を設定します。
infoGet	g_timer0.p_api->infoGet(g_timer0.p_ctrl, &info) タイマが期限切れになるまでの時間をクロックカウント単位で取得し、指定されたポインタ info に格納します。
close	g_timer0.p_api->close(g_timer0.p_ctrl) ドライバーを再設定できるようになります。
versionGet	g_timer0.p_api->versionGet(&version) バージョンを取得し、指定されたポインタ version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	操作が正常に行われました。
SSP_ERR_ASSERTION	パラメータが NULL であるか、構成設定が許可されていません。
SSP_ERR_IN_USE	指定したチャネルはすでに開かれています。
SSP_ERROR_NOT_OPEN	チャネルが開かれていません。
SSP_ERR_INVALID_ARGUMENT	無効な引数が提供されました。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.19.2 GPT HAL モジュールの動作の概要

GPT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。時間が経過した時点で、CPU の割り込み、ポートピンの切り替え、DMAC または DTC を使用したデータ転送の開始、別のペリフェラルの動作開始のトリガなどを行うことができます。

次の図では、指定した時間後にポートピンの切り替えまたは CPU 割り込みの生成を行うためのフローチャートを示します。

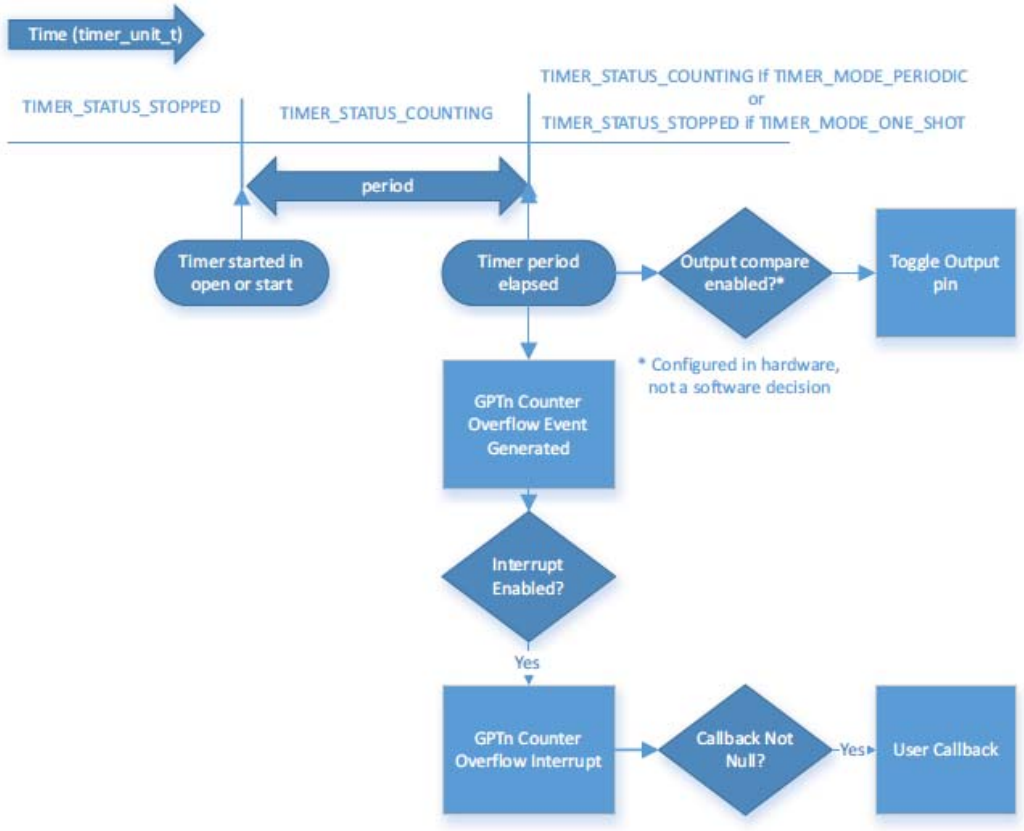


図 370: GPT タイマ - 周期またはワンショットモード

SSP では、GPT と AGT の 2 種類のタイマモジュールがサポートされます。以下のセクションでは、開発者が特定のアプリケーションに対する各モジュールの機能を比較して対照できるように、両方のモジュールについての情報を提供します。 AGT の追加情報については、『AGT ユーザーズガイド』を参照してください。

GPT モジュールは、ほとんどの汎用タイマアプリケーションに推奨されますが、基本的なタイマ機能にはどちらのモジュールでも使用できます。以下のユースケースでは、一方のタイマモジュールが他方より推奨される理由を説明します。

GPT タイマモジュールの選択

GPT モジュールは、PCLKD のみでクロック供給可能な、高解像度の 32 ビットカウンタを使用します。Synergy デバイスでは AGT チャネルより GPT チャネルの方が多いので、GPT を使用することでリソースが競合する可能性が低くなります。

AGT タイマモジュールの選択

AGT モジュールは、PCLKB、LOCO、または Fsub でクロック供給可能な 16 ビットカウンタを使用します。LOCO または Fsub でクロック供給された場合、AGT 割り込みを使用して MCU をスリープモードから起動できます。 2 つのチャネルがあり、チャネル 1 はチャネル 0 のアンダーフローによってクロック供給できるため、実質的に 32 ビットのカスケードされたタイマが作成されます。

GPT HAL モジュールの動作に関する重要な注意事項と制限事項

最大の時間間隔は、タイマの種類と入力クロック周波数に依存します。

- 120MHz で動作する PCLKD を使用する 32 ビット分解能の GPT では、最大期間は約 36650 秒であり、10 時間を少し超えます （AGT カウントクロックは PCLKD/1024 です）。
- 32MHz で動作する PCLKD を使用する 16 ビット分解能の GPT では、最大期間は約 2.09 秒です （AGT カウントクロックは PCLKD/1024 です）。
- 60MHz で動作する PCLKB を使用する 16 ビット分解能の AGT では、最大期間は約 8.7 ミリ秒です（AGT カウントクロックは PCLKB/8 です）。
- 32kHz で動作する Fsub を使用する 16 ビット分解能の AGT では、最大期間は約 2.0 秒です（AGT カウントクロックは AGTSCLK/128 です）。

以下の状況では、選択した使用チャネルに対する AGT カウンタアンダーフロー割り込みを BSP で有効にする必要があります。

1) タイマ期間が経過したらソフトウェア割り込みを取得するため。
2) ワンショットモードを使用するため。

BSP で AGTn AGTI 割り込みが有効になっている場合、対応する ISR がタイマドライバーで定義されます。ISR は、ユーザーコールバック関数が open で登録されている場合は、それを呼び出します。

注: *irq が TRANSFER_IRQ_END. に設定された DTC とともに使用した場合、割り込みがスキップされることがあります。*

GPT 出力タイマ信号

タイマ出力が設定されている場合（GTIOCA/B 出力有効に True が設定されている場合）、出力ピンは GTIOCA/B 停止レベルで開始され、期間が経過するたびに切り替わります。最初の切り替えは、タイマ起動後に初めて期間が経過したときに起こります。

ワンショットモードでは、タイマのカウントが開始されるときにも切り替わるように出力が設定されます。これによりパルスが生成されます。タイマは、カウントが始まるときに停止レベルから切り替わり、カウントが終了するときに停止レベルに戻ります。

タイマ期間の計算

タイマ期間は、タイマが期限切れになるまでの時間として定義されます。出力比較を使用する場合、出力ピンが期間あたり 1 回トグルされるため、従来の期間（立ち上がりエッジから立ち上がりエッジ）は、ソフトウェアで指定された期間の 2 倍になります。

現在のクロック設定に基づく実行時の時間間隔の計算は、 open および periodSet から使用できます。

指定されたタイマ時間間隔が raw カウントと異なる場合は、現在のタイマクロック周波数を使用して時間間隔が計算されます（「GPT クロックの設定」または 「AGT クロックの設定」を参照してください）。現在のタイマクロック周波数を確認するには、 systemClockFreqGet を使用します。この周波数は、次の表の適切な式で、 clk_freq_hz. として使用されます。

タイマ期間の計算

Timer Units	Formula
TIMER_UNIT_PERIOD_NSEC	Counts = (period * clk_freq_hz) / 1000000000

Timer Units	Formula
TIMER_UNIT_PERIOD_USEC	Counts = (period * clk_freq_hz) / 1000000
TIMER_UNIT_PERIOD_MSEC	Counts = (period * clk_freq_hz) / 1000
TIMER_UNIT_PERIOD_SEC	Counts = (period * clk_freq_hz)
TIMER_UNIT_FREQUENCY_HZ	Counts = (clk_freq_hz) / period
TIMER_UNIT_FREQUENCY_KHZ	Counts = (clk_freq_hz) / 1000 * period

必要な時間間隔がカウンタのサイズ（32 ビットまたは 16 ビット）より長い場合、ドライバーは結果がカウンタのサイズに収まる最も小さい除数を選択します。カウンタ値が、最大の除数 (1024) を持つカウンタサイズよりも大きい場合は、エラーコード (SSP_ERR_INVALID_ARGUMENT) が返されます。

GPT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_GPTn_COUNTER_OVERFLOW（n は GPT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、DMAC または DTC のガイドを参照してください。

***注:** DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマの時間間隔が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH, に設定するか、DMAC を使用して転送します。*

GPT を使用した ELC イベントのトリガー

GPT タイマは、他のペリフェラルの起動をトリガできます。ELC のガイドには、すべての使用可能なペリフェラルのリストがあります。

AGT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_AGTn_AGTI（n は AGT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、「転送インタフェース」を参照してください。

***注:** DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマの時間間隔が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH, に設定するか、DMAC を使用して転送します。*

AGT を使用した ELC イベントのトリガー

AGT タイマは、他のペリフェラルの起動をトリガできます。ELC のガイドには、elc_peripheral_t. に列記されているすべての使用可能なペリフェラルのリストがあります。詳細については、イベントとペリフェラルの定義を参照してください。

32 ビットタイマを作成するカスケードされた AGT タイマ

AGT チャネル 1 は AGT チャネル 0 のアンダーフローによってクロック供給することができ、カスケードされた 32 ビットタイマが作成されます。

GPT の電源オフの場合、GPT モジュールは closeAPI で GPT のモジュールストップビット（MSTP）を設定しません。これは意図的なもので、GPT モジュールストップビットは複数の GPT チャネルを制御し、GPT モジュールには他の GPT モジュールがアプリケーションで使用されているかどうかを判断することはできないた

めです。GPT にモジュールストップビットを設定し、使用中の GPT チャネルがない場合に電力消費量を削減するには、次の手順を使用します。

1) お使いの MCU のハードウェアマニュアルの「Low Power Modes」の章で、モジュールストップコントロールレジスタ D に関する「Register Descriptions」の章を確認します。
2) MSTPD5 および MSTPD6 を確認し、アプリケーションで使用するチャネルがどちらのビットに含まれているかを調べます。
3) Synergy 構成ツールの [Threads] タブでプロジェクトに LPM ドライバーを追加します。
4) [New] > [Driver] > [Power] > [Low Power Modes Driver on r_lpm.]。
5) アプリケーションコードで、手順 2 で特定した MSTP ビット内の他の GPT チャネルが、アプリケーションにより現在使用されていないことを確認します。使用中のチャネルがない場合は、 LPM API を使用して GPT チャネルセットの電源をオフにします。

   a) g_lpm0, という名前の LPM モジュールで MSTPD5 によって制御されている GPT チャネルの電源をオフにするには、次の関数を呼び出します。g_lpm0.p_api->moduleStop(LPM_MSTP_GPT_CH7_0); // 列挙値のチャネル番号は無視します – これは MSTPD5 用です

   a) MSTPD6 によって制御されている GPT チャネルの電源をオフにするには、次の関数を呼び出します。g_lpm0.p_api->moduleStop(LPM_MSTP_GPT_CH13_8); // 列挙値のチャネル番号は無視します – これは MSTPD6 です

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.19.3 アプリケーションへの GPT HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに GPT HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

GPT HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（タイマドライバーのデフォルト名は g_timer0 です。この名前は、対応する [Properties] ウィンドウで変更できます。）

GPT の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_timer0 Timer Driver on r_gpt	Threads -> HAL/Common	New Stack> Driver> Timers> Timer Driver on r_gpt

次の図に示すように、GPT HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

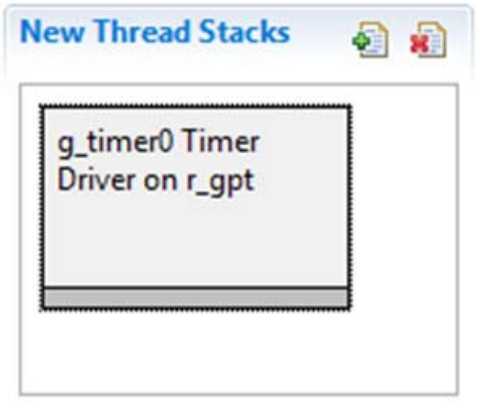


図 371: GPT HAL モジュールのスタック

5.2.19.4 GPT HAL モジュールの構成

ユーザーは必要な動作に合わせて GPT HAL モジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を設定するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これらのプロパティ設定は正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータの選択。

ISDE Property	Value	説明
Name	g_timer0	モジュール名。
Channel	0	物理ハードウェアチャネル。 S7G2 の場合は 0 ～ 13、S3A7 の場合は 0 ～ 9、S124 の場合は 0 ～ 6
Mode	Periodic, One Shot, PWM (Default: Periodic)	モードの選択。 ***注:*** *ワンショット機能は、GPT ハードウェアでは使用できません。そのため、期限が切れたときに呼び出される ISR 内でタイマを停止することにより、ソフトウェアで実装されています。そのため、ワンショットモードでは、コールバックを使用しない場合でも ISR を有効にする必要があります。*
Period Value	10	期間値の選択。前の「タイマ期間の計算」セクションを参照してください。
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz (Default: Milliseconds)	期間単位の選択。前の「タイマ期間の計算」セクションを参照してください。
Duty Cycle Value	50	デューティサイクル値の選択。
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 (Default: Unit Raw Counts)	デューティサイクル単位の選択。
Auto Start	True, FALSE (Default: True)	オートスタートの選択。構成後にタイマを開始するには、true に設定します。start が呼び出されるまでタイマを停止したままにするには、false に設定します。

ISDE Property	Value	説明
GTIOCA Output Enabled	True, False (Default: False)	GTIOCA 出力有効の選択。GPT 用に設定されたポートピン上でタイマ信号を出力するには、true を設定します。タイマ信号を出力しない場合は false を設定します。
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	タイマが停止したときの出力ピン レベルを制御します。
GTIOCB Output Enabled	True, False (Default: False)	GTIOCB 出力有効の選択。
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained (Default: Pin Level Low)	GTIOCB 停止レベルの選択。
Callback	NULL	コールバックの選択。ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、タイマの期間が経過するたびに割り込みサービスルーチン(ISR)から呼び出されます。
Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Disabled)	割り込み優先順位の選択。

***注:** 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、デフォルトとは異なるクロック分周器を選択すると役に立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

GPT HAL モジュールのクロック構成

GPT タイマは、PCLKD 周波数に基づいてクロック供給されます。PCLKD の周波数は、ISDE の [Configuring Clocks] タブで、または実行時に CGC インタフェースで、クロックコンフィギュレータを使用して設定できます。

GPT HAL モジュールのピン構成

GPT 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。最初の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では対応するピンの選択例を示します。

***注:** 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

GPT HAL ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
GPT	Pins	Select Peripherals > Timer: GPT>GPT0

***注:** 上記の選択シーケンスでは、GPT0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

GPT HAL ドライバーのピン構成設定

Property	Value	説明
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	ピンのグループマッピングを選択します。
Operation Mode	Disabled, GTIOCA or GTIOCB, GTIOCA and GTIOCB (Default: Disabled)	タイマ動作モードを選択します。
GTIOCA:	None, P300, P512 (Default: P512)	GTIOCA ピン。
GTIOCB:	None, P108, P511 (Default: P511)	GTIOCB ピン。

***注:** 前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.19.5 アプリケーションでの GPT モジュールの使用

アプリケーションで GPT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 GPT HAL モジュールを初期化します。
2) start API を呼び出して、 GPT HAL モジュールを開始します。
3) 必要に応じて、タイマコールバックに応答します（ユーザーコード）。

注: *アプリケーションのニーズに基づいて、GPT 期間とデューティサイクルのパラメータを再構成できます。*

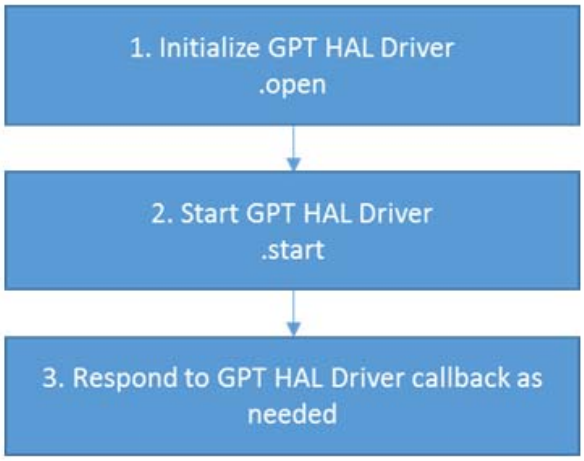


図 372: 一般的な GPT HAL モジュールアプリケーションのフローチャート

5.2.20 I2C SCI ドライバー

- I2C SCI 動作のサポート
- スレーブ I2C SCI デバイスに対し、次の動作のサポート
  - 読み取り
  - 書き込み
  - リセット
- コールバックのサポート
  - 転送中断
  - 送信完了（送信済みバイト数が提供されます）
  - 受信完了（受信済みバイト数が提供されます）

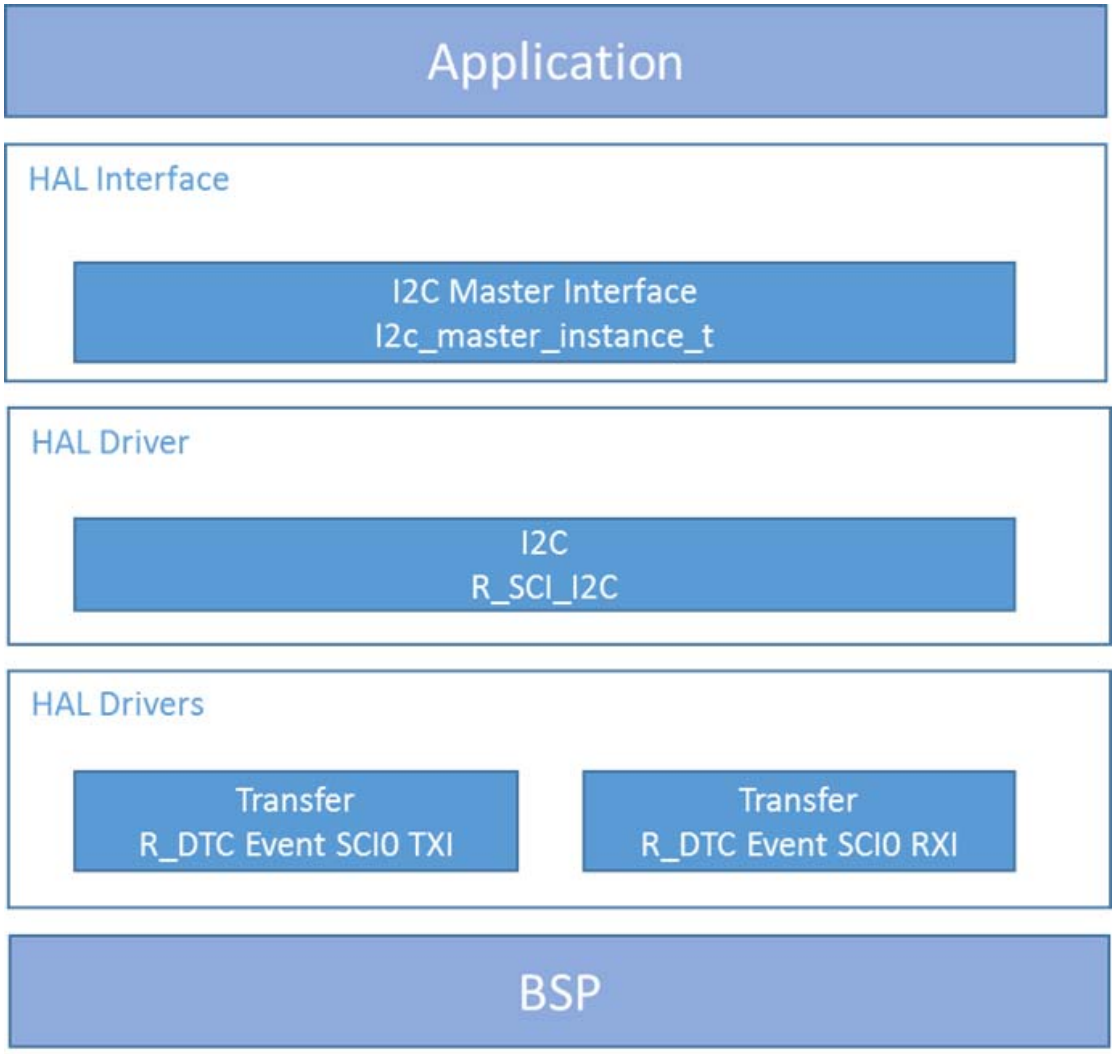


図 373: I2C SCI HAL モジュールのブロック図

5.2.20.1 I2C SCI HAL モジュールの API の概要

I2C SCI HAL モジュールでは、マスタ I2C デバイスを使用したリードとライトのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

I2C SCI HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); インスタンスを開き、ハードウェアを初期化します。
close	g_i2c.p_api->close(g_i2c.p_ctrl); ドライバーを閉じ、 I2C デバイスを解放します。
read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで読み取り操作を実行します。
write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで書き込み操作を実行します。
reset	g_i2c.p_api->reset(g_i2c.p_ctrl); ペリフェラルをリセットします。
versionGet	g_i2c.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの* API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_IN_USE	既に開いているデバイスインスタンスを開こうとしました。
SSP_ERR_ABORTED	転送が進行中にデバイスが閉じられました。

Name	説明
SSP_ERR_INVALID_RATE	要求されたレートを設定できません
SSP_ERR_ASSERTION	p_ctrl パラメータは NULL です。
SSP_ERR_NOT_OPEN	デバイスが開かれていません。
SSP_ERR_IRQ_BSP_DISABLED	イベント情報が見つかりませんでした。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの **API リファレンス** *を参照してください。*

5.2.20.2 I2C SCI HAL モジュールの動作の概要

I2C SCI HAL モジュールは、I2C スレーブデバイスとの通信をサポートします。送信が完了または中断したとき、または受信が完了したときに CPU に割り込むためのコールバックが提供されます。I2C SCI HAL モジュールは、バッファ内の受信バイト数または送信バイト数、ユーザーが指定したコンテキストへのポインタ、およびイベント 2c_event_t. を示す引数 i2c_callback_args_t を使用して、コールバックを呼び出します。

I2C SCI HAL モジュールの動作に関する重要な注意事項と制限事項

I2C SCI HAL モジュールの動作に関する注意事項

割り込み

- 選択したチャネルの I2C 割り込み （SCI エラー （EEI）、受信バッファフル （RXI）、送信バッファエンプティ （TXI）、および送信終了 （TEI））を、ユーザーがコールバックを使用するかどうかにかかわらず、ボードサポートパッケージ （BSP）で有効にする必要があります。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

IIC レート計算

- I2C SCI HAL モジュールは、構成されている転送レートに基づいて内部ボーレートの設定を計算し、それを open に渡します。現在の PCLKB 設定で達成可能な最も近いボーレート （要求されたレートより小さいか等しいもの）が計算されて使用されます。
- 有効なクロックレートを計算できなかった場合は、エラーが返されます。

IIC を使用した DMAC/DTC のトリガ

- DTC 転送のサポートは、コンフィギュレータにおいてデフォルトで追加されますが、CPU 転送を行う場合は削除できます。 DTC はモジュール内で構成されます。これに関してユーザーの構成は必要ありません。
- DMA 転送はサポートされていません。

IIC を使用した ELC イベントのトリガ

- I2C SCI HAL モジュールは、他のペリフェラルの開始をトリガできます。詳細については、『ELC User Guide』を参照してください。

バス上の複数のデバイス

- API slaveAddressSet() を使用すると、アプリケーションでバスを再構成することなく（閉じて開きなおす必要なしに）スレーブデバイスを切り替えることができ、同じバスで複数のスレーブデバイスを使用できます。
- 制御インスタンスとバスの構成は同じままですが、スレーブアドレスとアドレシングは変わります。
- 最初のデバイスを閉じて新しいデバイスを開くことにより、異なるスレーブデバイスと他の通信方法を使用できます。これは、スレーブの構成が異なる場合に推奨されます。
- 同じチャネルに接続された複数のデバイスを使用するアプリケーションでは、プロジェクトのプリプロセッサの設定で以下のマクロを定義する必要があります（定義しないと、プロジェクトが正しくビルドされない可能性があります）。

SSP_SUPPRESS_ISR_<device_name>

<device_name> は、同じチャネルに接続される追加デバイスの名前です。

IRQ モードの I2C SCI HAL モジュールは、特定のスレーブデバイスでは動作しない場合があります。そのようなデバイスでモジュールを動作させるには、 DTC 転送モードを有効にする必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.20.3 アプリケーションへの I2C SCI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2C SCI HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの手順に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な手順を管理する方法を確認してください。*

I2C SCI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（I2C SCI HAL ドライバーのデフォルト名は g_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

I2C SCI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Master Driver on r_sci_i2c	Threads	New Stack> Driver> Connectivity> I2C Master Driver on r_sci_i2c

次の図に示すように、r_sci_i2c の I2C SCI HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

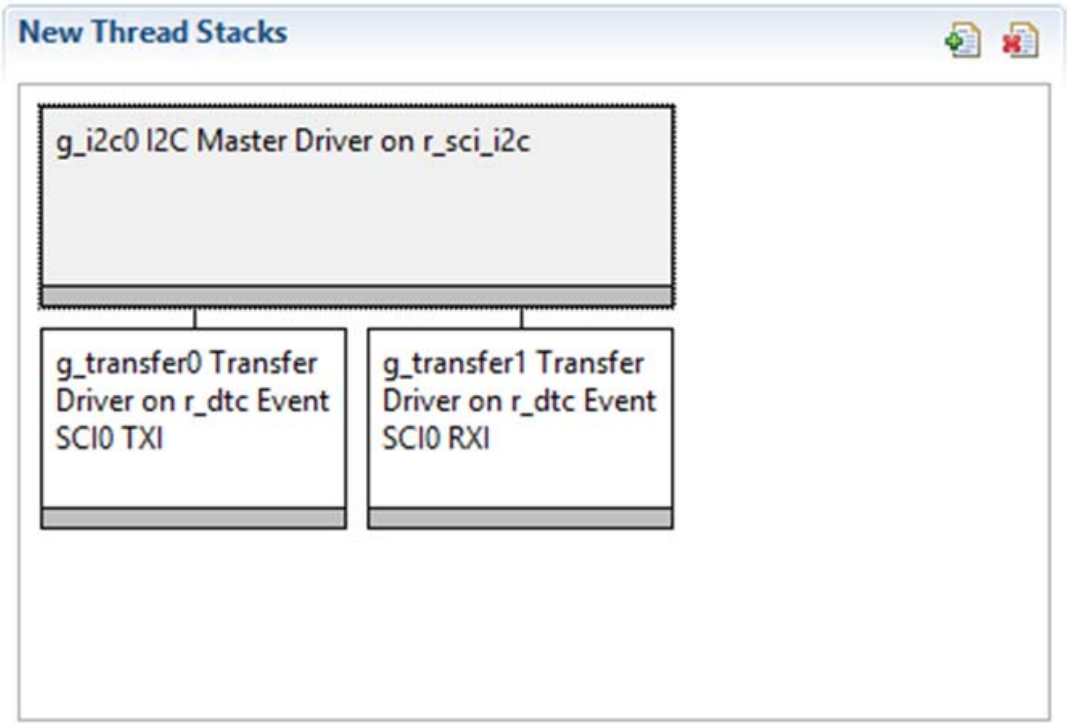


図 374: I2C SCI HAL モジュールのスタック

5.2.20.4 I2C SCI HAL モジュールの構成

必要な動作に合わせて I2C SCI HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータの [Properties] タブにあります。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。このステップは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_sci_i2c での I2C SCI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_i2c0	モジュール名
Channel	0	チャネル番号
Rate	Standard, Fast mode, Fast mode plus (Default: Standard)	レートの選択
Slave Address	0x00	スレーブアドレス
Address Mode	7-bit, 10-bit (Default: 7-bit)	アドレスモード
Callback	NULL	コールバック関数
Receive Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	受信割り込み優先順位
Transmit Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	送信割り込み優先順位
Error Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	エラー割り込み優先順位

***注:** 設定例とデフォルトは、Synergy S7G2 グループを使用するプロジェクトに対するものです。その他のMCUのデフォルト値と使用可能な構成設定は異なる可能性があります。*

スタックモジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なるスレーブアドレスやアドレスモードを選択するときに役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

***注：** モジュールのプロパティ設定の大半は、直観的であり、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。*

r_dtc イベント SCI0 TXI での転送ドライバー

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択

ISDE Property	Value	説明
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Disabled)	ELC イベントの割り込み優先順位

注: *設定例とデフォルトは、Synergy S7 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI での転送ドライバー

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled (Default: Disabled)	ソフトウェアスタートの選択

ISDE Property	Value	説明
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest) -15 (lowest), Disabled (Default: Disabled)	ELC イベントの割り込み優先順位

I2C SCI HAL モジュールのクロック構成

SCI 周辺モジュールは、PCLKA をそのクロックソースとして使用します。実際の I2C 転送レートは、ドライバーにより（選択された転送レートに応じて）計算されて内部的に設定されます。選択された内部レートを実現できないように PCLKB が構成されている場合は、ドライバーを初期化するときにエラーが返されます。

I2C SCI HAL モジュールのピン構成

SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能な周辺信号が決まり、それに従って I2C SCI HAL モジュールで必要な MCU ピンが決定されます。*

I2C SCI HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > Connectivity: SCI > SCIn

注: *選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

I2C SCI HAL モジュールのピン構成設定

Property	Value	説明
Pin Group Selection	_A only, _B only, Mixed (Default: _A only)	ピングループの選択
Operation Mode	Enabled, Disabled (Default: Disabled)	周辺モジュールを有効または無効にします
SDA	None, P401, P407 (Default: None)	SDA ピン
SCL	None, P400, P204 (Default: None)	SCL ピン

注: *設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.20.5 アプリケーションでの I2C SCI HAL モジュールの使用

アプリケーションで I2C SCI HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 I2C SCI HAL モジュールを初期化して開きます。
2) write API を使用して、スレーブにデータを転送します。
3) read API を使用して、スレーブからデータを受信します。
4) 受信したデータをアプリケーションの必要に応じて操作します。
5) reset API を使用して、インスタンスをリセットします（必要な場合）。
6) スレーブデバイスでトランザクションを実行します（必要な場合）。
7) close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

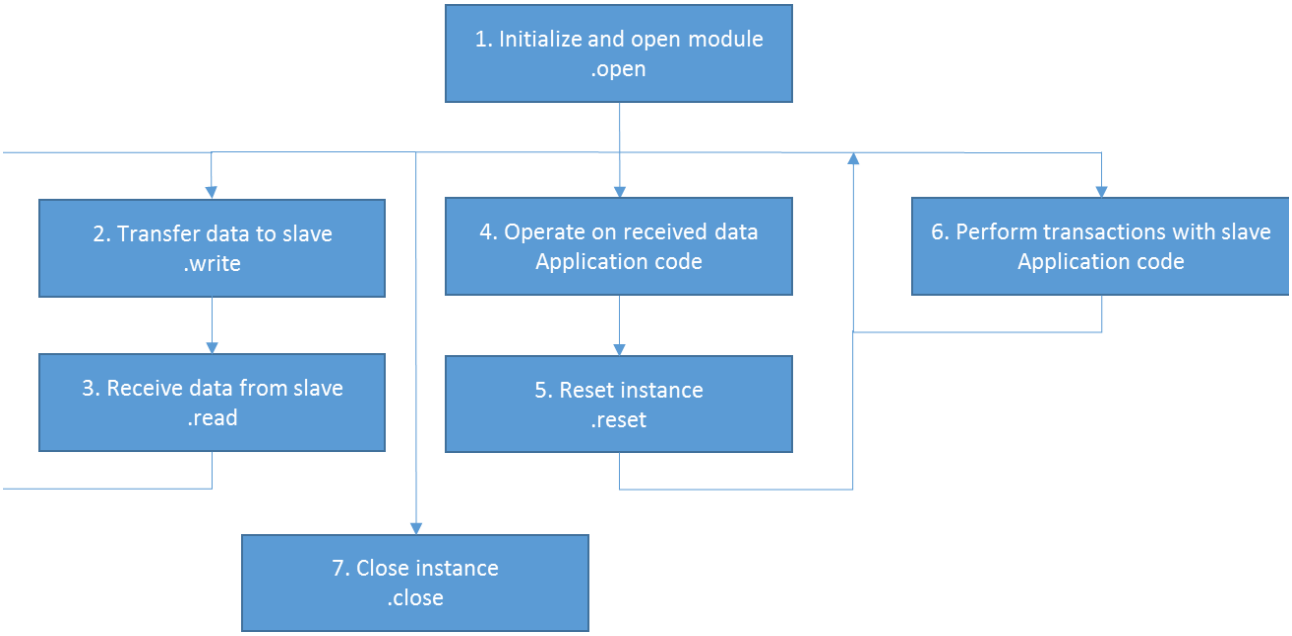


図 375: 一般的な I2C SCI HAL モジュールアプリケーションのフロー

5.2.21 I2C マスタドライバー

- I2C RIIC 動作のサポート
  - I2C ノーマルモード （最大 100Kbps）
  - I2C 高速モード （最大 400Kbps）
  - I2C 高速モードプラス （S7G2 および S5D9 MCU の選択チャネルで 1Mbps）
- RIIC モジュールの初期化
- スレーブデバイスへの書き込み
- I2C のリセット
- このインタフェースでは、コールバックもサポートしています。
- スレーブデバイスのアドレスの設定
- コールバックのサポート
  - 転送中断
  - 送信完了 （送信済みバイト数が提供されます）
  - 受信完了 （受信済みバイト数が提供されます）

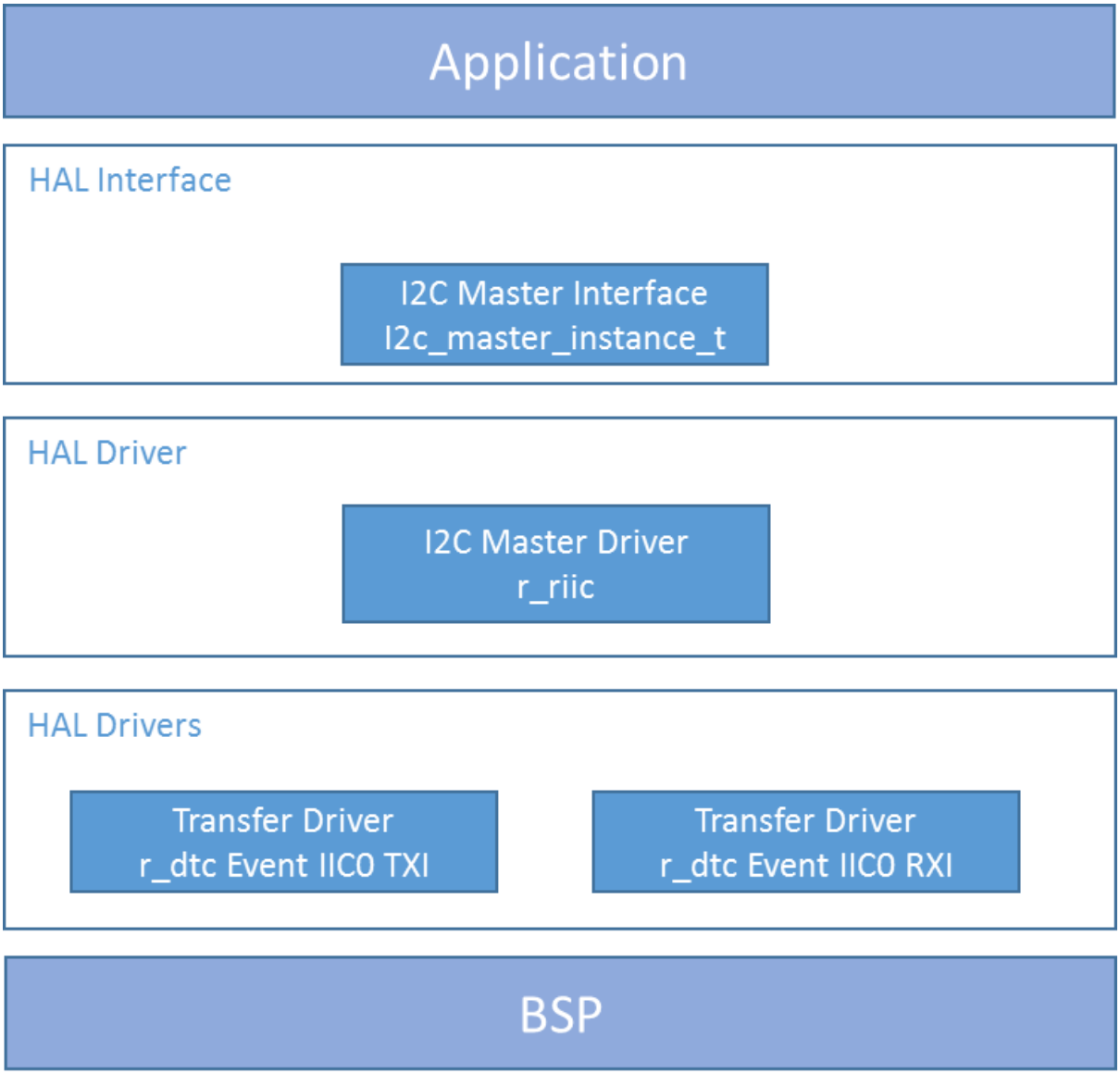


図 376: I2C マスタ HAL モジュールのブロック図

5.2.21.1 I2C マスタ HAL モジュールの API の概要

RIIC（I2C RIIC）上の I2C マスタ HAL モジュールでは、マスタ I2C デバイスを使用したリードとライトのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

I2C マスタ HAL モジュールの API の要約の表

Function Name	API の呼び出し例と説明
open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); インスタンスを開き、ハードウェアを初期化します。
close	g_i2c.p_api->close(g_i2c.p_ctrl); ドライバーを閉じ、 I2C デバイスを解放します。
read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで読み取り操作を実行します。
write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで書き込み操作を実行します。
reset	g_i2c.p_api->reset(g_i2c.p_ctrl); ペリフェラルをリセットします。
versionGet	g_i2c.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_POINTER	ポインタが NULL です
SSP_ERR_IN_USE	既に開いているデバイスインスタンスを開こうとしました。

Name	説明
SSP_ERR_ABORTED	転送が進行中にデバイスが閉じられました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です
SSP_ERR_INVALID_RATE	要求されたレートを設定できません

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.21.2 I2C マスタ HAL モジュールの動作の概要

RIIC 上の I2C マスタ HAL モジュールは、I2C スレーブデバイスとの通信をサポートします。送信または受信が完了したときに CPU に割り込むためのコールバックが提供されます。 RIIC HAL モジュールは、バッファ内の受信バイト数または送信バイト数、ユーザーが指定したコンテキストへのポインタ、およびイベント i2c_event_t. を示す引数 i2c_callback_args_t を使用して、コールバックを呼び出します。

I2C マスタ HAL モジュールの動作に関する重要な注意事項と制限事項

割り込み

- 選択した使用チャネルの RIIC エラー（EEI）、受信バッファフル （RXI）、送信バッファエンプティ（TXI）、および送信終了（TEI）割り込みを、ユーザーがコールバックを使用するかどうかにかかわらず、選択したデバイスのプロパティで有効にする必要があります。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

IIC レート計算

- I2C マスタモジュールは、構成されている転送レートに基づいて内部ボーレートの設定を計算し、それを open に渡します。現在の PCLKB 設定で達成可能な最も近いボーレート（要求されたレートより小さいか等しいもの）が計算されて使用されます。
- 有効なクロックレートを計算できなかった場合は、エラーが返されます。

IIC を使用した DMAC/DTC のトリガ

- DTC 転送のサポートは、コンフィギュレータにおいてデフォルトで追加されます。CPU 転送の場合は、これを削除できます。 DTC はモジュール内で構成されます。これに関してユーザーの構成は必要ありません。
- DMA 転送はサポートされていません。

IIC を使用した ELC イベントのトリガ

- I2C マスタモジュールは、他のペリフェラルの開始をトリガできます。詳細については、『ELC User Guide』のイベントとペリフェラルの定義を参照してください。

バス上の複数のデバイス

- 複数のデバイスが同じバスに接続されている場合、一度に開くことのできるデバイスは 1 つのみです。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.21.3 アプリケーションへの I2C マスタ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2C RIIC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

I2C マスタドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（I2C RIIC HAL モジュールのデフォルト名は g_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます）。

I2C マスタ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Master Driver on r_riic	Threads	New Stack> Driver> Communications> I2C Master Driver on r_riic

***注：** 次の図に示すように、r_riic の I2C RIIC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。*

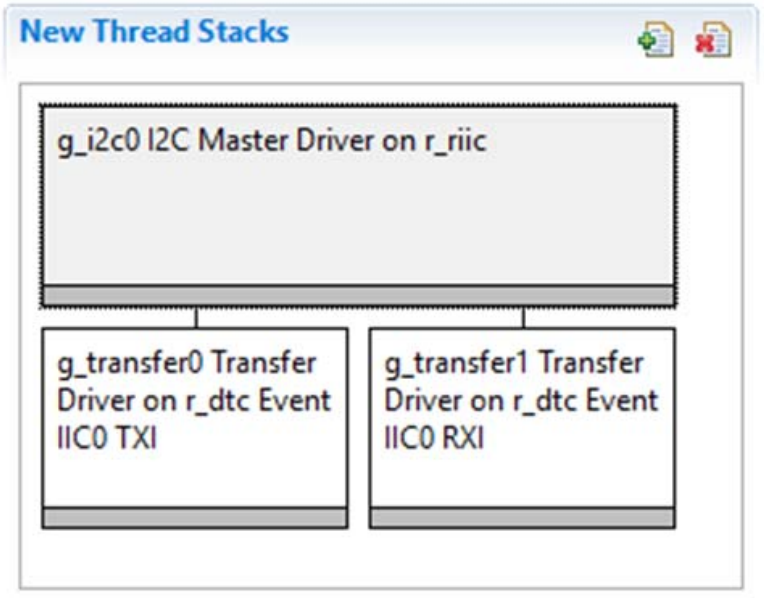


図 377: I2C マスタ HAL モジュールのスタック

5.2.21.4 I2C マスタ HAL モジュールの構成

必要な動作に合わせて I2C RIIC HAL モジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、 ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_riic 上の I2C マスタ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_i2c0	モジュール名。
Channel	0, 1, or 2	この設定で使用するIICチャネルを指定します。
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	標準、高速および高速プラス。（「IICレート計算」を参照。）
Slave Address	0x00	I2C マスターが通信するスレーブデバイスのアドレスを設定します。
Address Mode	7-Bit, 10-Bit	現在、7ビットアドレスのみがサポートされています。

ISDE Property	Value	説明
	Default: 7-Bit	
Callback	NULL	ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、 i2c_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン (ISR) から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキングコールを使用したり、長時間処理することはしないでください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信終了割り込み優先順位の選択。

ISDE Property	Value	説明
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	エラー割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント IIC0 TXI での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Enabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択

ISDE Property	Value	説明
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IICI0 TXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント IIC0 RXI での RTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択。
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクション。
Name	g_transfer1	モジュール名

ISDE Property	Value	説明
Mode	Normal	モードの選択
Transfer Size	1 Byte	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event IIC0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なるスレーブアドレスやアドレスモードを選択するときに役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

I2C マスタ HAL モジュールのクロック構成

IIC 周辺モジュールは、 PCLKB をそのクロックソースとして使用します。実際の I2C 転送レートは、選択された転送レートに応じて、ドライバーにより計算されて内部的に設定されます。選択された内部レートを実現できないような設定が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

I2C マスタ HAL モジュールのピン構成

IIC 周辺モジュールは、 MCU のピンを使用して外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

I2C マスタ HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
IIC	Pins	Select Peripherals > Connectivity: IIC > IIC0

注: *選択シーケンスでは、IIC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

I2C マスタ HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Pin Group Selection	_A only, _B only, Mixed (Default: _A only)	ピングループの選択
Operation Mode	Enabled, Disabled (Default: Disabled)	周辺モジュールを有効または無効にします
SDA	None, P401, P407 (Default: None)	SDA ピン
SCL	None, P400, P204 (Default: None)	SCL ピン

注: *設定例は、Synergy S7G2 MCU を使用するプロジェクトに対するものです。他の Synergy MCU では、使用可能なピン構成設定が異なる場合があります。*

5.2.21.5 アプリケーションでの I2C マスタ HAL モジュールの使用

アプリケーションで I2C RIIC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 I2C RIIC HAL モジュールを初期化して開きます
2) write API を使用して、スレーブにデータを転送します
3) read API を使用して、スレーブからデータを受信します

4) reset API を使用して、インスタンスをリセットします（必要な場合）

5) close API を使用して、チャネルを閉じます。

注: *アプリケーションで、バスを開いたり閉じたりすることなしにデバイスを切り替える必要がある場合は、slaveAddressSet API を使用します。g_i2c.p_ctrl は、最後に開いたデバイスで使用されていたものと同じ制御インスタンスです。モジュールは、同じバス構成を使用して新しいデバイスと通信します。このとき、新しいスレーブアドレスを設定することで、同じ制御インスタンスを使用してさまざまなスレーブデバイスと通信できます。また、同じ制御インスタンスでリード API やライト API を呼び出すことができます。*

スレーブデバイスと通信するためのこれらの一般的な手順を、次の図の通常の動作フローで示します。

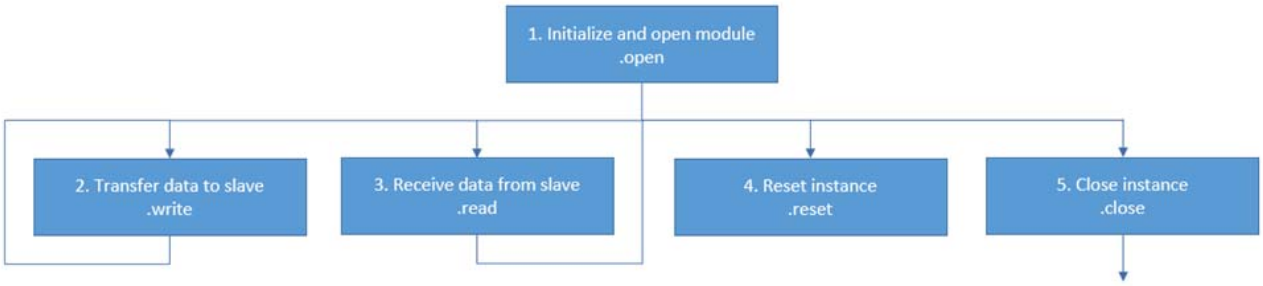


図 378: 一般的な I2C マスタ HAL モジュールアプリケーションのフロー図

5.2.22 I2C スレーブドライバー

- I2C スレーブ動作のサポート
- I2C マスタデバイスとの以下の通信のサポート
  - 読み取り
  - 書き込み
- コールバックのサポート
  - 送信完了（送信済みバイト数が提供されます）
  - 受信完了（受信済みバイト数が提供されます）

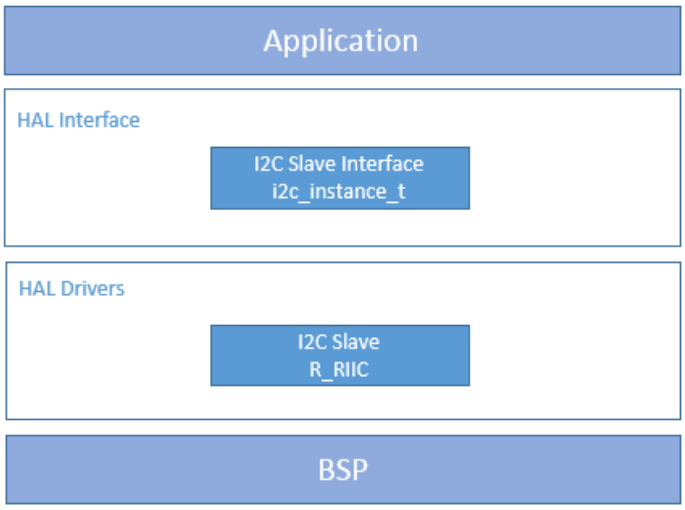


図 379: I2C RIIC スレーブ HAL モジュールの編成、オプション、スタックの実装

5.2.22.1 I2C スレーブ HAL モジュールの API の概要

I2C スレーブ HAL モジュールでは、マスタ I2C デバイスとのリードとライトのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

I2C スレーブ HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); インスタンスを開き、ハードウェアを初期化します。
.close	g_i2c.p_api->close(g_i2c.p_ctrl); ドライバーを閉じ、I2C デバイスを解放します。
.read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで読み取り操作を実行します。

Function Name	API の呼び出し例と説明
.write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); I2C デバイスで書き込み操作を実行します。
.reset	g_i2c.p_api->reset(g_i2c.p_ctrl); ペリフェラルをリセットします。
.versionGet	g_i2c.p_api->versionGet(&version); version ポインタを使用して、API のバージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの* API リファレンス *を参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_POINTER	ポインタが NULL です
SSP_ERR_IN_USE	既に開いているデバイスインスタンスを開こうとしました。
SSP_ERR_ABORTED	転送が進行中にデバイスが閉じられました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの* **API リファレンス** *を参照してください。*

5.2.22.2 I2C スレーブ HAL モジュールの動作の概要

I2C RIIC スレーブ HAL モジュールは、I2C マスタデバイスへの転送をサポートします。送信が完了したとき、または受信が完了したときに CPU に割り込むためのコールバックが提供されます。

I2C スレーブ HAL モジュールの動作に関する重要な注意事項と制限事項

- 選択した使用チャネルの RIIC エラー (EEI)、受信バッファフル (RXI)、送信バッファエンプティ (TXI)、および送信完了 (TEI) 割り込みは、ユーザーがコールバックを使用するかどうかにかかわらず、BSP で有効にする必要があります。

- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

これは I2C RIIC スレーブドライバーの最初のバージョンであり、基本的な機能だけが実装されています。以下の制限事項があることがわかっています。

以下のいずれかの動作が発生した場合、ドライバーは I2C バスをロックアップします。

- マスタが読み取っているバイト数が 「M」であるのに対し、スレーブが書き込むことのできるバイト数が 「N」であるとき。この場合、（M<N）または （M>N）です。
- マスタとスレーブの両方が同時にバスに書き込んでいるとき。
- マスタとスレーブの両方が同時にバスから読み取っているとき。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.22.3 アプリケーションへの I2C スレーブ HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに I2C RIIC スレーブ HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

I2C RIIC スレーブ HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（I2C RIIC スレーブ HAL モジュールのデフォルトの名前は g_i2c0 であり、これは次の表にも示されています。この名前は、対応する [Properties] ウィンドウで変更できます）。

I2C RIIC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Slave Driver on r_riic_slave	Threads	New Stack> Driver> Communications> I2C Slave Driver on r_riic_slave

次の図に示すように、r_riic_slave の I2C スレーブ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。

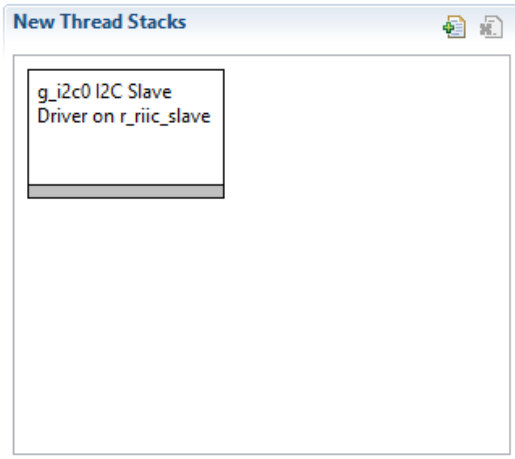


図 380: I2C スレーブ HAL モジュールのスタック

5.2.22.4 I2C スレーブ HAL モジュールの構成

必要な動作に合わせて、I2C RIIC スレーブ HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注:** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_i2c での I2C スレーブ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択

ISDE Property	Value	説明
Name	Default: g_i2c0	モジュール名
Channel	0	チャネル番号
Rate	Standard, Fast mode, Fast mode plus (Default: Standard)	レートの選択
Slave Address	0x00	スレーブアドレス
Address Mode	7-bit, 10-bit (Default: 7-bit)	アドレスモード
Callback	NULL	コールバック関数
Receive Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	受信割り込み優先順位
Transmit Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	送信割り込み優先順位
Error Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: 2)	エラー割り込み優先順位

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

スタックモジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、異なるスレーブアドレスやアドレスモードを選択するときに役立つ場合があります。

I2C スレーブ HAL モジュールのクロック構成

RIIC 周辺モジュールは、 PCLKB をそのクロックソースとして使用します。

I2C スレーブ HAL モジュールのピン構成

RIIC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表ではピンの選択例を示します。

注: *一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

I2C RIIC スレーブ HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
I2C	Pins	Select Peripherals > Connectivity: IIC > IIC0

注: 上記の選択シーケンスでは、IIC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_riic_slave での I2C RIIC スレーブ HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Pin Group Selection	_A only, _B only, Mixed (Default: _A only)	SCI 上の I2C の動作モードとして簡易 I2C を選択します
Operation Mode	Enabled, Disabled (Default: Disabled)	周辺モジュールを有効または無効にします
SDA	None, P401, P407 (Default: None)	SDA ピン
SCL	None, P400, P204 (Default: None)	SCL ピン

注: 前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.2.22.5 アプリケーションでの I2C スレーブ HAL モジュールの使用

アプリケーションで I2C RIIC スレーブ HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、I2C スレーブ HAL モジュールを初期化して開きます
2) write API を使用して、マスタにデータを転送します
3) read API を使用して、マスタからデータを受信します
4) close API を使用して、チャネルを閉じます。

上の一般的な手順を、次の図の通常の動作フローに示します。

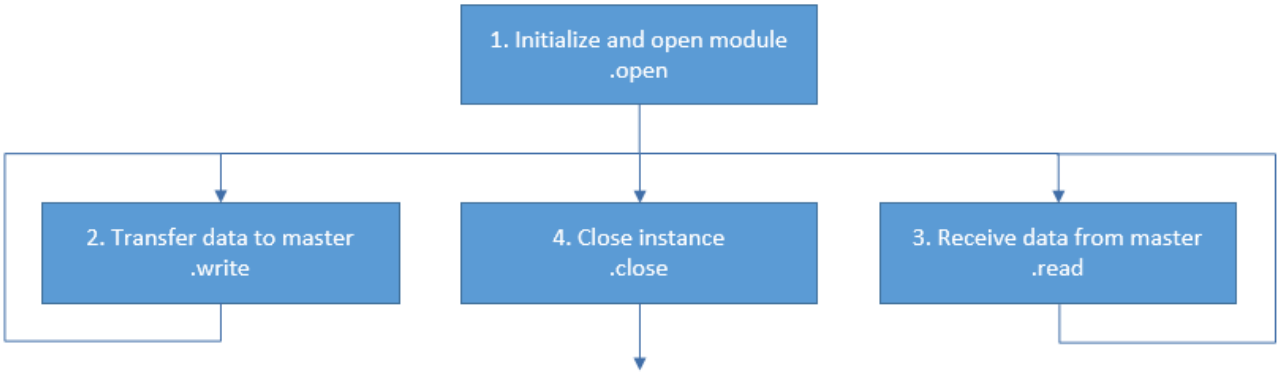


図 381: 一般的な I2C RIIC スレーブ HAL モジュールアプリケーションのフロー図

5.2.23 I2S ドライバー

I2S マスターモードの SSI で使用される I2S ドライバーは、標準的な I2S プロトコルに加えて、次の機能をサポートしています。

- 全二重 I2S 通信（SSI チャネル 0 のみ）
- 割り込み駆動型のデータ送信と受信
- DTC 転送モジュールとの統合。
- ユーザー定義のコールバックを作成し、追加データが必要な場合に対応できます。

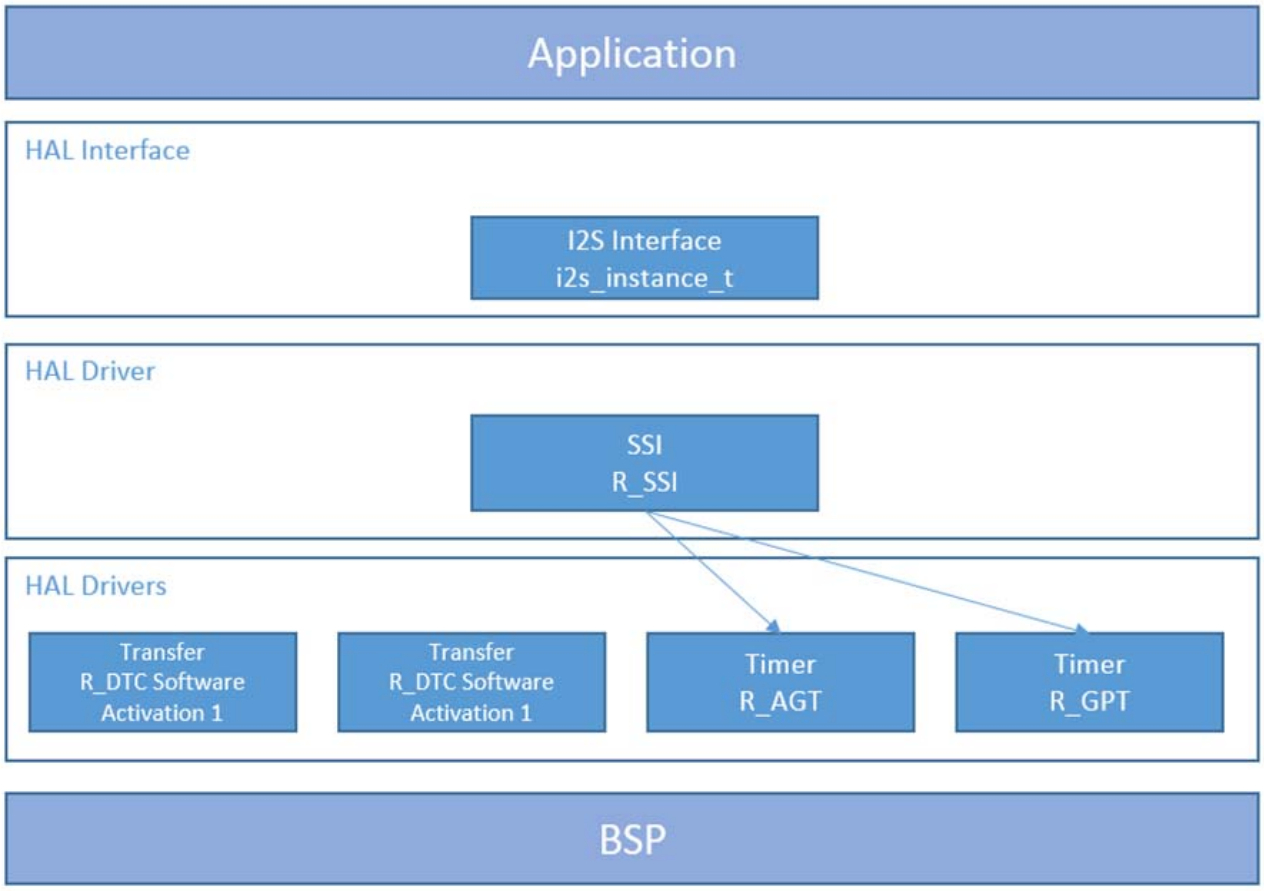


図 382: I2S HAL モジュールのブロック図

5.2.23.1 I2S HAL モジュールの API の概要

I2S HAL モジュールでは、オープン、ミュート、ライト、リードなどの動作のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

I2S HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_i2s0.p_api->open(g_i2s0.p_ctrl, g_i2s0.p_cfg); 初期設定。
stop	g_i2s0.p_api->stop(g_i2s0.p_ctrl, direction_to_stop); 通信を停止します。 I2S_EVENT_IDLE でコールバックを呼び出した場合、送信が停止します。 I2S_EVENT_RX_EMPTY でコールバックを呼び出した場合、受信が停止します。
mute	g_i2s0.p_api->mute(g_i2s0.p_ctrl, mute_enable); ミュートを有効化 / 無効化します。
write	g_i2s0.p_api->write(g_i2s0.p_ctrl, &data, bytes); I2S データを書き込みます。 I2S_EVENT_TX_EMPTY でコールバックを呼び出した場合、すべての送信データがキューイングされます。 I2S_EVENT_IDLE でコールバックを呼び出した場合、送信が完了します。
read	g_i2s0.p_api->read(g_i2s0.p_ctrl, &data, bytes); I2S データを読み取ります。 I2S_EVENT_RX_EMPTY でコールバックを呼び出した場合、受信が完了します。
writeRead	g_i2s0.p_api->writeRead(g_i2s0.p_ctrl, &source, &destination, bytes); I2S データの書き込みと読み取りを同時に行います。 I2S_EVENT_IDLE でコールバックを呼び出した場合、送信と受信が完了します。

Function Name	API の呼び出し例と説明
infoGet	g_i2s0.p_api->infoGet(g_i2s0.p_ctrl,&info); インスタンス固有の情報を取得し、指定されたポインタ info に格納します。
close	g_i2s0.p_api->close(g_i2s0.p_ctrl); ドライバーを再設定できるようになります。
versionGet	g_i2s0.p_api->versionGet(&version); バージョンを取得し、指定されたポインタ version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_OUT_OF_MEMORY	一度に開くストリームの数は、SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS に制限されています。この数値を超えると、メモリ不足のエラーが発生します。
SSP_ERR_TIMEOUT	再生が終了する前にタイムアウトが発生しました。
SSP_ERR_ASSERTION	この関数を呼び出す前に、呼び出し側が制御ハンドルをクリアする必要があります。
SSP_ERR_IN_USE	チャネルは実行中 / ビジーです
SSP_NOT_OPEN	要求したチャネルは構成されていないか、API が開かれていません

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.23.2 I2S HAL モジュールの動作の概要

I2S HAL モジュールは、I2S プロトコルを使用するオーディオ通信をサポートします。このドライバーは、I2S マスタモードの Synergy MCU 上の SSI をサポートします。圧縮されていないオーディオデータを送信および受信できます。全二重の I2S 通信（SSI チャネル 0 のみ）、割り込み駆動型のデータの送信と受信、および DTC 転送モジュールとの統合を提供します。

I2S HAL モジュールの動作に関する重要な注意事項と制限事項

チャネル 0 でオーディオデータを受信できるようにするには、SSI0 RXI 割り込みを有効にします。チャネル 0 でオーディオデータを送信できるようにするには、SSI0 TXI 割り込みを有効にします。チャネル 0 で送信も受信もできるようにするには、SSI0 TXI と SSI0 RXI の両方の割り込みを有効にします。チャネル 1 で送信または受信できるようにするには、SSIn TXI RXI 割り込みを有効にします。すべての場合に、SSIn INT 割り込みを有効にします。

割り込みが BSP で有効になっている場合、対応する ISR が I2S ドライバーで定義されます。ISR は、open で登録されたユーザーコールバック関数を呼び出します。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.23.3 アプリケーションへの I2S HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2S HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

I2S HAL ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（I2S HAL ドライバーのデフォルトの名前は g_i2s0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

I2S HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2s0 I2S Driver on r_ssi	Threads-> HAL/Common Stacks	New Stack> Driver> Connectivity> I2S Driver on r_ssi

次の図に示すように、r_ssi の I2S HAL モジュールが HAL/ 共通スタックまたはスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

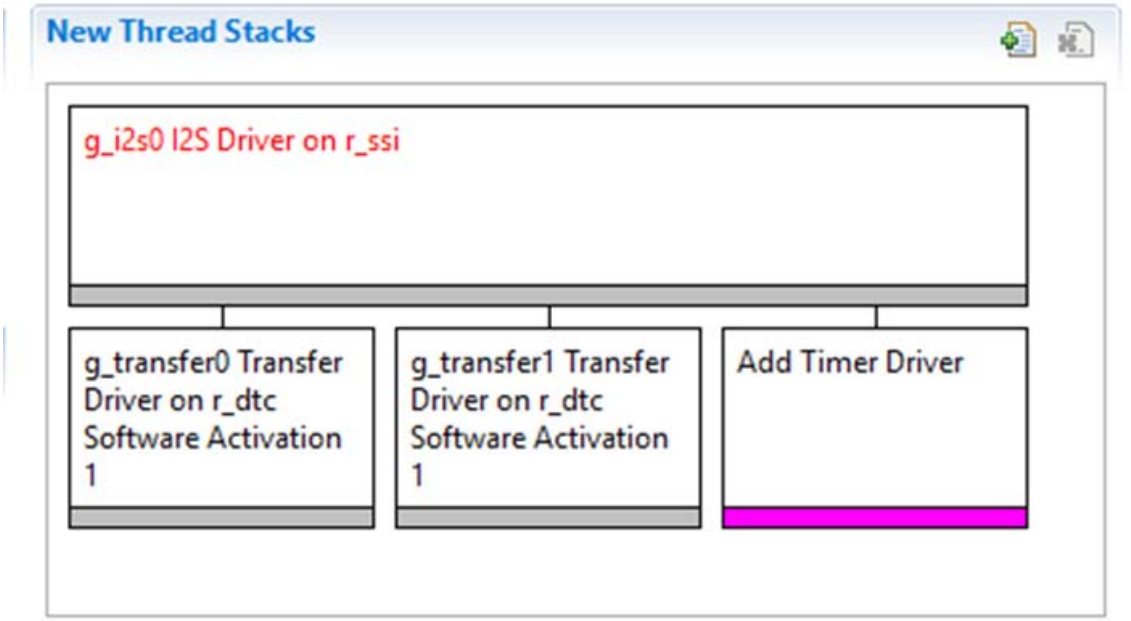


図 383: I2S HAL モジュールのスタック

5.2.23.4 I2S HAL モジュールの構成

ユーザーは必要な動作に合わせて I2S HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_ssi 上の I2S HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェックを有効化または無効化します。
Name	g_i2s0	モジュール名。
Channel	0	物理ハードウェアチャネル。
Audio Clock Frequency (Hertz)	2822400	I2S クロックの作成に用いられる入力オーディオクロック周波数。1 ～ 128(sampling_freq_hz* word_length_in_bits) の範囲で設定する必要があります。
Sampling Frequency (Hertz)	44100	オーディオデータのサンプリング周波数。
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	オーディオデータのビット深度（オーディオデータのサンプル 1 点のビットサイズ）。
Word Length	8 bits, 16, 24, 32 Default: 16 bits	オーディオデータのワード長、少なくともビット深度と同じサイズである必要があります（データビットフィールド）。
WS Continue Mode	Enabled, Disabled Default: Disabled	WS 続行モードを有効化すると、ペリフェラルがアイドル状態になった場合でも、ワード選択ラインの出力が継続して行われます。ペリフェラルがアイドル状態になった場合にワード選択ラインの出力を停止するには、これを無効化します。

ISDE Property	Value	説明
Name of I2S callback function to be defined by user	NULL	ユーザーコールバック関数はopenで登録する必要があります。全送信データの送信後に送信FIFO最高値に達した場合、あるいは受信が完了した（要求されたバイト数が受信された）場合に、割り込みサービスルーチン(ISR)からコールバックが呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信割り込み優先順位の選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	受信割り込み優先順位の選択
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	アイドル/エラー割り込み優先順位の選択

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルまたは I2S チャネルの選択が役立つ場合があります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *モジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

I2S HAL モジュールの低レベルモジュールの構成設定

I2S HAL モジュールで使用されるタイマドライバーの実装には 2 つの選択肢があり、それぞれのオプションで、関連するローレベルドライバーに必要な構成設定が異なります。これら 2 つのオプションの構成オプションを以下の表に示します。

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

表 5 r_dtc Software Activation での DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択。
Name	g_transfer0	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Fixed	宛先アドレスモードの選択。
Source Address Mode	Incremented	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。

ISDE Property	Value	説明
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

注: 設定例とデフォルトは、Synergy S7 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択。
Name	g_transfer1	ドライバー名。
Mode	Normal	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Incremented	宛先アドレスモードの選択。
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode)	Destination	リピートエリアの選択。
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択。

ISDE Property	Value	説明
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	0	転送回数の選択。
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択。
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

***注:** 設定例とデフォルトは、Synergy S7 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択

ISDE Property	Value	説明
Mode	Periodic	モードの選択
Period Value	2822400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Auto Start	False	オートスタートの選択
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSUB Default: PCLKB	クロックソースの選択
AGTO Output Enabled	True, False Default: False	AGTO 出力の選択
AGTIO Output Enabled	True, False Default: False	AGTIO 出力の選択
Output Inverted	True, False Default: False	出力の反転の選択
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

注: *設定例とデフォルトは、Synergy S7 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_gpt でのタイマドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択
Name	g_timer0	モジュール名
Channel	0	チャネルの選択
Mode	Periodic	モードの選択
Period Value	2822400 *2	期間値の選択
Period Unit	Hertz	期間単位の選択
Duty Cycle Value	50	デューティサイクル値の選択
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	デューティサイクル単位の選択
Auto Start	False	オートスタートの選択
GTIOCA Output Enabled	True, False Default: False	GTIOCA 出力有効の選択
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA 停止レベルの選択
GTIOCB Output Enabled	True, False Default: False	GTIOCB 出力有効の選択
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB 停止レベルの選択

ISDE Property	Value	説明
Callback	NULL	コールバックの選択
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

***注:** 設定例とデフォルトは、Synergy S7 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

I2S HAL モジュールのクロック構成

SSI モジュールでは、[Clock configuration] ウィンドウにある周辺クロック（PCLKB）を使用します。また、AUDIO_CLK ピンへの外部クロック入力も使用します。

I2S HAL モジュールのピン構成

SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。

選択した SSI チャネルに対し、SSI RX ピンと SSI TX ピンのどちらか一方または両方を構成します（[Pins] タブ > [Peripherals] > [SSI] > [SSIn] > [SSITXD0]/[SSIRXD0]）。チャネル 0 の場合は、これらのピンのうちいずれかまたは両方を有効化します。チャネル 1 の場合は、SSIDATA1 ピンを有効化します。

ワード選択およびクロックピンを設定します（[Pins] > [Peripherals] > [SSI] > [SSIn] > [SSITWSn および SSISCKn]）。このピンは大抵、どちらとも必要となります。必要なピンについては、使用する I2S デバイスのデータシートを確認してください。

オーディオクロックピンを SSI 用に設定します（[Pins] > [Peripherals] > [SSI] > [SSI0_SSI1_AUDIO_CLK]）。外部オーディオクロックをこのクロック入力ピンに接続します。オーディオクロックの作成に GPT タイマを使用する場合は、GPT タイマ出力ピンを設定し（[Pins] > [Peripherals] > [GPT1] > [GPTn] > [GTIOCx]）、使用する GPT 出力ピンをオーディオクロック入力ピンに接続します。

次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は関連するピンの選択例を示しています。

I2S HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
I2S	Pins	Select Peripherals > Connectivity:SSI > SSI0/SSI1

Resource	ISDE Tab	Pin selection Sequence
I2S	Pins	Select Peripherals > Connectivity:SSI > SSI0_SSI1_AUDIO_CLK

注: 選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

SSI での I2S ドライバーのピン構成設定

Pin Configuration Property	設定値	説明
Pin Group Selection	_A only, Mixed	I2S ポートのピングループ。
Operation Mode	Custom, Disabled	動作の選択。
SSISCK	None, P400 (Default: None)	AUDIO_CLK pin(P400)、このアプリケーションで使用されます

注：前の設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

SSI0 での I2S ドライバーのピン構成設定

Pin Configuration Property	設定値	説明
Pin Group Selection	_A only, _B only, Mixed	I2S ポートのピングループ。
Operation Mode	Enabled, Custom, Disabled	動作の選択。
SSISCK	None, P112 (Default: None)	SSISCK pin(P112)、このアプリケーションで使用されます
SSIWS	None, P113 (Default: None)	SSIWS pin(P113)、このアプリケーションで使用されます
SSITXD	None, P115 (Default: None)	SSITXD pin(P115)、このアプリケーションで使用されます

Pin Configuration Property	設定値	説明
SSIRXD	None,P114 (Default:None)	SSIRXD pin(P114)、このアプリケーションで使用されます

***注：** 前の設定例は、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

SSI1 での I2S ドライバーのピン構成設定

Pin Configuration Property	設定値	説明
Pin Group Selection	_A only, _B only, Mixed	I2S ポートのピングループ。
Operation Mode	Enabled, Custom, Disabled	動作の選択。
SSISCK	None, P204 (Default: None)	SSI シリアルクロック、このアプリケーションでは使用されません。
SSIWS	None, P205 (Default: None)	SSI ステレオピン選択、このアプリケーションでは使用されません。
SSIDATA	None, P206 (Default: None)	SSI データピン選択、このアプリケーションでは使用されません。

***注：** 前の設定例は、 Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.23.5 アプリケーションでの I2S HAL モジュールの使用

アプリケーションで I2S HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 I2S HAL モジュールを開きます。
2) write API を使用して、 I2S バスにオーディオデータを書き込みます。
3) I2S_EVENT_TX_EMPTY でのコールバックを待ち、ソースバッファを解放します。
4) read API を使用して、 I2S バスからデータを読み取ります。

5) I2S_EVENT_RX_FULL でのコールバックを待ってから、宛先バッファにアクセスするか、次のバッファを読み取ります。
6) 必要に応じて、アプリケーションで他の API を使用します。
7) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

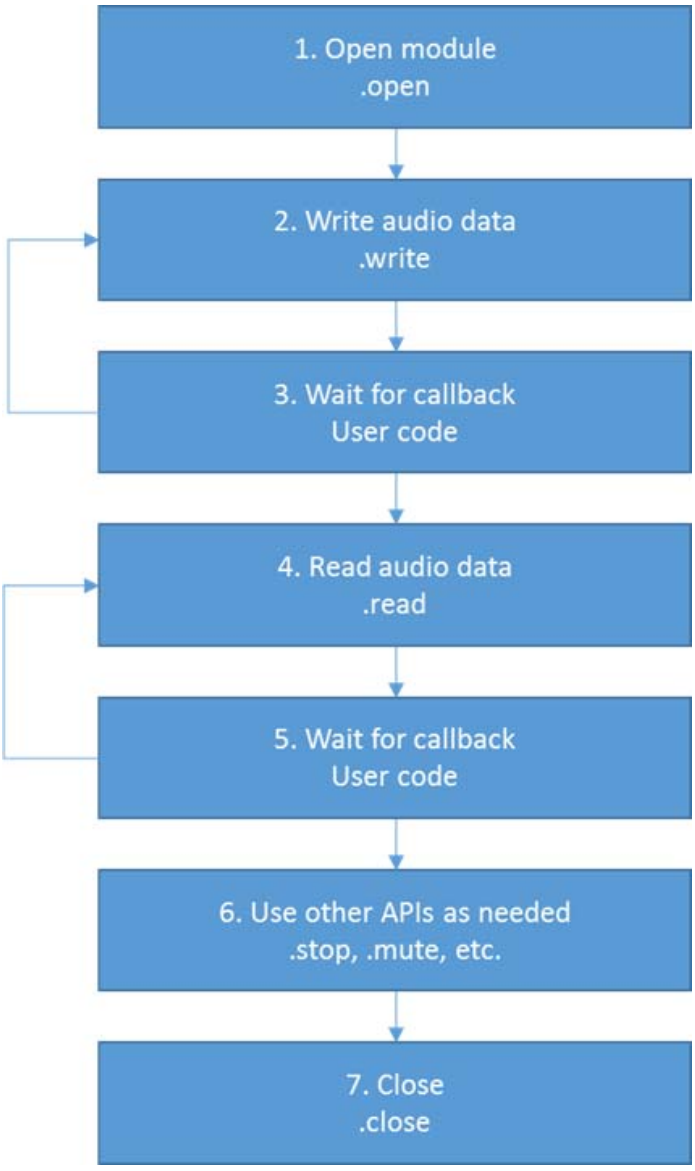


図 384: 一般的な I2S HAL モジュールアプリケーションのフロー図

5.2.24 入力キャプチャドライバー

入力キャプチャ HAL モジュールは、入力キャプチャ機能用の GPT を構成します。

- 入力キャプチャ HAL を使用すると、次のタスクを実行できます。
  - モジュールを初期化する
  - 入力キャプチャの測定を有効にする
  - 入力キャプチャの測定を無効にする
  - 測定カウンタの状態（実行中かどうか）を取得する
  - 最後にキャプチャされたタイマ / オーバーフローカウンタの値を取得する
  - 入力キャプチャ動作を閉じる
- 入力キャプチャ HAL モジュールは以下のものをサポートします。
  - パルス幅の測定のみ
  - 立ち上がりエッジまたは立ち下がりエッジの測定の開始
  - ワンショットモードまたは周期モード
  - キャプチャを有効にするためのハードウェアイネーブル信号（ローイネーブル / ハイイネーブル）
  - 以下のイベントでのコールバック関数 :
  - **カウンタオーバーフロー**
  - **入力キャプチャ発生**
- 割り込みイベントに関するデータを提供するコールバック構造体（input_capture_callback_args_t）。発生した割り込みおよび関連するカウンタ値を含みます。

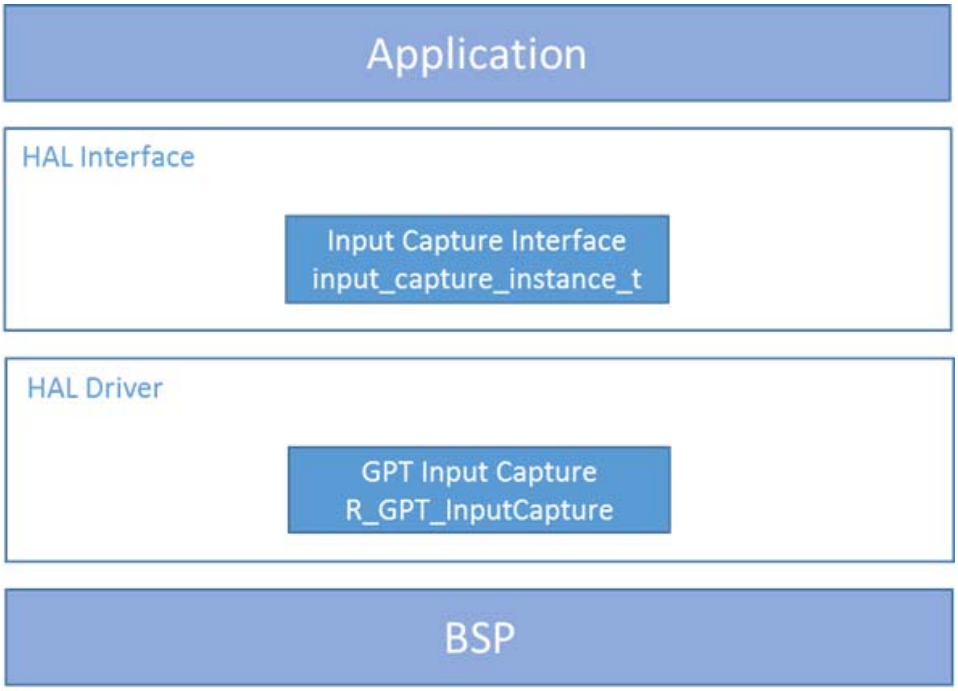


図 385:　入力キャプチャ HAL モジュールのブロック図

5.2.24.1 入力キャプチャ HAL モジュールの API の概要

入力キャプチャ HAL モジュールインタフェースでは、オープン、クローズ、有効化、無効化、状態情報へのアクセス、入力キャプチャでの汎用 PWM タイマ（GPT）を使用した最後のキャプチャ値へのアクセスのための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は、 HAL モジュールの API 要約の後にあります。

入力キャプチャ HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_input_capture.p_api->open(g_input_capture.p_ctrl, g_input_capture.p_cfg); 入力キャプチャ HAL を開き、構成を初期化します。
close	g_input_capture.p_api->close(g_input_capture.p_ctrl); 入力キャプチャ動作を閉じます。ドライバーを再構成して、電力消費を減らすことができます。

Function Name	API の呼び出し例と説明
enable	g_input_capture.p_api->enable(g_input_capture.p_ctrl); 入力キャプチャ測定を有効化します。
disable	g_input_capture.p_api->disable(g_input_capture.p_ctrl); 入力キャプチャ測定を無効化します。
infoGet	g_input_capture.p_api->infoGet(g_input_capture.p_ctrl, &input_capture_info); 測定カウンタのステータス（実行中かどうか）を取得します。
lastCaptureGet	g_input_capture.p_api->lastCaptureGet(g_input_capture.p_ctrl, &input_capture_counter); 最後にキャプチャされたタイマ / オーバーフローカウンタの値を取得します。
versionGet	g_input_capture.p_api->versionGet(&input_capture_version); input_capture_version ポインタで API のバージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『Synergy Software Platform（SSP）ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの 1 つが NULL であるか、p_cfg パラメータで要求されたチャネルが r_bsp_cfg.h, で選択されたデバイスで使用できない可能性があるか、または p_cfg -> mode が無効です。

Name	説明
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効であるか、または ISR が有効になっていません。
SSP_ERR_IN_USE	既に開いているデバイスインスタンスを開こうとしました。
SSP_ERR_NOT_OPEN	チャネルは開かれていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.24.2 入力キャプチャ HAL モジュールの動作の概要

入力キャプチャ HAL モジュールは、ユーザーの構成に従って、Synergy マイクロコントローラ上の GPT HAL モジュールユニットを制御します。RTOS 要素を使用しないで GPT ハードウェアに直接アクセスし、開発が簡単になる便利な API を提供します。

コールバックを使用できる場合は（割り込みが有効）、通常の測定が完了すると、入力キャプチャ HAL モジュールはコールバックを呼び出して引数 input_capture_callback_args_t を渡します。

引数 input_capture_callback_args_t では、チャネル、 input_capture_event_t イベント、割り込み発生時にキャプチャされたタイマの値、この測定の間に発生したカウンタオーバーフローの回数が示されています。

割り込みが有効になっていない場合は、API はメインループで最後にキャプチャされたタイマ / オーバーフローカウンタの値を取得します。

入力キャプチャ HAL モジュールの動作に関する重要な注意事項と制限事項

入力キャプチャ測定モードを取得する

入力キャプチャインタフェースでは、選択可能なモード、ワンショット測定、および周期的な測定が提供されています。GPT ハードウェアは、ワンショット機能をネイティブにはサポートしていません。タイマを停止およびクリアするためのソフトウェアサポートは割り込みサービスルーチン（ISR）内にあります。そのため、コールバックを使用しない場合でも、ワンショットモードのために ISR を有効にする必要があります。

GPT 入力キャプチャ信号

入力キャプチャ測定は、入力キャプチャ信号ピン（GTIOCA/GTIOCB）で入力キャプチャ信号エッジ（立ち下がりまたは立ち下がり）が検出され、かつイネーブル条件が満たされていると、開始されます。イネーブル条件は、イネーブルレベルによって定義されており、無効化するか（なし）、入力キャプチャイネーブルピン（GTIOCA/GTIOCB）のレベル（Low または High）を指定することができます。入力キャプチャイネーブルピンは、入力キャプチャ信号ピンとして使用されていないピンです。

測定カウントの時間への変換

測定が完了すると、未加工のカウントデータとオーバーフロー数がコールバック関数でユーザーに返されます。

必要に応じて、未加工の測定データをコールバックまたはユーザーのアプリケーション内で論理的な時間単位に変換できます。未加工のデータを変換するには、現在の PCLKD クロック周波数とプリスケーラの値、オ

ーバーフロー数、最大カウンタ値、測定カウントを考慮する必要があります。測定カウントとオーバーフロー数は、コールバック引数 input_capture_callback_args_t で提供されます。

現在の PCLKD 周波数を取得するには、systemClockFreqGet API を使用する方法が推奨されます。入力クロック周波数は PCLKD 周波数であり、プリスケーラの値で分周されます。次に示す「入力キャプチャの時間計算」の表では、 clk_freq_hz として表されます。

S7G2（全チャネル）、S3A7（全チャネル）、S124（チャネル 0）の最大カウンタ値は、0xFFFFFFFF です。S124（チャネル 1-6) の最大カウンタ値は 0xFFFF です。次の表では、この最大カウンタ値に 1 を加えた値（カウンタは 0 から始まるため）が max_counts として示されています。

入力キャプチャの時間計算

Desired Time Units	Formula
Nanoseconds (ns)	time_ns = ((overflows * max_counts) + counter) * 1000000000 / clk_freq_hz
Microseconds (µs)	time_ns = ((overflows * max_counts) + counter) * 1000000 / clk_freq_hz
Milliseconds (ms)	time_ns = ((overflows * max_counts) + counter) * 1000 / clk_freq_hz
Seconds (s)	time_ns = ((overflows * max_counts) + counter) / clk_freq_hz

- 現在、入力キャプチャ HAL モジュールがサポートしているのはパルス幅の測定だけです。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.24.3 アプリケーションへの入力キャプチャ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに入力キャプチャ HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

入力キャプチャドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（入力キャプチャドライバーのデフォルトの名前は g_input_capture. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

入力キャプチャ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_input_capture Input Capture Driver on r_gpt_input_capture	**Threads-> HAL/Common Stacks**	Highlight **Threads > HAL/Common Stacks** and select **New Stack > Driver > Timers > Input Capture Driver on r_gpt_input_capture**

次の図に示すように、r_gpt_input_capture の入力キャプチャ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは、スタンドアロンモジュールです。青い帯が表示されている項目は、共有モジュールまたは共通モジュールです。これらのモジュールは、1 回だけ追加すればよく、複数のスタックで使用できます。

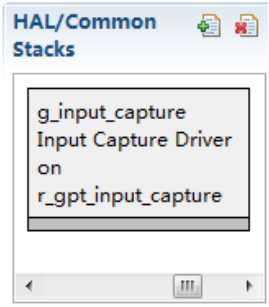


図 386:　入力キャプチャ HAL モジュールのスタック

5.2.24.4 入力キャプチャ HAL モジュールの構成

ユーザーは、必要な動作に合わせて入力キャプチャ HAL モジュールを構成します。[SSP configuration] ウィンドウでは、低レベルモジュールが正常に作動するために構成する必要のある必須の構成選択項目（割り込みや動作モードなど）が、自動的に識別されます（ブロックが赤で強調表示されます）。変更しても競合が発生しないプロパティのみが変更可能になります。「ロック」されているプロパティは、ISDE の [Properties] ウィンドウではロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。SSP コンフィギュレータの [Properties] タブには、ユーザーがアクセスできるすべての使用可能なプロパティが表示されます。簡単に参照できるように、次の表ではこれらのプロパティの構成設定とデフォルトを示してあります。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウにあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。[Properties] ウィンドウの割り込み優先順位では、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性も示されていることに注意してください。

***注:** 次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、入力キャプチャ HAL モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_gpt_input_capture での入力キャプチャ HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択します。
Name	(Default: g_input_capture)	モジュールの名前。
Channel	0-13 for S7G2, 0-9 for S3A7, 0-6 for S124 (Default: 0)	物理ハードウェアチャネル。
Mode	Pulse width	信号エッジから反対側のエッジまでの入力を測定します。
Signal Edge	Rising, Falling (Default: Rising)	立ち上がりまたは立ち下がりエッジで測定を開始し、反対側のエッジで測定を停止します。
Repetition	One Shot, Periodic (Default: Periodic)	信号測定をキャプチャした後、enable API が呼び出されるまでキャプチャを無効化する（ワンショット）か、継続的に測定をキャプチャします（周期）。
Auto Start	True, False (Default: True)	構成後に測定を有効化するには、true に設定します。enable API が呼び出されるまで測定を無効状態にするには、false に設定します。
Callback	User-defined, call with arguments (Default: NULL)	ユーザーコールバック関数を open API で登録する必要があります。コールバックは、タイマ期間が経過するたびに ISR から呼び出されます。 ***注：****コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*

ISDE Property	Value	説明
Input Capture Signal Pin	GTIOCA, GTIOCB (Default: GTIOCA)	測定の開始をトリガする入力ピンを選択します。
GTIOCx Signal Filter	None, PCLK/1, PCLK/4, PCLK/16, PCLK/64 (Default: None)	ノイズフィルタは、PCLK をいずれかの値で割った間隔で外部信号をサンプリングします。3つ連続したサンプルが同じレベル（高または低）の場合、そのレベルは信号の観測された状態として渡されます。
Clock Divider	PCLK/1, PCLK/4, PCLK/16, PCLK/64, PCLK/256, PCLK/1024 (Default: PCLK/1)	測定カウンタのスケールに用いられるクロック分周器。
Input Capture Enable Level	None, Low, High (Default: None)	各 GPT チャネルは、2 つの I/O ピン（GPIOCA と GPIOCB）を備えています。そのうちの 1 つを、入力キャプチャ信号ピンとして選択する必要があります。もう一方の GPT I/O ピンは、キャプチャを有効化するためのハードウェアイネーブル信号として用いることができます。[None] を選択すると、キャプチャは常に有効になります。[Low] を選択すると、キャプチャはイネーブル入力ピンが Low の間に限り有効になります。[High] を選択すると、キャプチャはイネーブル入力ピンが High の間に限り有効になります。
Input Capture Enable Filter	None (No filtering), PCLK/1 (Fast sampling), PCLK/4, PCLK/16, PCLK/64 (Slow sampling), (Default: None (No filtering))	イネーブルフィルターは、PCLK をいずれかの値で割った間隔でイネーブル信号をサンプリングします。3つ連続したサンプルが同じレベル（高または低）の場合、そのレベルは信号の観測された状態として渡されます。

ISDE Property	Value	説明
Capture Interrupt Priority/Overflow Interrupt Priority	Priority 0 (Highest), Priority 1-2, Priority 3 (CM4: valid, CM0+: lowest – not valid if using ThreadX), Priority 4-14 (CM4: valid, CM0+: invalid), Priority 15 (CM4: lowest – not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	割り込みの優先順位を指定します。

注: *表で示されている設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

入力キャプチャ HAL モジュールのクロック構成

GPT HAL モジュールは、 PCLKD をそのクロックソースとして使用します。 PCLKD の周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clocks] タブを使用するか、実行時に CGC インタフェースを使用します。

入力キャプチャ HAL モジュールのピン構成

特定のチャネルとピンにアクセスするには、ISDE の [Pins] タブで GTIOCx ピンを設定する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では GTIOCx ピンの選択例を示します。

入力キャプチャ HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
GPT Input Capture	Pins	Select **Peripherals > Timer: GPT > GPT0**

注: *選択シーケンスでは、 GPT0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

入力キャプチャ HAL モジュールのピン構成設定

Property	Value	説明
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	ピングループの選択

Property	Value	説明
Operation Mode	Disabled, GTIOCA or GTIOCB, GTIOCA and GTIOCB (Default: Disable)	GPT での入力キャプチャの動作モードとして、GTIOCA または GTIOCB を選択します。
GTIOCA	None, P300, P512 (Default: None)	GTIOCA ピン
GTIOCB	None, P108, P511 (Default: None)	GTIOCB ピン

注: *設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.24.5 アプリケーションでの入力キャプチャ HAL モジュールの使用

モジュールを構成してファイルの生成が済むと、入力キャプチャ HAL モジュールはアプリケーションで使用できる状態になります。アプリケーションで入力キャプチャ HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、モジュールを初期化します
2) 目的の値は、 lastCaptureGet API を使用してメインループルーチン内で、または p_args を使用してコールバック関数内で、見つけることができます。
3) キャプチャ割り込みは、 disable API を使用して無効にできます
4) キャプチャ割り込みとオーバーフロー割り込みは、 enable API を使用して有効にできます
5) キャプチャされているカウンタの状態（実行中または停止）は、 infoGet API を使用して照会できます
6) 終了したら、 close API を使用してモジュールを閉じることができます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

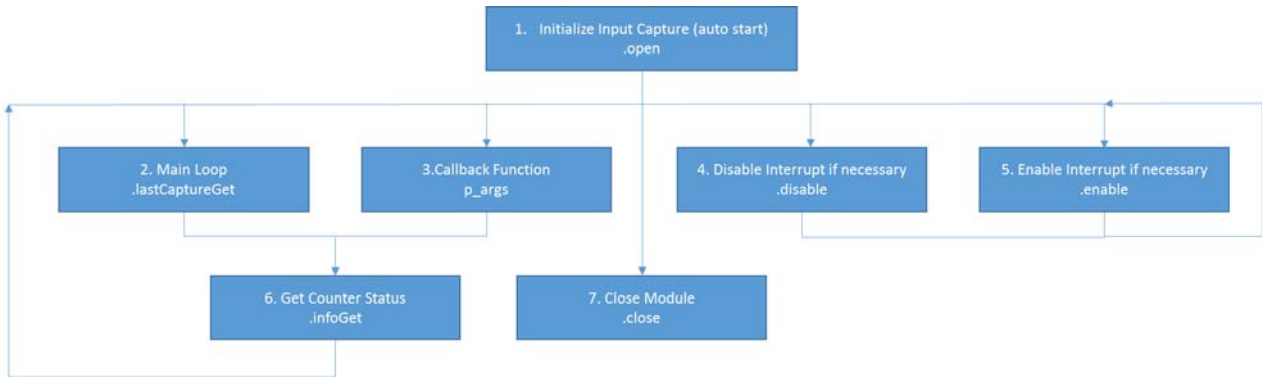


図 387: 通常の入力キャプチャ HAL モジュールアプリケーションのフロー図

5.2.25 I/O ポートドライバー

このモジュールは、1 つ以上の I/O ピンを構成します。ピンの方向は、以下に示す他のいくつかのオプションと共に設定できます。

- プルアップ
- NMOS/PMOS
- ドライブ強度
- イベントエッジトリガー（立ち下がり、立ち上がり、または両方）
- ピンを IRQ ピンとして使用するかどうか
- ピンをアナログピンとして使用するかどうか
- ピンをペリフェラルピンとして使用するかどうかと、そのペリフェラル

このモジュールは以下の機能も提供します。

- ポート上の 1 つ以上のピンの方向の変更
- ポート上の 1 つ以上のピンへのライト
- ポート上の 1 つ以上のピンからのリード
- イベント出力データの設定
- イベント入力データのリード

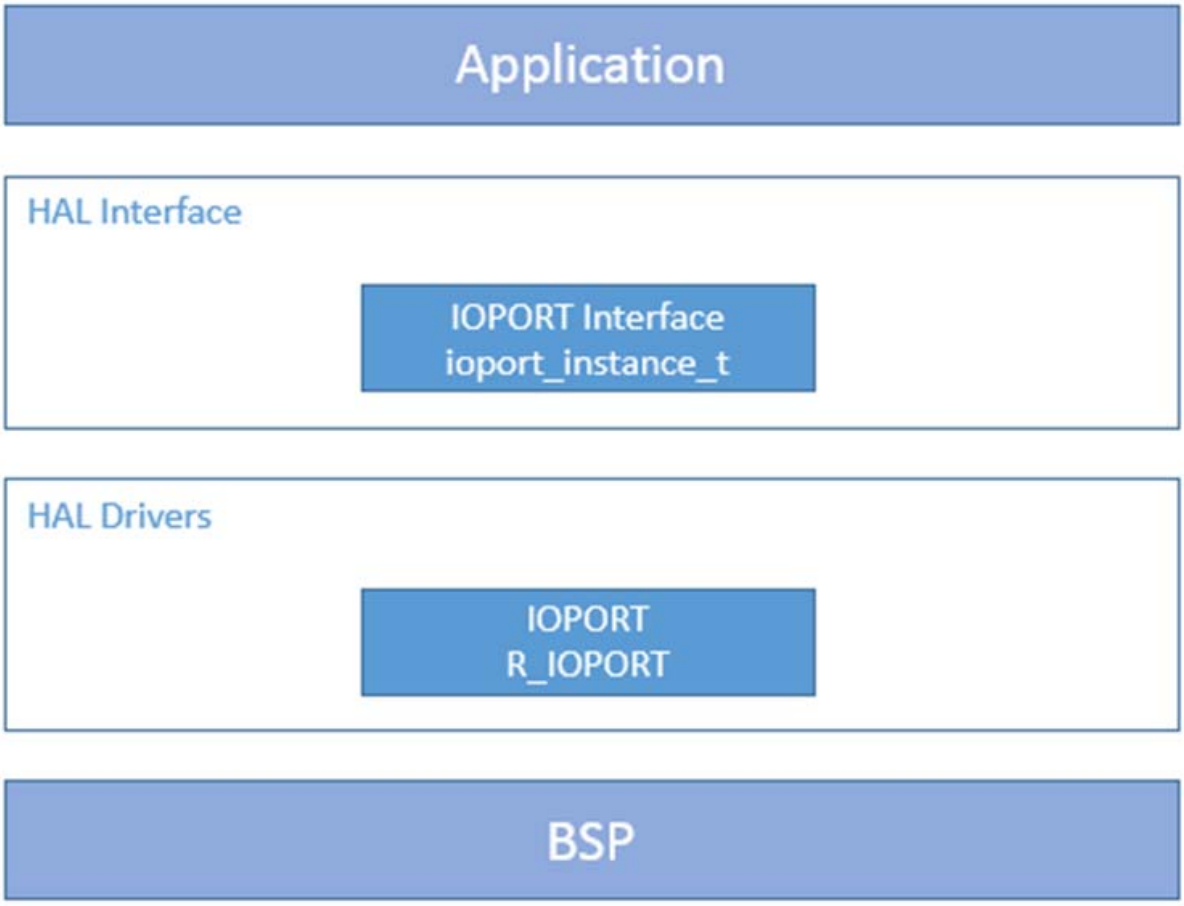


図 388: I/O ポート HAL モジュールのブロック図

5.2.25.1 I/O ポート HAL モジュールの API の概要

I/O ポート HAL モジュールでは、特定のピンとポートのリードおよびライトを行うための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。戻りステータス値の表は API 要約表の後にあります。

I/O ポート HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
init	g_ioport.p_api->init(g_ioport.p_cfg); 複数のピンの設定を初期化します。

Function Name	API の呼び出し例と説明
pinCfg	g_ioport.p_api->pinCfg(IOPORT_PORT_00_PIN_00, IOPORT_CFG_IRQ_ENABLE \ IOPORT_CFG_PORT_DIRECTION_INPUT); 1 本のピンの設定を指定します。
pinsCfg	g_ioport.p_api->pinsCfg(&pin_config); ピンのセットの設定を構成します。
pinDirectionSet	g_ioport.p_api->pinDirectionSet(IOPORT_PORT_00_PIN_00, IOPORT_DIRECTION_INPUT); ピンのピン方向を設定します。
pinEventInputRead	g_ioport.p_api->pinEventInputRead(IOPORT_PORT_00_PIN_00, &pin_level); 指定されたピンのイベント（ELC）入力データを読み取り、そのレベルを返します。
pinEventOutputWrite	g_ioport.p_api->pinEventOutputWrite(IOPORT_PORT_00_PIN_00, IOPORT_PIN_LEVEL_HIGH); ピンのイベント（ELC）データを書き込みます。
pinEthernetModeCfg	g_ioport.p_api->pinEthernetModeCfg(IOPORT_ETHERNET_CHANNEL_0, IOPORT_ETHERNET_MODE_MII); イーサネットチャネルの PHY モードを設定します。
pinRead	g_ioport.p_api->pinRead(IOPORT_PORT_00_PIN_00, &pin_level); ピンのレベルを読み取ります。

Function Name	API の呼び出し例と説明
pinWrite	g_ioport.p_api->pinWrite(IOPORT_PORT_00_PIN_00, IOPORT_PIN_LEVEL_HIGH); 指定されたレベルをピンに書き込みます。
portDirectionSet	g_ioport.p_api->portDirectionSet(IOPORT_PORT_00, direction_values, mask); ポート上の 1 つ以上のピンに対し、ピンの方向を設定します。
portEventInputRead	g_ioport.p_api->portEventInputRead(IOPORT_PORT_00, &pin_levels); ポートのキャプチャされたイベント（ELC）データを読み取ります。
portEventOutputWrite	g_ioport.p_api->portEventOutputWrite(IOPORT_PORT_00, pin_levels, mask); ポートのイベント（ELC）出力データを書き込みます。
portRead	g_ioport.p_api->portRead(IOPORT_PORT_00, &pin_levels); 指定されたポート上のピンの状態を読み取ります。
portWrite	g_ioport.p_api->portWrite(IOPORT_PORT_00, pin_levels, mask); ポート上の複数のピンに書き込みます。
versionGet	g_ioport.p_api->versionGet(&version); version ポインタを使用してバージョン情報を取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	ポート、ピン、マスク、方向、レベルなどが無効です。
SSP_ERR_ASSERTION	想定外の null ポインタです。
SSP_ERR_UNSUPPORTED	サポートされていない機能です。たとえば、イーサネット構成がデバイスでサポートされていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.25.2 I/O ポート HAL モジュールの動作の概要

I/O ポート HAL モジュールは、ビットとポートの両方のレベルでデバイスの I/O ポートにアクセスするための機能を提供します。ポートとピン両方の方向を変更できます。さらに、個々のピンの機能を変更するために数々の構成 API が提供されています。

I/O ポート HAL モジュールでは、ピンを構成するために以下の動作が提供されています。

- ドライバーを初期化します – init を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します。
  - VBATT ドメインのピン構成を処理します。
  - ピンの PFS レジスタを書き込みます。
- ピンを構成します – pinCfg API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin、 VBATT のサポートのチェック）。
  - ピンの PFS レジスタに書き込みます。
- ピンレベルを読み取ります – pinRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin のチェック）。
  - ピンの PFS レジスタを読み取ります。
- ポートのすべてのピンレベルを読み取ります – portRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port のチェック）。
  - 指定されたポートの PCNTR レジスタ値の現在の値を読み取ります。
- ピンレベルを書き込みます – pinWrite API を呼び出すことによって実行されます。

  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれるレベル level のチェック）。
  - ピンの PFS レジスタに書き込みます。
- ポートの複数のピンレベルを書き込みます – portWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、スコープ外のピンレベルを保持します。
- ポートの複数のピンの方向を設定します – portDirectionSet API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、スコープ外のピンの方向を保持します。
- ピンの方向を書き込みます – pinDirectionSet API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれる方向 direction のチェック）。
  - ピンの PFS レジスタに書き込みます。
- イベント（ELC）ポートの入力を読み取ります – portEventInputRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port のチェック）。
  - 指定されたポートの PCNTR レジスタ値の現在の値を読み取ります。
- イベント（ELC）ピンの入力を読み取ります – pinEventInputRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin のチェック）。
  - 指定されたピンのポートの PCNTR レジスタ値の現在の値を読み取ります。
  - PCNTR レジスタの値にピンマスクを適用することにより、ピンレベルを取得します。
- イベント（ELC）ポートの出力を書き込みます – portEventOutputWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask_value). のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、イベントスコープ外のピンレベルを保持します。

- イベント（ELC）ピンの出力を書き込みます – pinEventOutputWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれるレベル pin_value). のチェック）。
  - イベントスコープ外のピンレベルを保持するマスクに従って指定されたピンのポートの PCNTR レジスタにピンレベルを書き込みます。
- イーサネットチャネルの PHY モードを構成します – ethernetModeCfg API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（イーサネットチャネル channel およびモード mode のチェック）。
  - イーサネットコントロールレジスタ（PFENET）を更新します。

I/O ポート HAL モジュールの動作に関する重要な注意事項と制限事項

- ポートの特定のピンのリードやライトを行うには、16 ビットのビットマスクを適用する必要があります。ポートには、0（LSB）から 15（MSB）までの番号が付いています。
- イーサネットの構成は、一部のデバイスではサポートされていない場合があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.25.3 アプリケーションへの I/O ポート HAL モジュールの組み込み

e^2 studio 統合ソリューション開発環境（ISDE）では、必要な I/O ポートスタックがプロジェクトの HAL/ 共通スレッドに自動的に追加され、スタックはデフォルトで動作可能な状態になります。I/O ポートドライバーが意図せず削除された場合は、以下の方法でドライバーを HAL/ 共通スレッドに追加できます。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

I/O ポート HAL ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してプロジェクトスレッドに単純に追加します。(I/O ポートのデフォルト名は g_ioport.)

I/O ポート HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ioport I/O Port driver on r_ioport	Threads	Highlight HAL/Common and select New > Driver > System > I/O Port Driver on r_ioport

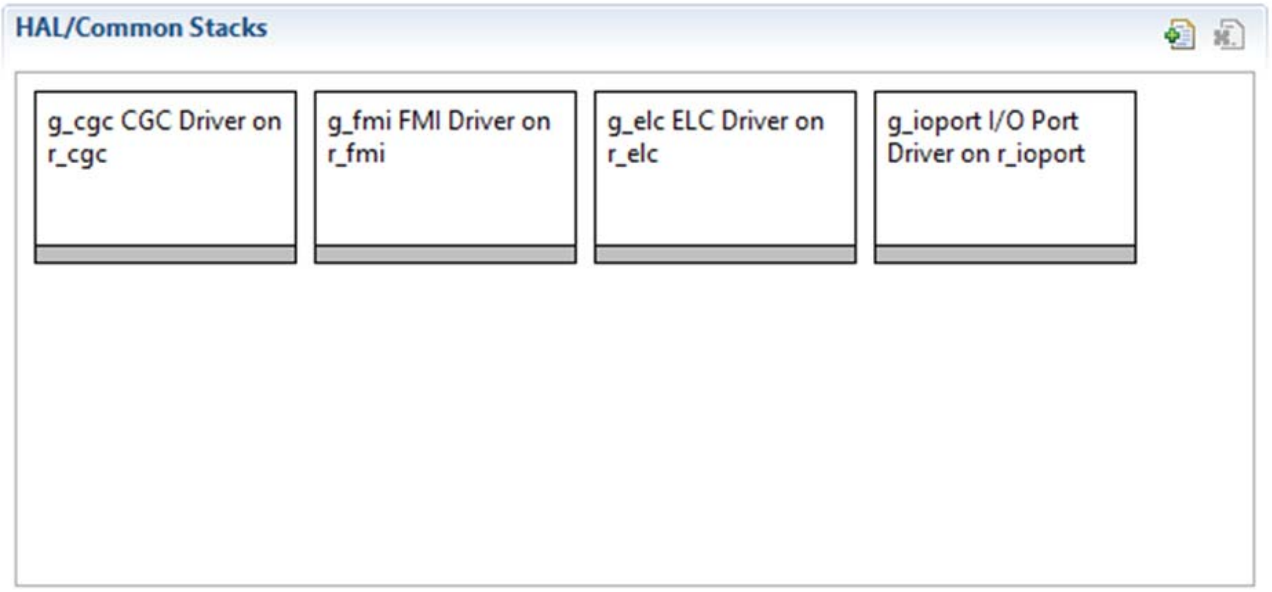


図 389: I/O ポート HAL モジュールのスタック

他の低レベルモジュールをさらに追加または構成する必要はありません。

5.2.25.4 I/O ポート HAL モジュールの構成

このモジュールに必要な他の構成はありません。すべてのピン構成は、 BSP に従って src/synergy_gen/pin_data.c ファイルで生成されます。

5.2.25.5 アプリケーションでの I/O ポート HAL モジュールの使用

アプリケーションで I/O ポート HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して、ドライバーを初期化します。
2) pinCfg API を使用して、ピンを構成します
3) pinRead または portRead API を使用して、指定したピンおよびポートから読み取ります。
4) pinWrite または portWrite API を使用して、指定したピンおよびポートに書き込みます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

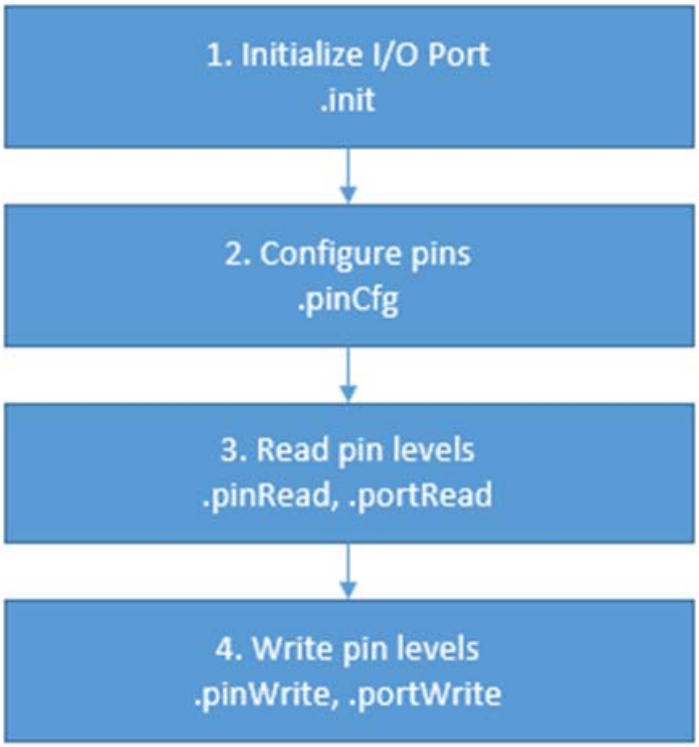


図 390: 通常の I/O ポート HAL モジュールアプリケーションのフロー図

5.2.26 独立ウォッチドッグドライバー

独立ウォッチドッグタイマ（IWDT）HAL モジュールは、ウォッチドッグタイマアプリケーションのための高レベル API であり、r_iwdt. に実装されています。ウォッチドッグタイマは、Synergy MCU の IWDT を使用します。ユーザー定義のコールバックを作成し、イベント通知に応答できます。

5.2.26.1 IWDT HAL モジュールの機能

IWDT HAL モジュールには、以下の主要な特長があります。

- WDT が許可されたリフレッシュウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。
  - デバイスのリセット
  - NMI の生成
- 独立ウォッチドッグタイマ（IWDT）をサポートします。IWDT は独自のクロックソースを持ち、安全性が向上します。
- リセット後の自動ハードウェア構成をサポートします。

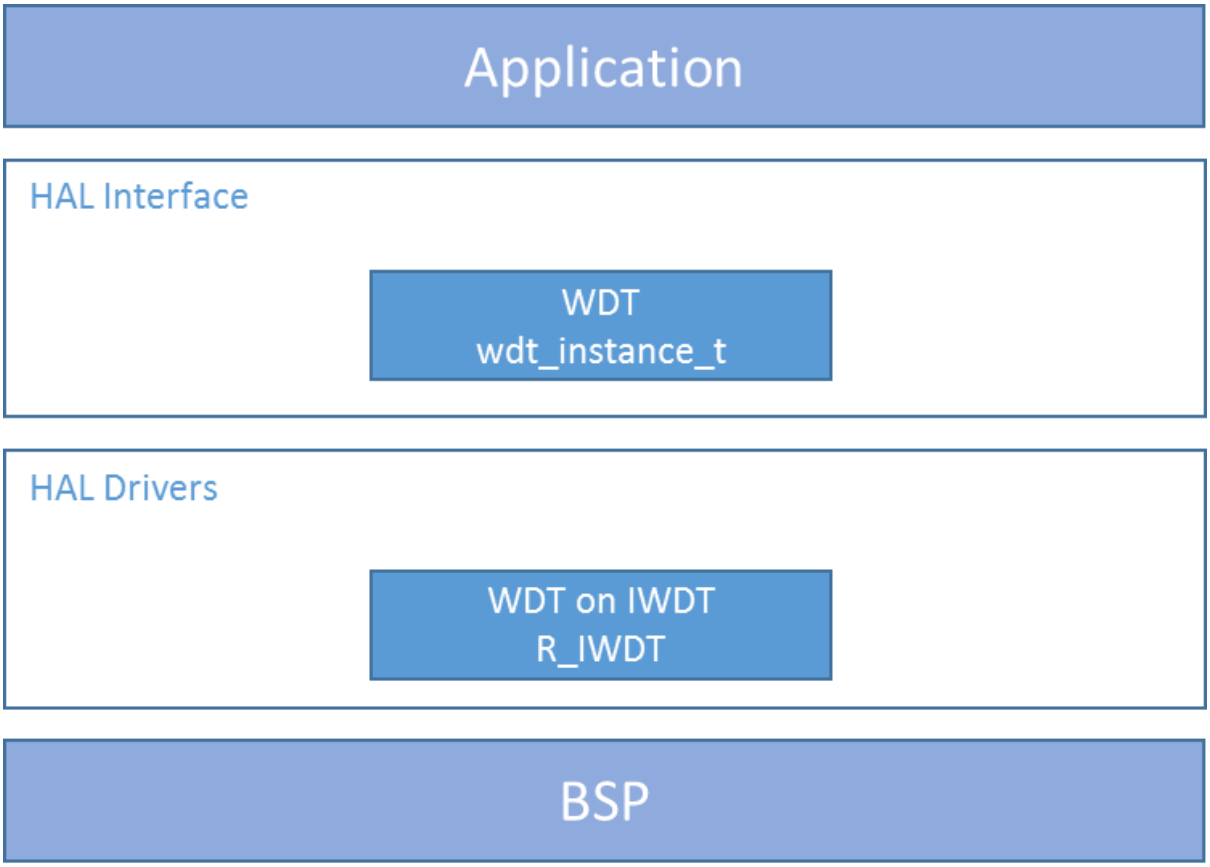


図 391: IWDT HAL モジュールの編成とスタックの実装

5.2.26.2 IWDT HAL モジュールの API の概要

IWDT HAL モジュールでは、状態のオープン、リフレッシュ、リード、取得のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

IWDT HAL モジュールタイマの API の要約

Function Name	API の呼び出し例と説明
cfgGet	g_wdt0.p_api->cfgGet(g_wdt0.p_ctrl, g_wdt0.p_cfg); iWDT をレジスタスタートモードで初期化します。NMI 出力を使用したオートスタートモードの場合は、NMI コールバックが登録されます。

Function Name	API の呼び出し例と説明
open	g_wdt0.p_api->open(g_wdt0.p_ctrl, g_wdt0.p_cfg); iWDT をレジスタスタートモードで初期化します。NMI 出力を使用したオートスタートモードの場合は、NMI コールバックが登録されます。
refresh	g_wdt0.p_api->refresh(g_wdt0.p_ctrl); ウォッチドッグタイマをリフレッシュします。
statusGet	g_wdt0.p_api->statusGet(g_wdt0.p_ctrl, &status); iWDT の状態を読み取ります。
statusClear	g_wdt0.p_api->statusClear(g_wdt0.p_ctrl, clear); iWDT の状態フラグをクリアします。
counterGet	g_wdt0.p_api->counterGet(g_wdt0.p_ctrl, &counter); 現在の iWDT カウンタの値を読み取ります。
timeoutGet	g_wdt0.p_api->timeoutGet(g_wdt0.p_ctrl, &timeout); ウォッチドッグのタイムアウト値を読み取ります。
versionGet	g_wdt0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	NULL ポインタ。
SSP_ERR_INVALID_ARGUMENT	1 つまたは複数の構成オプションが無効です。
SSP_ERR_INVALID_MODE	WDT をレジスタスタートモードで開こうとしましたが、OFS0 レジスタがオートスタートモードに設定されています。または WDT をオートスタートモードで開こうとしましたが、OSF0 レジスタがレジスタスタートモードに設定されています。
SSP_ERR_ABORTED	このウォッチドッグのクロック分周器が無効です

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.26.3 IWDT HAL モジュールの動作の概要

IWDT HAL モジュールは、IWDT インタフェースを構成します。IWDT が許可されたリフレッシュウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。

- デバイスのリセット
- NMI の生成

次の図は、IWDT の動作の例を示しています。カウンタの有効なリフレッシュ周期でリフレッシュされた場合、タイマカウントの値はリセットされます。カウントでアンダーフローまたは有効なリフレッシュ期間外のリフレッシュ発生が許可されている場合、IWDT はデバイスをリセットするか、NMI を生成します。

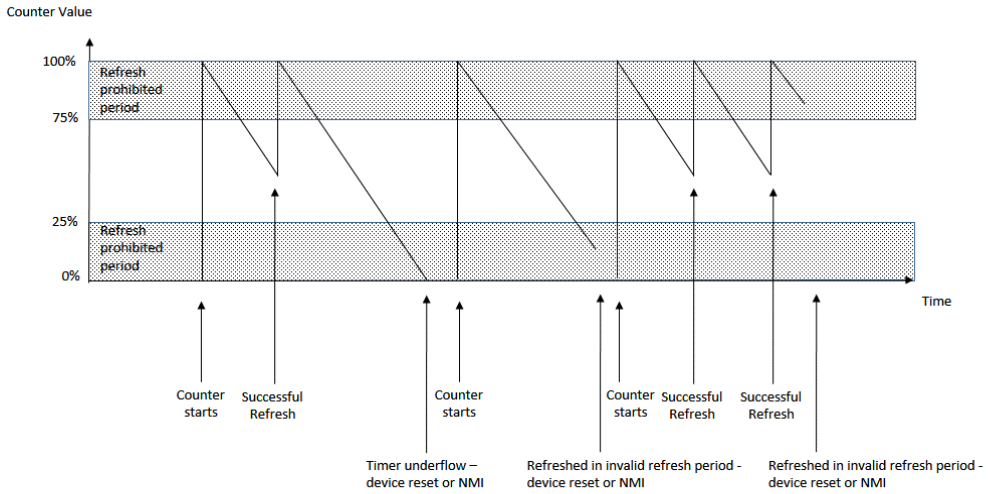


図 392:　独立ウォッチドッグタイマの動作の図

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後のペリフェラルの動作状態を設定できます。 OFS は、 IWDT、 WDT、 LVD、 CGC HOCO の状態の設定に使用できます。

次の表では、 OFS レジスタで構成できる IWDT のパラメータを示します。

***注:** IWDT は、OFS レジスタを使用することによってのみ構成できます。IWDT は、レジスタスタートモードをサポートしていません。*

Control	説明
IWDT Start Mode Select	有効にすると、リセット後に IWDT を自動的に開始します。
IWDT Timeout Period	IWDT タイムアウト（クロックサイクルの数）を指定します（128、 512、 1024、 2048 サイクル）
IWDT-Dedicated Clock Frequency Division Ratio	1 1/16 1/32 1/64 1/128 1/256
IWDT Window End Position	25%、50%、75%、100%（ウィンドウ終了位置の設定はありません）

Control	説明
IWDT Window Start Position	25%、50%、75%、100%（ウィンドウ開始位置の設定はありません）
IWDT Reset Interrupt Request	IWDT は、割り込み信号またはリセット信号を生成できます。
IWDT Stop Control	IWDT は、カウントを続けるか、ローパワーモードでカウントを停止することができます。

***注:** OFS0 レジスタの内容の詳細については、 Synergy MCU ハードウェアマニュアルを参照してください。*

OFS レジスタの値は、 Synergy 構成エディタの [BSP] タブのプロパティで設定します。

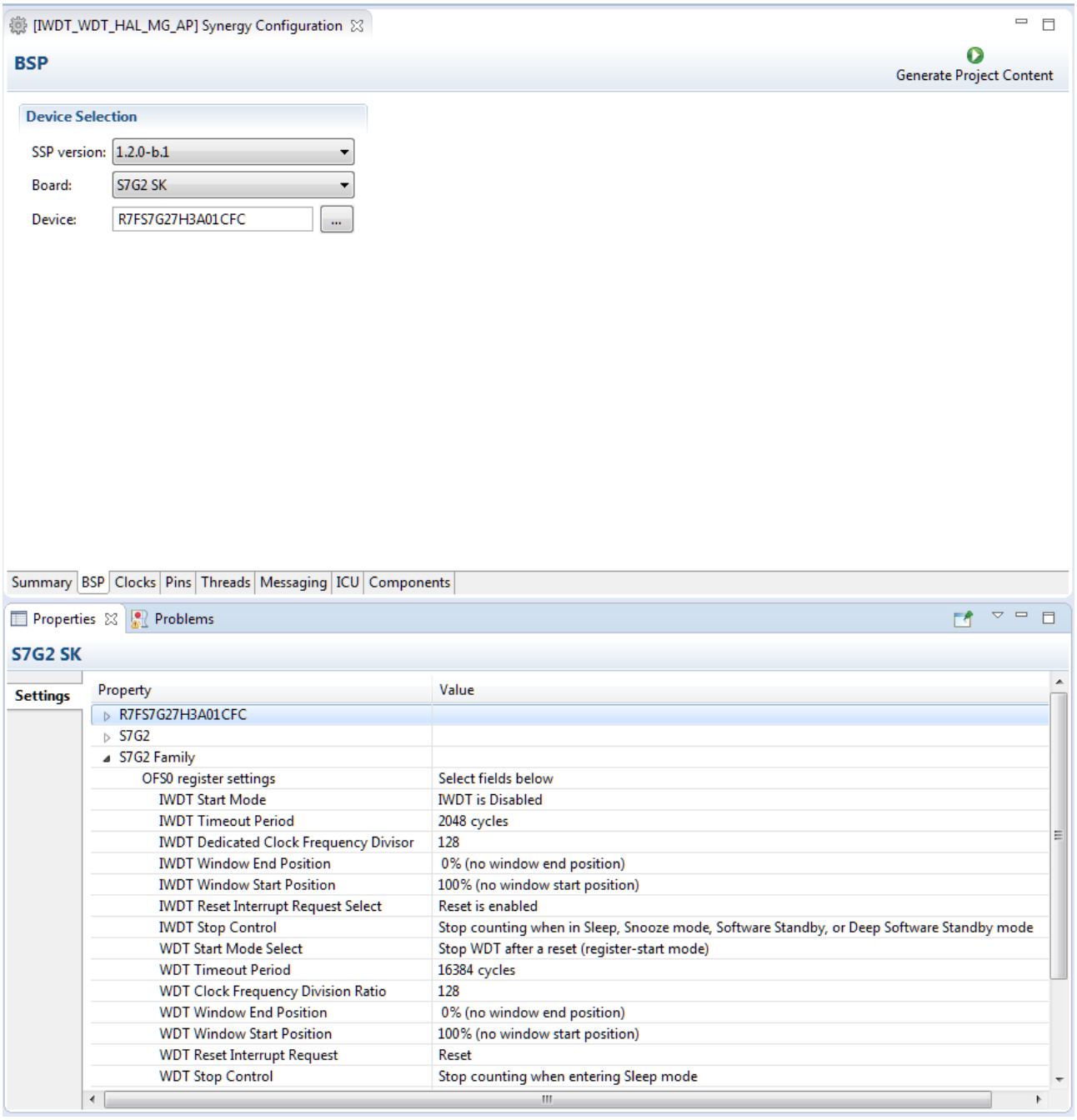


図 393: OFS レジスタの値の設定

5.2.26.4 IWDT HAL モジュールの動作に関する重要な注意事項と制限事項

5.2.26.5 IWDT HAL モジュールの期間の計算

IWDT は IWDTCLK から動作します。WDT のパラメータが最大で、IWDTCLK の周波数が 15kHz であると仮定した場合、最後のリフレッシュからデバイスのリセットまたは NMI の生成までの時間は、35 秒未満です。IWDT モジュールの期間に対するクロックの指定は以下のとおりです。

IWDTLCK = 15 kHz

クロック分割比 = IWTCLK/256

タイムアウト周期 = 2048 サイクル

IWDT クロック周波数 = 15 kHz / 256 = 58.59 Hz

サイクル時間 = 1 / 58.59 Hz = 17.07 マイクロ秒

タイムアウト = 17.07 マイクロ秒 x 2048 サイクル = 34.95 秒

5.2.26.6 IWDT HAL モジュールを使用した DMAC/DTC のトリガ

ウォッチドッグカウンタがアンダーフローした場合、または有効なリフレッシュ期間外にリフレッシュが試みられた場合に、 DMAC または DTC を使用してデータの転送をトリガするには、 activation_source を ELC_EVENT_IWDT_UNDERFLOW. に設定して DMAC/DTC 転送を構成します。詳細については、適切なモジュールのユーザーガイドを参照してください。

5.2.26.7 IWDT HAL モジュールを使用した ELC イベントのトリガ

IWDT は、イベントリンクコントローラ （ELC）で使用可能な他のペリフェラルの開始をトリガできます。詳細については、 ELC HAL モジュールのガイドを参照してください。

5.2.26.8 IWDT HAL モジュールの制限事項

- JLinkデバッガを使用していると、IWDTカウンタはカウントされないため、デバイスのリセットやNMIの生成は行われません。
- 実行できるアクティブなタスクがない場合、ThreadX は MCU をスリープモードにします。IWDT がローパワーモードでカウンタを停止するように構成されている場合、ThreadX RTOS で使用されているときは、アプリケーションでタイマを再度開始する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.26.9 アプリケーションへの IWDT HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに IWDT HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

IWDT HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（IWDT HAL モジュールのデフォルトの名前は、次の表で示されているように r_iwdt です。この名前は、対応する [Properties] ウィンドウで変更できます）。

ウォッチドッグタイマの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_wdt0 IWDT HAL on r_iwdt	Threads	New Stack> Driver> Monitoring> IWDT HAL on r_iwdt

次の図に示すように、r_iwdt の IWDT HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

図 394: IWDT HAL モジュールのスタック

5.2.26.10 IWDT HAL モジュールの構成

ユーザーは必要な動作に合わせて IWDT HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまで

スクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

以下の表では、 IWDT の実装の構成設定を示します。

r_iwdt での IWDT HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックコードを含みます
Name	g_wdt0	モジュール名
NMI Callback	NULL	コールバック。ユーザーコールバック関数を open API で登録できます。このコールバック関数が指定されている場合、IRQn がトリガされるたびに割り込みサービスルーチン(ISR)から呼び出されます。

コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。|

注: *上記の設定例とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

5.2.26.11 オプション機能選択レジスタ 0 （OFS0）の構成

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後のペリフェラルの動作状態を設定できます。 OFS は、 IWDT、 WDT、 LVD、 CGC HOCO の状態の設定に使用できます。

5.2.26.12 IWDT HAL モジュールの割り込みの構成

IWDT の割り込みの構成は、ISDE を使用し、IWDT モジュールの他のオプションの構成と同じ方法で行います。アンダーフローまたは無効なリフレッシュで NMI 割り込みを生成するよう IWDT が構成されている場合、割り込みは BSP で有効になっている必要があります。

注： *割り込みを有効にするには、[IWDT] > [IWDT NMIUNDF N] の優先順位を設定します。これにより、ssp_cfg/bsp/bsp_irq_cfg.h の BSP_IRQ_CFG_IWDT_UNDERFLOW が選択した優先順位レベルに設定されます。*

IWDT NMIUNDF N 割り込みが BSP で有効になっている場合、対応する ISR が定義されます。 ISR は、ユーザーコールバック関数が open API で登録されている場合、それを呼び出します。

5.2.26.13 IWDT HAL モジュールのクロック構成

IWDT の専用クロックは設定された周波数で動作しており、変更することはできません。

5.2.26.14 IWDT HAL モジュールのピン構成

5.2.26.15 IWDT は、動作するためにピンを必要としません。

5.2.26.16 アプリケーションでの IWDT HAL モジュールの使用

アプリケーションで IWDT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 IWDT NMI コールバックを登録します。
2) cfgGet API を使用して、 IWDT HAL モジュールの構成を読み取ります。
3) refresh API を使用して、独立ウォッチドッグタイマをリフレッシュします。
4) statusGet API を使用して、 IWDT の状態を読み取ります。
5) statusClear API を使用して、 IWDT の状態とエラーフラグをクリアします。
6) counterGet API を使用して、 IWDT の現在のカウント値を読み取ります。
7) timeoutGet API を使用して、 IWDT HAL モジュールのタイムアウト値を読み取ります。

上の手順を、次の図の通常の動作フローに示します。

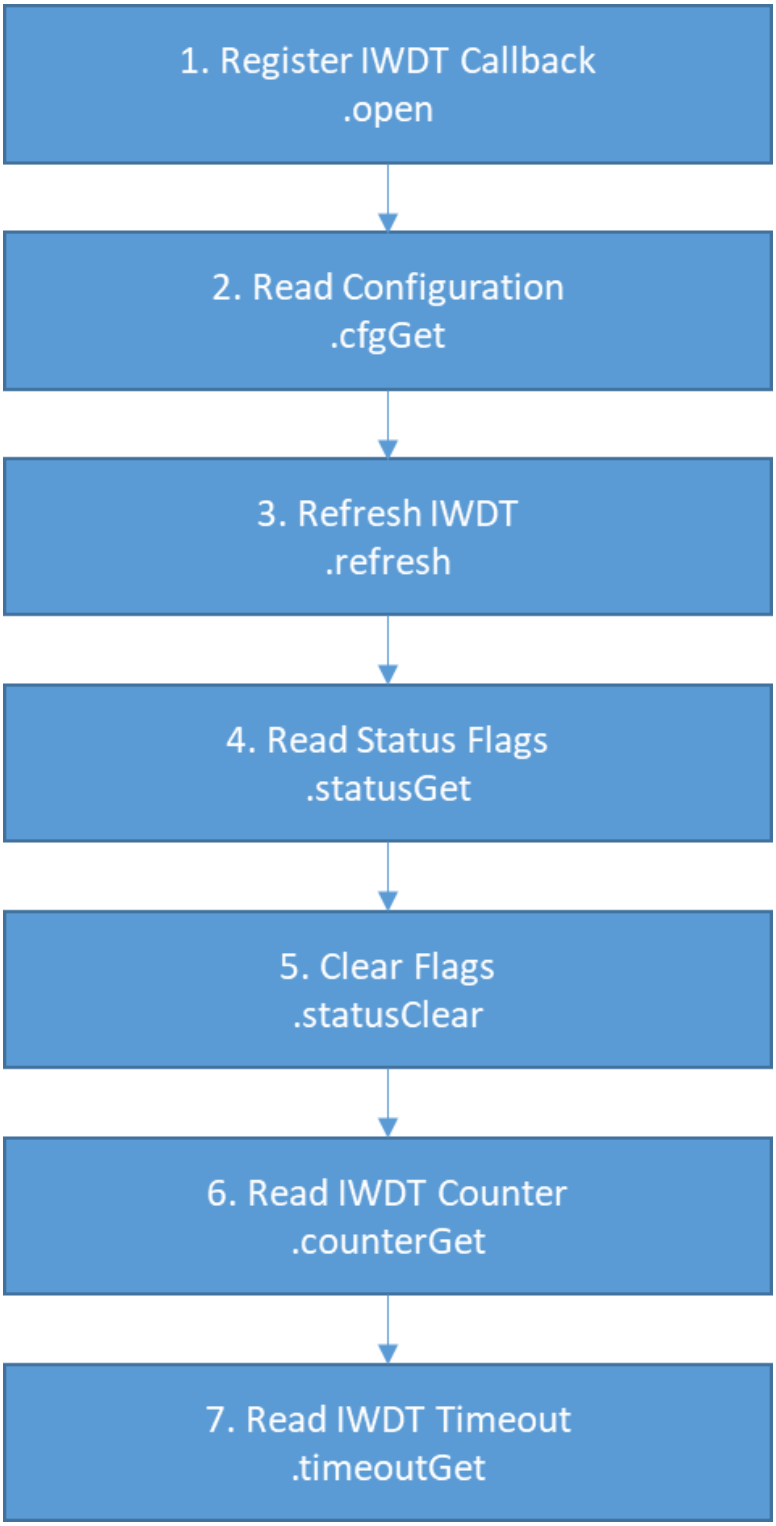

1. Register IWDT Callback
.open
2. Read Configuration
.cfgGet
3. Refresh IWDT
.refresh
4. Read Status Flags
.statusGet
5. Clear Flags
.statusClear
6. Read IWDT Counter
.counterGet
7. Read IWDT Timeout
.timeoutGet

図 395: 一般的な IWDT HAL モジュールアプリケーションのフロー図

5.2.27 JPEG デコードドライバー

- JPEG デコードをサポート。
- JPEG デコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直サブサンプル値、水平ストライド、デコードされたピクセルフォーマット、入力および出力データフォーマット、色空間などのパラメータを設定。
- イメージをデコードする前にそのサイズを取得。
- デコードされたイメージフレームの保管のため、コード化されたデータを入力バッファーおよび出力バッファーに格納する操作をサポート。
- JPEG デコーダーモジュールへのコード化されたデータのストリーム転送をサポート。この機能により、アプリケーションは、ファイルやネットワークからのコード化された JPEG イメージの読み取りを、イメージ全体をバッファーリングすることなく実行できます。
- デコードするイメージ行数を設定。この機能により、アプリケーションは、デコードされたイメージの処理を、フレーム全体をバッファーリングすることなくオンザフライで実行可能。
- 入力値のデコードされたフォーマットとして YCbCr444、YCbCr422、YCbCr420、YCbCr411 をサポート。
- 出力フォーマットとして ARGB8888、RGB565 をサポート。
- JPEG イメージのサイズ、高さ、および幅が要件を満たさない場合にエラーを返すことが可能。

5.2.27.1 e² studio による外部 JPEG デコードドライバーを使用するアプリケーションの作成

ドライバーは、e² studio の Synergy Software Package に組み込まれています（e² studio ISDE ユーザーガイドを参照）。

e² studio でプロジェクトの作成と設定を行い、ドライバーを追加します。

1) プロジェクトを作成します（Synergy プロジェクトの作成を参照）。
2) プロジェクトを設定します（プロジェクトの設定を参照）。
3) ドライバーを追加します（スレッドとドライバーの追加を参照）。

JPEG デコードドライバーを使用するアプリケーションでは、次のリソースが必要です。

Resource	ISDE Tab	Selection
JPEG Decode Driver	Threads	**Driver > Graphics > JPEG Decode Driver on r_jpeg**

Resource	ISDE Tab	Selection
Interrupts	Threads	SeeJPEG デコード割り込みの設定
Clock	Clocks	SeeJPEG デコードドライバーのクロックソースの設定
Pins	Pins	None

JPEG デコードドライバーのクロックソースの設定

e2 studio ISDE で、 [Clocks] タブを使用して JPEG クロックを設定します （クロックの設定を参照）。

JPEG デコードモジュールは、ペリフェラルモジュールクロック A(PCLKA) を使用して内部論理を実行します。ソースクロックの選択は、 e2 studio の Synergy 構成の [Clocks] タブで行うことができます。

JPEG デコード割り込みの設定

e2 studio ISDE を使用して、 [ICU] タブから JPEG 割り込みを設定します （割り込みの設定を参照）。

以降のセクションで説明するように、複数の JPEG デコード割り込みを設定することもできます。

解凍プロセス割り込み (JEDI)

JPEG デコードプロセス割り込みは、次のイベントが検出されたときに発生します。

1) 現在のデコードプロセスが正常に完了した。
2) デコードプロセス中にエラーが発生した。
3) イメージのサイズとピクセルフォーマットが正常に読み出された。

***注:** 割り込みを有効にするには、e2 studio のプロジェクトコンフィギュレーターの [ICU] または [Threads] タブで、 JPEG > JPEG JEDI 割り込みの優先度を設定します。*

データ転送インタフェース (JDTI)

JPEG データ転送割り込みは、次のイベントが検出されたときに発生します。

1) JPEG コードされたデータがすべて正常に完了した。
2) linesDecodedGet で指定された出力イメージデータ行数が転送された。
3) inputBufferSet で指定された入力イメージデータ行数が転送された。

***注:** 割り込みを有効にするには、e2 studio のプロジェクトコンフィギュレーターの [ICU] タブで、JPEG>JPEG JDTI の優先度を設定します。*

JPEG デコードステータス

JPEG デコードモジュールの制御ブロックにはステータスフラグがあります。ユーザーは statusGet を通じて現在のステータスを取得できます。ステータスの定義については以下の表をご覧ください。また、モジュールで特定のイベントが発生すると、ユーザーコールバック関数を通じてステータスがレポートされます。JPEG デコードコールバックを参照してください。

イベント名	イベントの発生条件
JPEG_DECODE_STATUS_FREE	JPEG デコードモジュールが閉じられた。
JPEG_DECODE_STATUS_IDLE	JPEG デコードモジュールが開いているが、動作していない。
JPEG_DECODE_STATUS_RUNNING	JPEG デコード操作が実行されている。
JPEG_DECODE_STATUS_DONE	JPEG デコード操作が正常に完了した。
JPEG_DECODE_STATUS_INPUT_PAUSE	JPEGデコードが追加の入力データを待つために一時停止した。
JPEG_DECODE_STATUS_OUTPUT_PAUSE	JPEG デコードが、ユーザーにより指定された行数をデコードした後に一時停止した。
JPEG_DECODE_STATUS_IMAGE_SIZE_READY	JPEG デコードが、デコードするデータのイメージサイズを取得して、一時停止した。
JPEG_DECODE_STATUS_ERROR	JPEG デコードモジュールでエラーが発生した。
JPEG_DECODE_STATUS_HEADER_PROCESSING	JPEG デコードモジュールが JPEG ヘッダーを処理中である。

JPEG デコードコールバック

ユーザーコールバック関数を open で登録できます。ユーザーコールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービスルーチン (ISR) からコールバック関数が呼び出されます。コールバック関数の引数 status は、デコーディングプロシージャにおいて発生したイベントをユーザーが識別できるように、以下に示す列挙値を受け取ることができます。

イベント名	イベントの発生条件
JPEG_DECODE_STATUS_ERROR	JPEG デコードモジュールでエラーが発生した。
JPEG_DECODE_STATUS_IMAGE_SIZE_READY	JPEG デコードが、デコードするデータのイメージサイズを取得して、一時停止した。
JPEG_DECODE_STATUS_INPUT_PAUSE	JPEGデコードが追加の入力データを待つために一時停止した。
JPEG_DECODE_STATUS_OUTPUT_PAUSE	JPEG デコードが、ユーザーにより指定された行数をデコードした後に一時停止した。

イベント名	イベントの発生条件
JPEG_DECODE_STATUS_DONE	JPEG デコード操作が正常に完了した。

注：ユーザーコールバック関数は ISR から呼び出されるため、ブロッキング呼び出しを使用することや、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

JPEG デコードドライバーパラメータの設定

e^2 studio ISDE を使用して、JPEG デコードドライバーパラメータを設定します。

RTOS を使用しないアプリケーションの場合 : HAL ドライバーの追加と設定。

ThreadX アプリケーションの場合 : ドライバーのスレッドへの追加とドライバーの設定。

JPEG デコードには、次のコンポーネントの設定が必要です。

- JPEG デコード :jpeg_decode_cfg_t

JPEG ドライバーデコードのビルド時構成

ISDE Property	Configuration	Setting	説明
Parameter Checking Enable	BSP_CFG_PARAM_CHECKING_ENABLE	Enabled (Default), Disabled	パラメータエラーチェックを有効または無効にします。
Name	Name prefix of instances.	Arbitrary C Symbol (Default: "g_jpeg_decode")	JPEGデコードモジュールインスタンスに使用する名前。
Byte Order for Input Data Format	input_data_format	Normal byte order - (1)(2)(3)(4)(5)(6)(7)(8)(Default);Byte Swap - (2)(1)(4)(3)(6)(5)(8)(7); Word Swap - (3)(4)(1)(2)(7)(8)(5)(6); Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5); Longword Swap - (5)(6)(7)(8)(1)(2)(3)(4); Longword-Byte Swap - (6)(5)(8)(7)(2)(1)(4)(3); Longword-Word Swap -(7)(8)(5)(6)(3)(4)(1)(2); Longword-Word-Byte Swap - (8)(7)(6)(5)(4)(3)(2)(1)	入力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。

ISDE Property	Configuration	Setting	説明
Byte Order for Output Data Format	output_data_format	Normal byte order - (1)(2)(3)(4)(5)(6)(7)(8); Byte Swap (2)(1)(4)(3)(6)(5)(8)(7); Word Swap (3)(4)(1)(2)(7)(8)(5)(6); Word-Byte Swap -(4)(3)(2)(1)(8)(7)(6)(5); Longword Swap - (5)(6)(7)(8)(1)(2)(3)(4); Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3); Longword-Word Swap - (7)(8)(5)(6)(3)(4)(1)(2); Longword-Word-Byte Swap (8)(7)(6)(5)(4)(3)(2)(1)	出力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Pixel Format	pixel_format	ARGBB888 or RGB565 (Default)	出力データ フォーマットを指定します。
Alpha value to be applied to decoded pixel data	alpha_value	Value must be an integer greater than or equal to 0 and less than or equal to 255 (Default: 255)	出力データフォーマットのアルファ値を指定します（ARGB8888 フォーマットに対してのみ有効）。
Name of user callback function	p_callback	Arbitrary C Symbol (Default: NULL)	ユーザーコールバック関数の名前を指定します。

JPEG のユーザー定義のコールバック関数

JPEG デコードモジュールは、open の構成 p_callback に渡されたユーザー定義のコールバック関数を呼び出します。このコールバック関数を使用して、JPEG デコードモジュールのステータスを取得し、JPEG デコーディングのシーケンスを開始することができます。

```
volatile _Bool g_input_buffer_pause_flag = false;
volatile _Bool g_output_buffer_pause_flag = false;
volatile _Bool g_decode_done_flag = false;
volatile _Bool g_image_size_is_ready_flag = false;
volatile _Bool g_decode_error_flag = false;

void user_jpeg_callback (jpeg_decode_callback_args_t *p_args)
{
    if (p_args->status & JPEG_DECODE_STATUS_IMAGE_SIZE_READY)
    {
        g_image_size_is_ready_flag = true;
    }
    if (p_args->status & JPEG_DECODE_STATUS_INPUT_PAUSE)
    {
        g_input_buffer_pause_flag = true;
```

```
    }
    if (p_args->status & JPEG_DECODE_STATUS_OUTPUT_PAUSE)
    {
        g_output_buffer_pause_flag = true;
    }
    if (p_args->status & JPEG_DECODE_STATUS_DONE)
    {
        g_decode_done_flag = true;
    }
    if (p_args->status & JPEG_DECODE_STATUS_ERROR)
    {
        g_decode_error_flag = true;
    }
}
```

JPEG モジュールを初期化する

```
/* JPEG Decode module control block */
static jpeg_decode_ctrl_t g_jpeg_decode_ctrl;

/* JPEG Decode module configuration */
static const jpeg_decode_cfg_t g_jpeg_decode_cfg = {
    .input_data_format = JPEG_DECODE_DATA_FORMAT_NORMAL,
    .output_data_format = JPEG_DECODE_DATA_FORMAT_NORMAL,
    .pixel_format = JPEG_DECODE_PIXEL_FORMAT_RGB565,

    .alpha_value = 255,
    .p_callback = user_jpeg_callback
};

/* JPEG Decode module instance */
const jpeg_decode_instance_t g_jpeg_decode = {
    .p_api = (jpeg_decode_api_t const *)&g_jpeg_decode_on_jpeg_decode,
    .p_ctrl = &g_jpeg_decode_ctrl,
    .p_cfg = &g_jpeg_decode_cfg
};
```

JPEG モジュールを開く

JPEG エンコードされたデータのデコードは、open を呼び出すことで開始できます。モジュールを開くには、JPEG モジュールインスタンスを使用します。このインスタンスには、API 関数のポインタ、制御ブロックへのポインタ、および e^2 studio のプロジェクトコンフィギュレーターを通じて生成可能な静的構成が含まれています。

```
/* JPEG Decode driver open */
g_jpeg_decode.p_api->open(jpeg_decode.p_ctrl, jpeg_decode.p_cfg);
```

JPEG モジュールを閉じる

JPEG デコードモジュールを停止するには、 close を呼び出します。

```
/* JPEG Decode driver close */
g_jpeg_decode.p_api->close(jpeg_decode.p_ctrl);
```

JPEG 入力バッファストリーミングモード

次の例は、JPEG エンコードされたデータを JPEG 入力バッファーストリーミングモードでデコードする方法を示しています。JPEG デコードモジュールは、JPEG モジュールを開くで記述した構成を使用してあらかじめ開かれていなければなりません。

```
    …

/* Set the vertical and horizontal sample. */

g_jpeg_decode0.p_api->imageSubsampleSet(g_jpeg_decode0.p_ctrl,
*JPEG_DECODE_OUTPUT_NO_SUBSAMPLE*,
*JPEG_DECODE_OUTPUT_NO_SUBSAMPLE*);

/* Set the decoding image horizontal stride. Here, assuming it is 800
in pixels */
g_jpeg_decode0.p_api->horizontalStrideSet(g_jpeg_decode0.p_ctrl,
800);

/* Set output buffer address. */

g_jpeg_decode0.p_api->outputBufferSet(g_jpeg_decode0.p_ctrl,
output_buffer, output_buffer_size);

/* Get an input buffer address and the size */
input_buffer = user_get_input_buffer_address();
input_buffer_size = user_get_input_buffer_size();

/* Set the input buffer address. */

g_jpeg_decode0.p_api->inputBufferSet(g_jpeg_decode0.p_ctrl,
input_buffer, input_buffer_size);

while(1)

{   /* Wait until the JPEG hardware being ready to get image size */

    if(true == g_image_size_is_ready_flag)

    {

        break;

    }
```

```
}

while(1)

{

    /* Wait until either of flags is set. Decode done or Input buffer pause. */

    if(true == g_decode_done_flag)

    {

        break;

    }

    if(true == g_input_buffer_pause_flag)

    {

        input_buffer = user_get_input_buffer_address();
        input_buffer_size = user_get_input_buffer_size();

        /* Set input buffer address. */

        input_buffer = input_buffer + input_size;

        err = g_jpeg_decode0.p_api->inputBufferSet(g_jpeg_decode0.p_ctrl,
input_buffer, input_buffer_size);

    }

}

    …
```

JPEG 出力バッファストリーミングモード

次の例は、JPEG エンコードされたデータを JPEG 出力バッファーストリーミングモードでデコードする方法を示しています。JPEG デコードモジュールは、JPEG モジュールを開くで記述した構成を使用してあらかじめ開かれていなければなりません。

```
…

/* Set the vertical and horizontal sample. */

g_jpeg_decode0.p_api->imageSubsampleSet(g_jpeg_decode0.p_ctrl,
*JPEG_DECODE_OUTPUT_NO_SUBSAMPLE*,
*JPEG_DECODE_OUTPUT_NO_SUBSAMPLE*);
```

```
/* Set the decoding image horizontal stride. Here, assuming it is 800
in pixels */
g_jpeg_decode0.p_api->horizontalStrideSet(g_jpeg_decode0.p_ctrl,
800);

/* Set output buffer address. */

g_jpeg_decode0.p_api->outputBufferSet(g_jpeg_decode0.p_ctrl,
output_buffer, output_buffer_size);

/* Set the input buffer address. */

g_jpeg_decode0.p_api->inputBufferSet(g_jpeg_decode0.p_ctrl,
                                     input_buffer, input_buffer_size);

while(1)

{   /* Wait until the JPEG hardware being ready to get image size */

    if(true == g_image_size_is_ready_flag)

    {

        break;

    }

}

uint16_t horizontal_size;

uint16_t vertical_size;

g_jpeg_decode0.p_api->imageSizeGet(g_jpeg_decode0.p_ctrl,

&horizontal_size, &vertical_size);

/* Calculate the size of image in bytes. Here is the case of 2bpp(RGB565) */

uint32_t total_bytes = (uint32_t)(horizontal_size * vertical_size * 2);

uint32_t decoded_bytes = 0;

while(1)

{

    /* Wait until either of flags is set. Decode done or Output buffer pause. */
```

```
    if(true == g_decode_done_flag)

    {

        break;

    }

    if(true == g_output_buffer_pause_flag)

    {

        /* Set input buffer address. */

        output_buffer = output_buffer + output_size;

        g_jpeg_decode0.p_api->outputBufferSet(g_jpeg_decode0.p_ctrl,
                    output_buffer, output_size);

        decoded_bytes = decoded_bytes + output_size;

        if (total_bytes <= decoded_bytes)

        {

            break;

        }

        g_output_buffer_pause_flag = false;

    }

}

…
```

5.2.27.2 JPEG デコードドライバーの制限事項

JPEG デコードドライバーは、JPEG エンコード処理をサポートしていません。

5.2.27.3 JPEG デコードドライバーでサポートされるデバイス

モジュールは S7G2 ファミリでサポートされています。

5.2.28 Key Matrix ドライバー

この Key Matrix HAL モジュールは、キー割り込み （KINT）を構成および制御します。次のキー機能を実装します：

- KINT チャネルでの立ち上がりエッジと立ち下がりエッジの両方をサポートします
- 割り込みベースのイベント通知をサポートします
- 複数のイベントを効率よくキャプチャするためのビットマスク機能をサポートします
- 任意の 2 つのチャネルにエッジのあるマトリックスキーパッドをサポートします

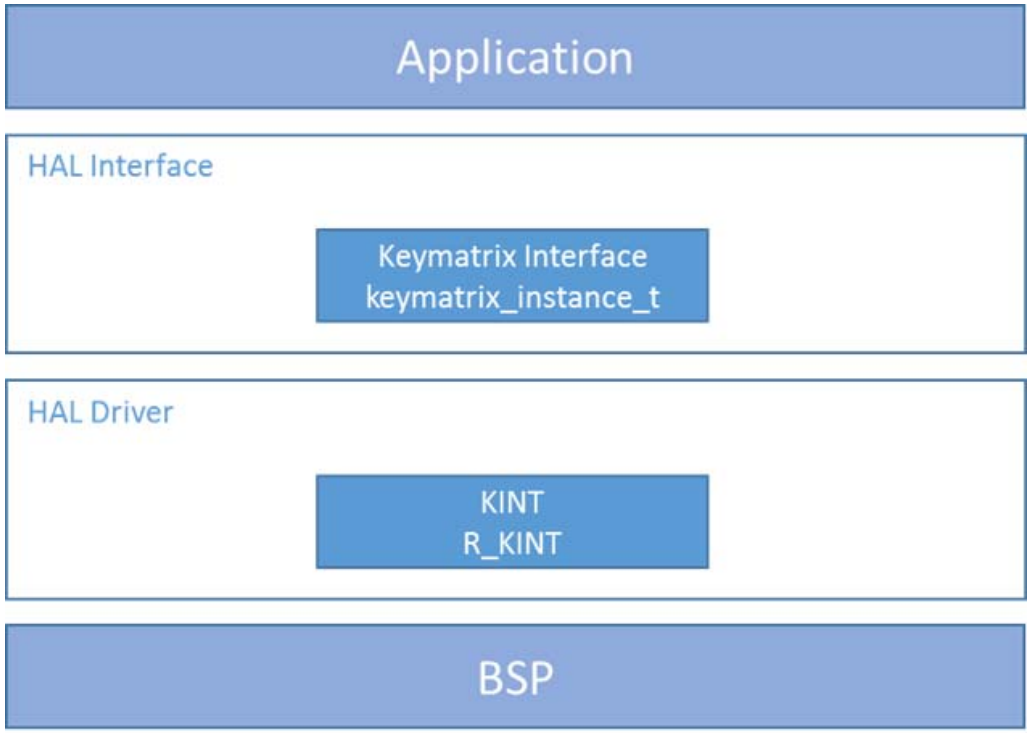


図 396: Key Matrix HAL モジュールのブロック図

5.2.28.1 Key Matrix HAL モジュールの API の概要

Key Matrix HAL モジュールでは、キー割り込み機能のオープン、クローズ、有効化、無効化のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

Key Matrix HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_keymatrix_on_kint.p_api->open(&g_kint.p_ctrl, g_kint.p_cfg_cfg) 初期設定。
enable	g_keymatrix_on_kint.p_api->enable(&g_kint.p_ctrl) キー割り込みを有効にします。
disable	g_keymatrix_on_kint.p_api->disable(g_kint.p_ctrl) キー割り込みを無効にします。
triggerSet	g_keymatrix_on_kint.p_api->triggerSet() キー割り込みのトリガーを設定します。
close	g_keymatrix_on_kint.p_api->close(&g_ keymatrix) ドライバーを再設定できるようになります。電力消費を低減できます。
versionGet	g_keymatrix_on_kint.p_api->versionGet(&version) バージョンを取得し、指定されたポインタ version に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に完了しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。

Name	説明
SSP_ERR_INVALID_ARGUMENT	引数が無効です。
SSP_ERR_HW_LOCKED	API が既に開かれています。もう一度開く前に閉じる必要があります。
SSP_ERR_NOT_OPEN	ペリフェラルが開かれていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.28.2 HAL モジュールの動作の概要

Key Matrix HAL モジュールは、いずれかのキー割り込み（KINT）チャネルでの立ち上がりエッジまたは立ち下がりエッジを検出するように、KINT を構成します。設定されているいずれかのピンで該当するイベントが検出された場合、このモジュールは割り込みを生成します。その後、割り込みはユーザーコールバック (p_callback) を呼び出します。そのときに渡されるコールバック引数 keymatrix_callback_args_t では、ビットマスクを使用してエッジが検出された channel(s) が指定されています。

いずれか 1 つのチャネルでエッジが検出されると割り込みが生成されますが、他のいずれかのピンでもエッジが検出された場合は、その時点でトリガされたすべてのピンのビットマスク keymatrix_channels_t がコールバックで返されます。そのため、コールバックが呼び出される前に他のピンでもエッジが検出された場合、ピンごとのエッジ検出に対して必ずしも割り込みが生成されるとは限りません。コールバックが呼び出された後で新しいエッジが検出されると、割り込みが再度トリガされ、新たにコールバックが呼び出されます。

このモジュールを使用すると、押された実際のキーを示す 2 つのチャネル上のエッジを使用した、マトリックスキーパッドを実装できます。また、このモジュールは 1 つの入力ピン上のエッジを検出する単一入力としても使用できます。

Key Matrix HAL モジュールの動作に関する重要な注意事項と制限事項

- トリガエッジが検出されたときに、DMAC または DTC を使用するデータの転送をトリガするには、activation_source を ELC_EVENT_KEY_INT. に設定して DMAC/DTC 転送を構成します。
- KINT モジュールは、ELC に対して使用可能な他のペリフェラルの開始をトリガできます。詳細については、『*SSP ユーザーズマニュアル*』の ELC に関するユーザーガイドを参照してください。
- このモジュールが動作するためには、open の呼び出しでコールバックが使用されているかどうかにかかわらず、BSP で KINT（INTKR）割り込みを有効にする必要があります。
- このモジュールは、ポーリングモードの動作をサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.28.3 アプリケーションへの Key Matrix HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して Key Matrix HAL モジュールをアプリケーションに組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

Key Matrix ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（Key Matrix ドライバーのデフォルトの名前は g_kint0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

Key Matrix HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_kint0 Key Matrix Driver on r_kint	**Threads**	**New Stack> Driver> Input> Key Matrix Driver on r_kint**

次の図に示すように、r_kint の Key Matrix ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

図 397: Key Matrix HAL モジュールのスタック

5.2.28.4 Key Matrix HAL モジュールの構成

ユーザーは必要な動作に合わせて、r_kint の Key Matrix HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、統合ソリューション開発環境（ISDE）の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアク

セスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_kint での Key Matrix HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_kint0	open API を使用して、ユーザーコールバック関数を登録できます。このコールバック関数が指定されている場合、いずれかのチャネルで設定されているエッジが検出された場合に、割り込みサービスルーチン（ISR）から呼び出されます。 ***注:*** *コールバックなしでは、アプリケーションはイベントが発生したかどうかを判定することはできません。コールバックは ISR からコールされるため、ブロッキングコールを使用したり、長時間処理することはしないでください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。*
Keymatrix Channel Mask	Select Channels Below	各ビットがそのチャネルを有効にするかどうかを指定するビットマスク。使用するチャネルを選択します。

ISDE Property	Value	説明
Channel 0:7	Unused, Used (Default: Unused)	チャネル 0:7 の選択。
Trigger Type	Rising Edge, Falling Edge (Default: Rising Edge)	有効化されたチャネルが、立ち上がりエッジと立ち下がりエッジのどちらを検出するか指定します。***注:*** 立ち上がりエッジを検出するすべてのチャネル、または立ち下がりエッジを検出するすべてのチャネルのいずれか。
Interrupt enabled after initialization	True, False (Default: False)	モジュール割り込みを open の呼び出しの一部として有効にする必要があるかどうかを指定します。
Callback	NULL	コールバックの選択。
Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

Key Matrix HAL モジュールのクロック構成

Key Matrix HAL モジュールには、固有のクロック構成は必要ありません。

Key Matrix HAL モジュールのピン構成

KINT 周辺モジュールは、 MCU のピンを使用して外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では KINT ピンの選択例を示します。

r_kint での Key Matrix HAL モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
KINT	Pins	Select **Peripherals > Input:KINT> KINT0**

注 : 選択シーケンスでは、 KINT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_kint での Key Matrix HAL モジュールのピン構成設定

Property	Value	説明
Operation Mode	Disabled, Custom (Default: Disabled) Custom	動作モードとして [Custom] を選択します
KRM0:7	None, Pnn (Default: None)	キー割り込みピンの選択

注 : 設定例は、 Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

5.2.28.5 アプリケーションでの Key Matrix HAL モジュールの使用

アプリケーションで Key Matrix HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 Key Matrix HAL モジュールを初期化します
2) オートスタートの構成設定が true の場合、モジュールはすぐに動作を開始します
3) オートスタートが設定されていない場合は、 enable API を使用して動作を有効にします
4) キー入力に応答します
5) disable API を使用して、動作を無効にします
6) 初期化後にトリガエッジを変更するには、 triggerSet API を使用します
7) close API を使用して、モジュールを閉じます

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

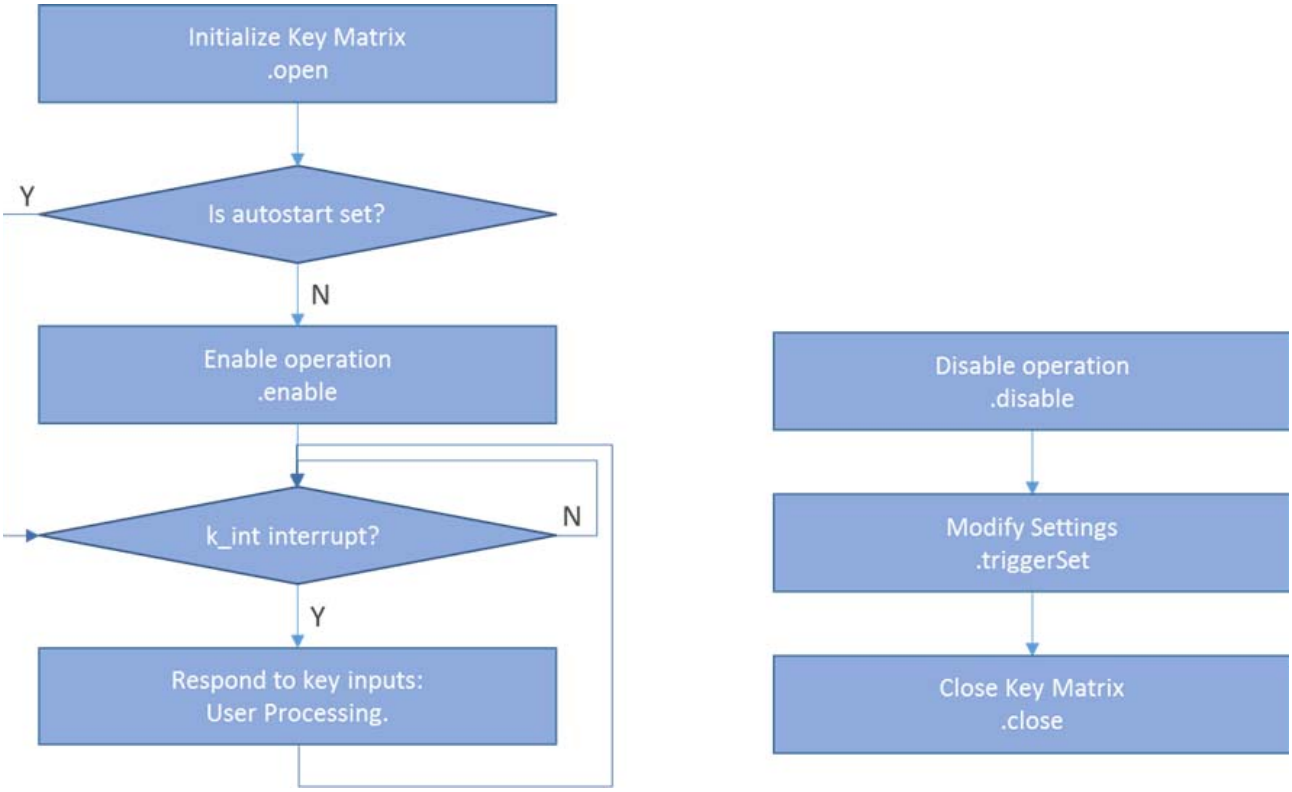


図 398: 通常の Key Matrix HAL モジュールアプリケーションのフロー図

5.2.29 ローパワーモードドライバー

LPM HAL モジュールは、ローパワーモードハードウェアペリフェラルを使用して、MCU 動作電力制御モードと MCU ローパワーモードの構成をサポートしています。

この LPM ドライバーは、動作電力制御モードの以下の機能をサポートしています。

- 低電圧モード
- 低速モード
- 中速モード
- 高速モード
- サブ発振器速度モード

この LPM ドライバーは、ローパワーモードの以下の機能をサポートしています。

- ディープソフトウェアスタンバイモード
- ソフトウェアスタンバイモード
- スリープモード
- スヌーズモード

*注：すべての MCU で、すべての動作モードが使用できるわけではありません。**注：**すべての MCU で、すべてのローパワーモードが使用できるわけではありません。**注：**これは非推奨のモジュールであるため、新しいプロジェクトには使用しないでください。代わりに lpm_v2 モジュールを使用してください。*

このドキュメントは以下のセクションに分かれており、個別に読んでも（経験豊富な Synergy 開発者向け）、続けて読んでも （Synergy プラットフォームの経験がない開発者向け）かまいません。

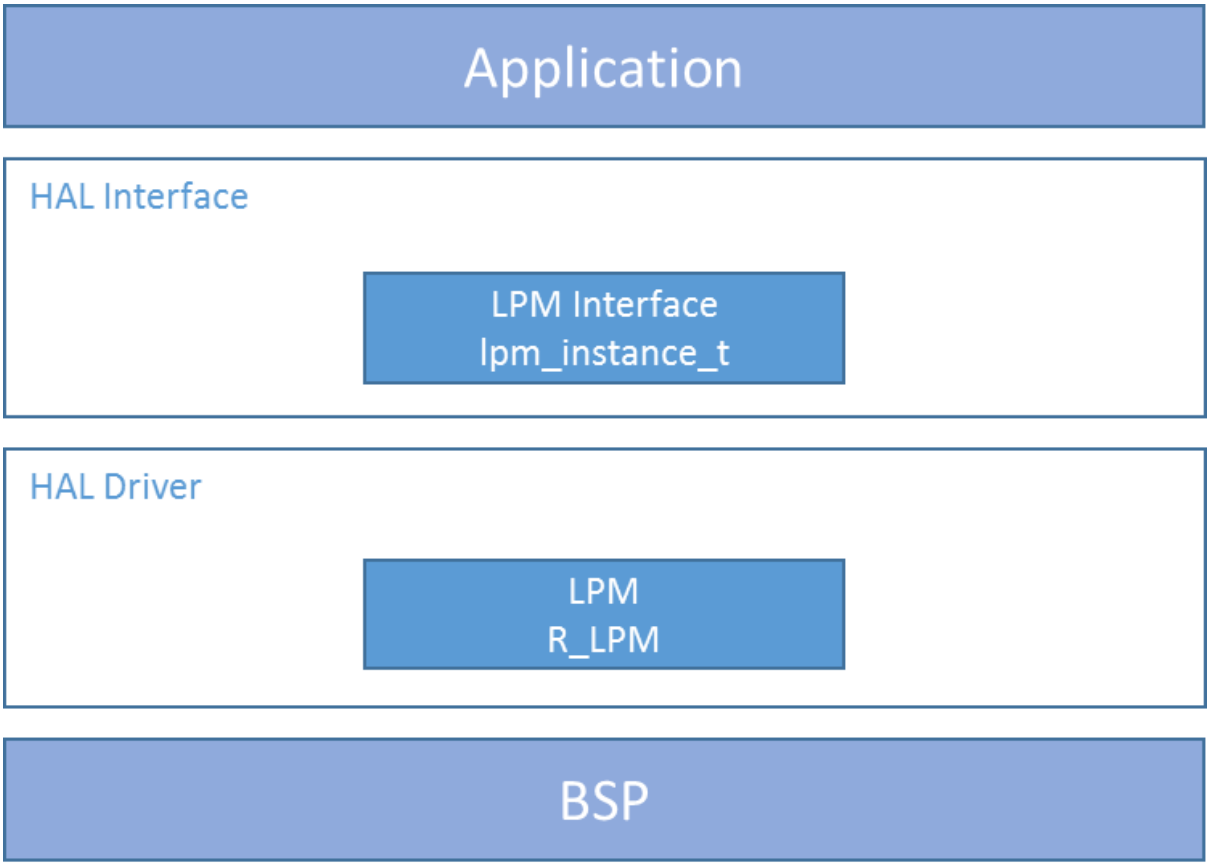


図 399: ローパワーモードドライバーの編成、オプション、スタックの実装

5.2.29.1 ローパワーモード HAL モジュールの API の概要

ローパワーモードドライバーでは、モジュールの初期化、モジュールの値の設定と取得、モジュールの停止のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表も提供します。

ローパワーモード HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
init	g_lpm0.p_api->init(g_lpm0.p_cfg); LPM ドライバーモジュールを開き、構成構造体で渡された値に従って LPM ブロックを初期化します。
mstpcrSet	g_lpm0.p_api->mstpcrSet(value1, value2, value3, value4); すべてのモジュールストップコントロールレジスタの値を設定します。
mstpcrGet	g_lpm0.p_api->mstpcrSet(&value1, &value2, &value3, &value4); すべてのモジュールストップコントロールレジスタの値を取得します。
moduleStop	g_lpm0.p_api->moduleStop(module); モジュールを停止します。
moduleStart	g_lpm0.p_api->moduleStart(module); 指定されたモジュールを実行します。
operatingPowerModeSet	g_lpm0.p_api->operatingPowerModeSet(power, osc); 動作電力モードと発振器を設定します。
snoozeEnable	g_lpm0.p_api->snoozeEnable(rdx0_mode, dtc_mode, requests, triggers); スヌーズモードを設定して有効にします。
snoozeDisable	g_lpm0.p_api->snoozeDisable(); スヌーズモードを無効にします。

Function Name	API の呼び出し例と説明
lowPowerCfg	g_lpm0.p_api->lowPowerCfg(power_mode, output_port_enable, power_supply, io_port_state); 低電力モードを設定します。
wupenSet	g_lpm0.p_api->wupenSet(wupen_value); Wake Up 割り込み有効化レジスタ WUPEN の値を設定します。
wupenGet	g_lpm0.p_api->wupenGet(&wupen_value); Wake Up 割り込み有効化レジスタ WUPEN の値を取得します。
deepStandbyCancelRequestEnable	g_lpm0.p_api-> deepStandbyCancelRequestEnable (pin_signal, rising_falling); ディープスタンバイのキャンセル要求を有効にします。
deepStandbyCancelRequestDisable	g_lpm0.p_api-> deepStandbyCancelRequestDisable (pin_signal); ディープスタンバイのキャンセル要求を無効にします。
lowPowerModeEnter	g_lpm0.p_api-> lowPowerModeEnter (); WFI マクロを使用して、ローパワーモード（スリープ / スタンバイ / ディープスタンバイ）に移行します。ローパワーモードが解除されると、関数が復帰します。
versionGet	g_lpm0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している SSP リファレンスマニュアルを参照してください。*

エラーステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータの値が無効です。
SSP_ERR_INVALID_PTR	p_version が NULL です。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。関連するすべてのエラーコードの定義については、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』を参照してください。*

5.2.29.2 ローパワーモード HAL モジュールの動作の概要

LPM HAL モジュールの API は、ローパワーモードハードウェアペリフェラルを使用して、MCU 動作電力制御モードと MCU ローパワーモードの構成をサポートしています。

LPM 初期化

他の LPM 関数を呼び出す前に、LPM API 関数 init を呼び出す必要があります。init 関数は、内部変数とロックの初期化を処理します。

スリープローパワーモード

デフォルトでは、電源投入時にローパワーモードとしてスリープモードが有効になっています。スリープモードは、通常のプログラム実行モードに戻すために（MCU をスリープから復帰させる適切な割り込みまたはイベントを構成して有効にすることを除き）特別な構成を必要としないため、使用できる最も便利なローパワーモードです。スリープモードでは、SRAM、プロセッサレジスタ、およびハードウェアペリフェラルの状態がすべて維持され、スリープへの移行とスリープからの復帰に必要な時間は最小限で済みます。ThreadX® スレッドスケジューラが使用する Systick 割り込みを含め、割り込みが発生すると MCU デバイスはスリープモードから復帰します。他の関数の前に、LPM API 関数 init を呼び出す必要があります。LPM API 関数 lowPowerCfg を使用すると、ローパワーモードとしてスリープを使用するように MCU を構成できます。スリープモードに直接移行するには、LPM API 関数 lowPowerModeEnter を使用する必要があります。

```
/* HAL-only entry function */

#include "hal_data.h"

void hal_entry(void)

{

/* Not shown: Enable any interrupt to wake from sleep mode */

/* Configure LPM peripheral for sleep mode */

g_lpm_on_hal.p_api->lowPowerCfg(LPM_LOW_POWER_MODE_SLEEP,
```

```
LPM_OUTPUT_PORT_ENABLE_RETAIN,

LPM_POWER_SUPPLY_DEEPCUT0,

LPM_IO_PORT_NO_CHANGE);

/* Enter sleep mode */

g_lpm_on_hal.p_api->lowPowerModeEnter();

}
```

「ソフトウェアスタンバイモード」

ソフトウェアスタンバイモードでは、CPU、オンチップペリフェラルの大部分、およびすべての内部発振器が停止します。ただし、CPU 内蔵レジスタと SRAM データの内容、およびオンチップペリフェラルと I/O ポートの状態は保持されます。ソフトウェアスタンバイモードでは大半の発振器が停止するため、このモードを使用すると、消費電力を大幅に削減できます。スリープモードと同様、スタンバイモードの場合にも、スタンバイモードからの復帰のために割り込みまたはイベントを設定して有効化する必要があります。MCU をスリープモードから復帰させることができるイベントと割り込みの完全な一覧については、『MCU Synergy ハードウェアユーザーズマニュアル』の表 「Interrupt Sources To Transition To Normal Mode from Snooze and Software Standby Mode」を参照してください。ソフトウェアスタンバイモードに入る前に、 Wake Up 割り込み有効化レジスタ (WUPEN) を変更しておく必要があります。詳細は、「Wake Up 割り込み有効化レジスタ（WUPEN）」および 『MCU Synergy ユーザーズマニュアル：マイクロコントローラ』のソフトウェアスタンバイモードのセクションをご覧ください。

***注:** ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU のローパワーモードとしてスリープを構成する必要があります。詳細については、次の使用上の注意を参照してください。*

```
/* HAL-only entry function */

#include "hal_data.h"

#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)

void hal_entry(void)

{

uint32_t wupen_value = 0;

/* Not shown: Configuration of the interrupt or event to wake from standby */

/* Set bit in WUPEN register to allow mcu to wake from standby through a specifi
c interrupt or event. AGT1 underflow is used in this example. */

/* Get current value of WUPEN register */

g_lpm.p_api->wupenGet(&wupen_value);
```

```
/* Set wake by AGT1 underflow bit in WUPEN register */

g_lpm.p_api->wupenSet(wupen_value | WUPEN_AGT1_UNDERFLOW_MASK);

/* Configure LPM peripheral for standby mode */

g_lpm.p_api->lowPowerCfg(LPM_LOW_POWER_MODE_STANDBY,

LPM_OUTPUT_PORT_ENABLE_RETAIN,

LPM_POWER_SUPPLY_DEEPCUT0,

LPM_IO_PORT_NO_CHANGE);

/* Enter standby mode */

g_lpm.p_api->lowPowerModeEnter();

}
```

ソフトウェアスタンバイモードを使用したスヌーズモード

スヌーズモードでは、MCU ペリフェラルを使用して、MCU をローパワーステータスで維持しながら、基本タスクを実行することができます。 ADC や DTC、その他のペリフェラルはスヌーズモードで有効にできます。次のコードはスヌーズモードの有効化処理とスヌーズ中の DTC の実行処理を示します。タイマ AGT1 の設定処理は記されていません。MCU をソフトウェアスタンバイモードにすると、AGT1 の比較一致 A が検出されたときにスヌーズに移行します。DTC の転送が完了すると、MCU はスヌーズモードではなくなります。DTC の詳細については、DTC の使用上の注意を参照してください。ISDE によって生成されるソースファイルに g_transfer_on_dtc ドライバーを組み込んでプロジェクトを生成すると、ISDE は以下のインスタンス構造を構成します。

注: *ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU のローパワーモードとしてスリープを構成する必要があります。詳細については、次の使用上の注意を参照してください。*

```
#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)

/* DTC settings */

g_transfer.p_cfg->p_info->p_src = &my_tx_data[0];

g_transfer.p_cfg->p_info->p_dest = &my_rx_data[0];

g_transfer.p_cfg->p_info->length = TRANSFER_SIZE;

g_transfer.p_api->open(g_transfer.p_ctrl, g_transfer.p_cfg);

/* snooze settings */

lpm_snooze_rxd0_t rdx0_mode = LPM_SNOOZE_RXD0_FALLING_EDGE_IGNORE;
```

```
lpm_snooze_dtc_t dtc_mode = LPM_SNOOZE_DTC_ENABLE;

lpm_snooze_request_t requests = LPM_SNOOZE_REQUEST_AGT1_COMPARE_A;

lpm_snooze_end_t triggers = LPM_SNOOZE_END_DTC_TRANS_COMPLETE;

g_lpm.p_api->snoozeEnable(rdx0_mode, dtc_mode, requests, triggers);

/* standby settings */

lpm_low_power_mode_t power_mode = LPM_LOW_POWER_MODE_STANDBY;

lpm_output_port_enable_t output_port_enable = LPM_OUTPUT_PORT_ENABLE_RETAIN;

lpm_power_supply_t power_supply = LPM_POWER_SUPPLY_DEEPCUT0;

lpm_io_port_t io_port_state = LPM_IO_PORT_RESET;

g_lpm.p_api->lowPowerCfg(power_mode, output_port_enable, power_supply, io_port_s
tate);

/* Clear WUPEN */

g_lpm.p_api->wupenSet(0);

/* Wake from AGT1 interrupt underflow */

g_lpm.p_api->wupenSet(WUPEN_AGT1_UNDERFLOW_MASK);
```

ディープソフトウェアスタンバイモード

ディープソフトウェアスタンバイモードは S7G2 MCU でのみ利用可能であり、ソフトウェアスタンバイよりもローパワーモードです。リセットピンのネゲート、または API typedef lpm_deep_standby_t で記述されるスリープ解除イベントのセットのいずれかによるリセットが発生した場合、MCU は毎回ディープソフトウェアスタンバイモードから復帰します。以下のコードは、AGT1 アンダーフローを使用して MCU をディープスタンバイモードからスリープ解除し、ディープソフトウェアスタンバイモードを有効にしてそのモードに移行する方法を示します。AGT1 が期限切れになり MCU のリセットを引き起こすと、MCU はリセットされます。

注: *ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU のローパワーモードとしてスリープを構成する必要があります。詳細については、次の使用上の注意を参照してください。*

```
#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)

/* Deep standby wake signal (wake will cause reset) */

g_lpm.p_api->deepStandbyCancelRequestEnable(LPM_DEEP_STANDBY_AGT1,

LPM_CANCEL_REQUEST_EDGE_RISING);
```

```
/* Deep standby settings */

lpm_low_power_mode_t power_mode = LPM_LOW_POWER_MODE_DEEP;

lpm_output_port_enable_t output_port_enable = LPM_OUTPUT_PORT_ENABLE_RETAIN;

lpm_power_supply_t power_supply = LPM_POWER_SUPPLY_DEEPCUT0;

lpm_io_port_t io_port_state = LPM_IO_PORT_RESET;

g_lpm.p_api->lowPowerCfg(power_mode, output_port_enable, power_supply, io_port_s
tate);

/* Clear WUPEN */

g_lpm.p_api->wupenSet(0);

/* Wake from AGT1 interrupt underflow */

g_lpm.p_api->wupenSet(WUPEN_AGT1_UNDERFLOW_MASK);

g_lpm.p_api->lowPowerModeEnter();
```

ローパワーモード HAL モジュールの動作に関する注意事項

このモジュールを使用して動作モードを変更するには、CGC モジュールを使用する必要があります。CGC ドライバーに関する使用上の注意も確認してください。MCU の動作モードを適切に変更するために必要なイベントのシーケンスの詳細については、『MCU Synergy ハードウェアユーザーズマニュアル』の「動作電力低減機能」セクションを参照してください。

このドライバーを使用し、割り込みに基づいて LPM を構成するには、BSP を使用して割り込みを構成および有効化する必要があります。

LPM API 関数 init の呼び出しによって設定されるのは、MCU の動作モードのみです。init 関数は、低電力モードの設定は行いません。

ローパワーモードを構成するには、LPM API 関数 lowPowerCfg を使用します。構成が完了すると、LPM API 関数 lowPowerModeEnter を使用して、いつでもローパワーモードに移行することができます。

ディープスタンバイまたはスヌーズを使用するには、LPM API 関数 deepStandbyCancelRequestEnable および snoozeEnable を使用して、LPM を追加構成する必要があります。

ソフトウェアスタンバイモードに入る前に、Wake Up 割り込み有効化レジスタ (WUPEN) を変更しておく必要があります。詳細は、「Wake Up 割り込み有効化レジスタ（WUPEN）」および『MCU Synergy ユーザーズマニュアル : マイクロコントローラ』のソフトウェアスタンバイモードのセクションを参照してください。

サブ発振器電力制御モードに移行するには、API 関数 operatingPowerModeSet を呼び出す前に、CGC モジュールを使用して MOCO クロックと HOCO クロックを停止する必要があります。

メイン発振器またはメイン発振器ソースを備えた PLL をシステムクロックに使用している場合、スタンバイモードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があり

ます。このレジスタ設定は、CGC ドライバープロパティのメイン発振器待ち時間設定から変更可能です。詳細については、「電気的特性」の表 「ウェイクアップのタイミングおよび時間」を参照してください。

重要 : プロジェクトで ThreadX を使用する場合、アプリケーションは API 関数 lowPowerModeEnter を呼び出す直前にのみ LPM_LOW_POWER_MODE_STANDBY または LPM_LOW_POWER_MODE_DEEP に切り替える必要があります。 LPM_LOW_POWER_MODE_STANDBY または LPM_LOW_POWER_MODE_DEEP が ThreadX で使用されている場合は、MCU が LPM_LOW_POWER_MODE_STANDBY. から復帰した直後にローパワーモードを LPM_LOW_POWER_MODE_SLEEP に戻す必要があります。

ローパワーモード HAL モジュールの制限事項

- フラッシュの停止（コードフラッシュの無効化）はサポートされていません。S124、S128、S3A3、または S3A7 の 『Synergy ハードウェアユーザーズマニュアル』の 「フラッシュ操作制御レジスタ（FLSTOP）」を参照してください。
- ソフトウェアスタンバイモードでの SRAM 保持領域の縮小はサポートされていません。 S3A3 または S3A7 の 『Synergy ハードウェアユーザーズマニュアル』の 「省電力メモリ制御レジスタ （PSMCR）」を参照してください。
- MCU はデバッガが付属した状態のソフトウェアスタンバイモードやディープソフトウェアスタンバイモードに移行したり、状態を維持したりすることはできません。代わりに、MCU はデバッガによって、ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードから復帰することができます。ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードを適切にテストして確認するには、デバッガをアタッチしないで行う必要があります。
- メイン発振器またはメイン発振器ソースを備えた PLL をシステムクロックに使用している場合、スタンバイモードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があります。このレジスタ設定は、 CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の表「ウェイクアップのタイミングおよび時間」を参照してください。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.29.3 アプリケーションへのローパワーモードドライバーの組み込み

このセクションでは、 SSP コンフィギュレータを使用してローパワーモードドライバーをアプリケーションに組み込む方法について説明します。

注 : *このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、 SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

ローパワーモードドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（ローパワーモードドライバーのデフォルトの名前は <name> であり、これは次の表に示されています。この名前は、対応する **[Properties]** ウィンドウで変更できます）。次の図に示すように、ドライバーが追加されるとスレッドに表示されます。

ローパワーモード HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lpm0 Low Power Modes Driver on r_lpm	**Threads**	**New Stack> Driver> Power> Low Power Modes on r_lpm**

次の図に示すように、r_lpm のローパワーモード HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、複数のスタックで使用できるため 1 回のみ追加する必要があります。

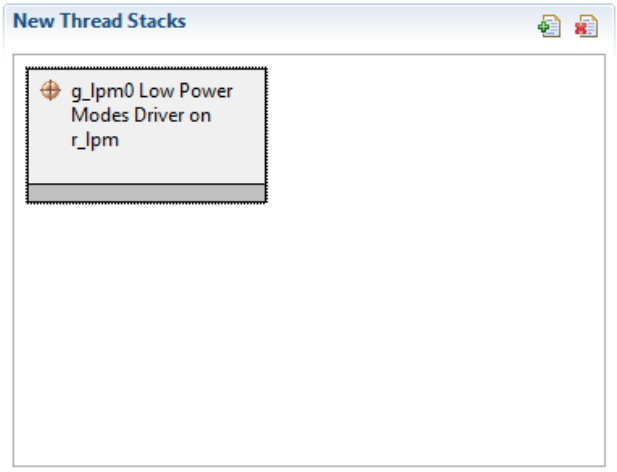


図 400: ローパワーモード HAL モジュールのスタック

5.2.29.4 ローパワーモード HAL モジュールの構成

ユーザーは必要な動作に合わせてローパワーモード HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータの **[Properties]** タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してく

ださい。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

***注：** 統合ソリューション開発環境を開き、ローパワーモードドライバーを作成し、次に示す構成テーブル設定を確認するのと並行してプロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_lpmv2 でのローパワーモードディープスタンバイドライバーの構成設定

ISDE Property	Setting	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します
Name	g_lpm0	モジュール名
Operating power mode	High speed operating mode, Middle speed operating mode, Low speed operating mode, Low voltage operating mode (Default: High speed operating mode)	動作電力モードを選択します
Sub oscillator mode	Sub oscillator mode not enabled, Sub oscillator mode enabled (Default: Sub oscillator mode not enabled)	サブ発振器モードを選択します

***注：** 上記の設定例とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

LPM のクロック設定

ローパワーモードモジュールには、構成可能なクロック入力はありません。

LPM のピン設定

ローパワーモードモジュールには、構成する必要のある特定の入力ピンと出力ピンはありません。

5.2.29.5 アプリケーションでのローパワーモードドライバーの使用

アプリケーションでローパワーモード HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して、ローパワーモード HAL モジュールを初期化します。
2) lowPowerCfg API を使用して、ローパワーモードを構成します。

3) lowPowerModeEnter API を使用して、ローパワーモードに移行します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

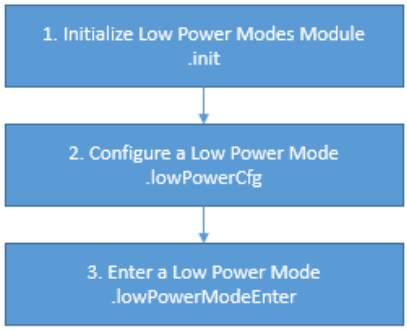


図 401: 通常のローパワーモードドライバーアプリケーションのフロー図

5.2.30 ローパワーモード V2 ドライバー

LPM V2 HAL モジュールは、ローパワーモードハードウェアペリフェラルを使用して、MCU 動作電力制御モードと MCU ローパワーモードの構成をサポートしています。

LPM V2 HAL モジュールは、以下のローパワーモードをサポートしています。

- ディープソフトウェアスタンバイモード
- ソフトウェアスタンバイモード
- スリープモード
- スヌーズモード

LPM V2 HAL モジュールは、ディープスタンバイモード時に、内部の電力供給制御と I/O ポートの状態の再設定を通じて消費電力の低減をサポートします。LPM V2 HAL モジュールは、MCU の他のハードウェアペリフェラルの無効化と有効化をサポートしています。

***注:** すべての MCU グループで、すべてのローパワーモード V2 モードを利用できるとは限りません。*

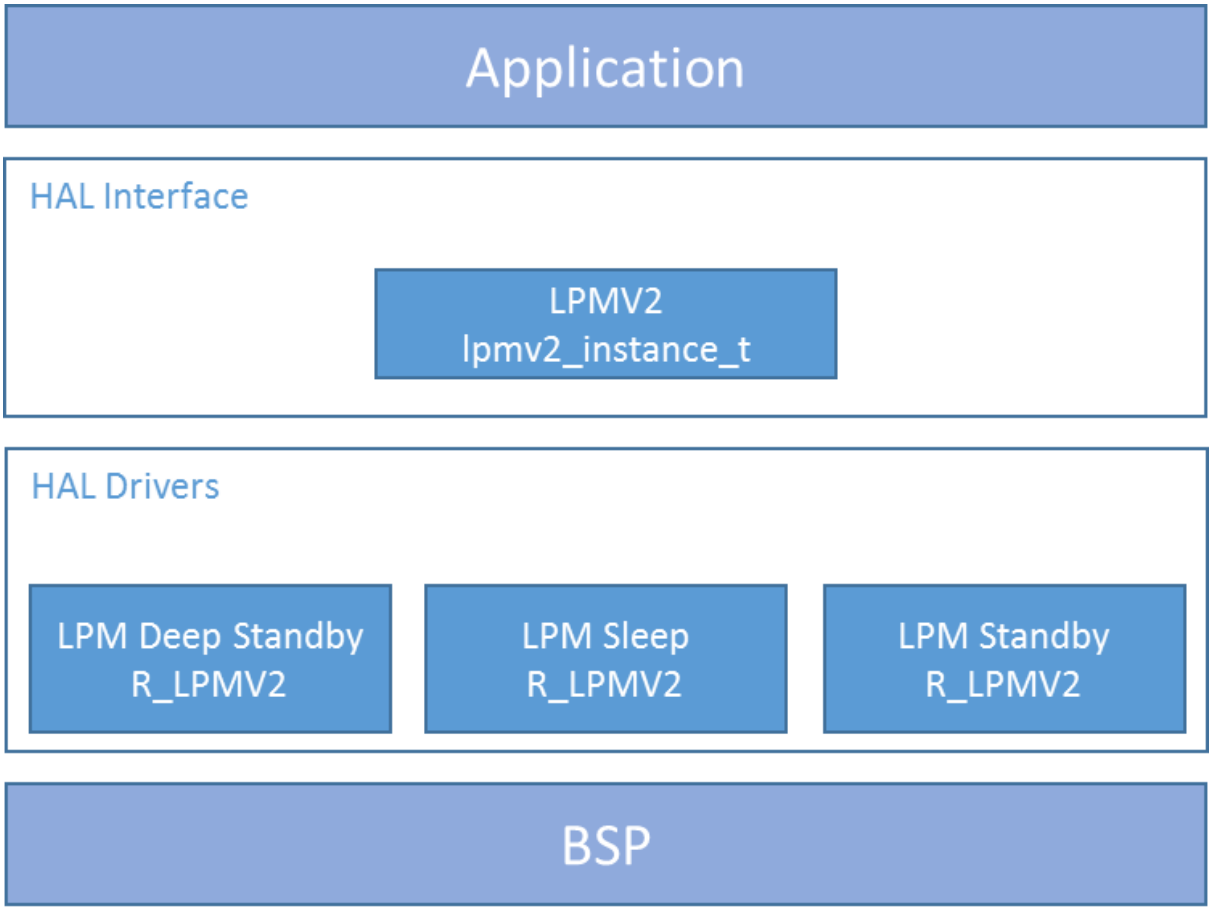


図 402: ローパワーモード V2 モジュールのブロック図

5.2.30.1 ローパワーモード V2 HAL モジュールの API の概要

ローパワーモード V2 HAL モジュールでは、動作の構成および低消費電力動作の有効化と無効化のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。戻りステータス値の表は API 要約表の後にあります。

注: ローパワーモード V2 HAL モジュールは、MCU の動作電力制御モードを処理しなくなります。この機能は、CGC HAL モジュールによって処理されます。

次の API の例では、スリープモードの使用を示します。「deep_standby」と「standby」を API の例の「sleep」に置き換えて、これらのモードの例を作成できます。

ローパワーモード V2 モジュールの API の要約

Function Name	API の呼び出し例と説明
init	g_lpmv2_sleep0.p_api->init(g_lpmv2_sleep0.p_cfg); LPM ドライバーモジュールを開き、構成構造体で渡された値に従って LPM ブロックを初期化します。
lowPowerCfg	g_lpmv2_sleep0.p_api->lowPowerCfg(g_lpmv2_sleep0.p_cfg); 低電力モードを設定します。
lowPowerModeEnter	g_lpmv2_sleep0.p_api->lowPowerModeEnter(void); WFI マクロを使用して、ローパワーモード（スリープ / スタンバイ / ディープスタンバイ）に移行します。ローパワーモードが解除されると、関数が復帰します。
versionGet	g_lpmv2_sleep0.p_api->versionGet(&version); ドライバーのバージョンを取得し、ポインタ version で示されている場所に格納します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_POINTER	ポインタが NULL です
SSP_ERR_INVALID_MODE	指定されたモードに対して設定が無効です

Name	説明
SSP_ERR_INVALID_HW_CONDITION	OPCMTSF フラグと SOPCMTSF フラグは、内部的に設定されたタイムアウト内ではクリアされません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.30.2 ローパワーモード V2 HAL モジュールの動作の概要

LPM V2 初期化

他の LPM V2 関数を呼び出す前に、LPM V2 API 関数 lpm_v2_api_t::init を呼び出す必要があります。init 関数は、内部変数とロックの初期化を処理します。

スリープローパワーモード

デフォルトでは、電源投入時にローパワーモードとしてスリープモードが有効になっています。スリープモードは、通常のプログラム実行モードに戻すために（MCU をスリープから復帰させる適切な割り込みまたはイベントを構成して有効にすることを除き）特別な構成を必要としないため、使用できる最も便利なローパワーモードです。任意の割り込みが発生すると、MCU デバイスはスリープローパワーモードから復帰します。スリープモードでは、SRAM、プロセッサレジスタ、およびハードウェアペリフェラルの状態がすべて維持され、スリープへの移行とスリープからの復帰に必要な時間は最小限で済みます。ThreadX® スレッドスケジューラが使用する Systick 割り込みを含め、割り込みが発生すると MCU デバイスはスリープモードから復帰します。他の関数の前に、LPM API 関数 lpm_v2_api_t::init を呼び出す必要があります。LPM API 関数 lpm_v2_api_t::lowPowerCfg, を使用すると、ローパワーモードとしてスリープを使用するように MCU を構成できます。スリープモードに直接移行するには、LPM API 関数 lpm_v2_api_t::lowPowerModeEnter を使用する必要があります。

ローパワーモードとしてスリープを構成し、ローパワーモードスリープに移行するコード例を、以下に示します。この例の LPM V2 スリープモジュールは、g_lpmv2_sleep0. という名前を使用します。

```
/* HAL-only entry function */
#include "hal_data.h"
void hal_entry(void)
{
ssp_err_t error = SSP_SUCCESS;
/* Initialize the LPM V2 Driver */
error = g_lpmv2_sleep0.p_api->init();
/* Handle error if any */
/* Configure LPM peripheral for sleep mode */
error = g_lpmv2_sleep0.p_api->lowPowerCfg(g_lpmv2_sleep0.p_cfg);
/* Handle error if any */
/* Entry sleep mode */
error = g_lpmv2_sleep0.p_api->lowPowerModeEnter();
/* Handle error if any */
}
```

LPM V2 のソフトウェアスタンバイモード

ソフトウェアスタンバイモードでは、CPU、オンチップペリフェラルの大部分、およびすべての内部発振器が停止します。保持されるのは、CPU 内蔵レジスタと SRAM データの内容、オンチップペリフェラルの状態、I/O ポートです。ソフトウェアスタンバイモードでは大半の発振器が停止するため、消費電力を大幅に削減できます。スリープモードと同様、スタンバイモードの場合にも、スタンバイモードからの復帰のために割り込みまたはイベントを構成して有効化する必要があります。

便宜上、スタンバイモードから復帰するために使用できるトリガが、**[Properties]** ウィンドウに列挙されています。複数のトリガを有効にできます。

ローパワーモードとしてスタンバイを構成し、ローパワーモードスタンバイに切り替えるコード例を、以下に示します。この例では、g_lpmv2_standby0 という名前の LPM V2 スタンバイモジュールを使用しています。スヌーズを有効にしたスタンバイモジュールを使用するためのコードもこれと同じです。

```
/* HAL-only entry function */
#include "hal_data.h"
void hal_entry(void)
{
ssp_err_t error = SSP_SUCCESS;
/* Initialize the LPM V2 Driver */
error = g_lpmv2_standby0.p_api->init();
/* Handle error if any */
/* Configure LPM peripheral for standby mode */
error = g_lpmv2_standby0.p_api->lowPowerCfg(g_lpmv2_standby0.p_cfg);
/* Handle error if any */
/* Entry standby mode */
error = g_lpmv2_standby0.p_api->lowPowerModeEnter();
/* Handle error if any */
}
```

LPM V2 のソフトウェアスタンバイモードを使用したスヌーズモード

スヌーズモードは、スタンバイモードの LPM V2 インスタンスで使用できます。**[Properties]** ウィンドウの [Choose the low power mode] で [Standby with Snooze Enabled] を選択します。スヌーズモードを MCU ペリフェラルで使用すると、MCU を低消費電力状態に維持しながら、基本タスクを実行することができます。スヌーズ設定は、**[Properties]** ウィンドウのスタンバイ設定の下にあります。ADC、DTC、他のペリフェラルを、スヌーズモードで有効にできます。レジスタ SELSR0 および IELSRn のイベントリンクコントローラ設定を除き、スヌーズのすべての設定は、スタンバイインスタンスの構成プロパティで使用できます。スヌーズは高度な機能とみなされています。

次の画面キャプチャで示すように、[Snooze Mode Settings]は、ローパワーモードの選択が**[Standby with Snooze Enabled]** である場合にのみ使用されます。

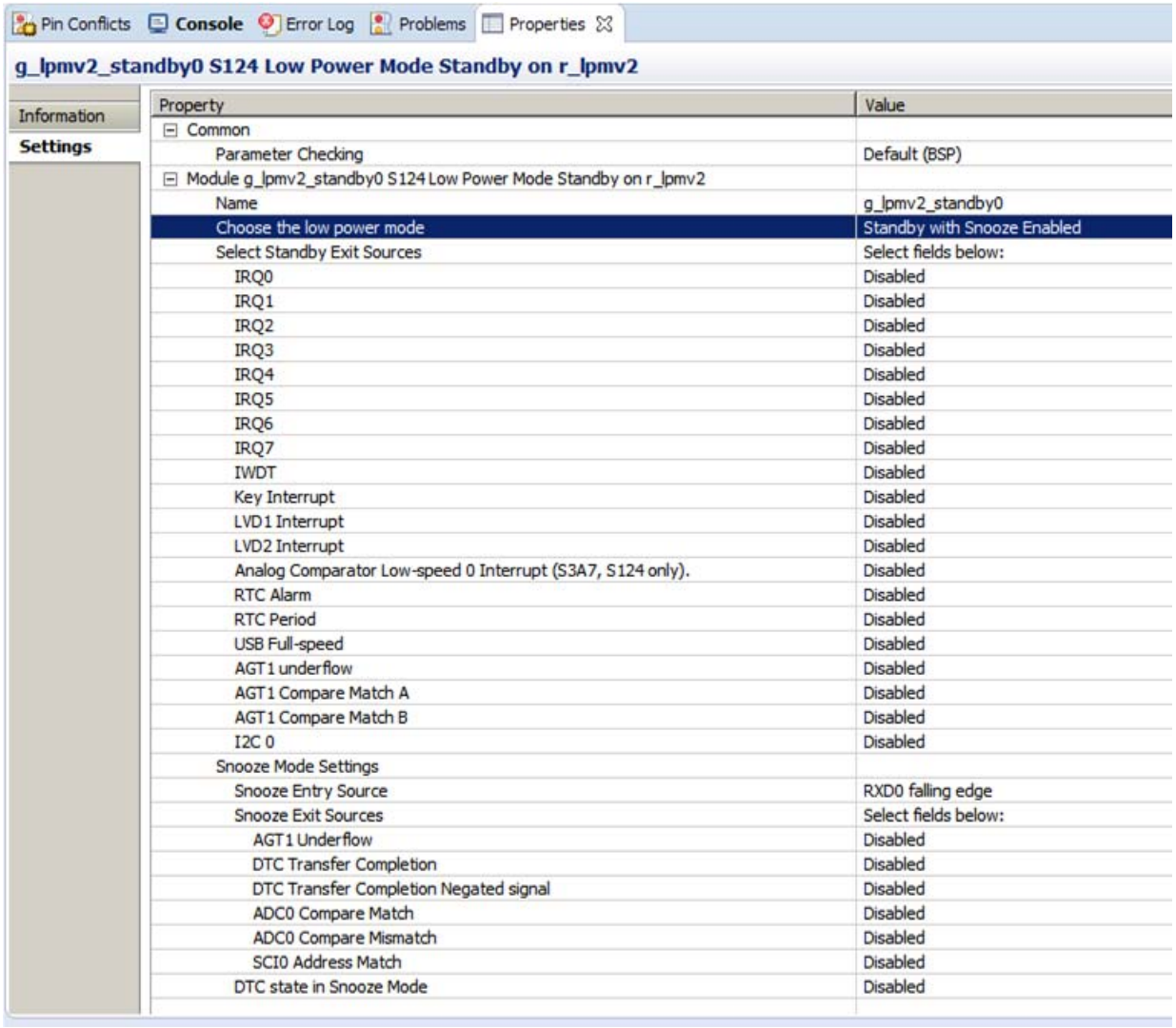


図 403: スヌーズの設定が有効になっているスタンバイモード

スヌーズはスタンバイモードの機能の 1 つです。これを使用すると、 MCU コアが命令を実行していなくても、一部のペリフェラルを動作させることができます。スヌーズモードに関連するローパワーモードペリフェラルのオプションを、次の図に示します。有効にできるスヌーズ開始ソースは 1 つだけです。複数のスヌーズ開始ソースを有効にすることはできません。 DTC ペリフェラルもスヌーズモードで有効にできます。

Snooze Mode Settings	
Snooze Entry Source	RXD0 falling edge
Snooze Exit Sources	Select fields below:
AGT1 Underflow	Disabled
DTC Transfer Completion	Disabled
DTC Transfer Completion Negated signal	Disabled
ADC0 Compare Match	Disabled
ADC0 Compare Mismatch	Disabled
SCI0 Address Match	Disabled
DTC state in Snooze Mode	Disabled

図 404: スヌーズモードの設定

LPM V2 のディープソフトウェアスタンバイモード

ディープソフトウェアスタンバイモードは、一部の MCU デバイスでのみ利用できます。リセットピンのネゲート、または LPM ディープスタンバイインスタンスの構成 **[Properties]** ウィンドウに表示されるスリープ解除イベントのセットのいずれかにより、MCU デバイスは常にリセットを経てディープソフトウェアスタンバイモードから復帰します。

便宜上、ディープスタンバイモードから復帰するために使用できるトリガが **[Properties]** ウィンドウに列挙されています。複数のトリガを有効にすることができます。一部のトリガには、関連するエッジタイプ（立ち上がりまたは立ち下がり）があります。これらのオプションについても、上および下のウィンドウに列挙されています。

ローパワーモードとしてディープスタンバイを構成し、ローパワーモードディープスタンバイに切り替えるコード例を、以下に示します。この例では、g_lpmv2_deep_standby0 という名前の LPM V2 ディープスタンバイモジュールを使用しています。

```
/* HAL-only entry function */
#include "hal_data.h"
void hal_entry(void)
{
ssp_err_t error = SSP_SUCCESS;
/* Initialize the LPM V2 Driver */
error = g_lpmv2_deep_standby0.p_api->init();
/* Handle error if any */
/* Configure LPM peripheral for deep sleep mode */
error = g_lpmv2_deep_standby0.p_api->lowPowerCfg(g_lpmv2_deep_standby0.p_cfg);
/* Handle error if any */
/* Entry deep sleep mode */
error = g_lpmv2_deep_standby0.p_api->lowPowerModeEnter();
/* Handle error if any */
```

ローパワーモード V2 HAL モジュールの動作に関する重要な注意事項と制限事項

このドライバーを使用して LPM ペリフェラルを構成し、割り込みによってスタンバイモードから MCU を復帰させるには、割り込みを構成し、その割り込みを使用するペリフェラルドライバーまたはフレームワークごとに割り込みを有効にする必要があります。たとえば、AGT1 アンダーフローによってスタンバイから復帰させるには、AGT タイマモジュールの構成で割り込みを有効にする必要があります。

メイン発振器またはメイン発振器ソースを備えた PLL をシステムクロックに使用している場合、スタンバイモードからの復帰時間は、MOSCWTCR レジスタでのメイン発振器待ち時間の設定により影響を受ける可能

性があります。このレジスタ設定は、CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の 「ウェイクアップのタイミングおよび時間」表を参照してください。

***注：** プロジェクトで ThreadX とローパワーモードのスタンバイ、ディープスタンバイ、またはスヌーズを有効にしたスタンバイを使用する場合、lpm_v2_api_t::lowPowerModeEnter API 関数を呼び出す直前に、lpm_v2_api_t::lowPowerCfg API 関数を呼び出す必要があります。これが必要なのは、ThreadX もアイドルループと tx_thread_sleep 関数でローパワーモードを使用するためです。ThreadX では、MCU デバイスがローパワーモードスリープ用に構成されていることが想定されています。プロジェクトで ThreadX とローパワーモードのスタンバイまたはスヌーズを有効にしたスタンバイを使用する場合、lpm_v2_api_t::lowPowerModeEnter 関数から戻った後、MCU デバイスがスタンバイから復帰してからローパワーモードをスリープに戻す必要があります。これが必要なのは、ThreadX もアイドルループと tx_thread_sleep 関数でローパワーモードを使用するためです。ThreadX では、MCU デバイスがローパワーモードスリープ用に構成されていることが想定されています。tx_thread_sleep 関数がプロジェクトで使用されているか、実行待ち状態のスレッドが常にあるとは限らない場合は、ローパワーモードをスリープに再構成するために、lpm_v2_api_t::lowPowerModeEnter の前に API 関数 lpm_v2_api_t::lowPowerCfg を再度呼び出す必要があります。*

動作状態およびローパワーモード V2 の MCU デバイスの予想消費電力の詳細については、『MCU Synergy ハードウェアユーザーズマニュアル』の 「電気的特性」セクションの 「動作電流とスタンバイ電流」を参照してください。

- フラッシュの停止 （コードフラッシュの無効化）はサポートされていません。『S1/S3 Synergy ハードウェアユーザーズマニュアル』の 「*フラッシュ操作制御レジスタ （FLSTOP）*」セクションを参照してください。
- ソフトウェアスタンバイモードでの SRAM 保持領域の縮小はサポートされていません。『S3 Synergy ハードウェアユーザーズマニュアル』の 「*省電力メモリ制御レジスタ （PSMCR）*」セクションを参照してください。
- MCU はデバッガが付属した状態のソフトウェアスタンバイモードやディープソフトウェアスタンバイモードに移行したり、状態を維持したりすることはできません。代わりに、MCU はデバッガによって、ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードから復帰することができます。ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードを適切にテストして確認するには、デバッガをアタッチしないで行う必要があります。
- メイン発振器またはメイン発振器ソースを備えた PLL をシステムクロックに使用している場合、スタンバイモードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があります。このレジスタ設定は、 CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の 「ウェイクアップのタイミングおよび時間」表を参照してください。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.30.3 アプリケーションへのローパワーモード V2 HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してローパワーモード V2 HAL モジュールをアプリケーションに組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない*

場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ローパワーモード V2 ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（ローパワーモードドライバーのデフォルトの名前は、g_lpmv2_<mode>0 です。この名前は、対応する **[Properties]** ウィンドウで変更できます）。

ローパワーモード V2 モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lpmv2_deep_standby0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	**New Stack> Driver> Power> Low Power Mode Deep Standby on r_lpmv2**
g_lpmv2_sleep0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	**New Stack> Driver> Power> Low Power Mode Sleep on r_lpmv2**
g_lpmv2_standby0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	**New Stack> Driver> Power> Low Power Mode Standby on r_lpmv2**

***注:** 選択シーケンスでは、S7G2 グループに使用できるモードの一覧が示されています。他の MCU については、ISDE で異なる選択シーケンスを使用できます。LPM V2 HAL モジュールを、複数のスレッド、または 1 つのスレッドと HAL/ 共通に追加する場合は、LPM V2 ドライバーのインスタンスごとに固有の名前を付けてください。名前を変更するには、[Modules] ウィンドウでドライバーのインスタンスをハイライトして、* **[PROPERTIES]** *ビューで名前のエントリを変更します。LPM V2 HAL モジュール構成の構造体のデフォルト設定も、このビューで変更できます。*

次の図に示すように、r_lpmv2 のローパワーモード V2 HAL モジュールがスレッドスタックに追加されると（図では、S7G2 MCU で使用できる 3 つのローパワーモードが示されています。これらのモジュールは、個別に、またはグループで追加できます。この図では完全を期すために 3 つとも示されています）、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。

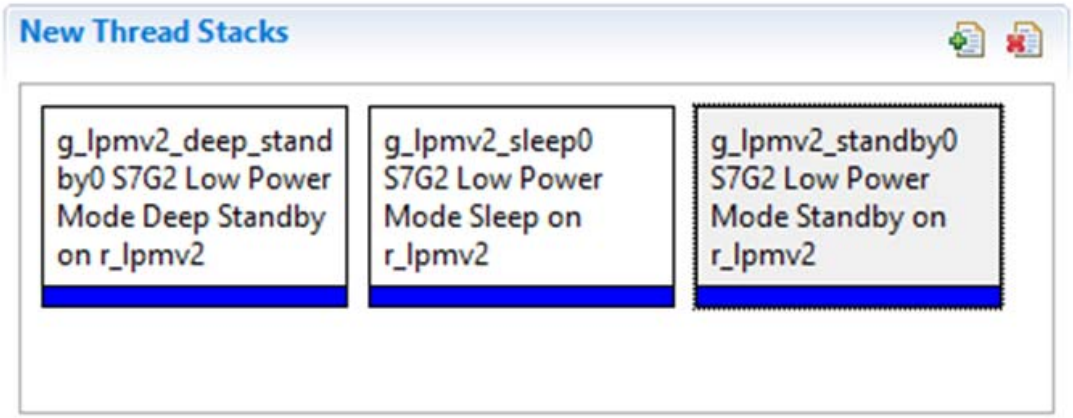


図 405: ローパワーモード V2 モジュールのスタック

5.2.30.4 ローパワーモード V2 HAL モジュールの構成

ユーザーは必要な動作に合わせてローパワーモード V2 ドライバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の **[Properties]** ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの **[Properties]** ウィンドウ内にあります。示されたモジュールを選択するだけで、**[Properties]** ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の **[Properties]** ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_lpmv2 でのローパワーモードディープスタンバイモジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します
Name	g_lpmv2_deep_standby0	モジュール名
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change (Default: No change)	スタンバイおよびディープスタンバイでの出力ポートの状態設定
Maintain or reset the IO port states on exit from deep standby mode	Maintain the IO port states, Reset the IO port states (Default: Maintain the IO port states)	出力ポート状態設定終了
Internal power supply control in deep standby mode	Maintain the internal power supply, Cut the power supply to standby RAM, low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit, Cut the power supply to LVDn, standby RAM, low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit (Default: Maintain the internal power supply)	ディープスタンバイモードでの内部電力供給制御の設定
Deep Standby Cancel Sources/Edges: Select Fields Below		
IRQ0	Enabled, Disabled (Default: Disabled)	IRQ0 の選択
IRQ0 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ0 エッジの選択

ISDE Property	Value	説明
IRQ1	Enabled, Disabled (Default: Disabled)	IRQ1 の選択
IRQ1 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ1 エッジの選択
IRQ2	Enabled, Disabled (Default: Disabled)	IRQ2 の選択
IRQ2 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ2 エッジの選択
IRQ3	Enabled, Disabled (Default: Disabled)	IRQ3 の選択
IRQ3 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ3 エッジの選択
IRQ4	Enabled, Disabled (Default: Disabled)	IRQ4 の選択
IRQ4 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ4 エッジの選択
IRQ5	Enabled, Disabled (Default: Disabled)	IRQ5 の選択

ISDE Property	Value	説明
IRQ5 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ5 エッジの選択
IRQ6	Enabled, Disabled (Default: Disabled)	IRQ6 の選択
IRQ6 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ6 エッジの選択
IRQ7	Enabled, Disabled (Default: Disabled)	IRQ7 の選択
IRQ7 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ7 エッジの選択
IRQ8	Enabled, Disabled (Default: Disabled)	IRQ8 の選択
IRQ8 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ8 エッジの選択
IRQ9	Enabled, Disabled (Default: Disabled)	IRQ9 の選択
IRQ9 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ9 エッジの選択

ISDE Property	Value	説明
IRQ10	Enabled, Disabled (Default: Disabled)	IRQ10 の選択
IRQ10 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ10 エッジの選択
IRQ11	Enabled, Disabled (Default: Disabled)	IRQ11 の選択
IRQ11 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ11 エッジの選択
IRQ12	Enabled, Disabled (Default: Disabled)	IRQ12 の選択
IRQ12 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ12 エッジの選択
IRQ13	Enabled, Disabled (Default: Disabled)	IRQ13 の選択
IRQ13 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ13 エッジの選択
IRQ14	Enabled, Disabled (Default: Disabled)	IRQ14 の選択

ISDE Property	Value	説明
IRQ14 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ14 エッジの選択
IRQ15	Enabled, Disabled (Default: Disabled)	IRQ15 の選択
IRQ15 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	IRQ15 エッジの選択
LVD1	Enabled, Disabled (Default: Disabled)	LVD1 の選択
LVD1 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	LVD1 エッジの選択
LVD2	Enabled, Disabled (Default: Disabled)	LVD2 の選択
LVD2 Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	LVD2 エッジの選択
RTC Interval	Enabled, Disabled (Default: Disabled)	RTC インターバルの選択
RTC Alarm	Enabled, Disabled (Default: Disabled)	RTC アラームの選択

ISDE Property	Value	説明
NMI	Enabled, Disabled (Default: Disabled)	NMI の選択
NMI Edge	Disabled, Rising Edge, Falling Edge (Default: Disabled)	NMI エッジの選択
USBFS	Enabled, Disabled (Default: Disabled)	USBFS の選択
UBSHS	Enabled, Disabled (Default: Disabled)	UBSHS の選択
AGT1	Enabled, Disabled (Default: Disabled)	AGT1 の選択

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_lpmv2 でのローパワーモードスリープドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します
Name	g_lpmv2_sleep0	モジュール名

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_lpmv2 でのローパワーモードスタンバイドライバーの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックを有効化または無効化します
Name	g_lpmv2_standby0	モジュール名
Choose the low power mode	Standby, Standby with Snooze enabled (Default: Standby)	ローパワーモードの選択
Output port state in standby and deep standby, applies to address output, data output, and other bus control pins	No Change, High Impedance state (Default: No Change)	出力ポート状態の選択
Select Standby Exit Sources: Select Fields below		
IRQ0:15	Enabled, Disabled (Default: Disabled)	IRQ0:16 の選択
IWDT	Enabled, Disabled (Default: Disabled)	IWDT の選択
Key Interrupt	Enabled, Disabled (Default: Disabled)	キー割り込みの選択
LVD1 Interrupt	Enabled, Disabled (Default: Disabled)	LVD1 の選択
LVD2 Interrupt	Enabled, Disabled (Default: Disabled)	LVD2 の選択

ISDE Property	Value	説明
Analog Comparator High-speed 0 Interrupt	Enabled, Disabled (Default: Disabled)	アナログコンパレータの選択
RTC Period	Enabled, Disabled (Default: Disabled)	RTC 周期の選択
RTC Alarm	Enabled, Disabled (Default: Disabled)	RTC アラームの選択
USBFS	Enabled, Disabled (Default: Disabled)	USBFS の選択
UBSHS	Enabled, Disabled (Default: Disabled)	UBSHS の選択
AGT1 Underflow	Enabled, Disabled (Default: Disabled)	AGT1 アンダーフローの選択
AGT1 Compare Match A	Enabled, Disabled (Default: Disabled)	AGT1 CMA の選択
AGT1 Compare Match B	Enabled, Disabled (Default: Disabled)	AGT1 CMB の選択
I2C 0	Enabled, Disabled (Default: Disabled)	I2C 0 の選択
Snooze Mode Settings		

ISDE Property	Value	説明
Snooze Entry Source	RXD0 falling edge, IRQ0:15, KINT (Key Interrupt), ACMPHS0 (High-speed Analog Comparator 0), RTC Alarm, RTC Period, AGT1 Underflow, AGT1 Compare Match A, AGT1 Compare Match B (Default: RXD0 falling edge)	スヌーズモード開始のソース
Snooze Exit Sources: Select fields below		
AGT1 Underflow	Enabled, Disabled (Default: Disabled)	AGT1 アンダーフローの選択
DTC Transfer Completion	Enabled, Disabled (Default: Disabled)	DTC 転送完了の選択
DTC Transfer Completion Negated signal	Enabled, Disabled (Default: Disabled)	DTC 転送完了ネゲート信号の選択
ADC0 Compare Match	Enabled, Disabled (Default: Disabled)	ADC0 比較一致の選択
ADC0 Compare Mismatch	Enabled, Disabled (Default: Disabled)	ADC0 比較不一致の選択
ADC1 Compare Match	Enabled, Disabled (Default: Disabled)	ADC1 比較一致の選択
ADC1 Compare Mismatch	Enabled, Disabled (Default: Disabled)	ADC1 比較不一致の選択

ISDE Property	Value	説明
SCI0 Address Match	Enabled, Disabled (Default: Disabled)	SCI0 アドレス一致の選択
DTC State Selection		
DTC state in Snooze Mode	Enabled, Disabled (Default: Disabled)	スヌーズモードの DTC の状態の選択

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

デフォルト以外の設定が望ましい場合もあります。たとえば、低消費電力状態の開始または終了に異なる状態を選択する場合に役立つことがあります。

ローパワーモード V2 HAL モジュールのクロック構成

ローパワーモード V2 周辺モジュールには、選択可能なクロックソースはありません。

ローパワーモード V2 HAL モジュールのピン構成

ローパワーモード V2 周辺モジュールでは、ピン配置は必要ありません。ピン機能の選択は、プロパティ構成ウィンドウで行います。

5.2.30.5 アプリケーションでのローパワーモード V2 HAL モジュールの使用

アプリケーションでローパワーモード V2 HAL モジュールを使用する一般的な手順は次のとおりです。

1) lpm_v2_api_t::init API を使用して、ローパワーモード V2 HAL モジュールを初期化します
2) lpm_v2_api_t::lowPowerCfg API を使用して、ローパワーモードを構成します
3) lpm_v2_api_t::lowPowerModeEnter API を使用して、ローパワーモードに移行します

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

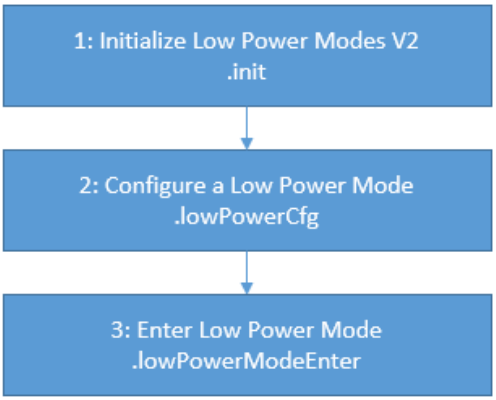


図 406: 通常のローパワーモード V2 モジュールアプリケーションのフロー図

5.2.31 低電圧検出ドライバー

LVD HAL モジュールは、次の機能をサポートしています。

- 電圧検出入力としての VCC
- 1 つのビルド時に構成可能な低電圧検出 （OFS1 レジスタを使用）
- 2 つの実行時に構成可能な低電圧検出
- 2 つの結果フラグ。 1 つはしきい値チェック用、もう 1 つは現在の状態用
- 割り込みとポーリングイベントチェックの両方をサポート

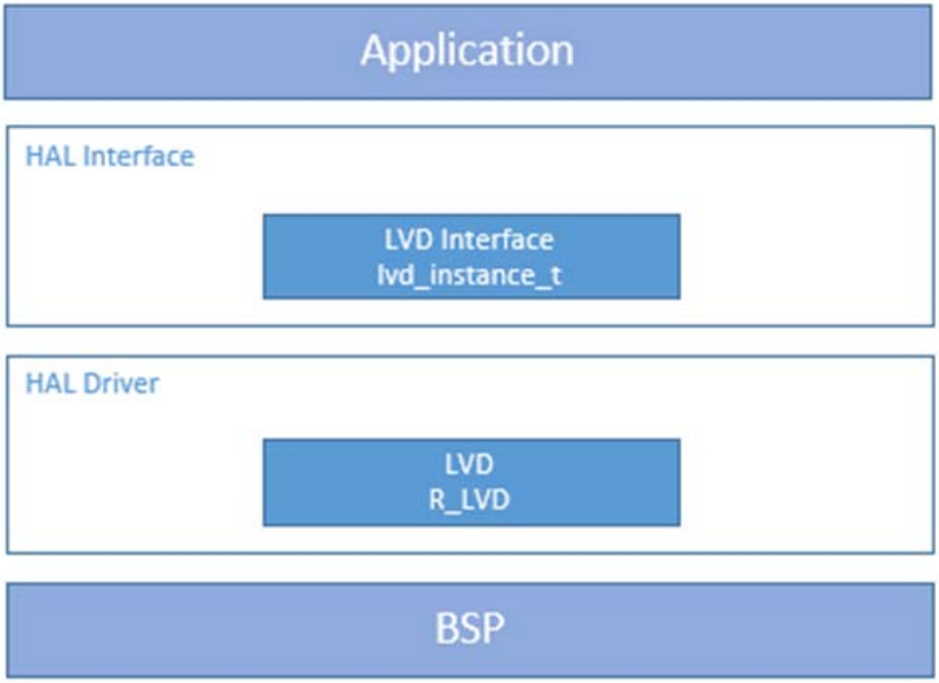


図 407: LVD HAL モジュールのブロック図

5.2.31.1 LVD HAL モジュールの API の概要

LVD HAL モジュールでは、オープン、クローズ、 statusGet、 statusClear のための API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

LVD HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_lvd.p_api->open(g_lvd.p_ctrl, g_lvd.p_cfg; 渡された構成構造体に従って、低電圧検出ドライバーを初期化します。構成構造体に従い、 LVD を有効化します。

Function Name	API の呼び出し例と説明
statusGet	g_lvd.p_api->statusGet(g_lvd.p_ctrl, &monitor_status); 監視の現在の状態を取得します（しきい値の超過が検出された、現在電圧が範囲内にある）。LVD 監視の状態のポーリングにいつでも使用できます。LVD_RESPONSE_NONE. に設定された lvd_response_t でペリフェラルが初期化された場合に、使用する必要があります。
statusClear	g_lvd.p_api->statusClear(g_lvd.p_ctrl); 監視のラッチ状態をクリアします。lvd_response_t を LVD_RESPONSE_NONE に設定した状態でペリフェラルを初期化した場合は、必ず使用する必要があります。
close	g_lvd.p_api->close(g_lvd.p_ctrl, g_lvd.p_cfg); LVD を無効化します。ドライバーインスタンスを閉じます。
versionGet	g_lvd.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_IN_USE	ドライバーは既に開かれているか、またはハードウェアロックを取得できません
SSP_ERR_NOT_OPEN	ユニットが開いていません。
SSP_ERR_ASSERTION	構成値が無効です
SSP_ERR_INVALID_MODE	この構成で試みたモードが無効の場合。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.31.2 LVD HAL モジュールの動作の概要

LVD HAL モジュールは、Synergy MCU での LVD 監視の構成と操作をサポートします。LVD HAL モジュールは、単一の LVD 監視を完全に構成するために必要なすべての情報を含む構成構造体を提供します。LVD0 の監視を除き、LVD 監視インスタンスごとに LVD HAL モジュールのインスタンスが 1 つ存在します。LVD0 監視は、実行時に構成することはできず、OFS1 レジスタによりコンパイル時に構成する必要があります。

LVD1 監視と LVD2 監視はどちらも、このモジュールを使用して実行時に構成できます。open 関数を使用すると、1 回の関数呼び出しで LVD 監視を構成して有効にすることができます。close 関数は LVD 監視を無効にします。statusGet 関数は、LVD 監視の現在のステータスを返します。モジュールがポーリングモードの場合は（つまり、LVD 監視の割り込みが有効になっていない場合）、statusGet 関数を使用する必要があります。監視のステータスは 2 つのフラグで構成されます。最初のフラグは crossing_detected と呼ばれるラッチフラグで、監視対象の電圧が電圧しきい値を超えたかどうかを示します。ポーリングモードでは、このフラグは statusClear への呼び出しを通じてクリアする必要があります。LVD 割り込みが使用されている場合は、このフラグを明示的にクリアする必要はありません。ユーザーコールバック関数が呼び出された後、このフラグは LVD 割り込みの中でドライバーのコードによりクリアされます。2 番目のフラグである current_state, は監視対象の電圧の電圧しきい値に対する瞬間的な状態です。このフラグはラッチされず、その状態は監視対象の電圧が変化すると変わります。

LVD HAL モジュールは、1 つまたは複数の LVD ペリフェラル割り込みを有効にするように構成できます。割り込みを使用する場合、ユーザーはその監視に対するコールバック関数を提供する必要があります。LVD 監視ごとに異なるコールバックルーチンを提供する必要があります。

LVD HAL モジュールには、BSP によって提供される機能が必要です。このドライバーは、BSP のハードウェアロックを使用して、レジスタのロックおよび割り込みの有効化とクリアを行います。

LVD HAL モジュールの動作に関する重要な注意事項と制限事項

- これらの LVD 設定に対して適切な値を選択した後、ユーザーはプロジェクトに対して LVD HAL モジュールの open API 関数を呼び出すコードを追加する必要があります。この関数は、アプリケーションの早い段階で 1 回呼び出す必要があります。
- LVD　監視の構成を変更する必要があるときは常に、モジュールを閉じて開くことができます。LVD open API 関数を呼び出すと、指定された LVD が監視する LVD ハードウェアペリフェラルが構成されて有効になります。
- close 関数は、LVD 監視を無効にして、ドライバーを閉じます。
- このモジュールを使用して LVD を構成し、割り込みを作成するには、モジュールの **[Properties]** タブで対応する割り込みが有効にされている必要があります。
- LVD 割り込みを使用するとき、コールバック関数は必須ではありませんが推奨されます。
- 各 LVD 割り込みに対する固有のコールバック関数は必須ではありませんが推奨されます。
- クロックシステムの初期化、構成、実行時の変更は、このモジュールの外で処理されます。このドライバーは、ユーザーが選択したサンプルクロック除数に基づいて、デジタルフィルタサンプルクロックを変更します。このデジタルフィルタサンプルクロックは、低速オンチップ発振器（LOCO）のシステムクロックから得られます。
- すべての MCU ですべての電圧しきい値が利用可能とは限りません。

- LVD 監視に対する VCC 入力のデジタルフィルタリングが、すべての Synergy MCU デバイスで利用可能であるとは限りません。
- LVD ドライバーには、BSP によって提供される機能が必要です。このドライバーは、BSP によって提供されるハードウェアロックを使用して、レジスタのロックおよび割り込みの有効化とクリアを行います。
- LVD 監視を構成して有効にするには、特定のタイミング制約とレジスタライト順序が必要です。このような制約のため、電圧監視を構成して有効にするプロセス全体は、単一の関数によって実行されるときが最も効率的です。 open API 関数は、構成を実行して監視を有効にし、タイミングとレジスタライト順序の制約を適切に強制します。
- すべての Synergy MCU シリーズに含まれる**オプション設定メモリ**を使用して、リセット後のペリフェラルの動作状態を設定できます。OFS（オプション関数選択）は、IWDT、WDT、LVD、および CGC 高速オンチップ発振器 （HOCO）の状態の設定に使用できます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

***注:** LVD0 は、OFS レジスタを使用することによってのみ構成できます。LVD0 は、レジスタスタートモードをサポートしていません。*

低電圧検出用の OFS の設定

Control	説明	
LVD0 detection level	S7 および S5 シリーズ 2.94V 2.87V 2.80V	S3 および S1 シリーズ 3.84V 2.82V 2.51V 1.90 1.70V
LVD0 detection start	有効にすると、リセット後に LVD0 を自動的に開始します	

OFS レジスタの値は、 Synergy 構成エディタの [BSP] タブの **[Properties]** ダイアログボックスで設定します。

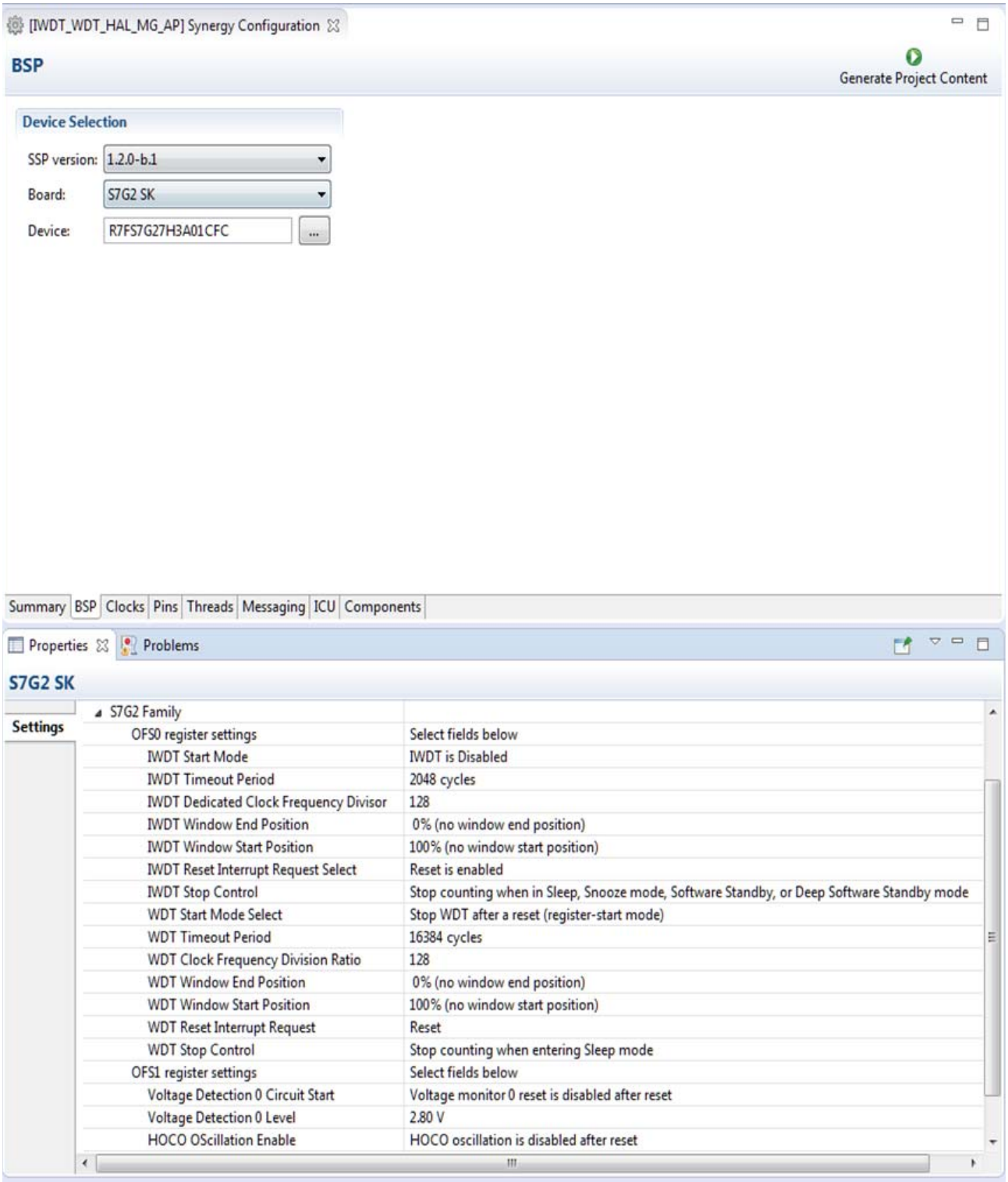

[IWDT_WDT_HAL_MG_AP] Synergy Configuration
BSP
Generate Project Content
Device Selection
SSP version: 1.2.0-b.1
Board: S7G2 SK
Device: R7FS7G27H3A01CFC
Summary BSP Clocks Pins Threads Messaging ICU Components
Properties Problems
S7G2 SK
Settings
S7G2 Family
OFS0 register settings Select fields below
IWDT Start Mode IWDT is Disabled
IWDT Timeout Period 2048 cycles
IWDT Dedicated Clock Frequency Divisor 128
IWDT Window End Position 0% (no window end position)
IWDT Window Start Position 100% (no window start position)
IWDT Reset Interrupt Request Select Reset is enabled
IWDT Stop Control Stop counting when in Sleep, Snooze mode, Software Standby, or Deep Software Standby mode
WDT Start Mode Select Stop WDT after a reset (register-start mode)
WDT Timeout Period 16384 cycles
WDT Clock Frequency Division Ratio 128
WDT Window End Position 0% (no window end position)
WDT Window Start Position 100% (no window start position)
WDT Reset Interrupt Request Reset
WDT Stop Control Stop counting when entering Sleep mode
OFS1 register settings Select fields below
Voltage Detection 0 Circuit Start Voltage monitor 0 reset is disabled after reset
Voltage Detection 0 Level 2.80 V
HOCO OScillation Enable HOCO oscillation is disabled after reset

図 408: OFS のレジスタ設定

5.2.31.3 アプリケーションへの LVD HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに LVD HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

LVD ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（LVD HAL モジュールのデフォルトの名前は g_lvd. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

表 4 LVD の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lvd Low Voltage Detection Driver on r_lvd	Threads	New Stack> Driver> Power> Low Voltage Detection Driver on r_lvd

次の図に示すように、r_lvd の LVD ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

図 409: LVD HAL モジュールのスタック

5.2.31.4 LVD HAL モジュールの構成

ユーザーは必要な動作に合わせて LVD HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を構成する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_lvd での LVD HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_lvd	LVD ドライバーモジュールの名前
Monitor Number	1	監視番号の選択
Digital Filter, enable by selecting a valid sample clock rate (S7G2 only).	Enabled with clock LOCO/2/4/8/16, Disabled (Default: Disabled)	デジタルフィルタの選択
Voltage Threshold	2.85V (Vdet1_13) (S7G2 only).	電圧しきい値レベルの選択
Detection Response, either reset, interrupt, non-maskable interrupt, or no response (polling mode).	Maskable interrupt is triggered when the voltage crosses the detection threshold.The non-maskable interrupt is triggered when voltage crosses the detection threshold.The MCU resets when the voltage falls below the detection threshold.No response: the driver is in polled mode (using statusGet and statusClear functions.)(Default: Maskable interrupt)	検出レスポンスの選択

ISDE Property	Value	説明
Voltage Slope	Threshold crossing detected with decreasing voltage.Threshold crossing detected with increasing voltage.Threshold crossing detected with increasing or decreasing voltage.(Default: Threshold crossing detected with decreasing voltage.)	しきい値検出の電圧スロープの方向
Negation of Monitor Signal	Negation of reset signal is based on delay from the reset.Negation of reset signal is based on delay from voltage returning to normal range.(Default: Negation of reset signal is based on delay from reset.)	ネゲートオプションの選択
Monitor Interrupt Callback	NULL	割り込みコールバック関数の名前
LVD Monitor Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	割り込み優先順位の設定

***注:** 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールの（デフォルト以外の）設定が望ましい場合もあります。たとえば、ターゲットシステムに適した電圧レベルを選択すると役に立つことがあります。

LVD HAL モジュールのクロック構成

クロックシステムのクロックの初期化、構成、実行時変更は、このモジュール外で処理されます。このモジュールは、ユーザーが選択したサンプルクロックの除数に基づいてデジタルフィルタサンプルクロックのみを変更します。このデジタルフィルターサンプルクロックは、 LOCO システムクロックから得られたものです。

LVD HAL モジュールのピン構成

LVD HAL モジュールは、 VCC ピンのみで電圧を測定し、構成する必要はありません。

5.2.31.5 アプリケーションでの LVD HAL モジュールの使用

アプリケーションで LVD HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 LVD HAL モジュールを初期化します。
2) ソフトウェアポーリングを使用する場合は、 statusGet API で LVD の状態フラグを監視し、適切に処理します。割り込みモードを使用する場合は、監視番号と状態の両方を返すコールバック関数の中で適切に処理します。
3) 必要に応じて処理します

4) close API を使用して、 LVD インスタンスを閉じます

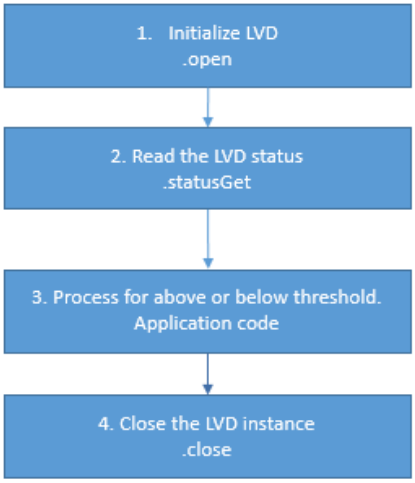


図 410: 一般的な LVD HAL モジュールのソフトウェアポーリングモードでのフロー図

5.2.32 PDC ドライバー

PDC HAL モジュールは、Synergy マイクロコントローラ上の PDC を制御し、以下の主要機能を備えています。

- 接続された構成済みのカメラからのキャプチャをサポートします。
- キャプチャの完了を CPU に通知するコールバックをサポートします
- キャプチャバッファへのポインタを提供します
- コールバックをトリガしたイベントを示す値を提供します

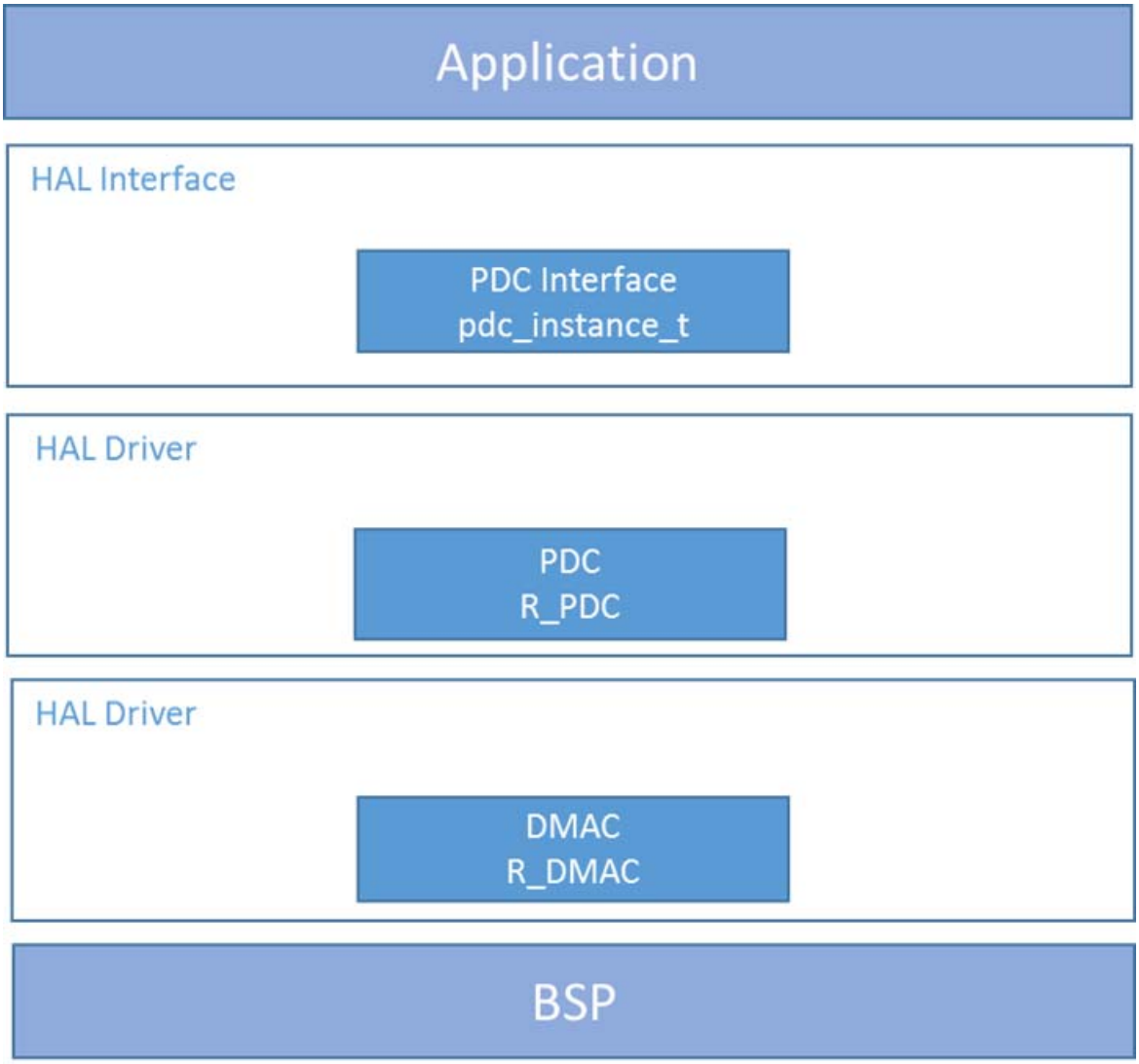


図 411: PDC HAL モジュールのブロック図

5.2.32.1 PDC HAL モジュールの API の概要

PDC HAL モジュールでは、データキャプチャをオープン、クローズ、開始するための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

PDC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_pdc.p_api->open(g_pdc.p_ctrl, g_pdc.p_cfg) 初期設定。
close	g_pdc.p_api->close(g_pdc.p_ctrl)) ドライバーを閉じ、再構成を許可します。電力消費を低減できます。
captureStart	g_pdc.p_api->captureStart(g_pdc.p_ctrl, &buffer) キャプチャを開始します。
stateGet	g_pdc.p_api->stateGet(g_pdc.p_ctrl, &state_data) VSYNC ピンおよび HSYNC ピンの状態を取得します。
versionGet	g_pdc.p_api->versionGet(&version) version ポインタを使用して API のバージョンを返します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_ASSERTION	パラメータ p_ctrl または p_sample が NULL です。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です。
SSP_ERR_NOT_OPEN	ユニットが開いていません。
SSP_ERR_ALREADY_OPEN	ユニットは既に開いています。
SSP_ERR_HW_LOCKED	BSP ハードウェアロックを予約できません。

Name	説明
SSP_ERR_TIMEOUT	リセット動作がタイムアウトしました。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.32.2 PDC HAL モジュールの動作の概要

キャプチャ動作には、Synergy マイクロコントローラに接続された構成済みの外部カメラが必要です。キャプチャを実行する前に、カメラが構成されていて、マイクロコントローラへの PIXCLK クロック入力が生成されていることが重要です。一部のインスタンスでは、カメラの構成には実行中のクロック入力が必要となる場合があります。

カメラを初期化する前に、 open を呼び出し、 PDC からカメラへの PCKO クロック出力を構成して開始します。カメラの構成が済んだら、 captureStart を呼び出してイメージをキャプチャできます。カメラモジュールの構成には、 I^2C または SPI インタフェースの使用が必要となる場合があります。

PDC HAL モジュールの動作に関する重要な注意事項と制限事項

ほとんどのインスタンスにおいて、カメラおよび PDC からのデータレートは、割り込みサービスルーチン（ISR）内の CPU による処理に対して速すぎます。そのため、このモジュールで PDC およびメモリからの高速転送を実行するには、 DMAC に転送ドライバーの実装が必要です。

以下の状況で割り込みを生成するには、PDC フレーム終了割り込みと PDC エラー割り込みの両方を有効にする必要があります。

- イメージがキャプチャされた場合の割り込み （フレーム終了）
- エラーが発生した場合の割り込み

データバッファの設定

p_buffer が NULL 以外に設定されている場合、イメージデータを格納するために 1 つまたは複数のデータバッファが作成されます。各バッファのサイズは、以下の式を使って計算されます。

バッファサイズ （バイト） = x_capture_pixels x y_capture_pixels x bytes_per_pixel

解像度の高いカメラの場合、キャプチャしたイメージのデータが膨大な量になることがあります。このため、 buffer(s) を外部メモリ （SDRAM など）に置くことが必要になる場合があります。外部メモリを使用する場合は、バス帯域を考慮してください。

たとえば、PDC を使用して高フレームレートカメラで SDRAM へのイメージキャプチャを行い、高リフレッシュレートの LCD ディスプレイ用にディスプレイバッファを保持するのに SDRAM を使用すると、 PDC からメモリへのデータボトルネックが発生し、オーバーランエラー状態になります。

注：p_buffer を NULL に設定した場合、キャプチャされたイメージデータの格納用にメモリは割り当てられません。ユーザーは責任をもって、PDC に十分なサイズの適切なメモリを確保する必要があります。PDC は接続された LCD パネルのディスプレイバッファに直接キャプチャを行うことができます。

PDC HAL モジュールの動作に関する制限事項については、最新の SSP リリースノートを参照してください。

5.2.32.3 アプリケーションへの PDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに PDC HAL モジュールを組み込む方法について説明します。

***注：** このプロセスを理解するには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

PDC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（PDC のデフォルトの名前は r_pdc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

CRC の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_pdc0 PDCDriver on r_pdc	Threads	New Stack> Driver> Graphics> PDC Driver on r_pdc

次の図に示すように、r_pdc の PDC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

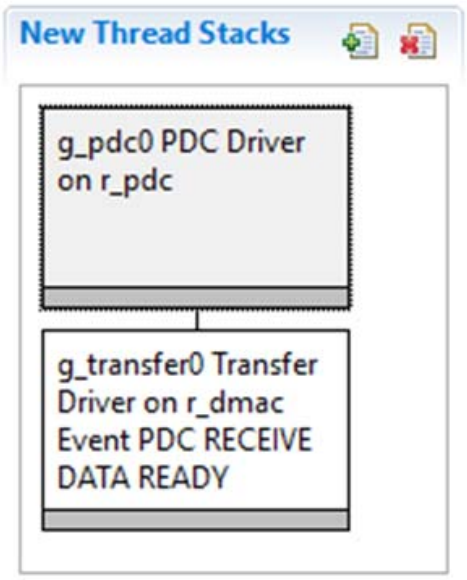


図 412: PDC HAL モジュールのスタック

5.2.32.4 PDC HAL モジュールの構成

ユーザーは必要な動作に合わせて PDC HAL モジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。変更できないプロパティは「ロック」されており、 ISDE の [Properties] ウィンドウでは「ロックされた」プロパティを示すロックアイコンで見分けられます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、 SSP での開発の表と裏を学習するための有用な実践的アプローチになる可能性があります。*

r_pdc での PDC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_pdc0	PDC モジュールインスタンスの名前。任意の C シンボルを指定します。
Name of the data buffer to store image data	g_user_buffer	作成するデータバッファの名前を指定するか、ユーザーが PDC ドライバー外にデータバッファを作成する場合は NULL に設定します。
Section where data buffer is allocated	sdram	イメージデータバッファの RAM セクションを指定します。通常は、BSS（内部 RAM）または SDRAM です。
Number of bytes per pixel	2	キャプチャするイメージデータのピクセルごとのバイト数を指定します。
Number of image data buffers	1	作成するバッファの数を指定します。

ISDE Property	Value	説明
Clock Divider	CLK/2, CLK/4, CLK/6, CLK/8, CLK10, CLK12, CLK14, CLK16 Default: CLK/2	PDC に対するクロック入力のクロック分周器を指定します。
Endian of image data	Little, Big Default: Little	キャプチャするイメージ データのエンディアンを指定します。
HYSNC signal polarity	High, Low Default: High	HSYNC 信号のアクティブ極性を指定します。
VSYNC signal polarity	High, Low Default: High	VSYNC 信号のアクティブ極性を指定します。
Number of pixels to capture horizontally	640	キャプチャする水平ピクセル数。
Number of lines to capture vertically	480	キャプチャする垂直ライン数。
Horizontal pixel to start capture from	0	イメージデータのキャプチャを開始する水平位置。カメラのネイティブ解像度を下回るイメージのキャプチャが可能になります。
Line to start capture from	0	イメージデータのキャプチャを開始する垂直ライン。カメラのネイティブ解像度を下回るイメージのキャプチャが可能になります。

ISDE Property	Value	説明
Callback	g_pdc_user_callback	ユーザーコールバック関数をopenで登録できます。このコールバック関数が指定されている場合、フレームがキャプチャされ処理可能な状態になるたびに、割り込みサービスルーチン(ISR)から呼び出されます。 警告：コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。 ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。
Frame End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ドライバーには有効な割り込み優先順位設定が必要です。無効な場合は機能しません。
PDC Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ドライバーには有効な割り込み優先順位設定が必要です。無効な場合は機能しません。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、デフォルトとは異なるクロック分周器を選択すると役に立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

PDC HAL モジュールのローレベルドライバー用の設定を構成するとき、通常は、ローレベルドライバーのデフォルト設定から変更する必要がある設定の数は少なく、スレッドスタックブロックでは赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

DMAC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータの選択。
Name	g_transfer0	ドライバー名。
Mode	Block	モードの選択。
Transfer Size	4 Bytes	転送サイズの選択。
Destination Address Mode	Incremented	宛先アドレスモードの選択。
Source Address Mode	Fixed	ソースアドレスモードの選択。
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択。
Destination Pointer	NULL	宛先ポインタの選択。
Source Pointer	NULL	ソースポインタの選択。
Number of Transfers	8	転送回数の選択
Number of Blocks (Valid only in Block Mode)	1	ブロック数の選択。
Activation Source (Must enable IRQ)	Event PDC RECEIVE DATA READY	アクティベーションソースの選択。
Auto Enable	FALSE	自動有効の選択。
Callback (Only valid with Software start)	NULL	コールバックの選択。
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	割り込み優先順位の選択

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

PDC HAL モジュールのクロック構成

PDC は PCLKB をそのクロックソースとして使用します。このクロックの構成に関する唯一の制限事項は、PPCLKB 周波数がこれに応じて設定されるよう、PIXCLCK が 0.6 x PCLKB 未満でなければならない点です。

PDC HAL モジュールのピン構成

PDC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では、 SSP 構成ウィンドウでピンを選択する方法を示します。その後の表では、 PDC ピンの選択例を示します。

***注:** 動作モードの選択によって、使用可能なペリフェラルの信号と必要な MCU ピンが決まります。*

PDC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
PDC	Pins	Select **Peripherals > Graphics: PDC > PDC0**

***注:** 選択シーケンスでは、 PDC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

PDC HAL モジュールのピン構成設定

Property	Value	説明
Pin Group Selection	Mixed, _A Only (Default: Mixed)	ピングループの選択
Operation Mode	Disabled, Custom, Enabled (Default: Disabled)	PDC の場合は、[Operation Mode] として [Enabled] を選択します
HSYNC	None, P704 (Default: None)	HSYNC のピン
PCKO	None, P511 (Default: P511)	PCKO のピン

Property	Value	説明
PIXCLK	None, P705 (Default: None)	PIXCLK のピン
VSYNC	None, P512 (Default: P512)	VSNC のピン
PIXD0:7	None, Pnnn (Default: None)	PIX Data0:7 のピン

***注:** 設定例は、Synergy S7G2 および DK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.32.5 アプリケーションでの PDC HAL モジュールの使用

アプリケーションで PDC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 PDC HAL モジュールを初期化します
2) 必要に応じてカメラを構成します
3) captureStart API を使用して、イメージのキャプチャを開始します
4) イメージがキャプチャされると、コールバックが呼び出されます
5) stateGet API を使用して、 HSYNC と VSTNC の状態を読み取ります
6) 必要に応じてデータを処理します
7) close API を使用して閉じます

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

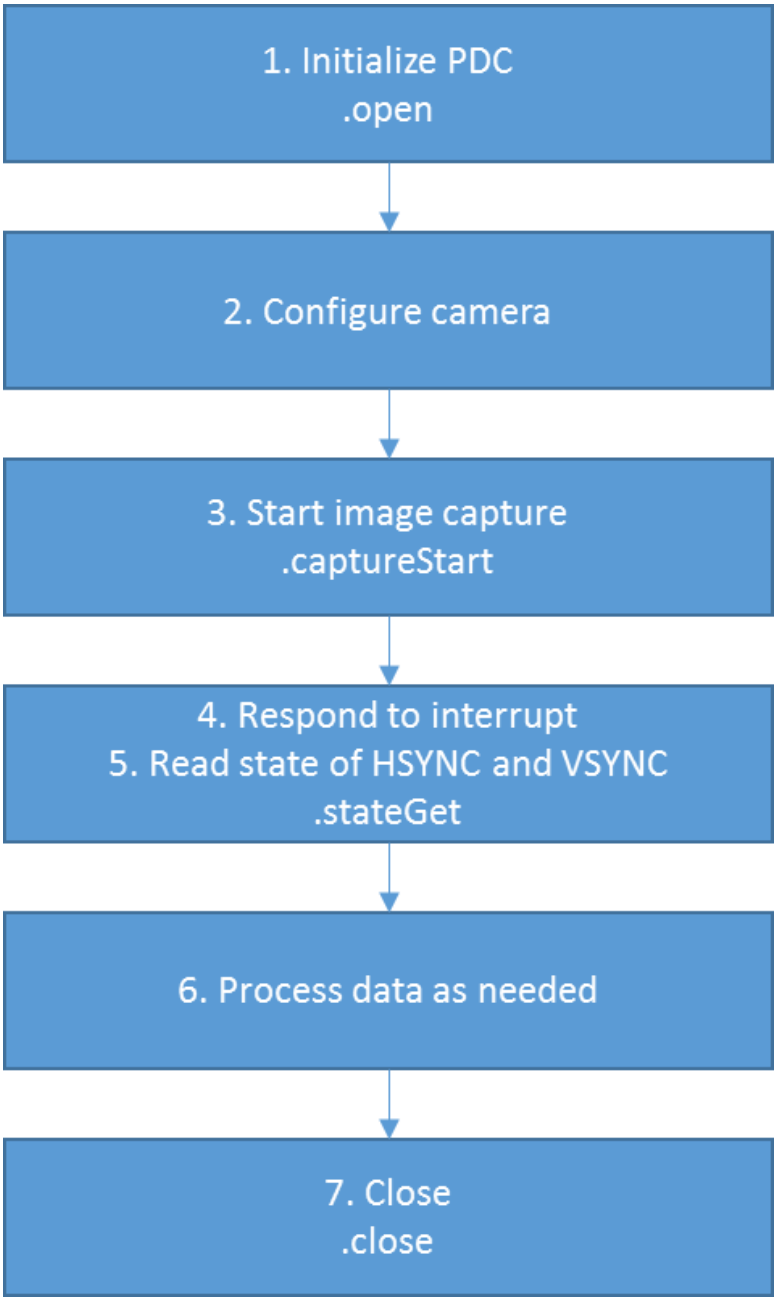


図 413:　一般的な PDC HAL モジュールアプリケーションのフロー図

5.2.33 QSPI ドライバー

QSPI HAL モジュールは、クワッド SPI インタフェースを通じてマイクロコントローラに接続された QSPI フラッシュデバイスの内容を消去およびプログラミングできる QSPI を初期化するために使用されます。主な特長は次のとおりです。

- 直接通信モードを使用したクワッド SPI フラッシュデバイスへのアクセス
- QSPI フラッシュデバイスのデータの読み取り
- QSPI フラッシュデバイスのページのプログラミング
- QSPI フラッシュデバイスのセクターの消去
- QSPI フラッシュデバイスへのアクセスを制御するバンクの選択

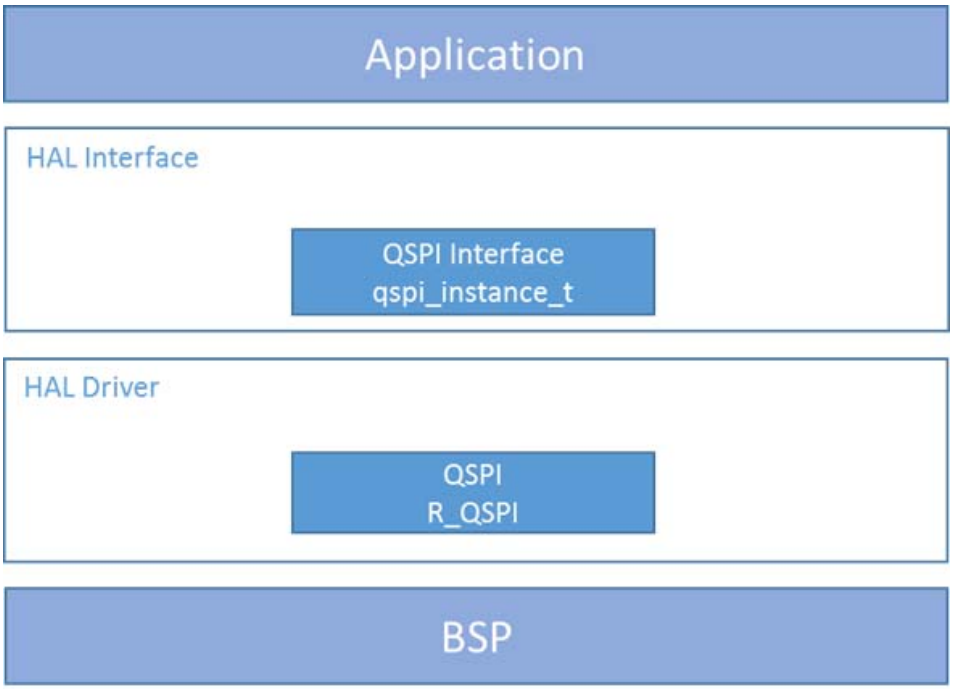


図 414: QSPI HAL モジュールのブロック図

5.2.33.1 QSPI HAL モジュールの API の概要

QSPI インタフェースでは、 QSPI HAL モジュールを使用したオープン、クローズ、リード、ライト、消去、デバイスバンク選択のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

QSPI HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg); QSPI HAL モジュールを開きます。
close	g_qspi0.p_api->close(g_qspi0.p_ctrl); QSPI HAL モジュールを閉じます。
read	g_qspi0.p_api->read(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, readBuffer, BUFFER_LENGTH); フラッシュからデータを読み取ります。
pageProgram	g_qspi0.p_api->pageProgram(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, writeBuffer, BUFFER_LENGTH); フラッシュデバイスにページ単位でデータをプログラミングします。
sectorErase	g_qspi0.p_api->sectorErase(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS); フラッシュ上のセクターを消去します。
erase	g_qspi0.p_api->erase(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, *BYTE_COUNT*); 入力引数「byte_count」に応じて、メモリのブロックを消去します
statusGet	g_qspi0.p_api->statusGet(g_qspi0.p_ctrl, &in_progress); フラッシュデバイスの書き込みまたは消去のステータスを取得します。

Function Name	API の呼び出し例と説明
bankSelect	g_qspi0.p_api->bankSelect(0); アクセスするバンクを選択します。
infoGet	g_qspi0.p_api->infoGet(g_qspi0.p_ctrl, &qspi_info); bsp_qspi.c での指定に従い、基になっている QSPI フラッシュに関する情報を提供します
versionGet	g_qspi0.p_api->versionGet(& ssp_version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	無効なパラメータが渡されました。
SSP_ERR_ASSERTION	p_cfg が NULL でした。
SSP_ERR_NOT_OPEN	ドライバーが開いていません。
SSP_ERR_UNSUPPORTED	ドライバーは、フラッシュメーカーの ID からメモリ容量とメモリタイプの情報を照会できません。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.33.2 QSPI HAL モジュールの動作の概要

QSPI HAL モジュールは、 Synergy デバイスが QSPI シリアルフラッシュデバイスと通信 （データのリード、ライト、消去）できるように、 QSPI を初期化するために使用されます。

このドライバーは、ページプログラム （ライト）、リード、消去の 3 つの動作モードをサポートします。

- *ページプログラム*は、単一のデータページをフラッシュデバイスにプログラムします。ページサイズはフラッシュメモリに固有であり、ベンダによって異なる場合があります。フラッシュの一般的なページサイズは、128,256 バイトと 512 バイトです。基になっているフラッシュデバイスでサポートされているページサイズを取得するには、 infoGet API を使用します。
- *リード*動作は、フラッシュからデータを読み取って、ユーザーが提供するバッファに格納します。
- *消去*動作は、データのブロックをフラッシュから消去します。基になっているフラッシュデバイスでサポートされている消去サイズを取得するには、 infoGet API を使用します。

***注:** すべての消去/ライト動作の後、次の動作を開始する前に、statusGet API を使用して動作の状態をポーリングすることをお勧めします。これを行わないと、ユーザーデータが壊れる可能性があります。*

QSPI HAL モジュールの動作に関する重要な注意事項と制限事項

SSP および BSP ベースのプロジェクト（SK-S7G2 および DK-S7G2）によってサポートされているボード、および QSPI メモリデバイスがプレインストールされているボードを使用する場合、BSP は初期化を行って、QSPI を XIP（インプレース実行）が有効な ROM アクセスモードにします。このプロセスにより、標準メモリのようにメモリを読み取ることができるようになります。つまり、QSPI HAL モジュールは、QSPI フラッシュデバイスの消去とプログラミングのときにのみ必要です。

一般的な QSPI アプリケーションは、QSPI フラッシュデバイスでデータをプログラムまたは消去します。このドライバーが開かれていない場合、 QSPI フラッシュデバイスの内容は 0x60000000 にマップされ、通常のメモリであるかのように読み取ることができます。

このドライバーは、QSPI ブロックおよび Micron N25Q256A QSPI フラッシュデバイスを使用して、Synergy マイクロコントローラグループ S7G2 および S3A7 でテストされています。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.33.3 アプリケーションへの QSPI HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに QSPI HAL モジュールを組み込む方法について説明します。

***注：** このプロセスを理解するには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

QSPI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します（QSPI ドライバーのデフォルトの名前は g_qspi0. です。この名前は、対応する [Properties] ウィンドウで変更できます）。

QSPI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_qspi0	Threads	Select HAL/Common and select New > Driver > Storage > QSPI HAL on QSPI

追加の構成情報を必要とするドライバーは、赤で強調表示されます。 QSPI には追加の構成は必要ありません。

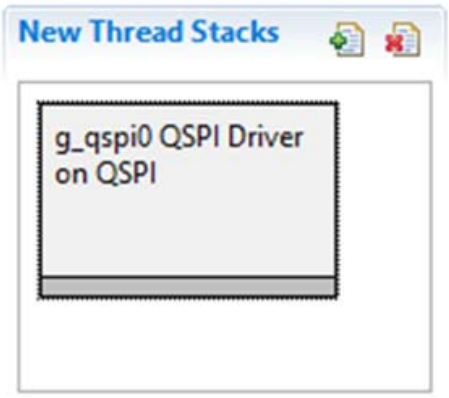


図 415: QSPI HAL モジュールのスタック

5.2.33.4 QSPI HAL モジュールの構成

QSPI HAL モジュールのコンポーネントオプションを変更する必要はありません。このドライバーは QSPI 割り込みを使用しないので、割り込みは有効になっていません。ソースコードを簡単に開発できるように、ドライバーインスタンスの名前の変更だけを検討してください。

[SSP configuration] ウィンドウでは、低レベルモジュールが正常に作動するために構成する必要のある選択項目（割り込みや動作モードなど）が、自動的に強調表示されます。ほとんどのアプリケーションでは、スタックの下位レイヤーのモジュールについてはデフォルト値のままで問題ありません。以下の表では、[Properties] ウィンドウで指定できる使用可能なオプションの詳細を示します。

注: *次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。この手順は正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチを提供します。*

r_qspi での QSPI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Default (BSP)	パラメータチェックのレベル。
Name	g_qspi0	QSPI HAL インスタンスの名前。必要に応じて、アプリケーション固有の名前に編集できます。

注: *設定に示されている例の値とデフォルトは、Synergy S7G2 MCU を使用するプロジェクトからのものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

QSPI HAL モジュールのピン構成

QSPI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では、[SSP configuration] ウィンドウでのピンの選択方法を示します。その後の表では、 QSPI ピンの選択シーケンスの例を示します。

QSPI HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI > QSPI0

***注:** 選択シーケンスでは、 QSPI0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

QSPI HAL モジュールのピン構成設定

Property	Value	説明
Pin Group Selection	- Mixed - _A only	ピングループの選択。
Operation Mode	- Disabled - Custom - Single or Dual - Quad	動作モード。
QSPCLK	None, P500	QSPI のクロック出力ピン。
QSSL	None, P501	QSPI のスレーブ選択ピン。
QIO0	None, P502	データ 0 の入力 / 出力。
QIO1	None, P503	データ 1 の入力 / 出力。
QIO2	None, P504	データ 2 の入力 / 出力。
QIO3	None, P505	データ 3 の入力 / 出力。

注: 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトからのものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

5.2.33.5 アプリケーションでの QSPI HAL モジュールの使用

モジュールを構成してファイルの生成が済むと、 QSPI はアプリケーションで使用できる状態になります。

アプリケーションで QSPI HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を呼び出して、 QSPI HAL モジュールを初期化します。
2) read API を呼び出して、データのブロックを読み取ります。
3) sectorErase API を呼び出して、データのセクターを消去します。
4) erase API を呼び出して、データの n バイトを消去します。
    a) infoGet を使用すると、基になっているフラッシュでサポートされている消去サイズを取得できます。
    b) statusGet API を使用すると、sectorErase API にも適用される消去動作の状態をポーリングできます。
5) pageProgram API を呼び出して、データのページをプログラミングします。
    a) 基になっているフラッシュでサポートされているページサイズを取得するには、infoGet API を使用します。
    b) statusGet API を使用すると、ライト動作の状態をポーリングできます。
6) close API を呼び出して、 QSPI HAL モジュールを閉じます。

注: フラッシュデバイスには標準のセクターサイズはなく、ベンダによって異なるので、sectorErase API ではなく erase API を使用することをお勧めします。

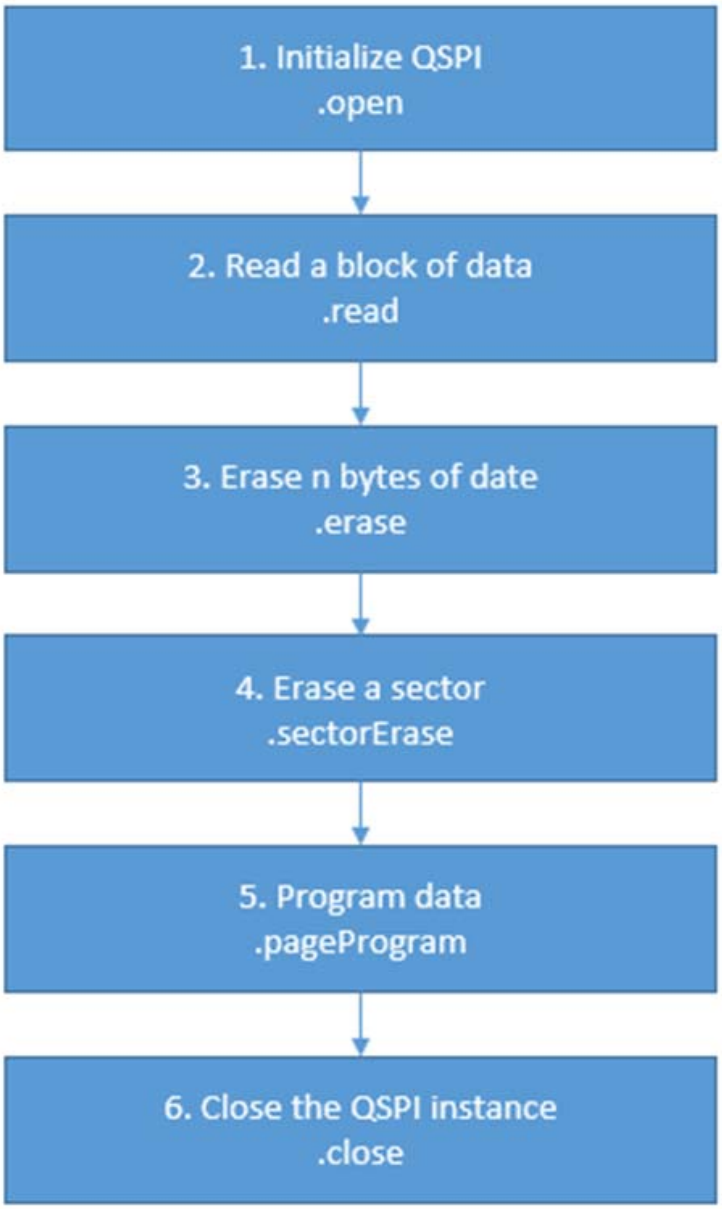


図 416: 一般的な QSPI HAL モジュールアプリケーションのフロー図

5.2.34 RTC ドライバー

RTC HAL モジュールは、リアルタイムクロックの以下の機能をサポートしています。

- RTC の設定

- RTC の時刻と日付の取得と設定
- RTC の時刻と日付のアラームの取得と設定
- RTC の時間カウンタの開始と停止
- RTC のアラームイベント、周期イベント、キャリーイベントの通知
- RTC のイベントタイプの有効化と無効化
- RTC のイベントレートの構成
- RTC のクロックソースの取得と設定

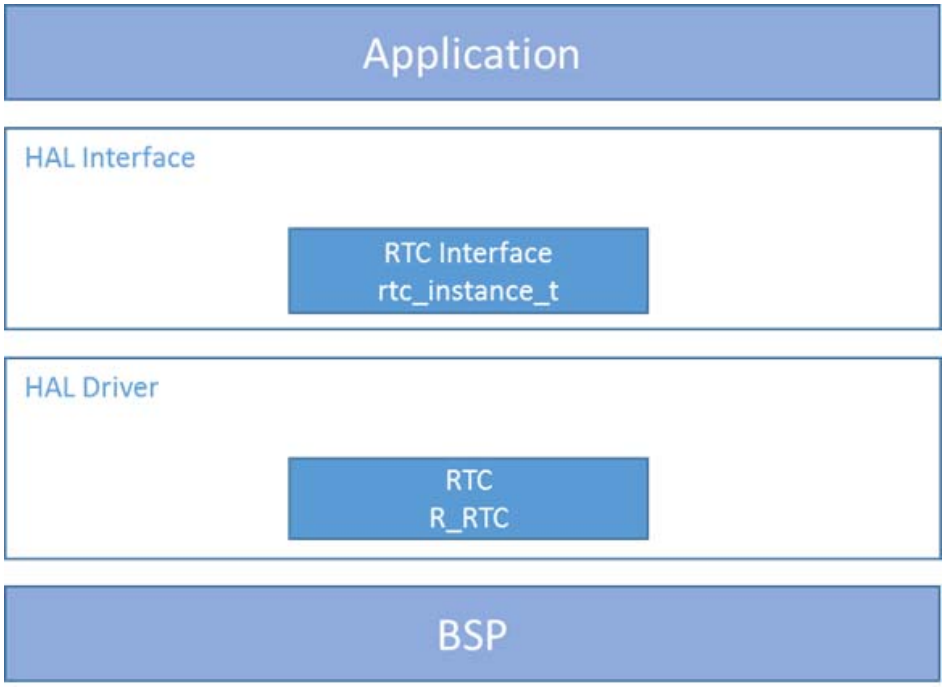


図 417: RTC HAL モジュールのブロック図

5.2.34.1 RTC の API の概要

RTC HAL モジュールでは、RTC 動作のオープン、クローズ、アラーム設定、開始、停止のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。その後で、該当するすべてのステータス戻り値の表を示します。

RTC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_rtc0.p_api->open(g_rtc0.p_ctrl, g_rtc0.p_cfg); RTC HAL を開きます。
close	g_rtc0.p_api->close(g_rtc0.p_ctrl); RTC HAL を閉じます。
calendarTimeSet	g_rtc0.p_api->calendarTimeSet(g_rtc0.p_ctrl, &start_time_struct_in, true); カレンダーの日付を設定します。
calendarTimeGet	g_rtc0.p_api->calendarTimeGet(g_rtc0.p_ctrl, &current_time_struct_out); カレンダーの時間を取得します。
calendarAlarmSet	g_rtc0.p_api->calendarAlarmSet(g_rtc0.p_ctrl, &in_alarm_time_struct_in, true); カレンダーのアラーム時間を設定します。
calendarAlarmGet	g_rtc0.p_api->calendarAlarmGet(g_rtc0.p_ctrl, &get_alarm_time_struct_out); カレンダーのアラーム時間を取得します。
calendarCounterStart	g_rtc0.p_api->calendarCounterStart(g_rtc0.p_ctrl); カレンダーカウンタを開始します。
calendarCounterStop	g_rtc0.p_api->calendarCounterStop(g_rtc0.p_ctrl); カレンダーカウンタを停止します。

Function Name	API の呼び出し例と説明
irqEnable	g_rtc0.p_api->irqEnable(g_rtc0.p_ctrl, rtc_event); アラーム irq を有効にします。
irqDisable	g_rtc0.p_api->irqDisable(g_rtc0.p_ctrl, rtc_event); アラーム irq を無効にします。
periodicIrqRateSet	g_rtc0.p_api->periodicIrqRateSet(g_rtc0.p_ctrl, Rate); 周期 IRQ レートを設定します。
infoGet	g_rtc0.p_api->infoGet(g_rtc0.p_ctrl, &clk_src); RTC に対して現在構成されているクロックソースを返します。
versionGet	g_rtc0.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました
SSP_ERR_ASSERTION	API に依存するエラーです
SSP_ERR_INVALID MODE	無効なモードです。
SSP_ERR_INVALID_PTR	無効なパラメータです。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.34.2 RTC HAL モジュールの動作の概要

RTC HAL モジュールは、 Synergy MCU のリアルタイムクロックモジュールの操作を制御します。一般的な RTC アプリケーションは、ユーザーによって指示されたシステム設定に基づき、リアルタイムクロックコントローラを定期的に設定します。共通の動作としては、時刻の設定、アラームの設定、周期割り込みの構成、動作の開始または停止などがあります。RTC アプリケーションは通常、RTC HAL モジュールの呼び出しと、ISR ハンドラからのオプションのコールバックで構成されます。

- RTC HAL モジュールは、 2 つのメインクロックソースを使用できます
  - 低速オンチップ発振器 （LOCO）は低消費電力ですが、精度は高くありません
  - サブクロック発振器は消費電力が多く、精度が高く、高コストです （外部水晶発振器が必要）
- RTC HAL モジュールは 3 種類の割り込みタイプをサポートしています
  - アラーム割り込みは、年、月、日、曜日、時、分、秒の任意の組み合わせに一致した場合に生成されます
  - 周期割り込みは、2、1、1/2、1/4、1/8、1/16、1/32、1/64、1/128、または 1/256second(s) ごとに生成されます
  - キャリー割り込みは、第 2 のカウンタへのキャリーが発生したとき、または 64Hz カウンタへのリードアクセス中に R64CNT カウンタへのキャリーが発生したときに、生成されます
- ユーザー定義のコールバック関数を （open API の呼び出しで）登録することができ、このコールバック関数はサポートされているいずれかの割り込みタイプの割り込みサービスルーチン（ISR）から呼び出されます。呼び出されたコールバック関数には、ユーザー定義のコンテキストポインタと発生した割り込みの種類を示す情報が格納された構造体 （rtc_callback_args_t）へのポインタが渡されます。

RTC HAL モジュールの動作に関する重要な注意事項と制限事項

他の RTC モジュール API を呼び出す前に、RTC HAL モジュールを開く必要があります。open の呼び出しに渡す構成構造体では、クロックソース、 ISR ハンドラからのユーザーコールバックの名前、コールバックに対するユーザー指定のコンテキストを指定します。設定構造体は、手動で定義するか、設定手順の中でのユーザー入力に基づいて ISDE によって生成します。

ドライバーの関数にアクセスするには、HAL レイヤーを直接呼び出すか、RTC インタフェース構造体を使用します。このインタフェース構造体の名前は、モジュールの構成で入力した名前の設定に基づきます。たとえば、名前が g_rtc, の場合、インタフェース構造体の名前は g_rtc_api. になります。

このモジュールでは、次の機能はサポートされていません。

- バイナリカウントモード
- バイナリアラームの取得と設定
- バイナリ時刻の取得と設定
- クロックエラーの訂正
- 1-Hz/64-Hz クロック出力
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.34.3 アプリケーションへの RTC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに RTC HAL モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

RTC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（RTC のデフォルトの名前は r_rtc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

RTC ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_rtc0 RTC HAL on r_rtc	Threads	New Stack> Driver> Timers> RTC HAL on r_rtc

次の図に示すように r_rtc の RTC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

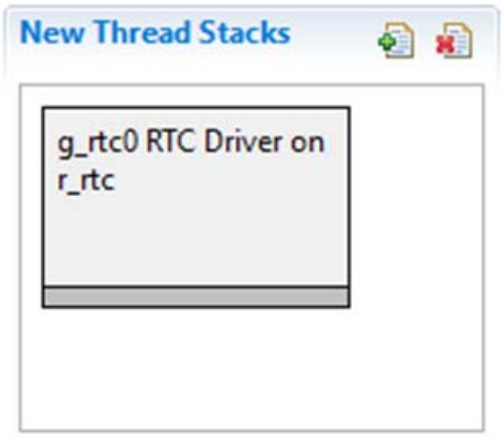


図 418: RTC HAL モジュールのスタック

5.2.34.4 RTC HAL モジュールの構成

ユーザーは必要な動作に合わせて RTC HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変

更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_rtc での RTC HAL モジュールの構成設定

Parameter	Value	説明
Parameter Checking Enable	BSP, Enabled, Disabled (Default: BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_rtc0	RTC モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。以下のコード例を参照してください
Clock Source	LOCO, Sub-clock (Default: LOCO)	RTC ブロックのクロックソース。
Error Adjustment Value	0	注意 : 非推奨の構成フィールド。0 を設定する必要があります。
Error Adjustment Type	None, Add prescaler, Subtract prescaler (Default: None)	注意 : 非推奨の構成フィールド。None を設定する必要があります。
Callback	NULL	3 つの割り込みのうちいずれかが発生したときに呼び出される ISR の名前。この ISR に渡される引数には、呼び出しの原因となった割り込みが示されます。

Parameter	Value	説明
RTC ALARM	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 5)	RTC アラーム割り込みの優先順位レベル
RTC PERIOD	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 6)	RTC 周期割り込みの優先順位レベル
RTC CARRY	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 7)	RTC キャリー割り込みの優先順位レベル

***注:** 設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

モジュールのデフォルト以外の設定が望ましい場合もあります。たとえば、デフォルトとは異なるクロック分周器を選択すると役に立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

RTC HAL モジュールのクロック構成

RTC HAL モジュールは、次のクロックソースを使用できます。

- LOCO（低速オンチップオシレーター）
  - 低い電力消費
  - 低い精度
- サブクロックオシレーター
  - 高い電力消費
  - 高い精度
  - 高コスト（水晶が必要）

LOCO が構成の間のデフォルトの選択です。

RTC HAL モジュールのピン構成

現在、 RTC は出力をサポートしていないので、出力ピンの選択はありません。

5.2.34.5 アプリケーションでの RTC HAL モジュールの使用

一般的な RTC アプリケーションは、ユーザーによって指示されたシステム設定に基づき、リアルタイムクロックコントローラを定期的に設定します。たとえば、時刻の設定、アラームの設定、周期割り込みの構成などです。RTC アプリケーションは、RTC モジュールの呼び出しと、オプションの ISR コールバックで構成されます。

他の API を呼び出す前に、RTC モジュールを開く必要があります。open の呼び出しに渡す構成構造体では、クロックソース、 ISR コールバックの名前、ハンドラのユーザー指定のコンテキストを指定します。設定構造体は、手動で定義するか、設定手順の中でのユーザー入力に基づいて ISDE によって生成します。モジュールの機能には、RTC インタフェース構造体を使用してアクセスできます。このインタフェース構造体の名前は、モジュールの構成で入力した名前の設定に基づきます。

アプリケーションで RTC の周期 IRQ を使用する一般的な手順は次のとおりです。

1) open API を使用して、 RTC を初期化します
2) periodicIrqRateSet API を使用して、周期 IRQ のレートを設定します
3) calendarCounterStart API を使用して、カレンダーカウンタを開始します
4) irqEnable API を使用して、割り込みを有効にします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

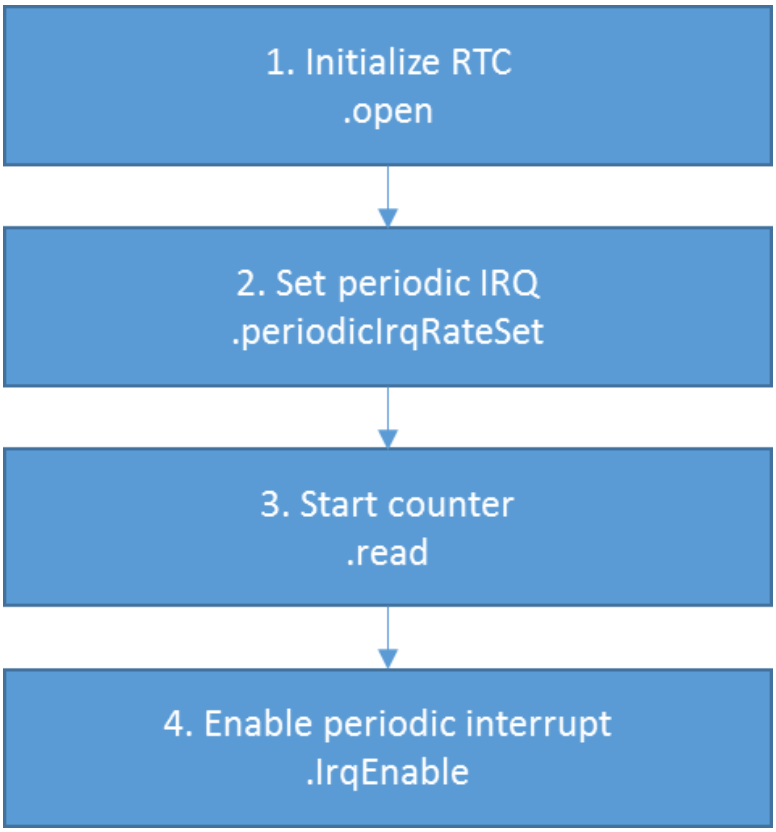


図 419: 一般的な RTC 周期割り込みアプリケーションのフロー図

アプリケーションで RTC のアラーム IRQ を使用する一般的な手順は次のとおりです。

1) open API を使用して、 RTC を初期化します
2) calendarTime Set API を使用して、カレンダー時刻を設定します

3) calandarAlarmSet API を使用して、アラーム時刻を設定します

4) calendarCounter Start API を使用して、カレンダーカウンタを開始します

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

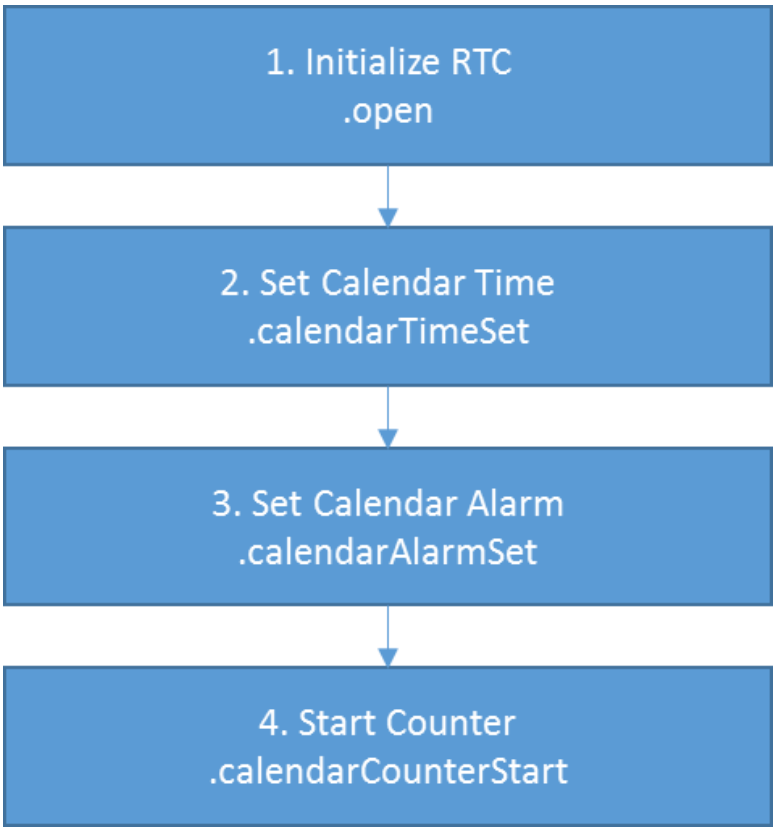


図 420: 一般的な RTC アラーム割り込みアプリケーションのフロー図

5.2.35 SD/MMC ドライバーおよび SDIO ドライバー

- SDSC（SD 標準容量）、SDHC（SD 大容量）、SDXC（SD 拡張容量）、eMMC（組み込みマルチメディアカード）の各メモリデバイスをサポートします
  - SD および eMMC メモリデバイスのリード、ライト、消去をサポート
  - 1 ビット、4 ビット、8 ビットデータバスをサポート（8 ビットバスは eMMC でのみサポート）
  - ハードウェアライトプロテクトの検出をサポート（SD カードのみ）
  - 下位互換モードと高速 SRD モード（eMMC）間を自動的に選択
- SDIO のサポート
  - SDIO シングルレジスタアクセスをサポート（CMD52）

– SDIO マルチレジスタアクセスをサポート （CMD53）
– SDIO 割り込みをサポート

• ホスト （MCU）とデバイスの両方でサポートされる最大クロックレートにクロックを自動的に構成

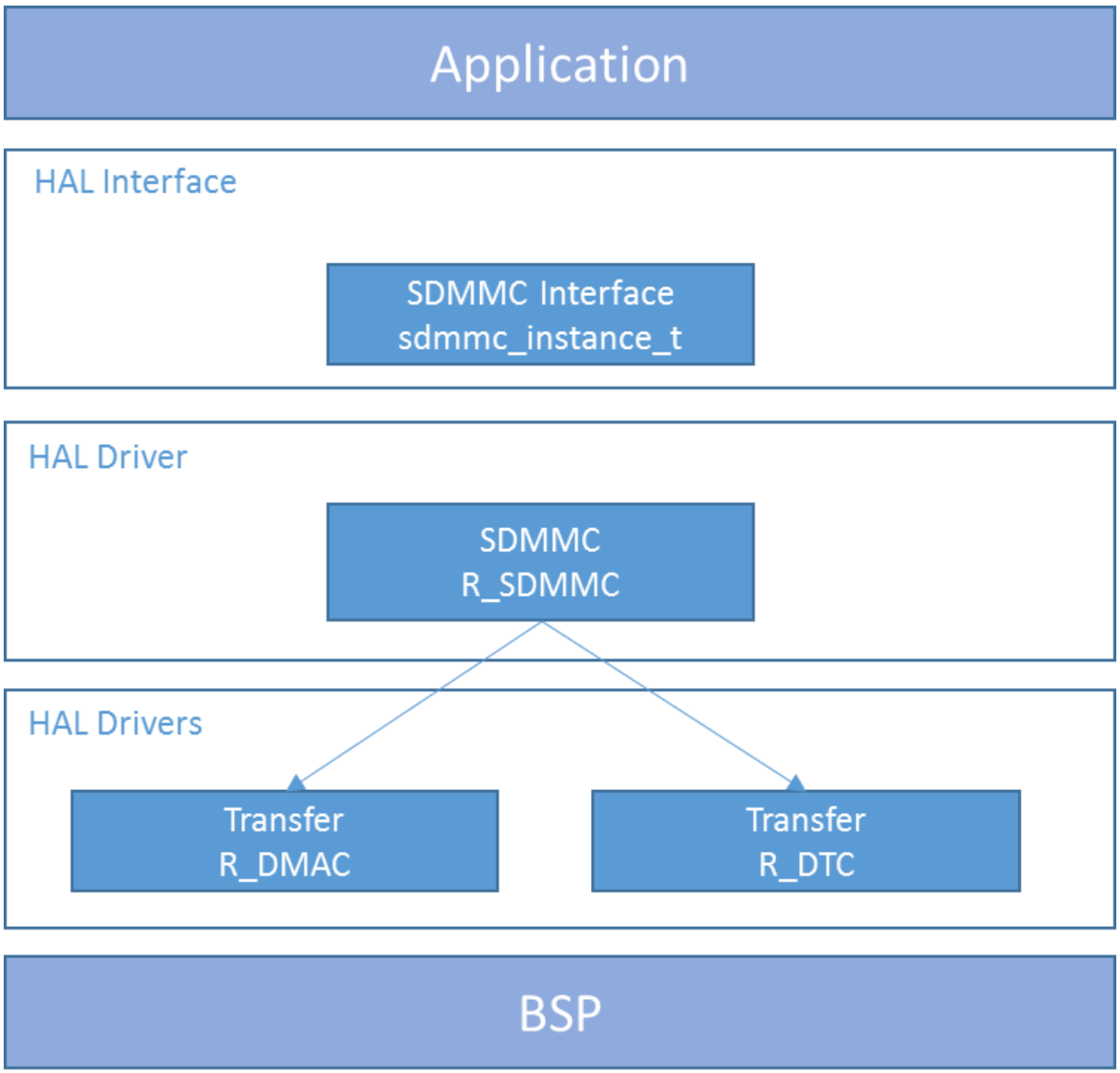


図 421: SDMMC HAL の編成、オプション、スタックの実装

5.2.35.1 SDMMC HAL モジュールの API の概要

このドキュメントは以下のセクションに分かれており、個別に読んでも（経験豊富な Synergy 開発者向け）、続けて読んでも（Synergy プラットフォームの経験がない開発者向け）かまいません。 SD/MMC ドライバーおよび SDIO ドライバーの API の概要

SD/MMC ドライバーの API の要約

Function Name	API の呼び出し例と説明
open	g_sdmmc.p_api->open(g_sdmmc.p_ctrl, g_sdmmc.p_cfg); 読み書きおよび制御のためにデバイスチャネルを開きます。最初の呼び出しで、ドライバーとハードウェアを初期化します。
read	g_sdmmc.p_api->read(g_sdmmc.p_ctrl, &destination, start, count); SD/MMC チャネルからデータを読み取ります。
write	g_sdmmc.p_api->write(g_sdmmc.p_ctrl, &source, start, count); SDMMC チャネルにデータを書き込みます。
control	g_sdmmc.p_api->control(g_sdmmc.p_ctrl, command, &data); SD/MMC ポートに制御コマンドを送信して、SD/MMC ポートのステータスを受信します。
close	g_sdmmc.p_api->close(g_sdmmc.p_ctrl); 開いている SDMMC チャネルを閉じます。開いている最後のチャネルである場合は、ハードウェアをオフにします。
versionGet	g_sdmmc.p_api->versionGet(&version); ブロックメディア SD/MMC ドライバーのバージョンを取得します。

Function Name	API の呼び出し例と説明
readIo	g_sdmmc.p_api->readIo(g_sdmmc.p_ctrl, &data, function, address); SDMMC チャネルから I/O データを読み取ります。
writeIo	g_sdmmc.p_api->writeIo(g_sdmmc.p_ctrl, &data, function, address, read_after_write); SDMMC チャネルに I/O データを書き込みます。
readIoExt	g_sdmmc.p_api->readIoExt(g_sdmmc.p_ctrl, &destination, function, address, count, transfer_mode, address_mode); SDMMC チャネルから I/O 拡張データを読み取ります。
writeIoExt	g_sdmmc.p_api->writeIoExt(g_sdmmc.p_ctrl, &source, function, address, count, transfer_mode, address_mode); SDMMC チャネルに I/O 拡張データを書き込みます。
ioIntEnable	g_sdmmc.p_api->ioIntEnable(g_sdmmc.p_ctrl, enable); SDMMC チャネルに対して、SDIO 割り込みを有効化します。
infoGet	g_sdmmc.p_api->infoGet(g_sdmmc.p_ctrl, &info); SDMMC チャネルの情報を取得します。
erase	g_sdmmc.p_api->erase(g_sdmmc.p_ctrl, start, count); SDMMC セクターを消去します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細な説明については、このドキュメントの「参考文献」セクションで説明している SSP リファレンスマニュアルを参照してください。FileX API ドキュメントについては、「参考文献」セクションで説明しているように Renesas Synergy の X-Ware コンポーネントと NetX コンポーネントのドキュメントも参照してください。*

エラーステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_INVALID_ARGUMENT	パラメータの値が無効です。
SSP_ERR_IN_USE	指定したチャネルはすでに開かれています。構成は変更されていません。関連する Close 関数を呼び出すか、関連する Control コマンドを使用してチャネルを再構成します。
SSP_ERR_ASSERTION	ポインタが NULL です。
SSP_ERR_WRITE_PROTECTED	SD または MMC カードが書き込み保護されています。
SF_INFO_NOT_AVAILABLE	カードが取り外されているか、故障しているために情報が入手できない可能性があります。
SSP_ERR_NOT_OPEN	チャネルは開かれていません。
SSP_ERR_HW_LOCKED	ハードウェアロックは既に取得されています。
SSP_ERR_TRANSFER_BUSY	転送が進行中です。
SSP_ERR_CARD_NOT_READY	カードの準備がまだできていません。
SSP_ERR_NOT_ENABLED	SDIO の割り込み要求イネーブルが失敗しました。
SSP_ERR_READ_FAILED	リード操作が失敗しました。
SSP_ERR_WRITE_FAILED	書き込み操作が失敗しました。

***注：** ローレベルドライバーによって共通のエラーコードが返される場合があります。関連するすべてのエラーコードの定義については、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』を参照してください。*

5.2.35.2 SDMMC HAL モジュールの動作の概要

以下では、SDMMC HAL モジュールを使用したときに操作できる機能と要件について説明します。

5.2.35.3 SDMMC HAL モジュールの割り込み構成

必要な割り込みは次のとおりです。

- DTC で SD/MMC を使用 :
  - アクセス割り込み

  - DTC 割り込み
- DMAC で SD/MMC を使用 :
  - アクセス割り込み
  - DMAC 割り込み （DMAC インスタンス内）
- DTC で SDIO を使用 :
  - アクセス割り込み
  - SDIO 割り込み
  - DTC 割り込み
- DMAC で SDIO を使用 :
  - アクセス割り込み
  - SDIO 割り込み
  - DMAC 割り込み （DMAC インスタンス内）

カード割り込みはオプションで、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン（n＝チャネル番号）が使用可能になっている MCU パッケージでのみ使用可能です。

5.2.35.4 SDMMC HAL モジュールのブロックサイズ構成

ブロックサイズ構成は、SDIO カードで使用する場合にのみ提供されます。SDIO カードの場合、ブロックサイズは 1 ～ 512 バイトに構成できます。ブロックサイズは、SD カードと e/MMC メモリデバイスではデフォルトの 512 バイトのままにする必要があります。

5.2.35.5 SDMMC HAL モジュールのカード検出構成

r_sdmmc ドライバーでカード検出を使用する前に、「カード検出の制限事項」を参照してください。カード検出が使用不能な場合や、アプリケーションに不要な場合は、 Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [Not Used] に設定する必要があります。カード検出を有効化するには、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [CD Pin] に、[Media Type] を [Card] に設定する必要があります。

5.2.35.6 SDMMC HAL モジュールのアクセス割り込み優先順位

データ転送が 4 バイトでアラインされていないか、4 バイトの倍数でない場合、ブロックサイズ （最大 512 バイト）のソフトウェアコピーはアクセス割り込みで実行されます。これにより、ソフトウェアコピーが完了するまで、プライオリティがアクセス割り込み以下の他の割り込みがブロックされます。

5.2.35.7 SDMMC HAL モジュールの一般的なタイミングに関する注意事項

このドライバーの一部の関数はブロックします。open と erase は、動作全体が完了するまでブロックします。read、write、readIo、writeIo、readIoExt、writeIoExt は、単一コマンドレスポンスサイクルにわたってブロックします。マルチスレッドシステムでは、これらのいずれかの関数呼び出し中にこのドライバーがブロックす

る可能性のある時間よりも短いレスポンスタイムを他のスレッドが必要とする場合、このドライバーをプライオリティの低いスレッドで使用するには注意が必要です。

5.2.35.8 SDMMC HAL モジュールの Open のタイミングに関する注意事項

open API は、デバイスの識別および構成プロセス全体を実行します。これには、1 ビットのバス幅と 400kHz 以下のバス速度でのいくつかのコマンドレスポンスサイクルが関係します。

5.2.35.9 SDMMC HAL モジュールの Read、 Write、 ReadIoExt、 WriteIoExt のタイミングに関する注意事項

メディアのリードおよびライト（read および write）と拡張リードおよびライト SDIO API（readIoExt および writeIoExt）は、デバイスからレスポンスを受信するまでブロックします。データ転送操作は非ブロックであり、割り込みおよび転送インスタンス（DMAC または DTC）を必要とします。これらの API は、初期動作が正常に開始したことを示す SSP_SUCCESS を返します。アプリケーションは、イベント SDMMC_EVENT_TRANSFER_COMPLETE または SDMMC_EVENT_TRANSFER_ERROR でコールバックを待って、リードまたはライトの完了を示す必要があります。

5.2.35.10 SDMMC HAL モジュールの ReadIo と WriteIo のタイミングに関する注意事項

SDIO readIo および writeIo API は、デバイスからレスポンスを受信するまでブロックします。リードまたはライト動作はこれらの API が戻ると完了します。

5.2.35.11 SDMMC HAL モジュールの Erase のタイミングに関する注意事項

erase API は、消去動作が完了するまでブロックします。これは、消去されるセクター数に応じて数秒以上の場合があります。

5.2.35.12 SDMMC HAL モジュールの SDIO 割り込み

多くの SDIO カードはレベル割り込みを使用するため、割り込みは割り込みがデバイスでクリアされるまでアサート解除されません。SDIO 割り込みが適切にクリアされるようにするには、次のいずれかの方法をお勧めします。

- コールバック関数がイベント SDMMC_EVENT_SDIO. で終了する前に SDIO 割り込みがデバイスでクリアされることを確認します。この方法を選択し、 SDIO API を割り込みで使用する必要がある場合は、アクセス割り込みのプライオリティを SDIO 割り込みよりも高くする必要があります。
- IoIntEnable を使用してイベント SDMMC_EVENT_SDIO の発生時のコールバック関数の SDIO 割り込みを無効化します。アプリケーションの他の場所の SDIO 割り込みをクリアしてから、必要に応じて IoIntEnable を使用して SDIO 割り込みを再有効化します。

5.2.35.13 動作に関する重要な注意事項と制限事項

SDMMC HAL モジュールの動作に関する注意事項

SD HALA への準拠

SD の仕様に準拠したホストデバイスを開発する場合、ホストは SD ホスト機器およびペリフェラルの使用許諾契約 （SD Host/Ancillary Product License Agreement）（SD HALA）に従う必要があります。

SDMMC HAL モジュールの制限事項

データのアラインメントとサイズ

データ転送は、4 バイトでアラインされる必要があり、可能な場合にはサイズが 4 バイトの倍数である必要があります。この推奨事項は、read()、write()、readIoExt()、writeIoExt() API に適用されます。データ転送が 4 バイトでアラインされ、4 バイトの倍数である場合、r_sdmmc ドライバーはゼロコピーで、DMAC または DTC によるハードウェア加速を十分に活用します。データ転送が 4 バイトでアラインされていないか、4 バイトの倍数でない場合、転送されるブロックごとに特別な CPU 割り込みが必要です（「アクセス割り込み優先順位」を参照）。

カード検出の制限事項

r_sdmmc ドライバーでのカード検出のサポートは制限されています。カード検出は open が完了した後でのみ使用可能で、open() はカードが挿入されていない限り完了できません。したがって、r_sdmmc ドライバーでのカード検出は、カードの取り外しと再挿入の検出にのみ役立ちます。再挿入が検出された後、ドライバーを終了して再開する必要があり、カード検出は再開が完了するまで使用可能になりません。カード検出は、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン （n = チャネル番号）が使用可能になっている MCU パッケージでのみ使用可能です。MCU に SDHIn CD ピンがないか、カード検出がアプリケーションに不要な場合は、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] でカード検出を無効化する必要があります。r_sdmmc ドライバーのカード検出機能の使用の代替方法は、外部 IRQ インスタンスを使用し、アプリケーションレイヤでカード検出を処理することです。カード検出がアプリケーションレイヤーで処理される場合は、カード挿入が検出された後に open を呼び出す必要があり、カード取り外しが検出された後に close を呼び出す必要があります。

このモジュールの最新の制限事項については、SSP の最新のリリースノートを参照してください。

5.2.35.14 アプリケーションへの SDMMC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SDMMC HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参考文献」セクションの説明に従って入手可能な『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

このモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（モジュールのデフォルトの名前は次の表に示してあります。この名前は、対応する [Properties] ウィンドウで変更できます）。次の図に示すように、ドライバーが追加されるとスレッドに表示されます。

モジュール選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sdmmc0 SD/MMC driver on r_sdmmc	Threads	New Stack> Driver> Storage> SD/MMC Driver on r_sdmmc

次の図に示すように、r_sdmmc の SD/MMC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、複数のスタックで使用できるため 1 回のみ追加する必要があります。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、このテキストを含むブロックで示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

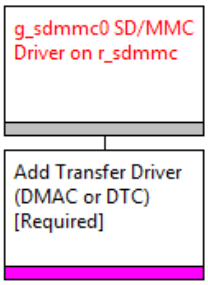


図 422: r_sdmmc スタックでの SD/MMC ドライバー

5.2.35.15 SDMMC HAL モジュールの構成

ユーザーは必要な動作に合わせてモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注: 次に示す表に目を通しながら、ISDE を開き、SDMMC HAL モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

SD/MMC および SDIO ドライバーの構成

ISDE Property	Setting	説明
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	Default: g_sdmmc0	モジュール名
Channel	0,1 (Default: 1)	SDMMC のチャネル（チャネル 0 または 1)
Media type	Embedded, Card (Default: Embedded)	メディアは、カードと組み込みデバイスのどちらかです。この設定により、カードの挿入/取り出しがあるのか、あるいは書き込み保護ピンが存在するのかをファームウェアに知らせます。
Bus Width	1 bit, 4 bits, 8 bits	ハードウェアインタフェースによって定義されたバス幅。（8 ビットは eMMC のみ）
Block Size	512	ブロックのサイズ
Callback	NULL (default) Name of user callback function.	（FileX を使用する場合は不要）ユーザーコールバック関数の名前に設定します。Provides event that caused interrupt: SDMMC_EVENT_CARD_REMOVED, SDMMC_EVENT_CARD_INSERTED, SDMMC_EVENT_ACCESS, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANSFER_COMPLETE, SDMMC_EVENT_TRANSFER_ERROR
Access Interrupt Priority	Priority level 0-15 (Default: Disabled)	SD および eMMC、SDIO への読み取り、書き込み、エラーに必要となる割り込み優先順位。有効にすることで、割り込みコールバックを呼び出します。

ISDE Property	Setting	説明
Card Interrupt Priority	Priority level 0-15 (Default: Disabled)	（オプション）カード取り外しの割り込み。カードが取り外された際、自動でデバイスを閉じます。有効にすることで、割り込みコールバックを呼び出します。
DMA Interrupt Priority	Priority level 0-15 (Default: Disabled)	SDIO の読み取り、書き込み、エラーに必要となる割り込みプライオリティレベル。有効にすることで、割り込みコールバックを呼び出します。メモリカードに使用されません。

注：示されている例とデフォルトは、Synergy S7 MCU を使用するプロジェクトに適用されます。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

5.2.35.16 SDMMC HAL モジュールのクロック構成

SDMMC MCU（SDHI）は、PCLKA をそのクロックソースとして使用します。データレートを最適化する必要がない限り、SDMMC 周辺デバイスに対して固有のクロックを設定する必要はありません。SDMMC ドライバーは、PCLKA 周波数と、SD、SDIO または eMMC デバイスで許可される最大クロックレート（これはメディアデバイスの初期化時に取得されます）に基づいて、適切な内蔵分周器を選択します。

5.2.35.17 SDMMC HAL モジュールのピン構成

SDMMC（SDHI）の I/O ピンを構成するには、ISDE ピンコンフィギュレータを使用します。ほとんどのボード、高速メモリ、SDIO デバイスにおいて、各品のドライブ能力は「中」または「高」に設定する必要があります。以下の表では、対応する [SSP configuration] ウィンドウでのピンの選択方法と構成方法を示します。

注: 動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

SDHI のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SDHI	Pins	Select **Peripherals > Storage:SHDI > SDHI0**

注: 選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

SDHI のピン構成設定

Pin Configuration Property	設定値	説明
Operation Mode	Disabled,	アプリケーションに従ってモードを選択します
	Custom,	
	SD_MMC 1 bit	
	SD_MMC 4 bit	
	MMC 8 bit (Default: Custom)	
CLK	None, P413 (Default: P413)	クロックピン
CMD	None, P412 (Default: P412)	コマンドピン
DAT0-7	None, PXXX (Default: PXXX)	データピン
CD	None, P903 (Default: P903)	カード検出ピン
WP	None, P414 (Default: P414)	カードライトプロテクションピン

***注:** 示されている設定は、Synergy S7G2 MCU および DK-S7G2 キットを使用するプロジェクトに適用されます。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.35.18 アプリケーションでの SDMMC HAL モジュールの使用

SD または eMMC メモリデバイスで r_sdmmc ドライバーを使用する一般的な手順は次のとおりです。

1) open API を使用して、ドライバーを開きます。
2) read API を使用してカードからデータを読み取るか、write API を使用してカードにデータを書き込みます。
3) read API または write API が呼び出された後、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR（操作が正常に完了しなかったことを意味します）のコールバックを待ちます。

次の図では、これらの一般的な手順を通常の動作フロー図に示します。

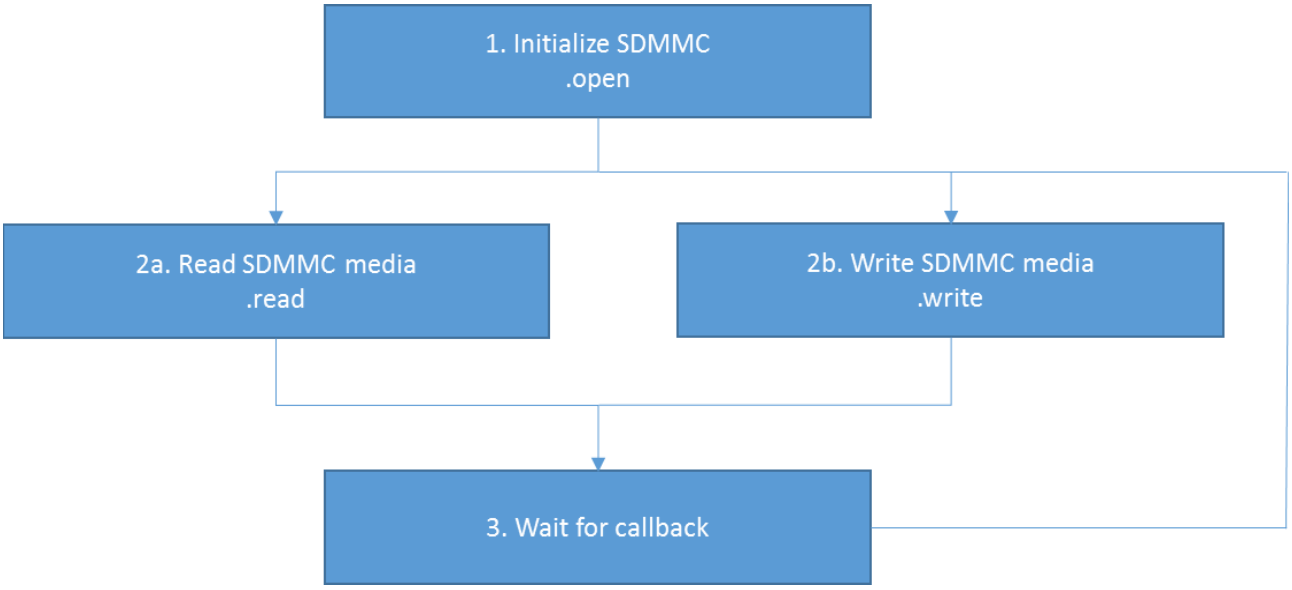


図 423: SD または eMMC メモリデバイスの一般的な SDMMC HAL モジュールアプリケーションのフロー

SDIO カードで r_sdmmc ドライバーを使用する一般的な手順は次のとおりです。

1) open API を使用して、ドライバーを開きます。
2) readIo API または writeIo API を使用して、単一レジスタを設定するか読み戻します。動作はこれらの呼び出しの直後に完了し、コールバックを待つ必要はありません。
3) readIoExt API または writeIoExt API を使用して、複数レジスタを設定するか読み戻します。必要な場合は、呼び出しの間に control API を使用して、ブロックサイズを変更できます。
4) readIoExt API または writeIoExt API が呼び出された後、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR（操作が正常に完了しなかったことを意味します）のコールバックを待ちます。
5) カードが割り込みを要求する場合は、イベント SDMMC_EVENT_SDIO. でコールバックが呼び出されます。カードのドキュメントの説明に従って SDIO 割り込みを処理します（このドキュメントの「SDIO 割り込み」セクションを参照してください）。カードからの SDIO 割り込みは、IoIntEnable API を使用していつでも有効または無効にできます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

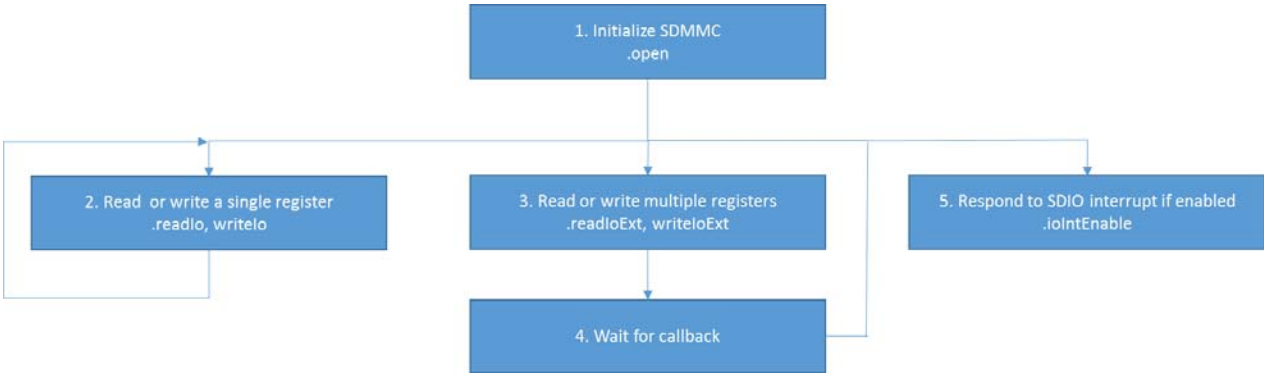


図 424: SDIO カードの一般的な SDMMC HAL モジュールアプリケーションのフロー

5.2.35.19 FileX ポートブロックメディアフレームワークのアプリケーションプロジェクト

このモジュールガイドに関連付けられているアプリケーションプロジェクトを見ると、上で示した設計の詳細な手順がわかります。プロジェクトについては、このドキュメントの最後にある「参考文献」セクションのリンクを参照してください。ISDE でアプリケーションプロジェクトをインポートして開き、FileX ポートブロックメディアフレームワークモジュールの構成設定を見ることができます。また、完全な設計におけるブロックメディア上の FileX API を示すために使用されているコード（sdmmc_thread_entry.c, 内にあります）を読むこともできます。

アプリケーションプロジェクトでは、ブロックメディア上の FileX フレームワーク API の一般的な使用方法が示されています。アプリケーションプロジェクトで自動生成されるコードは、FileX ポートブロックメディアフレームワークを使用する FileX フレームワークを初期化します。これは、ブロックメディアフレームワークを使用する FileX 固有アクセスを実装することを目的とするドライバーです。ブロックメディアフレームワークの基になる実装では、SD/MMC ドライバーが使用されます。具体的には、アプリケーションプロジェクトはオンボード eMMC メモリで動作するように設計されていますが、SD カードにアクセスするように構成設定を簡単に変更できます。次の表では、アプリケーションプロジェクトで使用されている関連するソフトウェアとハードウェアのターゲットバージョンを示します。

アプリケーションプロジェクトで使用されているソフトウェアリソースとハードウェアリソース

Resource	Revision	説明
e^2 Studio	5.3.1.002	統合ソリューション開発環境
IAR Workbench	7.71.1.11989	Synergy プラットフォーム用 IAR Workbench IDE
SSP	1.2.0	Synergy Software Platform
SSC	5.3.1.002	Synergy Standalone Configurator
DK-S7G2	3.0	開発キット

アプリケーションプロジェクトの簡単なフロー図を次に示します。

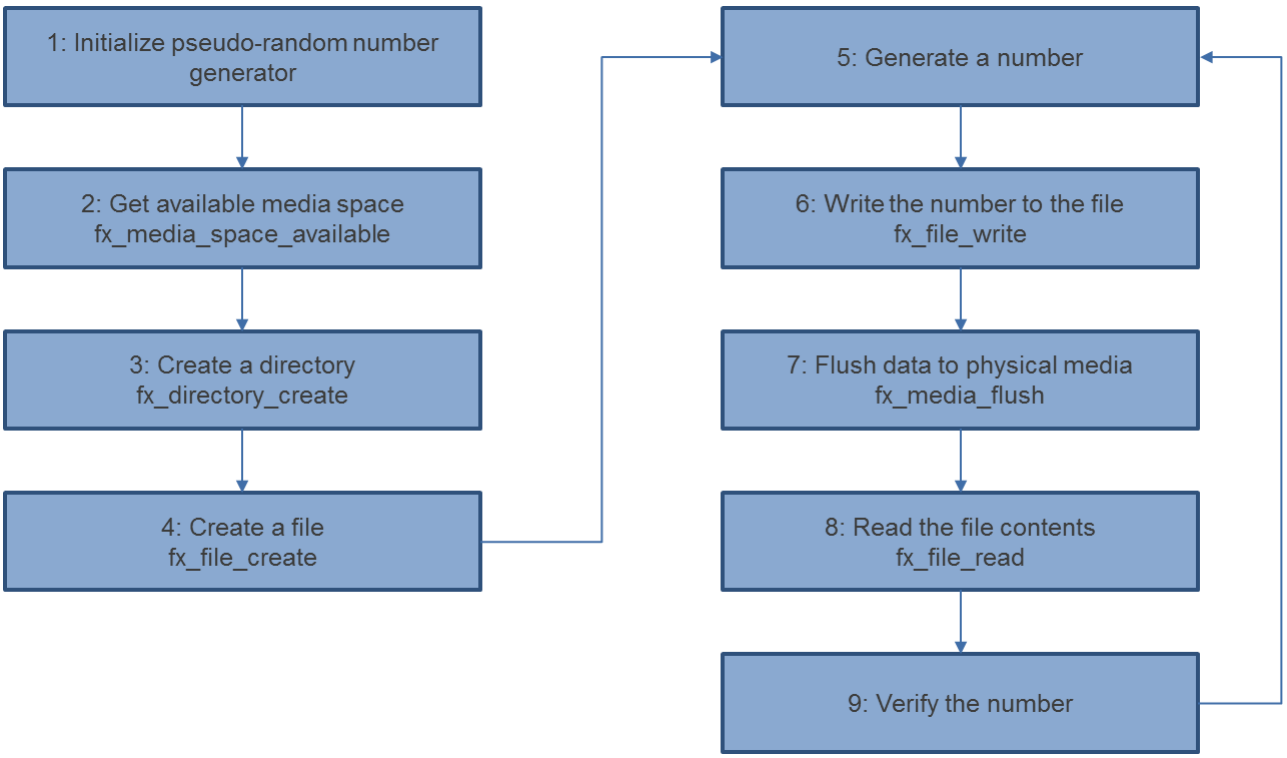


図 425: FileX ポートブロックメディアアプリケーションプロジェクトのフロー図

***注:** FileX の初期化は、生成されるコードで自動的に行われます。*

完全なアプリケーションプロジェクトについては、このドキュメントの最後にある「参考文献」セクションのリンクを参照してください。プロジェクトを ISDE にインポートすると、sdmmc_thread_entry.c ファイルがプロジェクトに配置されます。このファイルを ISDE で開き、以下の説明を読みながら見ることで、API の重要な使用方法がわかります。

sdmmc_thread_entry.c の最初のセクションには、FileX のメディア構造体を参照するヘッダファイルと、printf() を使用して結果を表示するためのセミホスティングが可能なコードセクションがあります。次のセクションには、マクロ定数と変数の定義が含まれます。その後に、関数のプロトタイプがあります。最初の関数は疑似乱数を生成します。次に、整数と文字列の間の双方向の変換を行うための 2 つの関数が定義されています。エラー処理のための簡単なコードセクションもあります。セミホスティングが有効になっている場合、エラーコードが表示されます。次の関数は、ファイルに数値を書き込むために使用されます。最初に、数値は文字列に変換されます。次に、ファイルのオープン、クリア、ライト、クローズを行うための FileX API にアクセスして、物理メディアにデータをフラッシュします。最後のステップは、FileX の操作ではバッファが使用されており、リセット後に最近の変更がすべて反映されていない可能性があるために必要です。または、FileX は fx_media_close が呼び出されるとデータをフラッシュします。その後の関数は前の関数とよく似ていますが、ライトを実行するのではなく、指定されたファイルから数値を読み取るために使用される点が異なります。

最後のセクションは、スレッドエントリ関数セクションです。最初に、事前に定義されている値を使用して、疑似乱数ジェネレータが初期化されます。セミホスティングが有効になっている場合、使用可能なスペース

が表示されます。次に、アプリケーションはファイルを作成する必要がありますが、ディレクトリは事前に作成されています。その後、ソフトウェアは「無限の」 while ループに入り、数値の生成を開始します。この数値は、前に作成されたファイルに書き込まれます。その後、数値は読み取られて、生成された数値と検証されます。エラーがある場合は、メッセージが提供されます。続いて、スレッドスリープ関数によって ThreadX タイマ数ティックだけ実行が一時停止された後、 while ループの関数が繰り返されます。

注: *上の説明では、 Synergy Software Package の Debug Console での printf() の使用に慣れているものと想定しています。慣れていない場合は、このドキュメントの最後にある「参考文献」セクションの「How do I Use Printf() with the Debug Console in the Synergy Software Package」を参照してください。または、デバッグモードで変数を見て結果を確認することもできます。*

ターゲットボードと MCU の必要な動作と物理特性をサポートするため、このアプリケーションプロジェクトではいくつか重要なプロパティが構成されています。以下では、そのようなプロパティと特定のプロジェクトで設定される値を示します。実践的な練習として、アプリケーションプロジェクトを開いてプロパティウィンドウでこれらの設定を見ることもできます。

アプリケーションプロジェクトでの FileX ポートブロックメディアフレームワークの構成設定

ISDE Property	Value Set
Name	g_fx_media
Format System is on SDMMC	True
Volume Name	Volume 1
Number of FATs	1
Directory Entries	256
Hidden Sectors	0
Total Sectors	500000
Bytes per Sector	512
Sectors per Cluster	1
Working media memory size	512

アプリケーションプロジェクトの r_sdmmc 構成設定での SD/MMC ドライバー

ISDE Property	Value Set
Parameter Checking Enable	デフォルト (BSP)
Name	g_sdmmc

ISDE Property	Value Set
Channel	0
Media Type	Embedded
Bus Width	8 Bits
Block Size	512
Callback	NULL
Access Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Card Interrupt Priority	無効
DMA Request Interrupt Priority	無効

アプリケーションプロジェクトの r_dmac 構成設定での転送ドライバー

ISDE Property	Value Set
Parameter Checking Enable	デフォルト (BSP)
Name	g_sdmmc
Channel	0
Mode	Normal
Transfer Size	1 Byte
Destination Address Mode	Fixed
Source Address Mode	Incremented
Repeat Area (Unused in Normal Mode)	Source
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	0
Number of Blocks (Valid only in Block Mode)	0
Activation Source	Software Activation

ISDE Property	Value Set
Auto Enable	False
Callback	NULL
Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)

5.2.36 セグメント LCD ドライバー

SLCDC HAL モジュールは、セグメント LCD コントローラ（SLCDC）を使用して、データをセグメント LCD に表示します。ドライバーは LCD を初期化してデータを表示し、ドライブ電圧ジェネレータ、ディスプレイ波形、タイムスライス数、および LCD を稼働するバイアスメソッドを構成します。このモジュールでは、指定されたセグメントのセットにデータを表示するための関数、既存のセグメントデータを更新するための関数、ディスプレイを有効化および無効化するための関数、ディスプレイエリアを設定するための関数、およびコントラストを縫製するための関数を提供します。

このモジュールでは、次の機能がサポートされています。

- LCD ドライバー電圧ジェネレータの内部電圧ブースト : 容量分割方式または外部抵抗分割方式を選択します。
- 表示バイアス : 1/2 バイアス法、 1/3 バイアス法、または 1/4 バイアス法を選択します。
- 表示のタイムスライス : 静的、 2 時分割、 3 時分割、 4 時分割、または 8 時分割を選択します。
- 表示波形 : 波形 A または波形 B を選択します。
- 表示データ領域 : A パターン、B パターン、または点滅を選択します。表示データ領域は切り替えることができます。
- RTC 周期割り込み (PRD) を使用して、 A パターンまたは B パターンの点滅表示を生成します。
- 参照電圧（電圧ブースト回路を稼働すると生成されます）を 16 ステップ（コントラスト調整）で調整します。

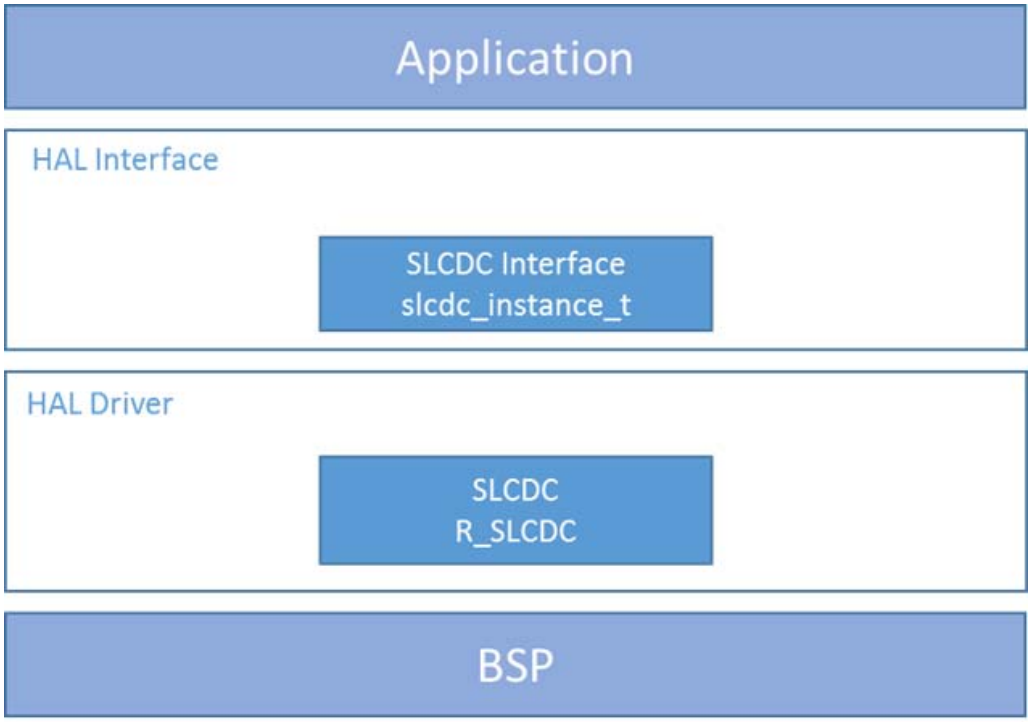


図 426: SLCD HAL モジュールのブロック図

5.2.36.1 SLCDC HAL モジュールの API の概要

セグメント LCD コントローラ HAL モジュールでは、オープン、ライト、開始、変更、クローズなどの機能の API が定義されています。使用可能なすべての API のリスト、 API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

SLCDC HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_slcdc.p_api->open(g_slcdc.p_ctrl, g_slcdc.p_cfg); SLCD デバイスを開きます。

Function Name	API の呼び出し例と説明
write	g_slcdc.p_api->write(g_slcdc.p_ctrl, start_segment, &data, segment_count); SLCD セグメントにデータを書き込みます。初期表示データを指定します。 data パラメータは、少なくとも segment_count 個の項目で構成されるバイト配列へのポインタです。各バイトは、1 つのセグメントデータレジスタと関連付けられています。タイムスライスの数が静的、2、3、または 4 の場合、データの下位 4 ビットは A パターン領域になり、上位 4 ビットは B パターン領域になります。表示領域の設定については、「.setdisplayArea」を参照してください。
modify	g_slcdc.p_api->modify(g_slcdc.p_ctrl, segment, data_mask, data); SLCD セグメントのデータを書き換えます。LCD の表示データがビット単位で書き換えられます。指定されたビット以外の値はそのまま保持されます。書き換えるデータを指定します。
start	g_slcdc.p_api->start(g_slcdc.p_ctrl); SLCD での表示を有効にします。指定されたデータが LCD に表示されます。データを表示する前に、セグメントにデータを書き込んでおく必要があります。
stop	g_slcdc.p_api->stop(g_slcdc.p_ctrl); SLCD での表示を無効化します。これにより、SLCD でのデータの表示が停止します。

Function Name	API の呼び出し例と説明
contrastIncrease	g_slcdc.p_api->contrastIncrease(g_slcdc.p_ctrl); 表示コントラストを上げます。コントラストを 1 単位ずつ上昇させます。この関数は、駆動電圧生成回路に内部昇圧方式を使用している場合に選択できます。
contrastDecrease	g_slcdc.p_api->contrastDecrease(g_slcdc.p_ctrl); 表示コントラストを下げます。コントラストを 1 単位ずつ低下させます。この関数は、駆動電圧生成回路に内部昇圧方式を使用している場合に選択できます。
setdisplayArea	g_slcdc.p_api->setdisplayArea(g_slcdc.p_ctrl, display_area); LCD の表示領域を設定します。この関数では、指定に従って、表示領域が A パターン領域か B パターン領域のいずれかに設定されます。この関数を使用して、A パターン領域と Bパターン領域のデータを交互に表示する点滅表示を設定できます。点滅を使用する場合、この関数を実行する前に RTC を操作する必要があります。 RTC を設定するには、次の手順を実行します。 1）RTC を開きます。 2）周期的割り込み要求を ½ 秒に設定します。3）RTC カウンタを開始します。4）IRQ、RTC_EVENT_PERIODIC_IRQ を有効にします。詳細な手順については、『ユーザーズマニュアル：マイクロコントローラ』を参照してください。
close	g_slcdc.p_api->close(g_slcdc.p_ctrl); 表示デバイスを閉じます。

Function Name	API の呼び出し例と説明
versionGet	g_slcdc.p_api->versionGet(&version); バージョンポインタを使用して API バージョンを取得します。

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました
SSP_ERR_ASSERTION	アサートエラーです
SSP_ERR_INVALID_ARGUMENT	無効な引数です
SSP_ERR_HW_LOCKED	SLCDC リソースがロックされています
SSP_ERR_NOT_OPEN	デバイスが開いていないか、初期化されていません。
SSP_ERR_UNSUPPORTED	サポートされていない操作です。
SSP_ERR_NOT_ENABLED	RTC が点滅操作に対して有効になっていません。

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.36.2 SLCDC HAL モジュールの動作の概要

このモジュールはセグメント LCD コントローラ（SLCDC）を使用して、データをセグメント LCD に表示します。ドライバーは LCD を初期化してデータを表示し、ドライブ電圧ジェネレータ、ディスプレイ波形、タイムスライス数、および LCD を稼働するバイアスメソッドを構成します。このモジュールでは、指定されたセグメントのセットにデータを表示するための関数、既存のセグメントデータを変更するための関数、ディスプレイを有効化および無効化するための関数、表示領域を設定するための関数、およびコントラストを調整するための関数が提供されています。LCD に表示される情報は、LCD 表示データレジスタの内容を変更することによって変更できます。

SLCDC HAL モジュールの動作に関する重要な注意事項と制限事項

- このドライバーは HAL ドライバーであり、 ThreadX RTOS に依存しません。必要な場合は、セグメント LCD HAL モジュールを ThreadX RTOS のスレッドに追加することができます。
- セグメントのシーケンスを書き込むには、スタートセグメント番号と書き込み API に書き込むセグメントの数を渡します。
- SLCDC HAL モジュールには既知の制限事項はありません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.36.3 アプリケーションへの SLCDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにセグメント LCD コントローラ HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

セグメント LCD コントローラ HAL ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（セグメント LCD HAL モジュールのデフォルトの名前は g_slcdc0. です。この名前は、対応する **[Properties]** ウィンドウで変更できます。）

SLCD HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_slcdc0 Segment LCD Driver on r_slcdc	**Threads**	**New Stack> Driver> Graphics> Segment LCD Driver on r_slcdc**

次の図に示すように r_slcdc のセグメント LCD コントローラ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

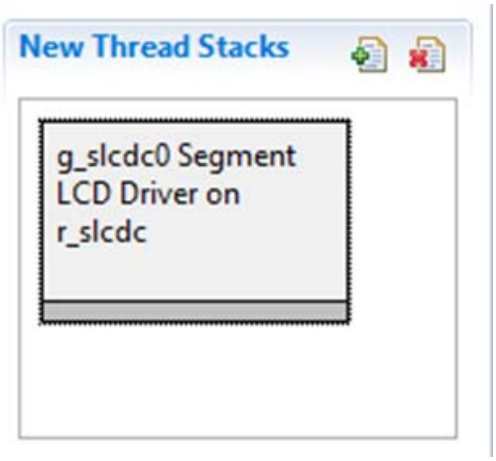


図 427: SLCD HAL モジュールのスタック

5.2.36.4 SLCDC HAL モジュールの構成

ユーザーは必要な動作に合わせて SLCD コントローラ HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

***注:** 次の表に示す構成テーブルの設定に目を通しながら、ISDE を開き、セグメント LCD コントローラ HAL モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_slcdc での SLCDC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータ値をチェックするためのコードを追加するかどうかを指定します
Name	g_slcdc0	モジュール名

ISDE Property	Value	説明
Slcdc Clock	Clock LOCO, Clock SOSC, Clock MOSC, Clock HOCO (Default: Clock HOCO)	SLCD クロックソース (LCDSCKSEL)。
Slcdc Clock Divisor	Clk Divisor LOCO 4/8/16/32/64/128/256/512/1,024 Clk Divisor HOCO 256/1,024/2,048/4,096/8,192/16,384/32,768/65,536/13,1072/262,144/524,288 (Default: Clk Divisor Hoco 16,384)	LCD クロック設定 (LCDC0)、クロック除算
Bias Method	Bias 2, Bias 3, Bias 4 (Default: Bias 2)	LCD の表示バイアス法選択 （LBAS ビット）
Time Slice	Static, Slice 2, Slice 3, Slice 4 (Default: Static)	LCD 表示の時分割数選択 （LDTY ビット）
Wave Form	Wave A, Wave B (Default: Wave A)	LCD 表示波形の選択（LWAVE ビット）
Slcdc Drive Voltage Generator	External resistance division, Internal voltage boosting, Capacitor Split (Default: External resistance division)	LCD 駆動電圧生成回路の選択 （MDSTET ビット）

注: *設定例とデフォルトは、Synergy S3A7 MCU および DK-S3A7 キットを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

SLCDC HAL モジュールのクロック構成

SLCDC クロックは **[Clocks]** タブからは構成できません。クロックの構成は、g_slcdc ドライバーの **[Properties]** ウィンドウで行います。セグメント LCD HAL モジュールの動作クロックは、 **[Properties]** ウィンドウの [SLCDC Clock] および [SLCDC Clock Divisor] の設定で指定します。セグメント LCD HAL モジュールのソースクロックは、 ISDE コンフィギュレータを使用してメイン （MOSC）、 HOCO （高速クロック発振器）、

LOCO（低速クロック発振器）またはサブクロック （SOSC）として構成できます。 HOCO および LOCO の設定では、複数のクロック除数を使用できます。

SLCDC HAL モジュールのピン構成

SLCDC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

***注：** 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。*

SLCDC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SLCDC	Pins	Select **Peripherals > Graphics: SLCDC > SLCDC0**

***注：** 選択シーケンスでは、 SLCDC0 がドライバーに必要なハードウェアターゲットであることを想定しています。*

SLCDC HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Operation Mode	Disabled, Custom, Static, 2x Slice, 3x Slice, 4x Slice, 8x Slice (Default: Custom)	動作モードの有効または無効を選択します
CAPH	None, P111 (Default: None)	容量接続ピン
CAPL	None, P112 (Default: P112)	容量接続ピン
COM0:3	None, Pn (Default: P104:107)	共通ピン

Pin Configuration Property	Value	説明
COM4:7	None, Pn (Default: None)	共通ピン
VL1:4	None, Pn (Default: P100:103)	電力供給ピン
SEG00:02, SEG06:07, SEG16:17, SEG21:25, SEG46:51	None, Pn (Default: None)	セグメントピン
SEG03	None, P303 (Default: P303)	セグメントピン
SEG04:05	None, Pn (Default: P314:315)	セグメントピン
SEG08	None, P902 (Default: P902)	セグメントピン
SEG09:15	None, Pn (Default: P312:306)	セグメントピン

Pin Configuration Property	Value	説明
SEG18:19	None, Pn (Default: P808:809)	セグメントピン
SEG20	None, P313 (Default: P313)	セグメントピン
SEG26:27	None, Pn (Default: P806:807)	セグメントピン
SEG28:34	None, Pn (Default: P608:614)	セグメントピン
SEG35:41	None, Pn (Default: P606:600)	セグメントピン
SEG42:43	None, Pn (Default: P805:804)	セグメントピン
SEG44:45	None, Pn (Default: P800:801)	セグメントピン

注: *設定例は、 Synergy S3A7 MCU および DK-S3A7 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.36.5 アプリケーションでの SLCDC HAL モジュールの使用

アプリケーションでセグメント LCD コントローラ HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 SLCDC HAL モジュールを初期化します
2) write API を使用して、セグメントのシーケンスを書き込みます

3) setdisplayArea API を使用して、表示領域または点滅表示を変更します
4) start API を使用して、表示を有効にします
5) contrastIncrease または contrastDecrease API を使用して、コントラストを調整します
6) stop API を使用して、表示を無効にします
7) close API を使用して、ドライバーを閉じます

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

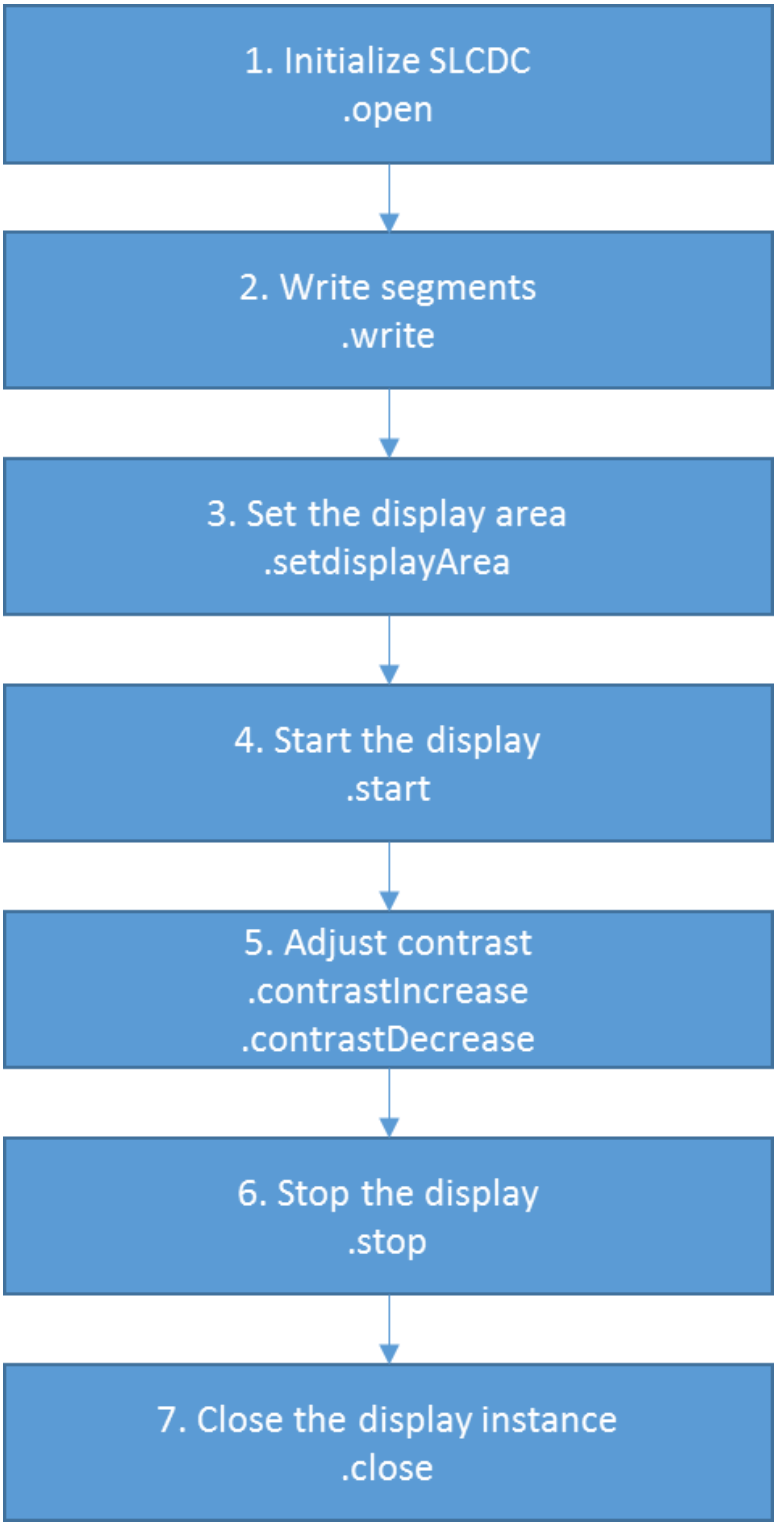

1. Initialize SLCDC
.open
2. Write segments
.write
3. Set the display area
.setdisplayArea
4. Start the display
.start
5. Adjust contrast
.contrastIncrease
.contrastDecrease
6. Stop the display
.stop
7. Close the display instance
.close

図 428: 一般的なセグメント LCDC HAL モジュールアプリケーションのフロー図

5.2.37 SCI SPI ドライバー

SCI SPI HAL モジュールは、 Synergy MCU のさまざまな SPI 機能の構成と制御をサポートします。主な特長を以下に示します。

- ドライバーの初期化
- 8 ビットデータ転送を使用した SPI 動作によるシリアル通信
- 構成可能な 4 つのクロックフェーズとクロック極性の設定
- コールバックのサポート。コールバック関数は、次のイベントで呼び出されます。
  - 転送中断
  - 転送完了
  - オーバーランエラー
- マスタモードでの SPI 通信。

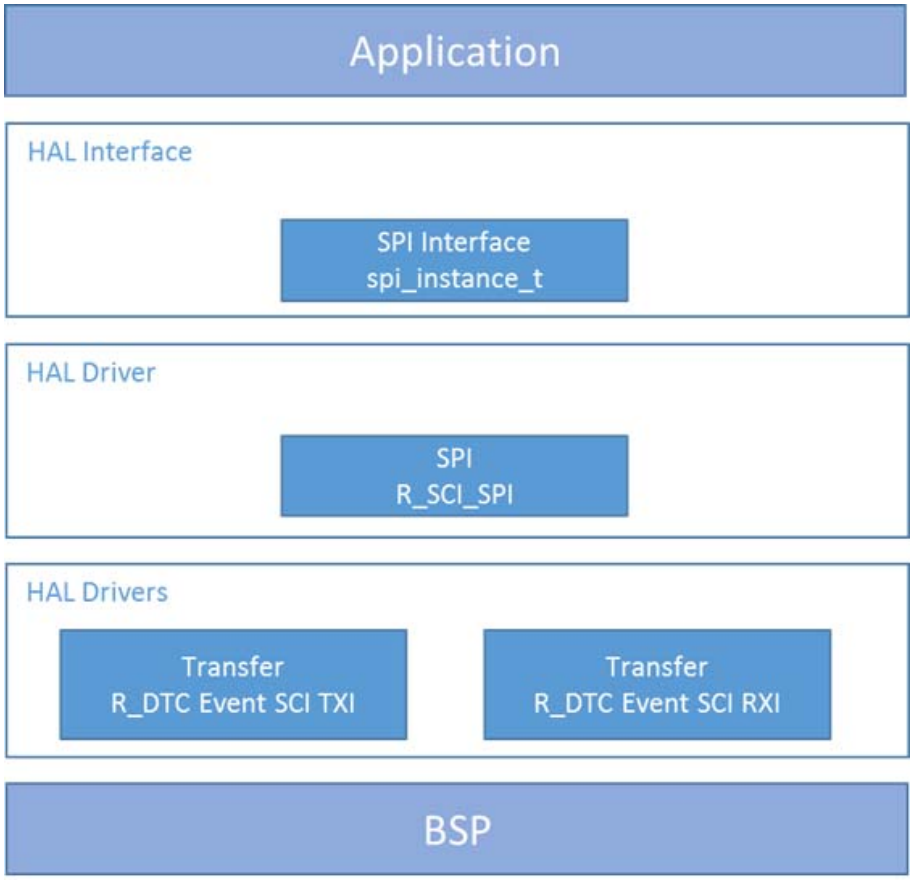


図 429: SCI SPI HAL モジュールのブロック図

5.2.37.1 SCI SPI モジュールの API の概要

SPI では、SCI のオープンとクローズおよびデータの送信と受信のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

SCI SPI HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_spi.p_api->open(g_spi.p_ctrl, g_spi.p_cfg); SPI ドライバーを開き、ハードウェアを初期化します。
close	g_spi.p_api->close(g_spi.p_ctrl); ドライバーを閉じ、SPI デバイスを解放します。

Function Name	API の呼び出し例と説明
read	g_spi.p_api->read(g_spi.p_ctrl, &read_buffer, number_of_bytes_to_read, bit_width); SPIデバイスでリード動作を実行します。
write	g_spi.p_api->write(g_spi.p_ctrl, &write_buffer, number_of_bytes_to_write, bit_width); SPI デバイスでライト動作を実行します。
writeRead	g_spi.p_api->writeRead(g_spi.p_ctrl, &read_buffer, &write_buffer, number_of_bytes_to_read_and_write, bit_width); SPI デバイスでリードとライトの（全二重）動作を実行します。
versionGet	g_spi.p_api->versionGet(&version); version ポインタを使用して API のバージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	API コールが成功しました。
SSP_ERR_IN_USE	既に開いているデバイスインスタンスを開こうとしたか、別の転送が進行中でした。
SSP_ERR_INVALID_POINTER	p_version が NULL です。
SSP_ERR_INVALID_ARGUMENT	チャネル番号が無効です。
SSP_ERR_HW_LOCKED	ロックが取得できませんでした。チャネルはビジーです。
SSP_ERR_CH_NOT_OPEN	チャネルが開かれていません。先にチャネルを開いてください。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.37.2 SCI SPI HAL モジュールの動作の概要

SCI SPI インタフェースでは、Synergy MCU の SPI 機能を構成して使用することができます。HAL モジュールでは、SPI 通信プロトコルを使用して周辺デバイスと通信できます。SCI SPI HAL モジュールのインスタン

スを開いた後は、SCI SPI モジュールのハンドルを使用して、さまざまな転送動作を実行します。デバイス制御ハンドルは、通信対象の特定の SCI SPI デバイスを示すために、 API コールの中で使用されます。

このドライバーでは、以下のことが可能です。

- ドライバーの初期化。
- SPI オペレーションによるシリアル通信です。SPI デバイスからのリードおよび SPI デバイスへのライト（および同時リード / ライト – 全二重）–read、write、writeRead API を呼び出すことによって実行されます。

ドライバーは、コールバックへのサポートも提供します。コールバック関数は、次のイベントで呼び出されます。

- 転送中断
- 転送完了
- オーバーランエラー

SCI SPI モジュールは、 8 ビットのデータ転送動作だけをサポートします。 SCI SPI モジュールは、チップセレクトとして構成された GPIO ピンを使用します。

クロックの設定 :

SCI SPI は、 PCLKA をそのクロックソースとして使用します。 PCLKA 周波数を設定するには、実行時に e^2 studio または CGC Interface のクロックコンフィギュレータを使用します。

IO ポートの設定 :

SPI を使用するには、出力ピンとして使用する I/O ポート pin(s) が、ピンコンフィギュレータで SCI SPI 周辺ピンとして構成されている必要があります。外部チップセレクトの場合は、GPIO 出力としてチップセレクトピンを構成します。

SCI SPI 割り込み :

SCI SPI の割り込みを有効にするには、e^2 studio のプロジェクトコンフィギュレータの [Threads] タブで、ドライバーモジュールを強調表示にして、 SCI RXI、 TXI、 TEI、および ERI の各割り込みの優先順位を設定します （「割り込みの設定」を参照）。

これにより、 ssp_cfg/bsp/bsp_irq_cfg.h の対応する割り込みに、選択した優先度が設定されます。

注 : *割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。*

SCI SPI HAL モジュールの動作に関する重要な注意事項と制限事項

- チップセレクト出力は、 GPIO を使用してサポートされます。
- SCI モジュールの SPI は、 8 ビットデータ転送のみを使用します。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。
- SCI SPI HAL モジュールは、MCU のデータトランスファコントローラモジュールを組み込むことにより、データ転送サポートありで有効化されます。これは、CPU の介入なしに、DTC を通じて SPI 転送を実行します。 DTC 転送はデフォルトで有効になっており、 IRQ モード転送を使用する場合は、構成から DTC 転送を削除する必要があります。
- SCI に SPI を実装するときは、マスタモードのみがサポートされます。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.37.3 アプリケーションへの SCI SPI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SPI HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。*

SPI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（SPI のデフォルトの名前は g_spi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

SPI の選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_spi0 SPI Driver on g_sci_spi	Threads	New Stack> Driver> Connectivity> SPI Driver on g_sci_spi

次の図に示すように、 SPI HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

次の図は、HAL/共通スレッドに追加されたSCIドライバーのSPIを示したものです。リソースの名前は g_spi0. です。 SPI HAL モジュールは転送ドライバーをサポートできますが、データトランスファコントローラを使用すると、ハードウェアペリフェラルと RAM のデータバッファの間でデータを送受信できます（CPU が転送を実行するのではありません）。転送ドライバーは、デフォルトで送信と受信の両方に追加されます。

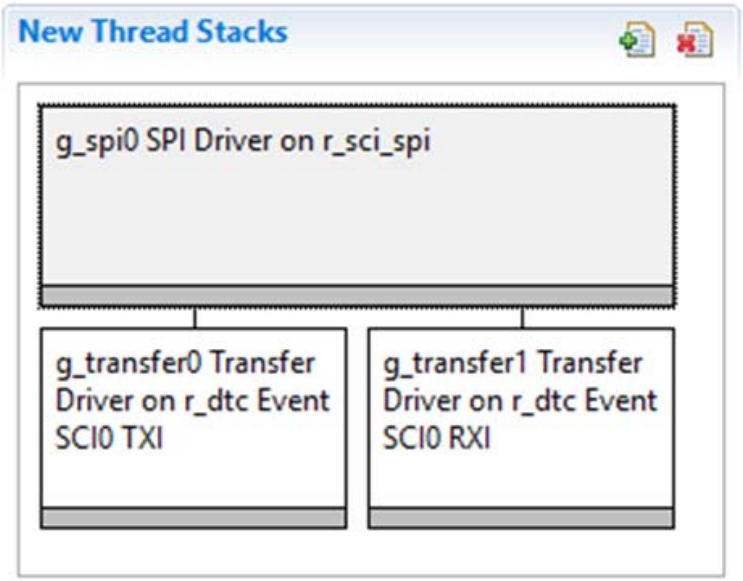


図 430: SCI SPI HAL モジュールのスタック

5.2.37.4 SCI SPI HAL モジュールの構成

ユーザーは必要な動作に合わせて SPI HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータの [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、SPI を作成して、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_sci_spi での SCI SPI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	Default (BSP), Enabled, Disabled	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	ドライバーのインスタンスの名前。ユーザーは名前を変更できます。

ISDE Property	Value	説明
Channel	0	SPI通信で使用するSCIチャネル番号。
Operating Mode	Master	SPI ドライバーの動作方法 – マスタまたはスレーブ （サポートされていません）。
Clock Phase	Data sampling on odd edge, data variation on even edge Data sampling on even edge, data variation on odd edge (Default: Data sampling on odd edge, data variation on even edge)	クロック信号エッジおよびデータ操作に関するクロック信号の動作方法。
Clock Polarity	Low when idle, High when idle (Default: Low when idle)	動作が実行されていないときは高出力。または、動作が実行されていないときは低出力。
Mode Fault Error	Disable, Enable (Default: Disable)	エラーの処理方法。
Bit Order	MSB First, LSB First (Default: MSB First)	送信 / 受信されるバイト内のビット順序。
Bitrate	100000	ビット / 秒で表された送信速度。
Bit Rate Modulation Enable	Enable, Disable (Default: Disable)	MDDR レジスタを使用して、ビットレートを均一になるように修正できます。これは、データ通信でのボーレートに関するエラーの削減に役立ちます。非標準の水晶周波数を使用するシステム、および低ボーレートの通信に、メリットがあります。
Callback	NULL	ユーザーコールバック関数をopenで登録できます。このコールバック関数が指定されている場合、spi_event_t. で定義されている条件のそれぞれに対し、割り込みサービスルーチン(ISR)から呼び出されます。
Receive Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	SPI ドライバーは割り込み駆動です。MCU の割り込み優先順位レベル。
Transmit Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	SPI ドライバーは割り込み駆動です。MCU の割り込み優先順位レベル。
Transmit End Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	SPI ドライバーは割り込み駆動です。MCU の割り込み優先順位レベル。

ISDE Property	Value	説明
Error Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	SPI ドライバーは割り込み駆動です。MCU の割り込み優先順位レベル。

注: *設定例とデフォルトは、Synergy S7 MCU ファミリを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

多くの場合、SPI を介した通信では、SPI ドライバーをデフォルト以外の設定にする必要があります。たとえば、ビットレート、クロックの極性、ビット順序に対する変更は、スレーブデバイスによって異なる場合があります。

SCI SPI HAL モジュールのクロック構成

SCI は、周辺クロック A（PCLKA）からクロックを供給されます。クロック周波数は、ISDE のコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。PCLKA の示されている値で必要な SPI ビットレートを実現できることを確認してください。指定したビットレートを実現できない場合、ISDE は表示されません。実行時に、SPI HAL モジュールは SCI を正しいビットレートに構成することを試み、目的のビットレートを設定できない場合はエラーを返します。ビットレートは、次の表の式を使用して計算されます。式の結果（N）が 0 ～ 255 の範囲内にある場合は、そのビットレートを実現できます。

ボーレートの計算式

SPI HAL	Bitrate calculation	説明
SPI on SCI	$N = \frac{PCLKA\ (MHz)}{8 * 2^{(2n-1)} * \left(\frac{256}{M}\right) * B} - 1$ **図 431: N = Peripheral register value.This must be in the range of 0 to 255 PLCKA = value of PLCKA in MHz n = 0, 1, 2 or 3 M = Bit Rate Modulation Index 128 < M < 256 If the Bit Rate Modulation is disabled, then M=256 B = Desired Bit Rate**	

SCI SPI HAL モジュールのピン構成

SCI は、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は SPI ピンの選択例を示しています。

SCI SPI HAL ドライバーのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SPI on SCI	Pins	Select Peripherals > Connectivity:SCI > SCIx Where x is the required SCI peripheral channel.

SCI 上の SPI のピン構成設定 – SCI0 を使用する場合の例

Pin Configuration Property	Value	説明
Pin Group Selection	Mixed, _A only, _B only	Synergy デバイスは、_A、_B によって示される複数のピンの場所を使用してペリフェラルをサポートします。[Mixed] を選択すると、ユーザーは場所 (_A と _B) の任意の組み合わせを選択できます。[_A] を選択すると、ユーザーは _A の場所のみを選択できます。[_B] を選択すると、ユーザーは _B の場所のみを選択できます。
Operation Mode	Disabled Custom Asynchronous UART Simple SPI Simple I2C Synchronous UART Smart Card	動作モードを設定します : 簡易 SPI。
TXD_MOSI	None, P411, P101	MOSI として使用するポートピンを指定します。
RXD_MISO	None, P410, P100	MISO として使用するポートピンを指定します。
SCK	None, P412, P102	CLK として使用するポートピンを指定します。
CTS_RTS_SS	None, P413, P103	SSとして使用するポートピンを指定します。

注: *設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.37.5 アプリケーションでの SCI SPI HAL モジュールの使用

アプリケーションで SCI SPI を使用する手順の例は次のとおりです。

g_spi.p_api->open() 関数を最初に呼び出す必要があります。残りの呼び出しは、アプリケーションの要件に応じて任意の順序で使用できます。

1) SCI SPI によって実装された SPI を使用して、SPI インスタンスを開きます。SPI インタフェースを通して SCI SPI ドライバーが呼び出されます。

```
g_spi.p_api->open (g_spi.p_ctrl, g_spi.p_cfg)
```

p_ctrl と p_cfg は、構成手順の後で自動生成される制御構造体と構成構造体のインスタンスです。

2) 以下を呼び出して、スレーブデバイスへの書き込みを初期化します

```
g_spi.p_api->write (g_spi.p_ctrl, source, length,
SPI_BIT_WIDTH_8_BITS);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

3) 以下を呼び出して、スレーブデバイスからの読み取りを初期化します

```
g_spi.p_api->read(g_spi.p_ctrl, dst, length,
SPI_BIT_WIDTH_8_BITS);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

4) 以下を呼び出して、スレーブデバイスとの間の双方向の同時転送を開始します

```
g_spi.p_api->writeRead(g_spi.p_ctrl, source,
   dst, length, SPI_BIT_WIDTH_8_BITS);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

5) SPI チャネルをクローズするには、次を呼び出します。

```
g_spi.p_api->close(g_spi,p_ctrl)
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御構造体です。

次のフロー図は、これまでに説明した一般的な手順を示したものです。

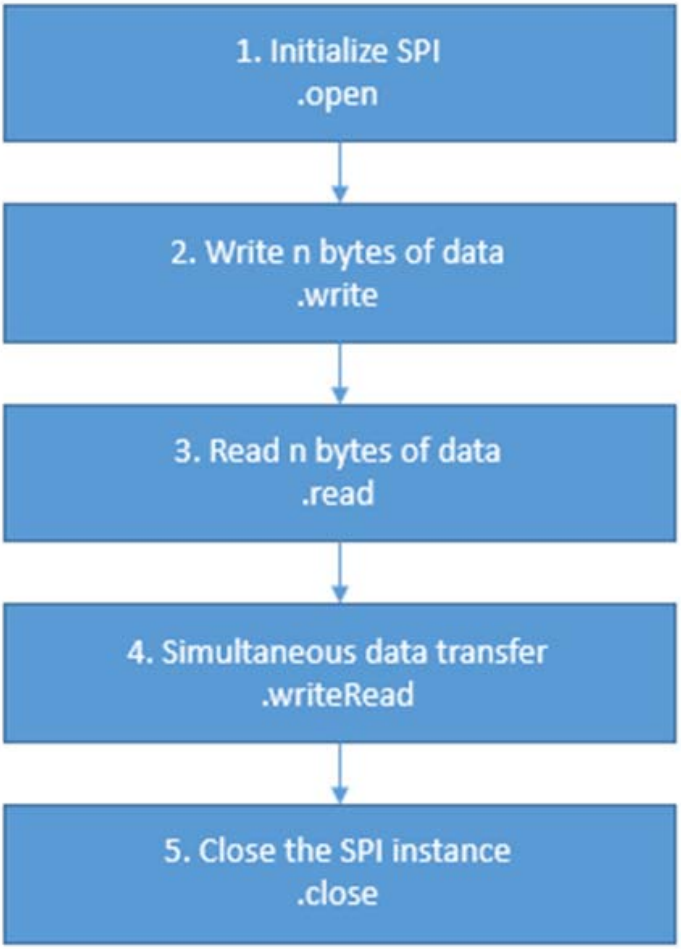


図 432: 一般的な SPI アプリケーションのフロー図

5.2.38 SPI ドライバー

SPI HAL モジュールは、以下の主要な機能をサポートします。

- ドライバーの初期化
- SPI 転送機能 :
  - 4 ワイヤ方式を使用する SPI 動作によるシリアル通信が可能です
  - マスタモードとスレーブモードでシリアル通信できます
  - シリアル転送クロックの極性の切り替え
  - シリアル転送クロックの位相の切り替え

- データ形式
  - MSB ファーストと LSB ファーストの選択が可能
  - 転送ビット長を 8、 16、 32 ビットから選択可能
- エラー検出
  - モード障害の検出
  - オーバーランエラーの検出
  - パリティエラーの検出
- SSL 制御機能
  - マスタモードでは、チャネルごとに最大 4 つの SSL 信号 （SSLn0 から SSLn3）を内部的に選択できます
  - 外部ハードウェアスレーブ選択をマスタモードで使用できます
- 割り込み
  - RSPI 受信割り込み （受信バッファフル）
  - RSPI 送信割り込み （送信バッファエンプティ）
  - RSPI エラー割り込み （モード障害エラー、オーバーランエラー、パリティエラー）
- 遅延
  - SPI クロック遅延の追加
  - スレーブ選択ネゲート遅延の追加
  - 次アクセス遅延の追加

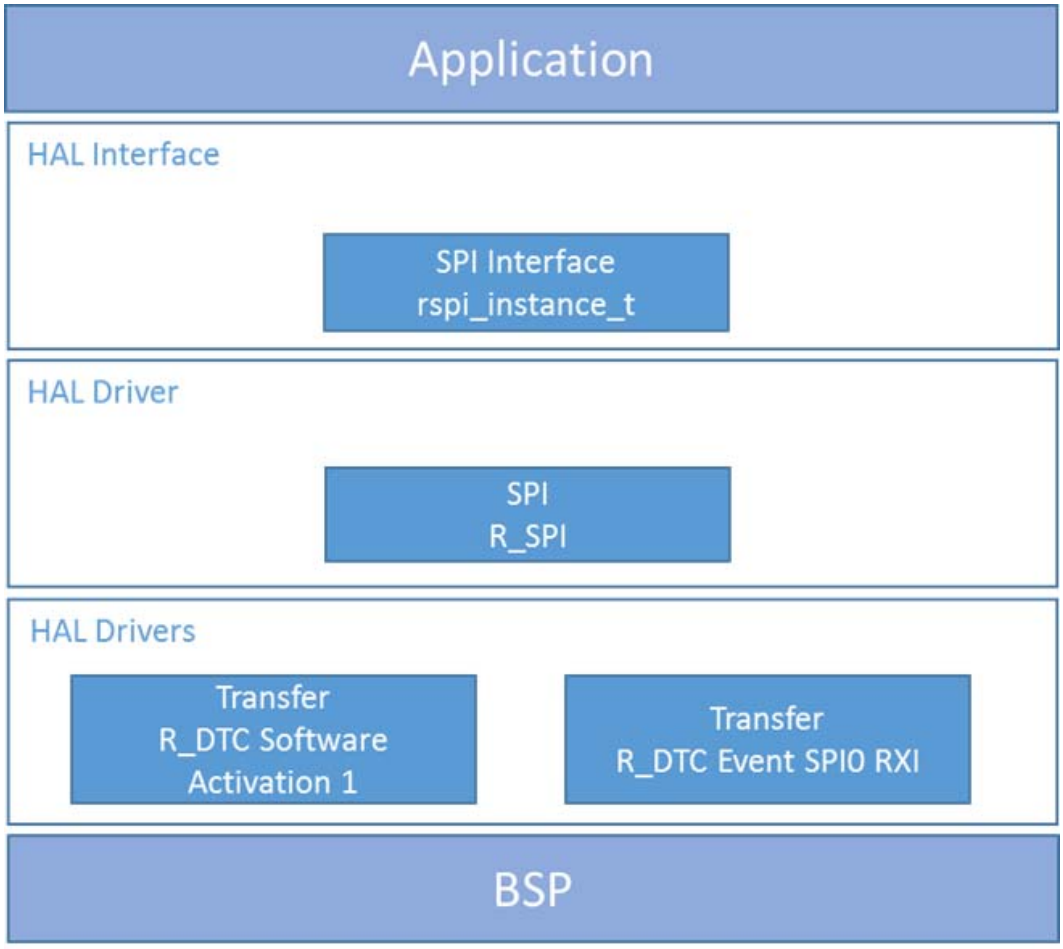


図 433: RSPI HAL モジュールのブロック図

5.2.38.1 RSPI HAL モジュールの API の概要

RSPI HAL モジュールでは、オープン、クローズ、リード、ライト、その他の有用な機能のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。その後で、ステータス戻り値の表を示します。

RSPI HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
open	g_spi.p_api ->open(g_spi.p_cntl, g_spi.p_cfg); 指定された SPI デバイスを開きます。
read	g_spi.p_api->read(g_spi.p_ctrl, dst16, length, SPI_BIT_WIDTH_16_BITS); SPI デバイスからデータを受信します。
write	g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS); SPI デバイスにデータを送信します
writeRead	g_spi.p_api ->writeRead (g_spi.p_cntl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); SPI デバイスとの間で、データの送信と受信を同時に実行します（全二重通信）。writeRead API は、ミューテックスオブジェクトをフェッチし、SPI HAL レイヤーで SPI データ送信を処理し、SPI HAL レイヤーからデータを受信します。またこの API は、イベントフラグ待ちを使用して、データ転送の完了と同期します。
close	g_spi.p_api->close(g_spi,p_ctrl) 制御ハンドルで指定された SPI デバイスを無効にします。またバスにデバイスが接続されていない場合は、RTOS サービスを終了します。この関数は、ハンドルで指定された SPI チャネルへの電源を遮断し、関連付けられた割り込みを無効にします。

Function Name	API の呼び出し例と説明
versionGet	g_spi.p_api ->versionGet (&version); 基礎となるドライバーのバージョン情報を取得します。

注: 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に完了しました
SSP_ERR_INVALID_MODE	無効なモード
SSP_ERR_INVALID_CHANNEL	無効なチャネル
SSP_ERR_IN_USE	使用中エラー
SSP_ERR_INVALID_ARGUMENT	無効な引数
SSP_ERR_QUEUE_UNAVAILABLE	キューが使用不能
SSP_ERR_INVALID_POINTER	無効なポインタ
SSP_ERR_INTERNAL	内部エラー
SSP_ERR_TRANSFER_ABORTED	転送中断
SSP_ERR_MODE_FAULT	モード障害
SSP_ERR_READ_OVF	リードオーバーフロー
SSP_ERR_PARITY	パリティエラー
SSP_ERR_OVERRUN	オーバーランエラー
SSP_ERR_UNDEF	不明なエラー
SSP_ERR_TIMEOUT	タイムアウトエラー
SSP_ERR_NOT_OPEN	デバイスが開かれていない

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.38.2 RSPI HAL モジュールの動作の概要

SPI HAL モジュールでは、 SPI 通信プロトコルを使用して周辺デバイスと通信できます。 SPI HAL モジュールのインスタンスを開いた後、SPI モジュールのハンドルを使用して、さまざまな転送動作を実行します。デバイス制御ハンドルは、通信対象の特定の SPI デバイスを示すために、 API コールの中で使用されます。

このドライバーでは、以下のことが可能です。

- ドライバーの初期化。
- SPI オペレーションによるシリアル通信です。

ドライバーは、コールバックへのサポートも提供します。コールバック関数は、次のイベントで呼び出されます（spi_event_t を参照）。

- 転送中断
- 転送完了
- モード障害
- エラーイベント

SPI モジュールは、8、16、32 ビットのデータ転送をサポートします。SPI モジュールは、チップセレクトとして構成された GPIO ピンをサポートします。さらに、 SPI は、専用チップ選択信号 SSLn0 から SSLn3 までをサポートします。 SPI で SSL ピンが有効になっていると、すべてのチップセレクト処理はハードウェアによって行われます。

クロックの設定 :

SPI は、PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、実行時に e^2 studio または CGC Interface のクロックコンフィギュレータを使用します。

***注：** S1 デバイスの場合、 SPI クロックソースは PCLKB です。*

IO ポートの設定 :

SPI を使用するには、出力ピンとして使用する I/O ポートピンが、ピンコンフィギュレーターで SPI ピンとして設定されている必要があります。外部チップ選択を使用するには、チップを構成して、ピンを GPIO 出力として選択します。

拡張構成 :

SPI ドライバーには、ハードウェアに固有の拡張設定が数多く存在します。

***注：** すべてのパラメータは、 SPI 拡張ドライバー構成構造体 spi_on_rspi_cfg_t で設定されます。*

RSPI HAL モジュールの動作に関する重要な注意事項と制限事項

SPI HAL ドライバーの構成中に、割り込みを異なる優先順位レベルに設定すると、適切に動作しなくなる可能性があります。

データ転送サポートでは、MCU のデータトランスファコントローラモジュールを取り込むことで、SPI HAL ドライバーモジュールが有効化されます。これで、CPU への介入なしで DTC を通じて SPI 転送を実行できます。

アプリケーションでは、DTC によるデータ転送は、通常の SPI 転送と同じ方法で使用されます。DTC 転送を有効にするには、 SPI HAL モジュールの下に DTC モジュールを追加します。

SPI モジュールは、CPU ベースと DTC ベース両方の転送モードで、8、16、32 ビットのデータ転送をサポートします。

重要な注意事項 : 16 ビットと 32 ビットの転送は、エンディアンスワップされます。

SPI HAL モジュールの性能に関する注意事項

R_RSPI モジュールは、複数の異なるモードに構成でき、それぞれ性能特性が異なります。 DTC をリセットするため DTC 転送のセットアップには CPU ベースの転送よりわずかに長い時間がかかりますが、DTC 転送は、 CPU による介入が必要ないため、 1 フレームより長い転送の場合は優れた性能を提供します。

ライト動作では、モジュールは送信専用モードに構成され、受信割り込みは無効になり、受信データは無視されます。高ビットレートの CPU ベースのライト動作では、送信 ISR が繰り返し呼び出されて、他のコードの実行をブロックする可能性があります。WriteRead 動作とリード動作では、モジュールが全二重モードに構成されます。

転送の間隔には 3 SPI クロックサイクルの下限が設けられているため、実質的なビットレートは構成より遅くなります。高ビットレートでは （特に、 CPU ベースの転送の場合）、転送と転送の間の時間がさらに長くなることがあります。

モジュールは、マスタモードのときはハードウェアがアイドル状態になるまで待ってから、スレーブモードのときはすべてのデータが送信または受信されるまで待ってから、コールバックを呼び出します。

- スレーブモードで R_RSPI を使用する場合は、データを偶数クロックエッジでサンプリングするか、またはマスタがフレームとフレームの間でスレーブ選択ラインをアサート解除する必要があります。これはハードウェアでの制限事項です。
- S124、 S128、 S3A6 は、 SSL レベルキープをサポートしていません。
- ドライバーが DTC を使用して開かれた後は、DTC に対して構成されている転送幅をすべての転送で使用する必要があります。
- 16 ビットと 32 ビットの転送は、エンディアンスワップされます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.38.3 アプリケーションへの RSPI HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに RSPI HAL モジュールを組み込む方法について説明します。

注 : *このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、このドキュメントの最後にある「参照文献」セクションに記載された『Getting Started Guide for SSP』を参照し、SSP ベースのアプリケーションを作成する際に重要な各手順を管理する方法を確認してください。*

RSPI HALドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（SPI フレームワークのデフォルトの名前は g_spi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

RSPI HAL ドライバーの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_spi0 SPI Driver on r_rspi	Threads	New Stack> Driver> Connectivity> SPI Driver on r_rspi

次の図に示すように、r_rspi の RSPI HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な低レベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、複数のスタックで使用できるため 1 回のみ追加する必要があります。ピンクの帯で示されるモジュールには、ローレベルドライバーの選択が必要です。これらはオプションの場合や推奨の場合があり、このテキストを含むブロックで示されます。ローレベルドライバーの追加が必要な場合は、モジュールの説明に 「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、 [New] アイコンが表示され、可能な選択肢が示されます。

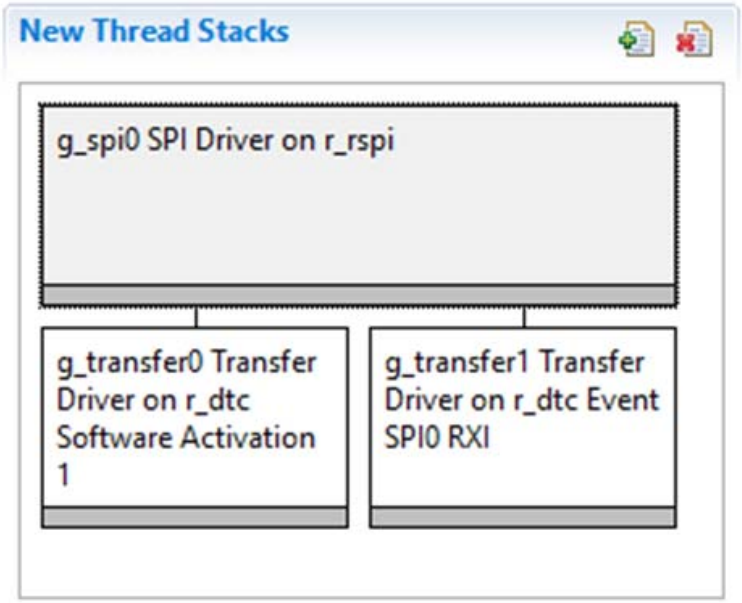


図 434: RSPI HAL モジュールのスタック

5.2.38.4 RSPI HAL モジュールの構成

r_rspi の RSPI HAL モジュールを必要な動作用に構成します。 [SSP configuration] ウィンドウでは、正常な動作のために必要な低レベルモジュールの構成（割り込みや動作モードなど）が自動的に識別されます（ブロックが赤で強調表示される）。変更しても競合が発生しないプロパティのみが変更可能になります。変更で

きないプロパティは「ロック」され、ISDE の [property] ウィンドウでは「ロックされた」プロパティにはロックアイコンが表示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

***注：** 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_rspi での RSPI HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_spi0	モジュール名
Channel	0	デバイスが接続されている SCI または SPI チャネル番号。
Operating Mode	Master, Slave Default: Master	マスターデバイスまたはスレーブデバイスとして設定します。 ***注：** SSP の現在のバージョンは、SPI マスターモードのみをサポートしています。*
Clock Phase	Data sampling on odd edge, data variation on even edge/Data sampling on even edge, data variation on odd edge Default: Data sampling on odd edge, data variation on even edge	奇数または偶数クロックエッジのいずれかのデータサンプリングを選択します。

ISDE Property	Value	説明
Clock Polarity	Low when idle, High when idle Default: Low when idle	アイドル時のクロックレベル。
Mode Fault Error	Enable, Disable Default: Disable	モード障害エラー（マスタおよびスレーブの競合）フラグを示します。
Bit Order	MSB First, LSB First Default: MSB First	MSBファーストとLSBファーストのいずれかの送信順序を選択します
Bitrate	500000	送信または受信レート。1秒あたりのビット。
Callback	NULL	オプションのコールバック関数ポインタ。
SPI Mode	SPI Operation, Clock synchronous operation Default: SPI Operation	SPI またはクロック同期の動作モードを選択します。
SPI Communication Mode	Full Duplex, Transmit Only Default: Full Duplex	通信方式を選択します（全二重または送信のみ）。
Slave Select Polarity(SSL0)	Active Low, Active High Default: Active Low	SSL0 の信号極性を選択します
Slave Select Polarity(SSL1)	Active Low, Active High Default: Active Low	SSL1 の信号極性を選択します
Slave Select Polarity(SSL2)	Active Low, Active High Default: Active Low	SSL2 の信号極性を選択します

ISDE Property	Value	説明
Slave Select Polarity(SSL3)	Active Low, Active High Default: Active Low	SSL3 の信号極性を選択します
Select Loopback1	Normal, Inverted Default: Normal	loopback1 を選択します
Select Loopback2	Normal, Inverted Default: Normal	loopback2 を選択します
Enable MOSI Idle State	Enable, Disable Default: Disable	MOSI アイドル固定値および選択肢を選択します
MOSI Idle State	MOSI Low, MOSI High Default: MOSI Low	MOSI アイドル固定値および選択肢を選択します
Enable Parity	Enable, Disable Default: Disable	パリティを有効または無効にします
Parity Mode	Parity Odd, Parity Even Default: Parity Odd	パリティを選択します
Select SSL(Slave Select)	SSL0, SSL1, SSL2, SSL3 Default: SSL0	使用するスレーブを選択します (0-SSL0、1-SSL1、2-SSL2、3-SSL3)
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep Default: SSL Level Do Not Keep	転送完了後の SSL レベルを選択します（0- ネゲート、 1- 維持）

ISDE Property	Value	説明
Clock Delay Enable	Clock Delay Enable, Clock Delay Disable Default: Clock Delay Disable	クロック遅延有効化の選択
Clock Delay Count	Clock Delay 1 thru 8 RSPCK Default: Clock Delay 1 RSPCK	クロック遅延カウントの選択
SSL Negation Delay Enable	Negation Delay Enable, Negation Delay Disable Default: Negation Delay Disable	SSL ネゲート遅延有効化の選択
Negation Delay Count	Negation Delay 1 thru 8 RSPCK Default: Negation Delay 1 RSPCK	ネゲート遅延カウントの選択
Next Access Delay Enable	Next Access Delay Enable, Next Access Delay Disable Default: Next Access Delay Disable	次アクセス遅延有効化の選択
Next Access Delay Count	Next Access Delay 1 thru 8 RSPCK Default: Next Access Delay 1 RSPCK	次アクセス遅延カウントの選択
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	受信割り込み優先順位の選択

ISDE Property	Value	説明
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	送信割り込み優先順位の選択
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Priority 2	エラー割り込み優先順位の選択

注: *上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc Software Activation 1 上の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択

ISDE Property	Value	説明
Linker section to keep DTC vector table	.ssp_dtc_vector_table	DTC ベクタテーブルを保持するリンカセクションの選択
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Software Activation 1	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

r_dtc イベント SCI0 RXI 時の DTC HAL モジュールの構成設定

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	4 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SPI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

RSPI HAL モジュールのクロック構成

SPI は、周辺クロック A（PCLKA）からクロックを供給されます。クロック周波数は、ISDE のコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。PCLKA の示されている値で必要な SPI ビットレートを実現できることを確認してください。指定したビットレートを実現できない場合、ISDE は表示されません。実行時に、SPI ドライバーは SPI を正しいビットレートに構成することを試み、目的のビットレートを設定できない場合はエラーを返します。ビットレートは、次の表の式を使用して計算されます。式の結果（n）が 0 ～ 255 の範囲内にある場合は、そのビットレートを実現できます。

ボーレートの計算式

SPI HAL	Bitrate calculation	説明
SPI on SPI	$n = \frac{PCLKA\ (MHz)}{2 * 2^N * B} - 1$	n = 周辺レジスタの値。 これは 0 ～ 255 の範囲でなければなりません PLCKA = PLCKA の値 （MHz 単位） N = 0、 1、 2、 3 B = 必要なビットレート

RSPI HAL モジュールのピン構成

SPI 周辺モジュールは、 MCU のピンを使用して外部デバイスと通信します。 I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では SPI ピンの選択例を示します。

RSPI HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
RSPI	Pins	Select Peripherals > RSPI > SPI0_Pin_Option_A/B

***注:** 上の選択シーケンスでは SPIO がドライバーに必要なハードウェアターゲットであることを想定し、下の選択シーケンスでは SPI_0 が必要なターゲットであることを想定しています。*

RSPI HAL モジュールのピン構成設定

Property	Value	説明
Operation Mode	Disabled, Custom, Enabled (Default: Disabled)	SPI 動作の場合は [Enabled] を選択します
MISO	None, P100, P410 (Default: None)	MISO ピンの選択
MOSI	None, P101, P411 (Default: None)	MOSI ピンの選択
RSPCLK	None, P102, P412 (Default: None)	RSPCLK ピンの選択
SSL0:3	None, P103:106, P413:415 (Default: None)	SSL0:3 ピンの選択

注: *例では、 Synergy S7G2 シリーズ 7 グループ MCU および SK-S7G2 キットを使用するプロジェクトに対する設定が示されています。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.38.5 アプリケーションでの RSPI モジュールの使用

SPI を使用して SPI アプリケーションを作成するには、次の手順を実行します。g_spi.p_api->open 関数を最初に呼び出す必要があります。残りの呼び出しは、アプリケーションの要件に応じて任意の順序で使用できます。

1) SPI によって実装された SPI を使用して、SPI インスタンスをオープンします。SPI ドライバーは、SPI インタフェースを通じて呼び出されます

```
g_spi.p_api->open
   (g_spi.p_ctrl, g_spi.p_cfg)
```

p_ctrl と p_cfg は、 SPI の設定手順の後に自動生成される制御および構成構造体のインスタンスです。

2) 以下を呼び出して、スレーブデバイスへの書き込みを初期化します

```
g_spi.p_api->write (g_spi.p_ctrl, source, length,
SPI_BIT_WIDTH_8_BITS);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

3) 以下を呼び出して、スレーブデバイスからの読み取りを初期化します

```
g_spi.p_api->read(g_spi.p_ctrl, dst16, length,
SPI_BIT_WIDTH_16_BITS);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

4) 以下を呼び出して、スレーブデバイスとの間の双方向の同時転送を開始します

```
g_spi.p_api->writeRead(g_spi.p_ctrl, source,
   dst16, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御インスタンスです。

5) SPI チャネルをクローズするには、次を呼び出します。

```
g_spi.p_api->close(g_spi,p_ctrl);
```

g_spi.p_ctrl は、 open 呼び出しで使用した同じ制御構造体です。

これらの一般的な手順を、次の図の通常の動作フローに示します。

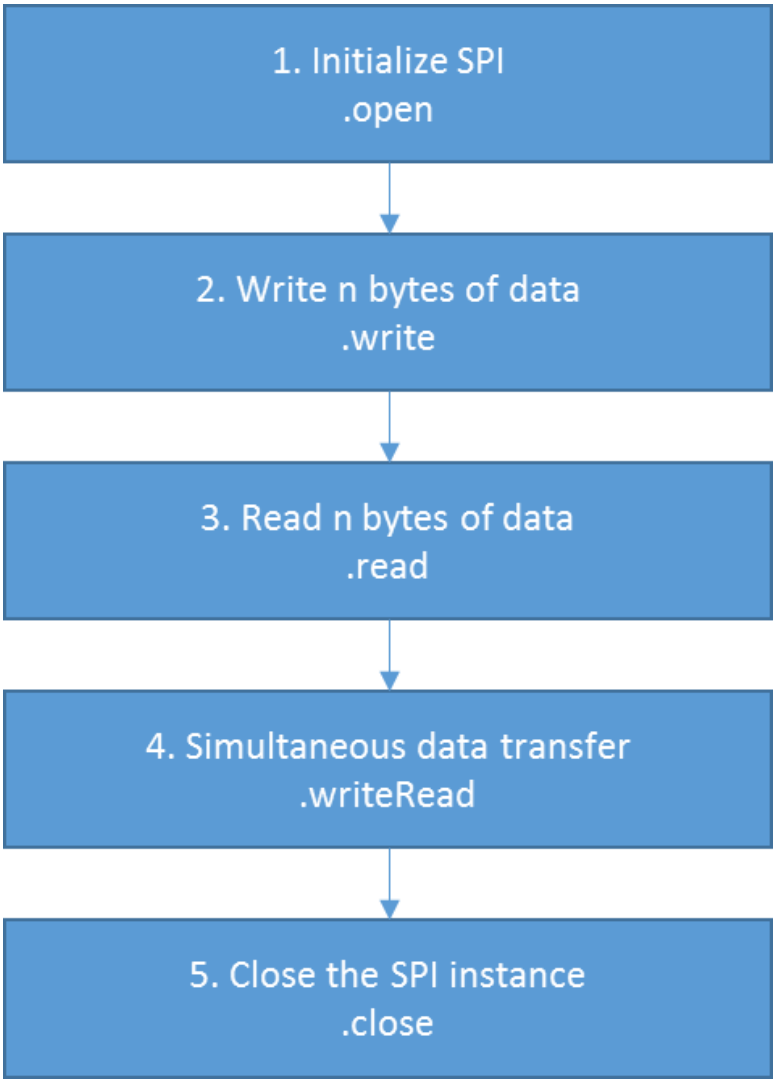


図 435:　一般的な RSPI HAL モジュールアプリケーションのフロー図

5.2.39 UART HAL モジュール

UART HAL モジュールは、標準の UART プロトコルをサポートします。UART モード（SCI 上の UART）の SCI と共に使用される UART HAL モジュールは、（標準的な UART プロトコルに加えて）次の機能をサポートします。

- 全二重 UART 通信
- 複数のチャネルを使用した同時通信
- 割り込み駆動型のデータ送信と受信
- イベントコードを引数として渡す、ユーザーコールバック関数の呼び出し

- 実行時のボーレートの変更
- UART トランザクション中のハードウェアリソースのロック
- CTS/RTS ハードウェアフロー制御（関連付けられた IOPORT ピンを使用し、ユーザー定義コールバック関数でサポート）
- DTC 転送モジュールとの統合

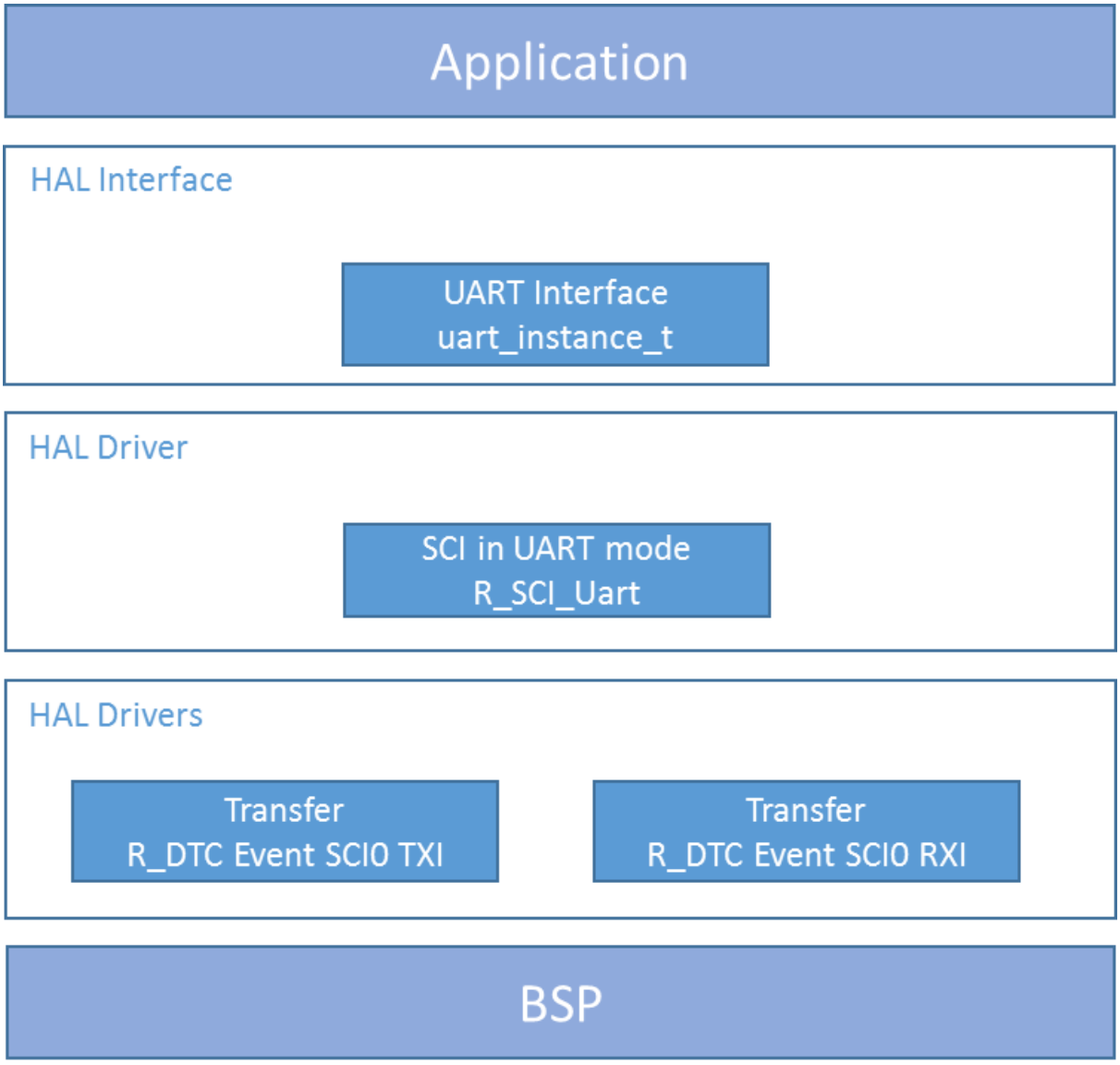


図 436: UART HAL モジュールのブロック図

5.2.39.1 UART HAL モジュールの API の概要

UART HAL モジュールのインタフェースでは、オープン、クローズ、リード、ライト、ボーレットの設定などの重要な機能のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

UART HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
.open	g_uart0.p_api->open(g_uart0.p_ctrl, g_uart0.p_cfg); UART デバイスを開きます。
.read	g_uart0.p_api->read(g_uart0.p_ctrl, uart0_buf, uart0_rcv_num); UART デバイスから読み取ります。転送インスタンスを受信に使用する場合、受信バイトは、読み取り入力バッファ uart0_buf. に直接保存されます。転送が完了すると、イベント UART_EVENT_RX_COMPLETE. により、コールバックが呼び出されます。アクティブな転送の外部で受信されるバイトは、イベント UART_EVENT_RX_CHAR. で指定されるコールバック関数で受信されます。
.write	g_uart0.p_api->write(g_uart0.p_ctrl, uart0_buf, uart0_send_num) UART デバイスに書き込みます。書き込みバッファは、書き込みが完了するまで使用されます。書き込みが完了するまでは、書き込みバッファの内容を上書きしないでください。ライトが完了すると（すべてのバイトが有線で完全に送信されると）、イベント UART_EVENT_TX_COMPLETE. でコールバックが呼び出されます。
.baudSet	g_uart0.p_api->baudSet(g_uart0.p_ctrl, (uint32_t)9600); ボーレートを変更します。
.infoGet	g_uart0.p_api->infoGet(g_uart0.p_ctrl, &uart_info); ドライバー固有の情報を取得します。

Function Name	API の呼び出し例と説明
.close	g_uart0.p_api->close(g_uart0.p_ctrl); UART デバイスを閉じます。
.versionGet	g_uart0.p_api->versionGet(&uart_version); バージョンポインタを使用して API バージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	チャネルは正常に動作しています。
SSP_ERR_IN_USE	制御ブロックが既に開かれているか、チャネルが別のインスタンスによって使用されています。
SSP_ERR_ASSERTION	UART 制御ブロックに対するポインタが NULL であるか、または構成構造体が NULL です。
SSP_ERR_HW_LOCKED	チャネルがロックされています。
SSP_ERR_INVALID_MODE	チャネルが非 UART モード用に使用されているか、設定されているモードが誤っています。
SSP_ERR_INVALID_ARGUMENT	設定構造体に無効なパラメータ設定値が見つかりました。あるいは、ソース / 宛先アドレスまたはデータサイズが、データ長に対して無効です。
SSP_ERR_NOT_OPEN	制御ブロックが開かれていません。
SSP_ERR_UNSUPPORTED	SCI_UART_CFG_RX_ENABLE が 0 に設定されています。

注: *ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

5.2.39.2 UART HAL モジュールの動作の概要

UART HAL モジュールは、標準の UART プロトコルを使用するデータフローを管理します。高レベルの API は、リード、ライト、および UART インタフェースに対するボーレートの設定に使用されます。さらに、通常、割り込みは下位レベルのアクティビティの管理を簡単にするために使用されます。

注: 以下の機能が正常に動作するためには、割り込みを有効にする必要があります。

SCI 上の UART の RXI 割り込み

RXI 割り込みは、 UART ポートから受信するデータのフローを制御するために使用されます。受信データ量が期待される読み取り長に達した場合、 ISR はユーザー定義のコールバック (p_callback) を呼び出して引数 uart_callback_args_t を渡し、受信データが完了したことを通知します。外部 RTS 動作オプションが有効になっている場合、ISR は RTS 外部ピン制御のための UART コールバック関数を 2 回呼び出します（ISR の先頭で 1 回、最後で 1 回）。コールバック関数を使用して、RTS 関数をエミュレートできます（「SCI 上の UART のハードウェアフロー制御」を参照）。この割り込みは、構成パラメータ SCI_UART_CFG_RX_ENABLE で受信が有効になっている限り、 open API でアクティブ化されます。

SCI 上の UART の TXI 割り込み

TXI 割り込みは、 write API による要求に従い、UART ポートへの連続するデータ転送を処理します。送信循環バッファにデータがなくなると、 ISR は TXI 割り込みを非アクティブ化し、 TEI 割り込みをアクティブ化して、データ送信の最後のシーケンスを処理します。この割り込みは、構成パラメータ SCI_UART_CFG_TX_ENABLE によって送信が有効になっている限り、 write API でアクティブ化されます。

SCI 上の UART の TEI 割り込み

TEI 割り込みは、write API によって要求された UART ポートへの最後のデータ送信を処理するために使用されます。この割り込みは、TXI ISR によりアクティブ化され、自動的に非アクティブ化されます。ISR はユーザー定義のコールバック (p_callback) を呼び出して引数 uart_callback_args_t を渡し、データが最後まで送信されたことを通知します。

SCI 上の UART の ERI 割り込み

ERI 割り込みは、 UART 受信で発生したエラーを処理するために使用されます。この割り込みは、構成パラメータ SCI_UART_CFG_RX_ENABLE によって受信が有効になっている限り、open API でアクティブ化されます。 ISR はユーザー定義のコールバック (p_callback) を呼び出して引数 uart_callback_args_t を渡し、uart_event_t がエラーの原因であることを通知します。

UART HAL モジュールの動作に関する重要な注意事項と制限事項

SCI 上の UART のハードウェアフロー制御

SCI ハードウェアモジュールは、同時に RTS 信号または CTS 信号のいずれか一方のみのハードウェアフロー制御をサポートします。CTS と RTS は、CTSn/RTSn ピンで多重化され、ユースケースに応じて、いずれかのハードウェアフロー制御信号を排他的に使用できるようになっています。UART HAL モジュールは、この仕様を拡張し、RTS 信号用に追加のピンを有効にすることで、CTS 信号と RTS 信号の両方を制御できます。このモードを有効にするには、以下のように SCI の構成で UART を設定します。

- SCI_UART_CFG_EXTERNAL_RTS_OPERATION を有効に設定します。
- ctsrts_en を CTS （true）に設定します。
- ユーザーコールバック関数の名前を p_extpin_ctrl. の [Name of UART callback function for the RTS external pin control] に指定します。

SCI 上の UART HAL モジュールは、処理の最初と最後で、RXI ISR からユーザーコールバック関数を呼び出します。

コールバック関数の引数 「level」は、選択した SCI チャネルの RTS ピンの信号レベルを表します。

***注：** HAL モジュールは、GPIO ピンの初期化の処理またはその制御を行いません。代わりに、ユーザーが、UART の受信を開始する前に GPIO ピンを初期化する必要があります。*

次の図は、RTS 信号として外部 GPIO ピンが使用されている RTS/RTS ハードウェアフロー制御のタイミング図です。

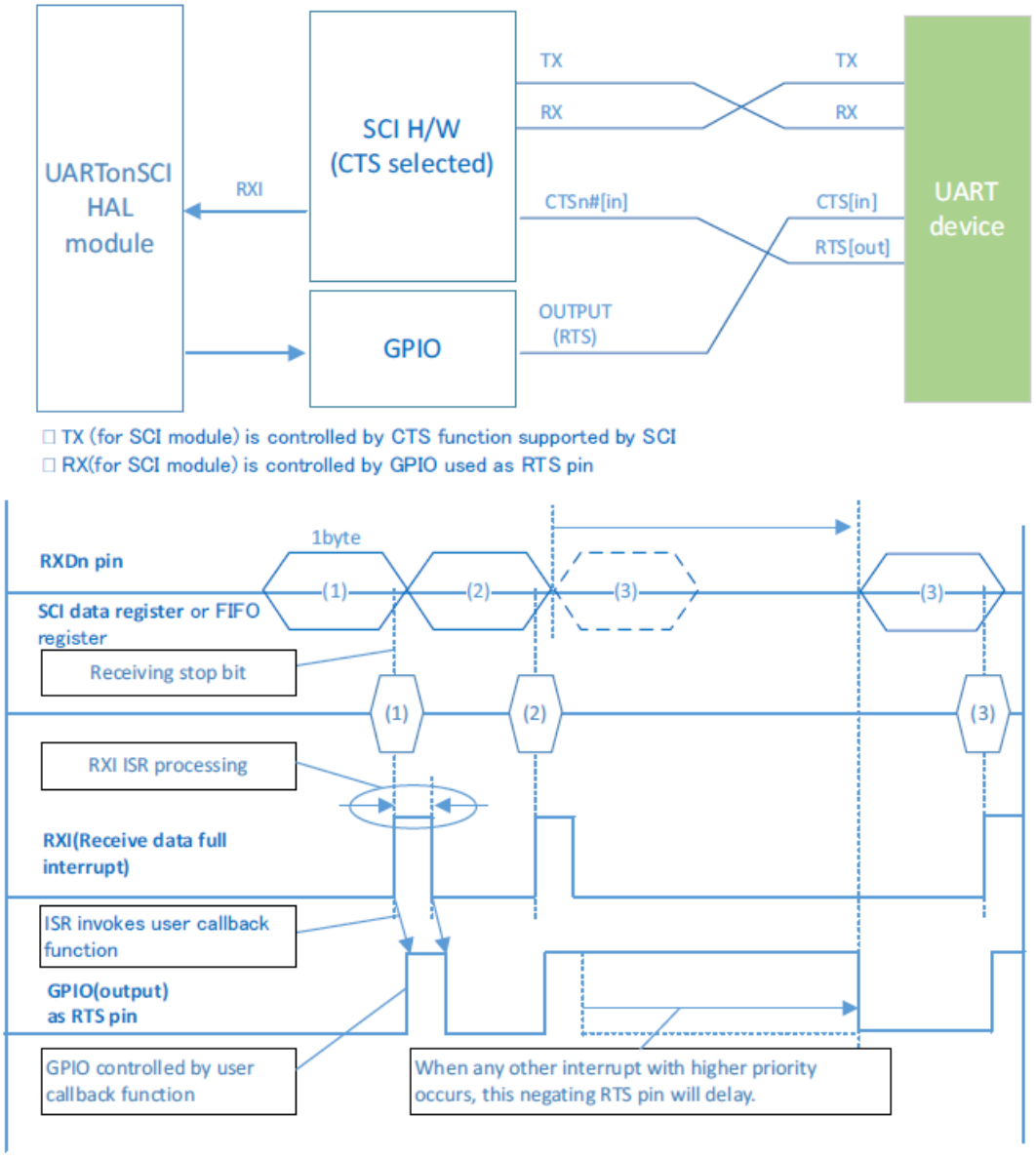


図 437: 外部 GPIO を使用する CTS/RTS ハードウェア制御

注 : *SK-S7G2 ボードの SCI 上の UART モジュールは、PORT8 の pin0 （ピン P800）および J8 を使用して、RS-232C トランシーバ上の RS232C ポートをアクティブ化します。J8 のピン 1 とピン 2 を接続します。ピン P800 を IOPORT ピンとして構成し、そのレベルを目的の動作に合わせて設定します。*

- このモジュールは、割り込みベースの動作をサポートしていますが、ポーリング UART モードはサポートしていません。
- このモジュールは、非バッファリング UART モードをサポートしていません。

- このモジュールは、イベントリンク機能をサポートしていません。
- DTC が使用されている場合、r_sci_uart ドライバーに送信できるブロックのサイズには 64k の制限があります。この制限は、 infoGet API を使用して取得できます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.39.3 アプリケーションへの UART HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに UART HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*

UART ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（UART HAL のデフォルトの名前は g_uart0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

UART ドライバースタックの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_uart0 UART on r_sci_uart	**Threads > HAL/Common Stacks**	Highlight **Threads > HAL/Common Stacks** and select **New Stack > Driver > Connectivity > UART Driver on r_sci_uart**

次の図に示すように、 r_sci_uart の UART HAL モジュールがスレッドに追加されると、コンフィギュレータは必要なローレベルドライバーを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、 1 回のみ追加すれば複数のスタックで使用できます。

赤い部分をマウスでポイントすると、構成を修正するために必要な操作が表示されます。指示に従って、[Properties] ウィンドウで SCI 受信割り込み（RXI）、SCI 送信割り込み（TXI）、SCI 送信終了割り込み（TEI）を有効にして、有効な構成を完成させてください。

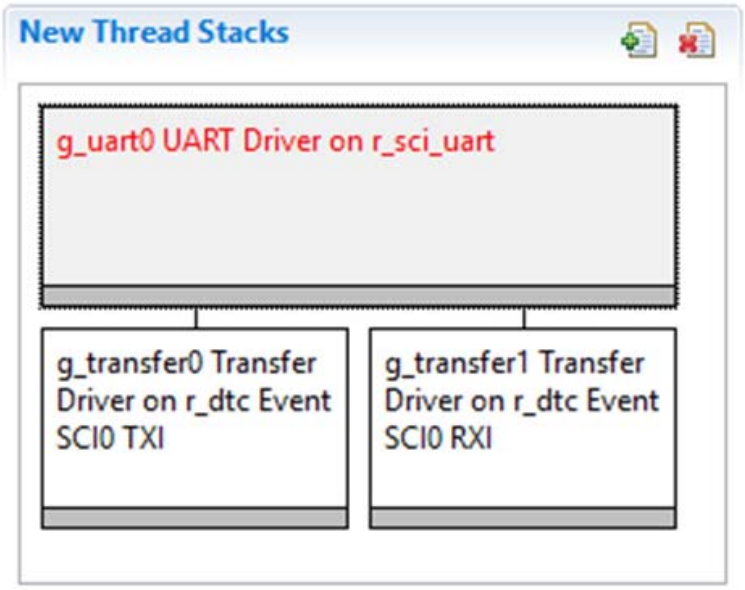


図 438: UART HAL モジュールのスタック

5.2.39.4 UART HAL モジュールの構成

ユーザーは必要な動作に合わせて UART HAL モジュールを構成する必要があります。 [SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、 ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、 SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注: *次に示す構成テーブルの設定に目を通しながら、 ISDE を開き、 UART HAL モジュールを作成して、プロパティ設定を調べることができます。これは正しい判断に役立ち、 SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

r_sci_uart での UART HAL モジュールの構成設定

ISDE Property	Value	説明
External RTS Operation	Enable, Disable Default: Disable	RTS 信号として使用する IOPORT ピンを有効にします。RTS 機能では、この構成パラメータを **[Enable]** に設定し、構成 **[Name of UART callback function for the RTS external pin control]** を指定します。
Reception	Enable, Disable Default: Enable	SCI 上のすべての UART チャネルについて UART の受信を有効または無効にします。この構成パラメータを **[Disable]** に設定すると、UART 受信のためのコード部分がコンパイルされないため、コードサイズが小さくなります。このパラメータは、個々の UARTT チャネルに対しては設定できません。
Transmission	Enable, Disable Default: Enable	SCI 上のすべてのチャネルについて UART 送信を有効または無効にします。この構成を **[Disable]** に設定すると、UART 送信のためのコード部分がコンパイルから除外されるため、コードサイズが小さくなりますが、この設定を **[Disable]** に設定できるのは、UART ポートとして機能する他の SCI チャネルが送信を行わない場合だけです。
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータエラーチェックを有効または無効にします。
Name	g_uart0	SCI 上の UART モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Channel	0-9	SCI チャネル番号。
Baud Rate	9600	ボーレートの選択。
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART データビット。

ISDE Property	Value	説明
Parity	None, Odd, Even Default: None	UART パリティビット。
Stop Bits	1 bit, 2 bits Default: 1 bit	UART ストップビット。
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	SCI チャネル n の CTSn/RTSn ピンに対して CTS または RTS を選択します。SCI ハードウェアは、このピン上で CTS と RTS のいずれかの制御信号をサポートしますが、両方はサポートしません。CTS と RTS の両方を使用するアプリケーションでは、この構成パラメータで **[CTS]** を選択し、構成 **[External RTS Operation]** を有効にして、構成 **[Name of UART callback function for the RTS external pin control]** を指定します。
Name of UART callback function to be defined by user	user_uart_callback	名前は有効な C シンボルである必要があります。
Name of UART callback function for the RTS external pin control to be defined by user	NULL	名前は有効な C シンボルである必要があります。
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	ボーレートクロック発信器ブロックで使用するクロックソースを選択します。
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	選択したチャネル n に対し SCKn ピン上でボーレートクロックを出力するためのオプション設定。
Start bit detection	Falling Edge, Low Level Default: Falling Edge	受信でのスタートビット検出モード。通常、この構成は **[Falling Edge]** に設定します。

ISDE Property	Value	説明
Noise Cancel	Enable, Disable Default: Disable	RXDn ピン上でデジタルノイズキャンセルを有効にします。SCI のデジタルノイズフィルターブロックは、2ステージのフリップフロップ回路で構成されます。詳細については、Renesas Synergy ハードウェアマニュアルのノイズキャンセルのセクションを参照してください。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調有効化の選択。
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信割り込み優先順位の選択。
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	送信終了割り込み優先順位の選択。

ISDE Property	Value	説明
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	エラー割り込み優先順位の選択。

注: *設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

低レベルモジュールではデフォルト以外の設定が望ましい場合もあります。たとえば、異なるノイズキャンセルの設定を選択する場合に役立つことがあります。低レベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: *下位レベルモジュールのプロパティ設定の大半は、かなり直観的であり、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。*

UART HAL モジュールの低レベルモジュールの構成設定

通常、低レベルモジュールでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_dtc イベント SCI0 TXI での転送ドライバーの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	開始モードを設定します
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの設定
Name	g_transfer0	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Bytes	転送サイズの選択

ISDE Property	Value	説明
Destination Address Mode	Fixed	宛先アドレスモードの選択
Source Address Mode	Incremented	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode)	Source	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 TXI	アクティベーションソースの選択
Auto Enable	True, False Default: True	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択

***注:** 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

r_dtc イベント SCI0 RXI での転送ドライバーの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled Default: BSP	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	開始モードを設定します
Linker section to keep DTC vector table	.ssp_dtc_vector_table	リンカセクションの設定
Name	g_transfer1	モジュール名
Mode	Normal	モードの選択
Transfer Size	1 Bytes	転送サイズの選択
Destination Address Mode	Incremented	宛先アドレスモードの選択
Source Address Mode	Fixed	ソースアドレスモードの選択
Repeat Area (Unused in Normal Mode	Destination	リピートエリアの選択
Interrupt Frequency	After all transfers have completed	割り込み頻度の選択
Destination Pointer	NULL	宛先ポインタの選択
Source Pointer	NULL	ソースポインタの選択
Number of Transfers	0	転送回数の選択
Number of Blocks (Valid only in Block Mode)	0	ブロック数の選択
Activation Source (Must enable IRQ)	Event SCI0 RXI	アクティベーションソースの選択
Auto Enable	FALSE	自動有効の選択
Callback (Only valid with Software start)	NULL	コールバックの選択

ISDE Property	Value	説明
ELC Software Event Interrupt Priority	Priority 0(highest), Priority 1-2, Priority 3 (CM4: valid, CM0+: lowest – not valid if using ThreadX), Priority 4-14 (CM4: valid, CM0+: invalid), Priority 15 (CM4: lowest, not valid if using Thread X, CM0: invalid), Disabled Default: Disable	ELC SW イベントの割り込み優先順位

注: *設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*

CTS 機能と RTS 機能で同時に UART を使用するときは、転送ドライバーを使用できません。ローレベルのすべての転送ドライバーを削除してください。削除した後、オプションの転送ドライバーは次の図のようにピンク色で表示され、これはドライバーが推奨ではあってもオプションであることを意味します。

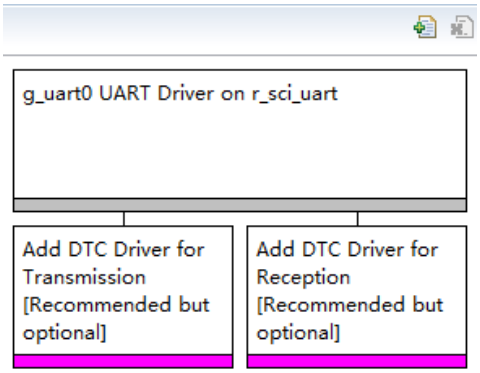


図 439: CTS 機能と RTS 機能での UART スタック

UART HAL モジュールのクロック構成

SCI UART は、PCLKA をクロックソースとして使用するか（S124 の PCLKB）、または選択されているチャネル n の SCKn ピンからの外部クロックを使用します。

UART HAL モジュールのピン構成

SCI UART は、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では UART ピンの選択例を示します。

注: *動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。*

SCI での UART HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select **Peripherals > Connectivity: SCI > SCI0**

SCI での UART HAL モジュールのピン構成設定

Pin Configuration Property	Value	説明
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	ピングループの選択
Operation Mode	Disabled, Custom, Asynchronous UART, Simple SPI, Simple I2C, Synchronous UART, SmartCard (Default: Simple SPI)	SCI での UART の動作モードを選択します
TXD_MOSI	None, P411, P101 (Default: P411)	TXD ピン
RXD_MISO	None, P410, P100 (Default: P410)	RXD ピン
SCK	None, P412, P102 (Default: P412)	SCK ピン
CTS_RTS_SS	None, P413, P103 (Default: None)	CTS ピン
SDA	Disabled	SDA ピン（簡易 I2C を使用している場合）

Pin Configuration Property	Value	説明
SCL	Disabled	SCL ピン（簡易 I2C を使用している場合）

注: *設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。*

5.2.39.5 アプリケーションでの UART HAL モジュールの使用

モジュールを構成してファイルの生成が済むと、UART HAL モジュールはアプリケーションで使用できる状態になります。アプリケーションで UART HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、 UART HAL モジュールを初期化します。
2) baudSet API を使用して、ボーレートを設定します （必要な場合）。
3) 必要に応じて、read および write API とコールバックを使用して、データのリードとライトを行います。
4) close API を使用して、 UART HAL モジュールを閉じます （必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

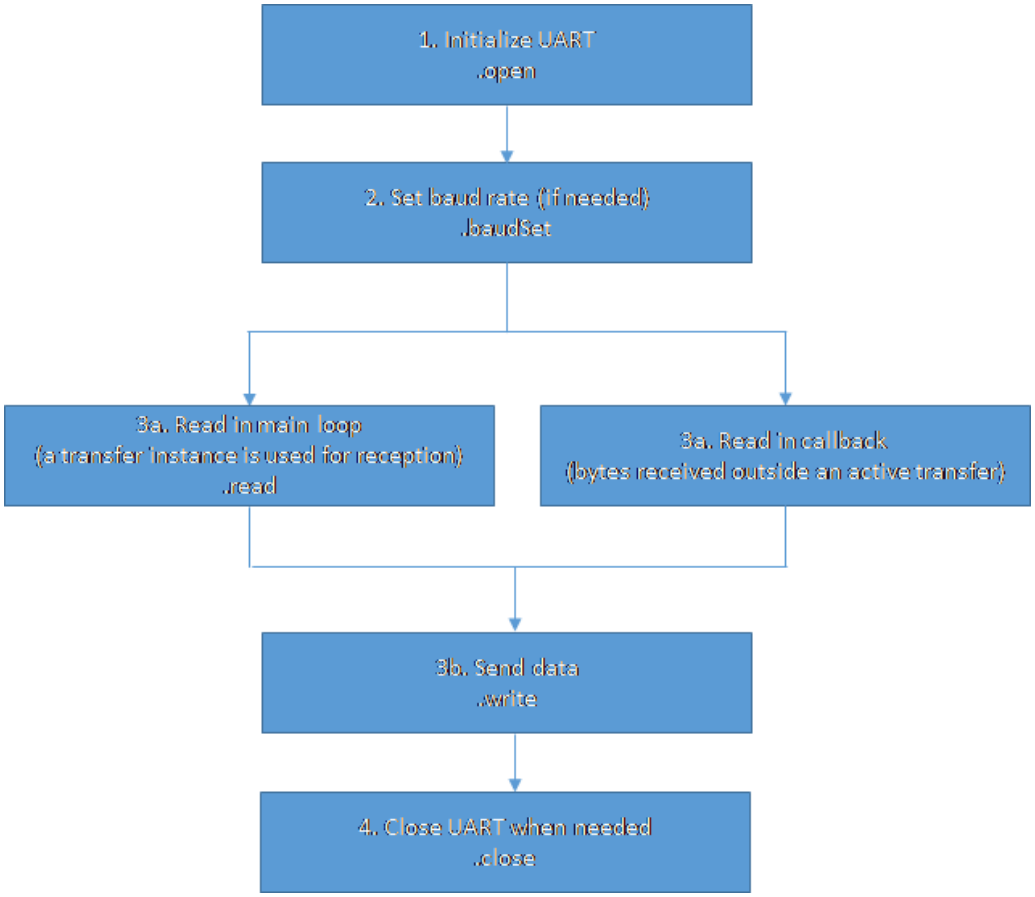


図 440: 一般的な UART HAL モジュールアプリケーションのフロー図

5.2.40 ウォッチドッグドライバー

WDT（ウォッチドッグタイマ）HAL モジュールは、WDT アプリケーションのための高レベル API であり、r_wdt. に実装されています。WDT HAL モジュールは、Synergy MCU 上の WDT を使用します。ユーザー定義のコールバックを作成し、イベント通知に応答できます。

5.2.40.1 WDT HAL モジュールの機能

WDT HAL モジュールには、以下の主要な機能があります。

- WDT が許可されたリフレッシュウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。
  - デバイスのリセット
  - NMI の生成
- ウォッチドッグタイマ（WDT）をサポートします。 WDT は外部クロックを使用します。

- WDT は、 WDT レジスタからレジスタスタートモードで設定できます。
- リセット後の自動ハードウェア構成をサポートします。
- WDT はアプリケーションから開始することができます。

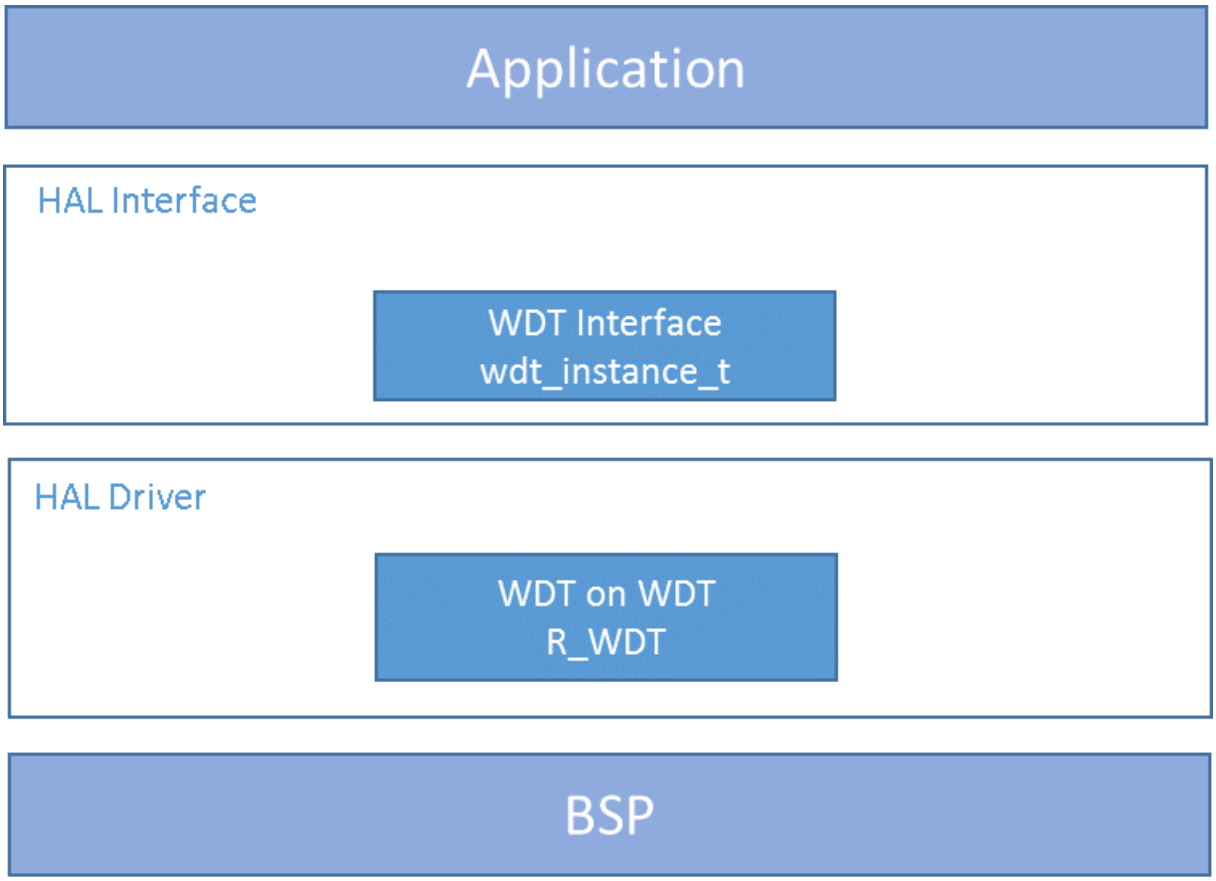


図 441: WDT HAL モジュールの編成、オプション、スタックの実装

5.2.40.2 WDT HAL モジュールの API の概要

WDT HAL モジュールでは、状態のオープン、リフレッシュ、リード、取得のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表はその後にあります。

WDT HAL モジュールの API の要約

Function Name	API の呼び出し例と説明
cfgGet	g_wdt0.p_api->cfgGet(g_wdt0.p_ctrl, g_wdt0.p_cfg); WDT をレジスタスタートモードで初期化します。NMI 出力を使用したオートスタートモードの場合は、NMI コールバックが登録されます。
open	g_wdt0.p_api->open(g_wdt0.p_ctrl, g_wdt0.p_cfg); WDT をレジスタスタートモードで初期化します。NMI 出力を使用したオートスタートモードの場合は、NMI コールバックが登録されます。
refresh	g_wdt0.p_api->refresh(g_wdt0.p_ctrl); ウォッチドッグタイマをリフレッシュします。
statusGet	g_wdt0.p_api->statusGet(g_wdt0.p_ctrl, &status); WDT の状態を読み取ります。
statusClear	g_wdt0.p_api->statusClear(g_wdt0.p_ctrl, clear); WDT の状態フラグをクリアします。
counterGet	g_wdt0.p_api->counterGet(g_wdt0.p_ctrl, &counter); WDT の現在のカウンタ値を読み取ります。
timeoutGet	g_wdt0.p_api->timeoutGet(g_wdt0.p_ctrl, &timeout); ウォッチドッグのタイムアウト値を読み取ります。
versionGet	g_wdt0.p_api->versionGet(&version); version ポインタを使用して API のバージョンを取得します。

注: *関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。*

ステータス戻り値

Name	説明
SSP_SUCCESS	関数が正常に実行されました。
SSP_ERR_ASSERTION	NULL ポインタ。
SSP_ERR_INVALID_ARGUMENT	1 つまたは複数の構成オプションが無効です。

Name	説明
SSP_ERR_INVALID_MODE	WDT をレジスタスタートモードで開こうとしましたが、OFS0 レジスタがオートスタートモードに設定されています。または WDT をオートスタートモードで開こうとしましたが、OSF0 レジスタがレジスタスタートモードに設定されています。
SSP_ERR_ABORTED	このウォッチドッグのクロック分周器が無効です

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

5.2.40.3 WDT HAL モジュールの動作の概要

Synergy MCU には、ウォッチドッグタイマ（WDT）と独立ウォッチドッグタイマ（IWDT）の 2 種類のウォッチドッグがあります。どちらを選択するかは、以下のことを考慮して決定します。

- WDT はアプリケーションから開始することができます。
- WDT は、WDT レジスタからレジスタスタートモードで設定できます。また WDT は、オプション関数選択レジスタ 0(OFS0) に格納されているパラメータを使用して、リセット後に自動的にハードウェアで設定することもできます。
- IWDT には、安全性を向上させる独自のクロックソースがあります。
- IWDT は、オプション関数選択レジスタ 0(OFS0) に格納されているパラメータを使用して、リセット後に自動的にハードウェアで設定することができます。

5.2.40.4 WDT HAL モジュールの動作

WDT HAL モジュールは、WDT インタフェースを構成します。WDT が許可されたリフレッシュウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。

- デバイスのリセット
- NMI の生成

下の図は、 WDT の動作の例を示しています。カウンタの有効なリフレッシュ周期でリフレッシュされた場合、タイマカウントの値はリセットされます。カウントがアンダーフローするか、有効なリフレッシュ周期外にリフレッシュされた場合、 WDT はデバイスをリセットするか、 NMI を生成します。

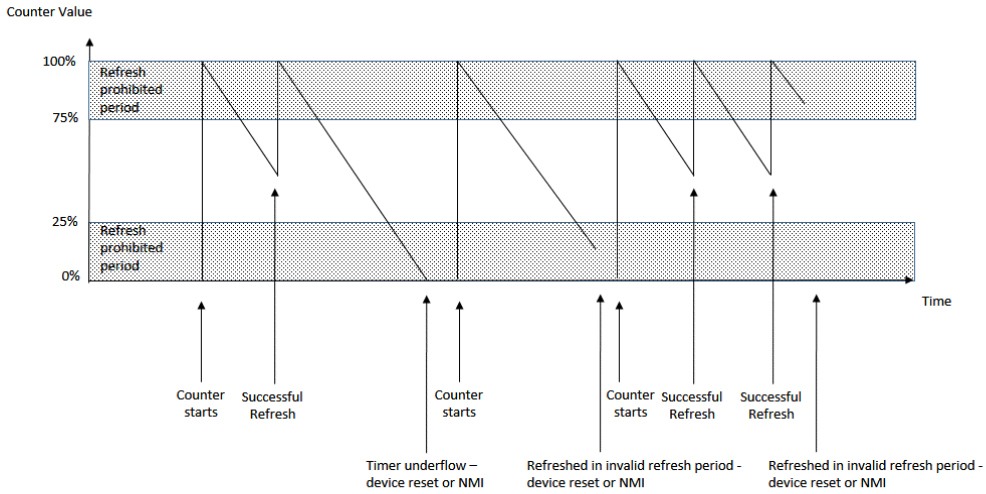


図 442: WDT HAL モジュールの動作

WDT は、WDT レジスタからレジスタスタートモードで設定できます。また、WDT は、次の表で示すように、オプション関数選択レジスタ 0（OFS0）に格納されているパラメータを使用して、リセット後にハードウェアで自動的に構成することもできます。

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後のペリフェラルの動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。

次の表では、OFS レジスタで構成できる WDT のパラメータを示します。

WDT のパラメータ

Control	説明
WDT Start Mode Select	有効にすると、リセット後に WDT を自動的に開始します。
WDT Timeout Period	WDT のタイムアウトを指定します（クロックサイクル数）。128、512、1024、2048 サイクル。
WDT-Clock Frequency Division Ratio (The WDT is clocked from PCLKB)	PCLKB / 1 PCLKB / 64 PCLKB / 128 PCLKB / 512 PCLKB / 2048 PCLKB / 8192

Control	説明
WDT Window End Position	25%、50%、75%、100%（ウィンドウ終了位置の設定はありません）
WDT Window Start Position	25%、50%、75%、100%（ウィンドウ開始位置の設定はありません）
WDT Reset Interrupt Request	WDT は、割り込み信号またはリセット信号を生成できます。
WDT Stop Control	WDT は、カウントを続けるか、ローパワーモードでカウントを停止することができます。

注: OFS0 レジスタの内容の詳細については、 Synergy MCU ハードウェアマニュアルを参照してください。

OFS レジスタの値は、 Synergy 構成エディタの [BSP] タブのプロパティで設定します。

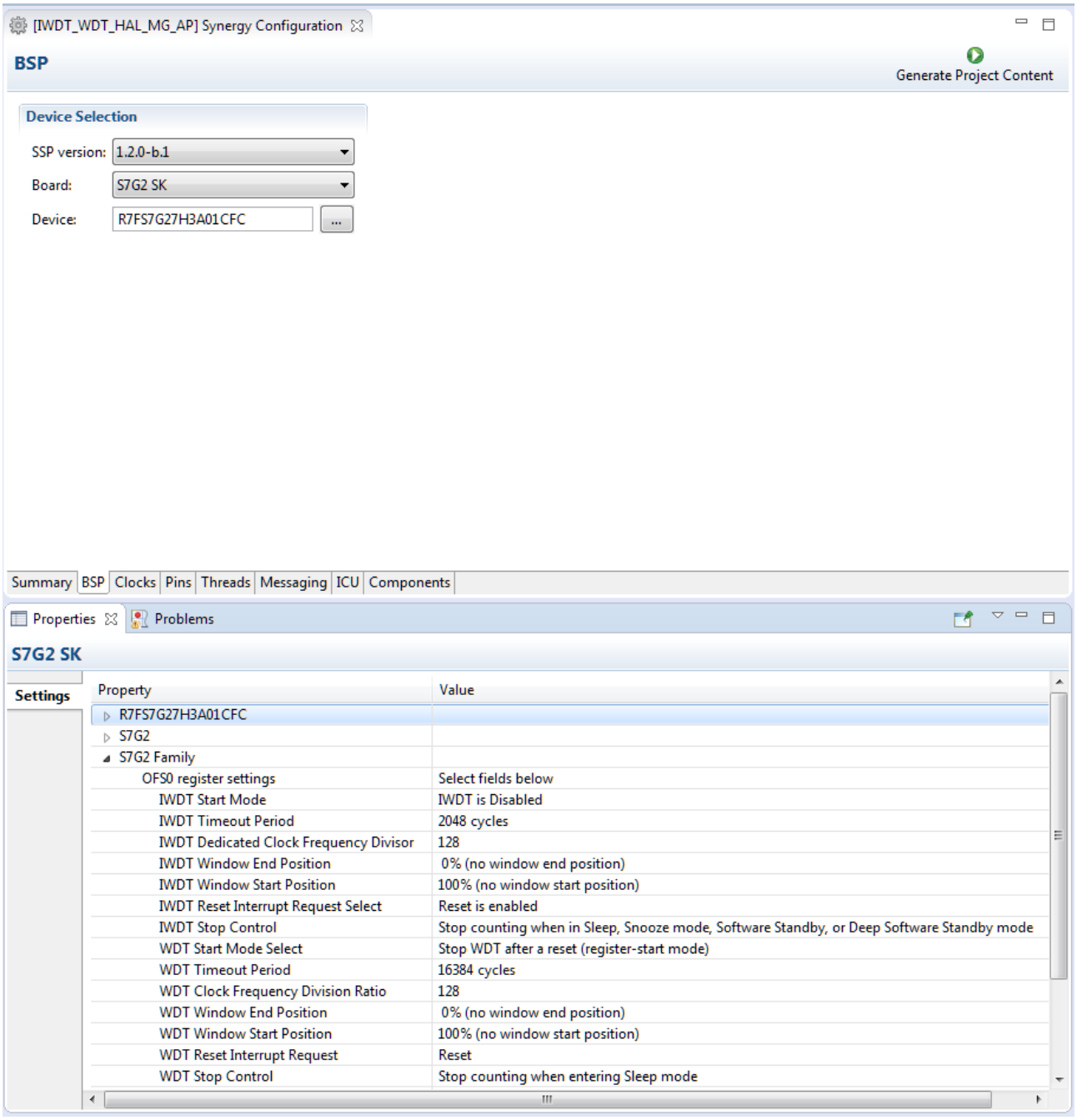


図 443: OFS レジスタの値の設定

5.2.40.5 WDT HAL モジュールの動作に関する重要な注意事項と制限事項

5.2.40.6 WDT HAL モジュールの期間の計算

WDT は PCLKB から動作します。WDT で最大のパラメータがあり、PCLKB が 60 MHz であると仮定した場合、最後のリフレッシュからデバイスのリセットまたは NMI の生成までの時間は、下記のように 2.2 秒をわずかに上回ります。

PLCKB = 60 MHz

クロック分割比 = PCLKB/8192

タイムアウト周期 = 16384 サイクル

WDT クロック周波数 = 60 MHz / 8192 = 7.324 kHz

サイクル時間 = 1 / 7.324 kHz = 136.53 マイクロ秒

タイムアウト = 136.53 マイクロ秒 x 16384 サイクル = 2.23 秒

5.2.40.7 WDT HAL モジュールを使用した DMAC/DTC のトリガ

WDT カウンタがアンダーフローした場合、または有効なリフレッシュ周期外にリフレッシュが試みられた場合に、 DMAC または DTC を使用してデータの転送をトリガするには、 NMI を生成するように WDT を設定し、 activation_source を ELC_EVENT_WDT_UNDERFLOW に設定して DMAC/DTC 転送を構成します。詳細については、対応するユーザーズガイド （DMAC、 DTC）を参照してください。

5.2.40.8 WDT HAL モジュールでのイベントリンクコントローライベントのトリガ

WDT は、イベントリンクコントローラ（ELC）を使用して別のペリフェラルの開始をトリガできます。使用可能なペリフェラルの完全なリストについては、 ELC のユーザーガイドを参照してください。

5.2.40.9 WDT HAL モジュールの制限事項

- JLink デバッガーを使用する場合、 WDT カウンタはカウントしないため、デバイスのリセットや NMI の生成は行われません。
- 実行できるアクティブなタスクがない場合、 ThreadX は MCU をスリープモードにします。 WDT がローパワーモードでカウンタを停止するように構成されている場合、ThreadX RTOS で使用されているときは、アプリケーションでタイマを再度開始する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

5.2.40.10 アプリケーションへの WDT HAL モジュールの組み込み

このセクションでは、 SSP コンフィギュレータを使用してアプリケーションに WDT HAL モジュールを組み込む方法について説明します。

***注：** このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない*

場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

WDT HAL モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します（WDT HAL モジュールのデフォルトの名前は、次の表で示されているように g_wdt0 です。この名前は、対応する [Properties] ウィンドウで変更できます）。

ウォッチドッグタイマの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_wdt0 Watchdog Driver on r_wdt	Threads	New Stack>Driver>Monitoring> Watchdog Driver on r_wdt

次の図に示すように r_wdt の WDT HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要な下位レベルモジュールを自動的に追加します。追加の構成情報を必要とするドライバーは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

図 444: WDT HAL モジュールのタイマスタック

5.2.40.11 WDT HAL モジュールの構成

ユーザーは必要な動作に合わせて WDT HAL モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、低レベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Property] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティの表には含まれませんが、割り込み優先順位を構成するときに ISDE で簡単に確認できます。

注: *次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。*

WDT HAL モジュールの構成

ISDE Property	Value	説明
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	パラメータチェックコードを含みます。
Name	g_wdt0	モジュール名。
Start Mode	Register, Auto (Default: Register)	スタートモードをレジスタスタートまたはオートスタートに設定します。
Start Watchdog After Configuration	True, False (Default: True)	WDTが初期化の際に開始するかどうかを制御します。
Timeout	1024 cycles, 4096 cycles, 8192 cycles, 16384 cycles (Default: 16384 cycles)	WDT タイムアウト期間。
Clock Division Ratio	PCLK/4, PCLK/64, PCLK/128, PCLK/512, PCLK/2048, PCLK/8192 (Default: PCLK/8192)	WDT クロック分周器。
Window Start Position	100% (Window Position Not Specified), 75%, 50%, 25% (Default: 100%)	許可されたリフレッシュ周期の開始位置。
Window End Position	0% (Window Position Not Specified, 25%, 50%, 75% (Default: 0%)	許可されたリフレッシュ周期の終了位置。
Reset Control	Reset Output, NMI Generated (Default: Reset Output)	WDT が MCU をリセットするか、NMI を生成するかを選択します。
Stop Control	WDT Count Enabled in Low Power Mode, WDT Count Disabled in Low Power Mode (Default: Disabled)	WDTが低電力モードでカウントを停止するかどうかを選択します。

ISDE Property	Value	説明
NMI Callback	NULL	コールバック。ユーザーコールバック関数を open で登録できます。このコールバック関数が指定されている場合、IRQnがトリガされるたびに割り込みサービスルーチン(ISR)から呼び出されます。コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

注: 上記の設定例とデフォルトは、Synergy S7G2 を使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

5.2.40.12 オプション機能選択レジスタ 0（OFS0）の構成

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後のペリフェラルの動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。このドキュメントで前述した動作の概要に関するセクションの説明を参照してください。

5.2.40.13 WDT HAL モジュールの割り込みの構成

WDT モジュールの他のオプション構成と同様に、WDT 割り込みを構成します。アンダーフローまたは無効なリフレッシュで NMI 割り込みを生成するよう WDT が構成されている場合、割り込みは BSP で有効になっている必要があります。

割り込みを有効にするには、[CWDT] > [CWDT NMIUNDF N n] で優先順位を設定します。これにより、ssp_cfg/bsp/bsp_irq_cfg.h の BSP_IRQ_CFG_WDT_UNDERFLOW が選択した優先順位レベルに設定されます。

CWDT NMIUNDF N 割り込みが BSP で有効になっている場合、対応する ISR が定義されます。ISR は、ユーザーコールバック関数が open API で登録されている場合、それを呼び出します。

5.2.40.14 WDT HAL モジュールのクロック構成

初期設定では、WDT クロックは PCLKB 周波数に基づいて動作します。ISDE のクロックコンフィギュレータを使用して、または実行時に CGC インタフェースを使用して、PCLKB の周波数を設定できます。PCLKB が 60 MHz で動作している状態での最大タイムアウト周期は約 2.2 秒です。

5.2.40.15 WDT HAL モジュールのピン構成

WDT は、動作のためにピンを必要としません。

5.2.40.16 アプリケーションでの WDT HAL モジュールの使用

アプリケーションで WDT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、レジスタスタートモードまたは自動スタートモードで WDT HAL モジュールを初期化します。
2) cfgGet API を使用して、レジスタスタートモードまたは自動スタートモードの WDT HAL モジュールの構成を読み取ります。
3) refresh API を使用して、ウォッチドッグタイマをリフレッシュします。
4) statusGet API を使用して、 WDT の状態を読み取ります。
5) statusClear API を使用して、 WDT HAL モジュールの状態フラグとエラーフラグをクリアします。
6) counterGet API を使用して、 WDT の現在のカウント値を読み取ります。
7) timeoutGet API を使用して、ウォッチドッグのタイムアウト値を読み取ります。

これらの手順を、次の図の通常の動作フロー図に示します。

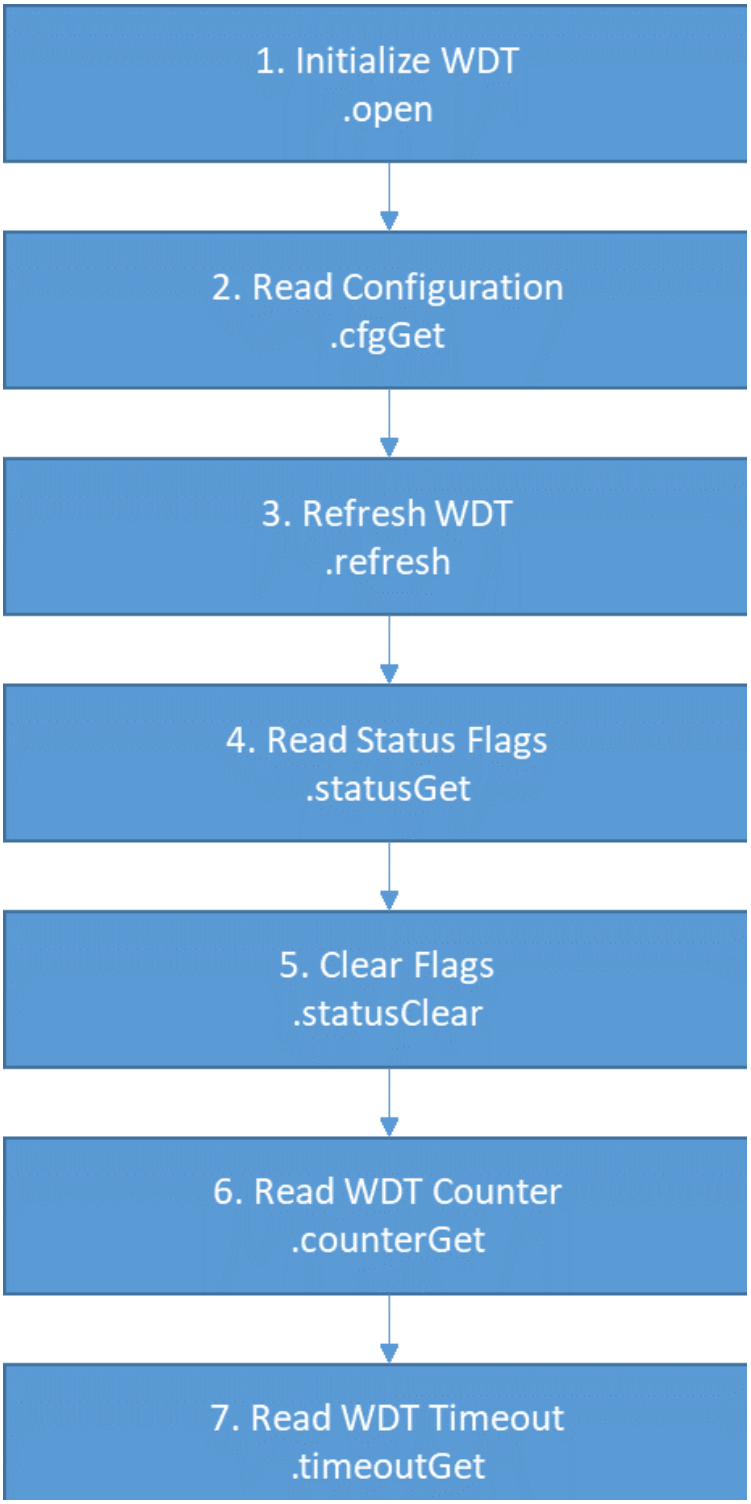


図 445:　一般的な WDT HAL モジュールアプリケーションのフロー図

Chapter 6 API Reference: Common

You can find error codes and version data structures that are common to the entire Synergy Software Package here.

6.1 Common Error Codes

All SSP code at every layer shares these common error codes.

6.1.1 User-defined enumerations

- ssp_err_t
- ssp_command_t

6.1.2 User-defined macros

- #define SSP_PARAMETER_NOT_USED

  Initial value:`(void) ((p))`

  This macro is used to suppress compiler messages about a parameter not being used in a function. The nice thing about using this implementation is that it does not take any extra RAM or ROM.
- #define SSP_CPP_HEADER

  Initial value:

  Determine if a C++ compiler is being used. If so, ensure that standard C is used to process the API information.
- #define SSP_CPP_FOOTER

  Initial value:
- #define SSP_HEADER

  Initial value:`SSP_CPP_HEADER`

  SSP Header and Footer definitions
- #define SSP_FOOTER

  Initial value:`SSP_CPP_FOOTER`

6.1.3 API Data

6.1.3.1 ssp_err_t

`ssp_err_t`

Detailed description

Common error codes

Enumerated values

Name	Description
SSP_SUCCESS	
SSP_ERR_ASSERTION	A critical assertion has failed.
SSP_ERR_INVALID_POINTER	Pointer points to invalid memory location.
SSP_ERR_INVALID_ARGUMENT	Invalid input parameter.
SSP_ERR_INVALID_CHANNEL	Selected channel does not exist.
SSP_ERR_INVALID_MODE	Unsupported or incorrect mode.
SSP_ERR_UNSUPPORTED	Selected mode not supported by this API.
SSP_ERR_NOT_OPEN	Requested channel is not configured or API not open.
SSP_ERR_IN_USE	Channel/peripheral is running/busy.
SSP_ERR_OUT_OF_MEMORY	Allocate more memory in the driver's cfg.h.
SSP_ERR_HW_LOCKED	Hardware is locked.
SSP_ERR_IRQ_BSP_DISABLED	IRQ not enabled in BSP.
SSP_ERR_OVERFLOW	Hardware overflow.
SSP_ERR_UNDERFLOW	Hardware underflow.
SSP_ERR_ALREADY_OPEN	Requested channel is already open in a different configuration.
SSP_ERR_APPROXIMATION	Could not set value to exact result.
SSP_ERR_CLAMPED	Value had to be limited for some reason.
SSP_ERR_INVALID_RATE	Selected rate could not be met.
SSP_ERR_ABORTED	An operation was aborted.
SSP_ERR_NOT_ENABLED	Requested operation is not enabled.
SSP_ERR_TIMEOUT	Timeout error.

Name	Description
SSP_ERR_INVALID_BLOCKS	Invalid number of blocks supplied.
SSP_ERR_INVALID_ADDRESS	Invalid address supplied.
SSP_ERR_INVALID_SIZE	Invalid size/length supplied for operation.
SSP_ERR_WRITE_FAILED	Write operation failed.
SSP_ERR_ERASE_FAILED	Erase operation failed.
SSP_ERR_INVALID_CALL	Invalid function call is made.
SSP_ERR_INVALID_HW_CONDITION	Detected hardware is in invalid condition.
SSP_ERR_INVALID_FACTORY_FLASH	Factory flash is not available on this MCU.
SSP_ERR_INVALID_FMI_DATA	Linked FMI data table is not valid.
SSP_ERR_INTERNAL	Internal error. Start of RTOS only error codes
SSP_ERR_WAIT_ABORTED	Wait.
SSP_ERR_FRAMING	Framing error occurs. Start of UART specific
SSP_ERR_BREAK_DETECT	Break signal detects.
SSP_ERR_PARITY	Parity error occurs.
SSP_ERR_RXBUF_OVERFLOW	Receive queue overflow.
SSP_ERR_QUEUE_UNAVAILABLE	Can't open s/w queue.
SSP_ERR_INSUFFICIENT_SPACE	Not enough space in transmission circular buffer.
SSP_ERR_INSUFFICIENT_DATA	Not enough data in receive circular buffer.
SSP_ERR_TRANSFER_ABORTED	The data transfer was aborted. Start of SPI specific
SSP_ERR_MODE_FAULT	Mode fault error.
SSP_ERR_READ_OVERFLOW	Read overflow.
SSP_ERR_SPI_PARITY	Parity error.
SSP_ERR_OVERRUN	Overrun error.
SSP_ERR_CLOCK_INACTIVE	Inactive clock specified as system clock. Start of CGC Specific

Name	Description
SSP_ERR_CLOCK_ACTIVE	Active clock source cannot be modified without stopping first.
SSP_ERR_STABILIZED	Clock has stabilized after its been turned on/off.
SSP_ERR_NOT_STABILIZED	Clock has not stabilized after its been turned on/off.
SSP_ERR_MAIN_OSC_INACTIVE	PLL initialization attempted when main osc is turned off.
SSP_ERR_OSC_STOP_DET_ENABLED	Illegal attempt to stop LOCO when Oscillation stop is enabled.
SSP_ERR_OSC_STOP_DETECTED	The Oscillation stop detection status flag is set.
SSP_ERR_OSC_STOP_CLOCK_ACTIVE	Attempt to clear Oscillation Stop Detect Status with PLL/MAIN_OSC active.
SSP_ERR_CLKOUT_EXCEEDED	Output on target output clock pin exceeds maximum supported limit.
SSP_ERR_USB_MODULE_ENABLED	USB clock configure request with USB Module enabled.
SSP_ERR_HARDWARE_TIMEOUT	A register read or write timed out.
SSP_ERR_PE_FAILURE	Unable to enter Programming mode. Start of FLASH Specific
SSP_ERR_CMD_LOCKED	Peripheral in command locked state.
SSP_ERR_FCLK	FCLK must be >= 4 MHz.
SSP_ERR_INVALID_CAC_REF_CLOCK	Measured clock rate < reference clock rate. Start of CAC Specific
SSP_ERR_CLOCK_GENERATION	Clock cannot be specified as system clock. Start of GLCD Specific
SSP_ERR_INVALID_TIMING_SETTING	Invalid timing parameter.
SSP_ERR_INVALID_LAYER_SETTING	Invalid layer parameter.
SSP_ERR_INVALID_ALIGNMENT	Invalid memory alignment found.
SSP_ERR_INVALID_GAMMA_SETTING	Invalid gamma correction parameter.
SSP_ERR_INVALID_LAYER_FORMAT	Invalid color format in layer.
SSP_ERR_INVALID_UPDATE_TIMING	Invalid timing for register update.

Name	Description
SSP_ERR_INVALID_CLUT_ACCESS	Invalid access to CLUT entry.
SSP_ERR_INVALID_FADE_SETTING	Invalid fade-in/fade-out setting.
SSP_ERR_JPEG_ERR	JPEG error. Start of JPEG Specific
SSP_ERR_JPEG_SOI_NOT_DETECTED	SOI not detected until EOI detected.
SSP_ERR_JPEG_SOF1_TO_SOFF_DETECTED	SOF1 to SOFF detected.
SSP_ERR_JPEG_UNSUPPORTED_PIXEL_FORMAT	Unprovided pixel format detected.
SSP_ERR_JPEG_SOF_ACCURACY_ERROR	SOF accuracy error: other than 8 detected.
SSP_ERR_JPEG_DQT_ACCURACY_ERROR	DQT accuracy error: other than 0 detected.
SSP_ERR_JPEG_COMPONENT_ERROR1	Component error1: the number of SOF0 header components detected is other than 1,3,or 4.
SSP_ERR_JPEG_COMPONENT_ERROR2	Component error2: the number of components differs between SOF0 header and SOS.
SSP_ERR_JPEG_SOF0_DQT_DHT_NOT_DETECTED	SOF0, DQT, and DHT not detected when SOS detected.
SSP_ERR_JPEG_SOS_NOT_DETECTED	SOS not detected: SOS not detected until EOI detected.
SSP_ERR_JPEG_EOI_NOT_DETECTED	EOI not detected (default)
SSP_ERR_JPEG_RESTART_INTERVAL_DATA_NUMBER_ERROR	Restart interval data number error detected.
SSP_ERR_JPEG_IMAGE_SIZE_ERROR	Image size error detected.
SSP_ERR_JPEG_LAST_MCU_DATA_NUMBER_ERROR	Last MCU data number error detected.
SSP_ERR_JPEG_BLOCK_DATA_NUMBER_ERROR	Block data number error detected.
SSP_ERR_JPEG_BUFFERSIZE_NOT_ENOUGH	User provided buffer size not enough.
SSP_ERR_JPEG_UNSUPPORTED_IMAGE_SIZE	JPEG Image size is not aligned with MCU.
SSP_ERR_CALIBRATE_FAILED	Calibration failed. Start of touch panel framework specific
SSP_ERR_IP_HARDWARE_NOT_PRESENT	Requested IP does not exist on this device. Start of FMI specific
SSP_ERR_IP_UNIT_NOT_PRESENT	Requested unit does not exist on this device.

Name	Description
SSP_ERR_IP_CHANNEL_NOT_PRESENT	Requested channel does not exist on this device.
SSP_ERR_NO_MORE_BUFFER	No more buffer found in the memory block pool. Start of Message framework specific
SSP_ERR_ILLEGAL_BUFFER_ADDRESS	Buffer address is out of block memory pool.
SSP_ERR_INVALID_WORKBUFFER_SIZE	Work buffer size is invalid.
SSP_ERR_INVALID_MSG_BUFFER_SIZE	Message buffer size is invalid.
SSP_ERR_TOO_MANY_BUFFERS	Number of buffer is too many.
SSP_ERR_NO_SUBSCRIBER_FOUND	No message subscriber found.
SSP_ERR_MESSAGE_QUEUE_EMPTY	No message found in the message queue.
SSP_ERR_MESSAGE_QUEUE_FULL	No room for new message in the message queue.
SSP_ERR_ILLEGAL_SUBSCRIBER_LISTS	Message subscriber lists is illegal.
SSP_ERR_BUFFER_RELEASED	Buffer has been released.
SSP_ERR_D2D_ERROR_INIT	Dave/2d has an error in the initialization. Start of 2DG Driver specific
SSP_ERR_D2D_ERROR_DEINIT	Dave/2d has an error in the initialization.
SSP_ERR_D2D_ERROR_RENDERING	Dave/2d has an error in the rendering.
SSP_ERR_D2D_ERROR_SIZE	Dave/2d has an error in the rendering.
SSP_ERR_QUEUE_FULL	Queue is full, cannot queue another data. Start of BYTEQ library specific
SSP_ERR_QUEUE_EMPTY	Queue is empty, no data to dequeue.
SSP_ERR_CTSU_SC_OVERFLOW	Sensor count overflowed when performing CTSU scan. User must clear the CTSUSCOVF bit manually.
SSP_ERR_CTSU_RC_OVERFLOW	Reference count overflowed when performing CTSU scan. User must clear the CTSURCOVF bit manually.
SSP_ERR_CTSU_ICOMP	Abnormal TSCAP voltage. User must clear the CTSUICOMP bit manually.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Auto tuning algorithm failed.

Name	Description
SSP_ERR_CTSU_SAFETY_CHECK_FAILED	Safety check failed
SSP_ERR_CARD_INIT_FAILED	SD card or eMMC device failed to initialize. Start of SDMMC specific
SSP_ERR_CARD_NOT_INSERTED	SD card not installed.
SSP_ERR_SDHI_FAILED	SD peripheral failed to respond properly.
SSP_ERR_READ_FAILED	Data read failed.
SSP_ERR_CARD_NOT_READY	SD card was removed.
SSP_ERR_CARD_WRITE_PROTECTED	Media is write protected.
SSP_ERR_TRANSFER_BUSY	Transfer in progress.
SSP_ERR_MEDIA_FORMAT_FAILED	Media format failed. Start of FX_IO specific
SSP_ERR_MEDIA_OPEN_FAILED	Media open failed.
SSP_ERR_CAN_DATA_UNAVAILABLE	No data available. Start of CAN specific
SSP_ERR_CAN_MODE_SWITCH_FAILED	Switching operation modes failed.
SSP_ERR_CAN_INIT_FAILED	Hardware initialization failed.
SSP_ERR_CAN_TRANSMIT_NOT_READY	Transmit in progress.
SSP_ERR_CAN_RECEIVE_MAILBOX	Mailbox is setup as a receive mailbox.
SSP_ERR_CAN_TRANSMIT_MAILBOX	Mailbox is setup as a transmit mailbox.
SSP_ERR_CAN_MESSAGE_LOST	Receive message has been overwritten or overrun.
SSP_ERR_WIFI_CONFIG_FAILED	WiFi module Configuration failed. Start of Crypto specific (0x10000) Refer to sf_cryoto_err.h for Crypto error code. Start of SF_WIFI Specific
SSP_ERR_WIFI_INIT_FAILED	WiFi module initialization failed.
SSP_ERR_WIFI_TRANSMIT_FAILED	Transmission failed.
SSP_ERR_WIFI_INVALID_MODE	API called when provisioned in client mode.
SSP_ERR_WIFI_FAILED	WiFi Failed.

Name	Description
SSP_ERR_CELLULAR_CONFIG_FAILED	Cellular module Configuration failed. Start of SF_CELLULAR Specific
SSP_ERR_CELLULAR_INIT_FAILED	Cellular module initialization failed.
SSP_ERR_CELLULAR_TRANSMIT_FAILED	Transmission failed.
SSP_ERR_CELLULAR_FW_UPTODATE	Firmware is uptodate.
SSP_ERR_CELLULAR_FW_UPGRADE_FAILED	Firmware upgrade failed.
SSP_ERR_CELLULAR_FAILED	Cellular Failed.
SSP_ERR_CELLULAR_INVALID_STATE	API Called in invalid state.
SSP_ERR_BLE_FAILED	BLE operation failed. Start of SF_BLE specific
SSP_ERR_BLE_INIT_FAILED	BLE device initialization failed.
SSP_ERR_BLE_CONFIG_FAILED	BLE device configuration failed.
SSP_ERR_BLE_PRF_ALREADY_ENABLED	BLE device Profile already enabled.
SSP_ERR_BLE_PRF_NOT_ENABLED	BLE device not enabled.
SSP_ERR_CRYPTO_COMMON_NOT_OPENED	Crypto Framework Common is not opened. Start of SF_CRYPTO specific
SSP_ERR_CRYPTO_HAL_ERROR	Cryoto HAL module returned an error.
SSP_ERR_CRYPTO_KEY_BUF_NOT_ENOUGH	Key buffer size is not enough to generate a key.

6.1.3.2 ssp_command_t

`ssp_command_t`

Detailed description

ioctl commands.

Enumerated values

Name	Description
SSP_COMMAND_GET_SECTOR_COUNT	Get media sector count.
SSP_COMMAND_GET_SECTOR_SIZE	Get sector size.

Name	Description
SSP_COMMAND_GET_BLOCK_SIZE	Get erase block size.
SSP_COMMAND_CTRL_ERASE_SECTOR	Erase sectors.
SSP_COMMAND_GET_WRITE_PROTECTED	Get Write Protection status.
SSP_COMMAND_SET_BLOCK_SIZE	Set block size.

6.2 Version control

6.2.1 ssp_version_t

6.2.1.1 Detailed description

Common version structure

6.2.1.2 Variables

version_id
code_version_minor
code_version_major
api_version_minor
api_version_major

6.2.2 version_id

```
uint32_t ::version_id
```

6.2.2.1 Detailed description

Version id

6.2.3 code_version_minor

```
uint8_t ::code_version_minor
```

6.2.3.1 Brief description

Code minor version.

6.2.4 code_version_major

```
uint8_t ::code_version_major
```

6.2.4.1 Brief description

Code major version.

6.2.5 api_version_minor

```
uint8_t ::api_version_minor
```

6.2.5.1 Brief description

API minor version.

6.2.6 api_version_major

```
uint8_t ::api_version_major
```

6.2.6.1 Brief description

API major version.

6.3 Pack Version Control

6.3.1 ssp_pack_version_t

6.3.1.1 Detailed description

SSP Pack version structure

6.3.1.2 Variables

version_id

build

patch

minor

major

6.3.2 version_id

```
uint32_t ::version_id
```

6.3.2.1 Detailed description

Version id

6.3.3 build

```
uint8_t ::build
```

6.3.3.1 Brief description

Build version of SSP Pack.

6.3.4 patch

```
uint8_t ::patch
```

6.3.4.1 Brief description

Patch version of SSP Pack.

6.3.5 minor

```
uint8_t ::minor
```

6.3.5.1 Brief description

Minor version of SSP Pack.

6.3.6 major

```
uint8_t ::major
```

6.3.6.1 Brief description

Major version of SSP Pack.

Chapter 7 API Reference: Framework Interfaces

7.1 API Reference: Framework Interfaces

The Framework Interface provides common APIs for functional Framework Layer applications. The Framework Interfaces can be implemented by one or more Framework Layer drivers.

- ADC Periodic Framework Interface
- Audio Framework Interface
- Audio Playback Framework Interface
- Audio Recording Framework Interface
- SF BLE Framework Interface
- SF BLE On-Board Profile Framework Interface
- SF BLE Alert Notification Profile Framework Interface
- SF BLE Battery Service Profile Framework Interface
- SF BLE Blood Pressure Profile Framework Interface
- SF BLE Current Time Service Profile Framework Interface
- SF BLE Find Me Profile Framework Interface
- SF BLE HID Over GATT Profile Framework Interface
- SF BLE Heart Rate Profile Framework Interface
- SF BLE Health Thermometer Profile Framework Interface
- SF BLE Immediate Alert Profile Framework Interface
- SF BLE Next DST Change Service Profile Framework Interface
- SF BLE Phone Alert Status Profile Framework Interface
- SF BLE Proximity Profile Framework Interface
- SF BLE Reference Time Update Service Profile Framework Interface
- SF BLE Scan Parameters Service Profile Framework Interface
- SF BLE Time Information Profile Framework Interface
- Block Media Framework Interface
- SF CELLULAR Framework Interface
- SF CELLULAR NSAL Framework Interface

- SF Socket CELLULAR Framework Interface
- Communications Framework Interface
- Console Framework Interface
- SSP Crypto Framework Common Module Interface
- SSP Crypto HASH Framework Interface
- SSP Crypto Key Framework Interface
- SSP Crypto TRNG Framework Interface
- GUIX Interface
- External IRQ Framework Interface
- I2C Framework
- JPEG Decode Framework Interface
- essaging Framework Interface
- Power Profiles Framework Interface
- SF Socket WIFI Framework Interface
- SPI Framework Interface
- Thread Monitor Framework Interface
- CTSU Framework Interface
- CTSU Button Framework Interface
- CTSU Slider Framework Interface
- Touch Panel Framework Interface
- SF WIFI Framework Interface
- SF WIFI NSAL Interface
- SF WIFI On-Chip Stack Interface
- SF WIFI NSAL on NetX
- SF_CELLULAR_COMMON

7.2 ADC Periodic Framework Interface

RTOS-integrated ADC Periodic Framework Interface.

7.2.1 Summary

This is a ThreadX aware generic Periodic ADC sampling framework intended to be used to sample the ADC at periodic intervals, buffer the specified number of samples and then notify the application. The driver will use hardware triggers to allow for time-synchronous sampling. After initial configuration and the scan process is started, the framework uses a hardware timer to trigger an ADC scan in one-shot mode. Each scan can consist of one or more channels. When each scan

is completed the ADC interrupt is intercepted by the DTC which moves the result of the scan into the user buffer. Each scan is defined as a sampling iteration and the number of samples generated for each scan will be equal to the number of channels if the channels are sequential eg: channels 1, 2, 3, 4. If the channels are not in sequence, eg: channels 1, 3, 4, 5, then the samples generated by each scan will also include data from the unused channels in between. Thus the second example here will result in 5 samples being stored to the user buffer each time even though only 4 channels have been configured for usage. The user specifies the total number of sample iterations that need to occur before being notified. When the specified number of sampling iterations have occurred and the data for each iteration has been stored into the user buffer, the user is notified via the callback function with an index for the valid data in the buffer and an event indicating that sampling for the specified number of iterations is complete. Unless the user stops the scan process using the stop API call, the scan will continue to be triggered by the timer and data will be written into the user buffer which is treated by the framework as a circular buffer. For this reason, the buffer length must be at least twice the total number of samples that will be generate after all the iterations are completed. In the second example, where there are 5 samples generated for each iteration, if the sample count is set to 3, this will result in 15 samples being available in the buffer before the callback is called. Thus in this example, the buffer length must be set to 30 or larger. The name and length of the buffer is specified via the framework configuration structure.

Implemented by: ADC Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

ADC Periodic Framework Interface description: ADC Periodic Framework

7.2.2 Interface API

sf_adc_periodic_api_t

Function name	Description
.open	Acquires mutex, then initializes driver at the HAL layer
.start	Starts the scan.
.stop	Stops the hardware trigger (timer) from triggering any more ADC scans.
.close	Releases channel mutex and closes channel at HAL layer.
.versionGet	Gets version and stores it in provided pointer p_version.

7.2.3 Data structures

- sf_adc_periodic_callback_args_t
- sf_adc_periodic_cfg_t

- sf_adc_periodic_instance_t

7.2.4 Enumerations

- sf_adc_periodic_event_t

7.2.5 Typedefs

- sf_adc_periodic_ctrl_t

7.2.6 Defines

- #define SF_ADC_PERIODIC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file
- #define SF_ADC_PERIODIC_API_VERSION_MINOR

  Initial value:`(3U)`

7.2.7 API Data

7.2.7.1 sf_adc_periodic_event_t

`sf_adc_periodic_event_t`

Detailed description

Options for the callback events.

Enumerated values

Name	Description
SF_ADC_PERIODIC_EVENT_NEW_DATA	New data is available in the buffer.

7.2.7.2 sf_adc_periodic_ctrl_t

`typedef void sf_adc_periodic_ctrl_t`

Detailed description

ADC periodic framework control block. Allocate an instance specific control block to pass into the ADC periodic framework API calls. Implemented as

- sf_adc_periodic_instance_ctrl_t

7.2.8 API Structures

7.2.8.1 sf_adc_periodic_callback_args_t

sf_adc_periodic_callback_args_t

Detailed description

ADC callback arguments definitions

Variables

- sf_adc_periodic_event_t event

  Periodic ADC callback event.

- uint32_t buffer_index

  Index to the buffer where the new data is stored.

- void const * p_context

  Placeholder for user data.

- adc_data_size_t * p_data_buffer

  Pointer to the buffer that will store the samples.

- uint32_t num_new_samples

  Number of new samples in the data buffer.

7.2.8.2 sf_adc_periodic_cfg_t

sf_adc_periodic_cfg_t

Detailed description

Configuration for RTOS integrated ADC driver

Variables

- adc_instance_t const *const p_lower_lvl_adc

  Pointer to the ADC instance.

- timer_instance_t const *const p_lower_lvl_timer

  Pointer to the Timer instance.

- transfer_instance_t const *const p_lower_lvl_transfer

  Pointer to the Transfer instance.

- adc_data_size_t * p_data_buffer

  Pointer to the buffer that will store the samples.

- uint32_t data_buffer_length

  Length of the data buffer that will store the samples.

- uint32_t sample_count

  Samples per channel to be buffered before notifying the app.

- elc_event_t scan_trigger

  The hardware trigger that starts the ADC scan.

- void(* p_callback)(sf_adc_periodic_callback_args_t *p_args)

  Callback function.

- void const * p_context

  Placeholder for user data.

- void const * p_extend

  Extension parameter for hardware specific settings.

7.2.8.3 sf_adc_periodic_api_t

sf_adc_periodic_api_t

Detailed description

Framework Periodic ADC API structure. Implementations will use the following API.

7.2.8.4 open

```
ssp_err_t(* sf_adc_periodic_api_t::open) (sf_adc_periodic_ctrl_t *const p_ctrl,
sf_adc_periodic_cfg_t const *const p_cfg)
```

Detailed description

Acquires mutex, then initializes driver at the HAL layer

Table 1: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. Elements initialized here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_adc_periodic_ctrl_t*const p_ctrl`

ADC periodic framework control block. Allocate an instance specific control block to pass into the ADC periodic framework API calls. Implemented assf_adc_periodic_instance_ctrl_t

Parameter p_cfg

Definition: `sf_adc_periodic_cfg_t const *const p_cfg`

Configuration for RTOS integrated ADC driver

- sf_adc_periodic_cfg_t::adc_instance_t

  Pointer to the ADC instance.
- sf_adc_periodic_cfg_t::timer_instance_t

  Pointer to the Timer instance.
- sf_adc_periodic_cfg_t::transfer_instance_t

  Pointer to the Transfer instance.
- sf_adc_periodic_cfg_t::adc_data_size_t

  Pointer to the buffer that will store the samples.
- sf_adc_periodic_cfg_t::data_buffer_length

  Length of the data buffer that will store the samples.
- sf_adc_periodic_cfg_t::sample_count

  Samples per channel to be buffered before notifying the app.
- sf_adc_periodic_cfg_t::elc_event_t

  The hardware trigger that starts the ADC scan.
- sf_adc_periodic_cfg_t::p_callback

  Callback function.
- sf_adc_periodic_cfg_t::p_context

  Placeholder for user data.
- sf_adc_periodic_cfg_t::p_extend

  Extension parameter for hardware specific settings.

7.2.8.5 start

```
ssp_err_t(* sf_adc_periodic_api_t::start) (sf_adc_periodic_ctrl_t *const p_ctrl)
```

Detailed description

Starts the scan.

ATTENTION: The driver will enable the ADC to be triggered via timer event; there will be a time delay from the time this function is called to the time the hardware timer count expires and triggers the scan.

Table 2: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in SF_ADC_PERIODIC_Open.

Parameter p_ctrl

Definition: `sf_adc_periodic_ctrl_t*const p_ctrl`

ADC periodic framework control block. Allocate an instance specific control block to pass into the ADC periodic framework API calls. Implemented assf_adc_periodic_instance_ctrl_t

7.2.8.6 stop

```
ssp_err_t(* sf_adc_periodic_api_t::stop) (sf_adc_periodic_ctrl_t *const p_ctrl)
```

Detailed description

Stops the hardware trigger (timer) from triggering any more ADC scans.

Table 3: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in SF_ADC_PERIODIC_Open.

Parameter p_ctrl

Definition: `sf_adc_periodic_ctrl_t*const p_ctrl`

ADC periodic framework control block. Allocate an instance specific control block to pass into the ADC periodic framework API calls. Implemented assf_adc_periodic_instance_ctrl_t

7.2.8.7 close

```
ssp_err_t(* sf_adc_periodic_api_t::close) (sf_adc_periodic_ctrl_t *const p_ctrl)
```

Detailed description

Releases channel mutex and closes channel at HAL layer.

Table 4: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in SF_ADC_PERIODIC_Open.

Parameter p_ctrl

Definition: `sf_adc_periodic_ctrl_t*const p_ctrl`

ADC periodic framework control block. Allocate an instance specific control block to pass into the ADC periodic framework API calls. Implemented assf_adc_periodic_instance_ctrl_t

7.2.8.8 versionGet

```
ssp_err_t(* sf_adc_periodic_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 5: Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.2.8.9 sf_adc_periodic_instance_t

sf_adc_periodic_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_adc_periodic_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_adc_periodic_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_adc_periodic_api_t const * p_api

  Pointer to the API structure for this instance.

7.3 Audio Framework Interface

RTOS-integrated Audio Framework Interface.

7.3.1 Summary

The Audio Interface is a ThreadX-aware Audio Framework Interface. The Interface is implemented by the Audio Framework using the Timer driver, the Transfer driver, and a choice of the following drivers for playback: DAC, PWM (to be implemented), or I2S (to be implemented).

Interfaces used:

- Transfer Interface
- DAC Interface

- Timer Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Audio Framework Interface description: Audio Playback DAC Framework

7.3.2 Interface API

sf_audio_playback_api_t

Function name	Description
.open	Configure the audio framework by creating a thread for audio playback and configuring HAL layer drivers used. This function must be called before any other audio functions.
.close	The close API handles cleans up internal driver data.
.start	Play audio. Currently only 16-bit mono PCM buffers are supported.
.pause	Pause audio playback. This stops the peripheral that triggers the DMA/DTC transfer and posts a flag to notify SF_AUDIO_PLAYBACK_Start() to pause any playback in progress.
.stop	Stop audio playback. Causes SF_AUDIO_PLAYBACK_Start() halt playback and return.
.resume	Resume audio playback. Posts a flag to notify SF_AUDIO_PLAYBACK_Start() to restart the peripheral that triggers the DMA/DTC transfer.
.volumeSet	Set software volume control. Software volume control is applied globally to all streams on the hardware.
.versionGet	Store version information in provided pointer.

7.3.3 Data structures

- sf_audio_playback_data_t

- sf_audio_playback_common_cfg_t
- sf_audio_playback_cfg_t
- sf_audio_playback_instance_t

7.3.4 Enumerations

- sf_audio_playback_status_t

7.3.5 Typedefs

- sf_audio_playback_common_ctrl_t
- sf_audio_playback_ctrl_t

7.3.6 Defines

- #define SF_AUDIO_PLAYBACK_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file
- #define SF_AUDIO_PLAYBACK_API_VERSION_MINOR

  Initial value:`(2U)`
- #define SF_AUDIO_PLAYBACK_MESSAGE_WORDS

  Initial value:`((sizeof(sf_message_payload_audio_t) + 3) / 4)`

  Audio playback message size in 4 byte words, rounded up.
- #define SF_AUDIO_PLAYBACK_MAX_VOLUME

  Initial value:`(255)`

  Macro defining the maximum volume.

7.3.7 API Data

7.3.7.1 sf_audio_playback_status_t

`sf_audio_playback_status_t`

Detailed description

Audio playback status.

Enumerated values

Name	Description
SF_AUDIO_PLAYBACK_STATUS_STOPPED	Stream is available to be used.
SF_AUDIO_PLAYBACK_STATUS_PAUSED	Stream is paused.
SF_AUDIO_PLAYBACK_STATUS_PLAYING	Stream is playing data.
SF_AUDIO_PLAYBACK_STATUS_WAITING	Stream is between packets, waiting for the next data message.

7.3.7.2 sf_audio_playback_common_ctrl_t

```
typedef void sf_audio_playback_common_ctrl_t
```

Detailed description

Audio playback common control block. Allocate an instance specific control block to pass to open in sf_audio_playback_cfg_t. Implemented as

- sf_audio_playback_common_instance_ctrl_t

7.3.7.3 sf_audio_playback_ctrl_t

```
typedef void sf_audio_playback_ctrl_t
```

Detailed description

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented as

- sf_audio_playback_instance_ctrl_t

7.3.8 API Structures

7.3.8.1 sf_audio_playback_data_t

sf_audio_playback_data_t

Detailed description

Audio data for playback.

Variables

- sf_message_header_t header

  Required common members of messaging framework payloads.

- sf_audio_playback_data_type_t type

  Data type. Must be uncompressed.

- uint32_t size_bytes

  Size of data in bytes.

- void const * p_data

  Pointer to data. Data start address must be 4-byte aligned.

- UINT loop_timeout

  ThreadX timeout, select TX_NO_WAIT to play the entire buffer once, TX_WAIT_FOREVER to loop until SF_AUDIO_PLAYBACK_Pause is called from another thread, or any timeout value from (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts to loop until the tick counts expire.

- bool stream_end

  This releases ownership of the stream and allows other threads to post data. Set to true if not more data will be sent as a part of this logical bitstream. Set to false if more packets are being prepared.

7.3.8.2 sf_audio_playback_common_cfg_t

sf_audio_playback_common_cfg_t

Detailed description

Common configuration for RTOS integrated audio framework. Shared by all streams.

Variables

- UINT priority

  Priority of the audio playback thread.

- sf_audio_playback_hw_instance_t const * p_lower_lvl_hw

  Hardware instance.

- sf_message_instance_t const * p_message

  Pointer to messaging framework instance used to post audio messages.

- TX_QUEUE * p_queue

  Pointer to the messaging framework queue specified for this audio stream. Must be subscribed to the SF_MESSAGE_EVENT_CLASS_AUDIO event class.

- void const * p_extend

  Implementation specific extension configuration.

7.3.8.3 sf_audio_playback_cfg_t

sf_audio_playback_cfg_t

Detailed description

Per stream configuration for RTOS integrated audio framework.

Variables

- void(* p_callback)(sf_message_callback_args_t *p_args)

  Callback called when playback of a buffer passed to sf_audio_playback_api_t::start is complete. Set to NULL for no callback.

- sf_audio_playback_common_ctrl_t * p_common_ctrl

  Pointer to the hardware control block used by this stream.

- sf_audio_playback_common_cfg_t const * p_common_cfg

  Pointer to common configurations shared by all streams using the same hardware.

- uint8_t class_instance

  Class instance used to identify the stream to the messaging framework.

7.3.8.4 sf_audio_playback_api_t

sf_audio_playback_api_t

Detailed description

Audio playback API structure. Audio playback implementations use the following API.

7.3.8.5 open

```
ssp_err_t(* sf_audio_playback_api_t::open) (sf_audio_playback_ctrl_t *const
p_ctrl, sf_audio_playback_cfg_t const *const p_cfg)
```

Brief description

Configure the audio framework by creating a thread for audio playback and configuring HAL layer drivers used. This function must be called before any other audio functions.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Open

Table 6: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a device structure allocated by user. The device control structure is initialized in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

Parameter p_cfg

Definition: `sf_audio_playback_cfg_t const *const p_cfg`

Per stream configuration for RTOS integrated audio framework.

- `sf_audio_playback_cfg_t`::`p_callback`

  Callback called when playback of a buffer passed to sf_audio_playback_api_t::start is complete. Set to NULL for no callback.

- `sf_audio_playback_cfg_t`::`sf_audio_playback_common_ctrl_t`

  Pointer to the hardware control block used by this stream.

- `sf_audio_playback_cfg_t`::`sf_audio_playback_common_cfg_t`

  Pointer to common configurations shared by all streams using the same hardware.

- `sf_audio_playback_cfg_t`::`class_instance`

  Class instance used to identify the stream to the messaging framework.

7.3.8.6 close

```
ssp_err_t(* sf_audio_playback_api_t::close) (sf_audio_playback_ctrl_t *const
p_ctrl)
```

Brief description

The close API handles cleans up internal driver data.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Close

Table 7: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for audio driver.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

7.3.8.7 start

```
ssp_err_t(* sf_audio_playback_api_t::start) (sf_audio_playback_ctrl_t *const
p_ctrl, sf_audio_playback_data_t *const p_data, UINT const timeout)
```

Brief description

Play audio. Currently only 16-bit mono PCM buffers are supported.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Start

NOTE: Call SF_MESSAGE_Open to configure the messaging framework control block and queues with the parameters specified in sf_audio_playback_cfg_t::p_message and sf_audio_playback_cfg_t::p_queue.

Table 8: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to device control block initialized in Open call for audio driver.
p_data	in	Pointer to data, description, timeout values, and synchronization options.
timeout	in	ThreadX timeout, represents the maximum amount of time to wait to post to the audio queue. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout values from 0x00000001 through 0xFFFFFFFE in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

Parameter p_data

Definition: `sf_audio_playback_data_t*const p_data`

Audio data for playback.

- `sf_audio_playback_data_t::header`

  Required common members of messaging framework payloads.

- `sf_audio_playback_data_t::type`

  Data type. Must be uncompressed.

- `sf_audio_playback_data_t::size_bytes`

  Size of data in bytes.

- `sf_audio_playback_data_t::p_data`

  Pointer to data. Data start address must be 4-byte aligned.

- `sf_audio_playback_data_t::loop_timeout`

  ThreadX timeout, select TX_NO_WAIT to play the entire buffer once, TX_WAIT_FOREVER to loop until SF_AUDIO_PLAYBACK_Pause is called from another thread, or any timeout value from (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts to loop until the tick counts expire.

- `sf_audio_playback_data_t::stream_end`

  This releases ownership of the stream and allows other threads to post data. Set to true if not more data will be sent as a part of this logical bitstream. Set to false if more packets are being prepared.

Parameter timeout

`const`

7.3.8.8 pause

```
ssp_err_t(* sf_audio_playback_api_t::pause) (sf_audio_playback_ctrl_t *const
p_ctrl)
```

Brief description

Pause audio playback. This stops the peripheral that triggers the DMA/DTC transfer and posts a flag to notify SF_AUDIO_PLAYBACK_Start to pause any playback in progress.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Pause

NOTE: Call SF_AUDIO_PLAYBACK_Start before using this function. Calling SF_AUDIO_PLAYBACK_Pause before SF_AUDIO_PLAYBACK_Start has no effect and does not return an error code.

Table 9: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for audio driver.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

7.3.8.9 stop

```
ssp_err_t(* sf_audio_playback_api_t::stop) (sf_audio_playback_ctrl_t *const
p_ctrl)
```

Brief description

Stop audio playback. Causes SF_AUDIO_PLAYBACK_Start halt playback and return.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Stop

NOTE: Call SF_AUDIO_PLAYBACK_Start before using this function. Calling SF_AUDIO_PLAYBACK_Stop before SF_AUDIO_PLAYBACK_Start has no effect and does not return an error code.

Table 10: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for audio driver.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

7.3.8.10 resume

```
ssp_err_t(* sf_audio_playback_api_t::resume) (sf_audio_playback_ctrl_t *const
p_ctrl)
```

Brief description

Resume audio playback. Posts a flag to notify SF_AUDIO_PLAYBACK_Start to restart the peripheral that triggers the DMA/DTC transfer.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_Resume

NOTE: Call SF_AUDIO_PLAYBACK_Pause before using this function. Calling SF_AUDIO_PLAYBACK_Resume before SF_AUDIO_PLAYBACK_Pause has no effect and does not return an error code.

Table 11: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for audio driver.

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

7.3.8.11 volumeSet

```
ssp_err_t(* sf_audio_playback_api_t::volumeSet) (sf_audio_playback_ctrl_t *const
p_ctrl, uint8_t const volume)
```

Brief description

Set software volume control. Software volume control is applied globally to all streams on the hardware.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_VolumeSet

ATTENTION: Software volume control reduces resolution and may require extra memory and processing bandwidth.

Table 12: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for audio driver.
volume	in	Volume level requested. Valid range is from 0 (muted, which will stop playback) to 255 (maximum volume, default on open).

Parameter p_ctrl

Definition: `sf_audio_playback_ctrl_t*const p_ctrl`

Audio playback framework control block. Allocate an instance specific control block to pass into the audio playback framework API calls. Implemented assf_audio_playback_instance_ctrl_t

Parameter volume

`uint8_t`

7.3.8.12 versionGet

```
ssp_err_t(* sf_audio_playback_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Store version information in provided pointer.

Detailed description

Implemented as

- SF_AUDIO_PLAYBACK_VersionGet

Table 13: Parameters

Name	Direction	Description
p_version	in	Pointer to device control block initialized in Open call for UART driver.

Parameter p_version

7.3.8.13 sf_audio_playback_instance_t

sf_audio_playback_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_audio_playback_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_audio_playback_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_audio_playback_api_t const * p_api

  Pointer to the API structure for this instance.

7.4 Audio Playback Framework Interface

RTOS-integrated Audio Playback Framework Interface.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

7.4.1 Summary

Audio playback driver to play buffers of audio data.

Implemented by: DAC Audio Playback Framework

Audio Framework Interface description: Audio Playback DAC Framework

7.4.2 Interface API

sf_audio_playback_hw_api_t

Function name	Description
.open	Open a device channel for read/write and control.
.start	Start audio playback hardware.
.stop	Stop audio playback hardware.
.play	Play audio buffer.
.dataTypeGet	Stores expected data type in provided pointer p_data_type.
.close	Close the audio driver.
.versionGet	Return the version of the driver.

7.4.3 Data structures

- sf_audio_playback_data_type_t
- sf_audio_playback_hw_callback_args_t
- sf_audio_playback_hw_cfg_t
- sf_audio_playback_hw_instance_t

7.4.4 Typedefs

- sf_audio_playback_hw_ctrl_t

7.4.5 Defines

- #define SF_AUDIO_PLAYBACK_HW_API_VERSION_MAJOR

  Initial value:`(1U)`

- #define SF_AUDIO_PLAYBACK_HW_API_VERSION_MINOR

  Initial value:(2U)

7.4.6 API Data

7.4.6.1 sf_audio_playback_hw_ctrl_t

```
typedef void sf_audio_playback_hw_ctrl_t
```

Detailed description

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented as

- sf_audio_playback_hw_dac_instance_ctrl_t
- sf_audio_playback_hw_i2s_instance_ctrl_t

7.4.7 API Structures

7.4.7.1 sf_audio_playback_data_type_t

sf_audio_playback_data_type_t

Detailed description

Audio data type.

Variables

- uint8_t scale_bits_max

  Maximum data resolution in bits.

- bool is_signed

  Set to 1 for signed samples, or 0 for unsigned samples.

7.4.7.2 sf_audio_playback_hw_callback_args_t

sf_audio_playback_hw_callback_args_t

Detailed description

Callback function parameter data

Variables

- void * p_context

  Placeholder for user data. Set in sf_audio_playback_hw_api_t::open function in sf_audio_playback_hw_cfg_t.

7.4.7.3 sf_audio_playback_hw_cfg_t

sf_audio_playback_hw_cfg_t

Detailed description

Audio playback driver configuration.

Variables

- void(* p_callback)(sf_audio_playback_hw_callback_args_t *p_args)

  Callback called when play is complete. Set to NULL for no callback.

- void * p_context

  Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

- void const * p_extend

  Hardware dependent configuration.

7.4.7.4 sf_audio_playback_hw_api_t

sf_audio_playback_hw_api_t

Detailed description

Audio playback API definition.

7.4.7.5 open

```
(* sf_audio_playback_hw_api_t::open) (sf_audio_playback_hw_ctrl_t *const p_ctrl,
sf_audio_playback_hw_cfg_t const *const p_cfg)
```

Detailed description

Open a device channel for read/write and control. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_Open

Table 14: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to memory allocated for control block.
p_cfg	in	Pointer to the hardware configurations.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

Parameter p_cfg

Definition: `sf_audio_playback_hw_cfg_t const *const p_cfg`

Audio playback driver configuration.

- `sf_audio_playback_hw_cfg_t`::`p_callback`

  Callback called when play is complete. Set to NULL for no callback.

- `sf_audio_playback_hw_cfg_t`::`p_context`

  Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

- `sf_audio_playback_hw_cfg_t`::`p_extend`

  Hardware dependent configuration.

7.4.7.6 start

`(* sf_audio_playback_hw_api_t::start) (sf_audio_playback_hw_ctrl_t *const p_ctrl)`

Detailed description

Start audio playback hardware. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_Start

Table 15: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized in open.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented
assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

7.4.7.7 stop

`(* sf_audio_playback_hw_api_t::stop) (sf_audio_playback_hw_ctrl_t *const p_ctrl)`

Detailed description

Stop audio playback hardware. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_Stop

Table 16: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized in open.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

7.4.7.8 play

```
(* sf_audio_playback_hw_api_t::play) (sf_audio_playback_hw_ctrl_t *const p_ctrl,
int16_t const *const p_buffer, uint32_t length)
```

Detailed description

Play audio buffer. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_Play

Table 17: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized in open.
p_buffer	in	Pointer to buffer with PCM samples to play. Data must be scaled for audio playback hardware.
length	in	Length of data in p_buffer.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

Parameter p_buffer

Parameter length

`uint32_t`

7.4.7.9 dataTypeGet

```
(* sf_audio_playback_hw_api_t::dataTypeGet) (sf_audio_playback_hw_ctrl_t *const
p_ctrl, sf_audio_playback_data_type_t *const p_data_type)
```

Detailed description

Stores expected data type in provided pointer p_data_type. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_DataTypeGet

Table 18: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized in open.
p_data_type	out	Pointer to audio sample data type required by hardware.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented
assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

Parameter p_data_type

Definition: `sf_audio_playback_data_type_t*const p_data_type`

Audio data type.

- `sf_audio_playback_data_type_t::scale_bits_max`

  Maximum data resolution in bits.

- `sf_audio_playback_data_type_t::is_signed`

  Set to 1 for signed samples, or 0 for unsigned samples.

7.4.7.10 close

```
(* sf_audio_playback_hw_api_t::close) (sf_audio_playback_hw_ctrl_t *const p_ctrl)
```

Detailed description

Close the audio driver. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_Close

Table 19: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized in open.

Parameter p_ctrl

Definition: `sf_audio_playback_hw_ctrl_t*const p_ctrl`

Audio playback hardware control block. Allocate an instance specific control block to pass into the audio playback hardware API calls. Implemented
assf_audio_playback_hw_dac_instance_ctrl_tsf_audio_playback_hw_i2s_instance_ctrl_t

7.4.7.11 versionGet

```
(* sf_audio_playback_hw_api_t::versionGet) ( *const p_version)
```

Detailed description

Return the version of the driver. Implemented as

- SF_AUDIO_PLAYBACK_HW_DAC_VersionGet

Table 20: Parameters

Name	Direction	Description
p_version	out	Pointer to variable that will be populated with version information.

Parameter p_version

7.4.7.12 sf_audio_playback_hw_instance_t

sf_audio_playback_hw_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_audio_playback_hw_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_audio_playback_hw_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_audio_playback_hw_api_t const * p_api

  Pointer to the API structure for this instance.

7.5 Audio Recording Framework Interface

RTOS-integrated Audio Recording Framework Interface.

7.5.1 Summary

The Audio Record Interface is a ThreadX-aware Interface for Audio Recording. The Interface is implemented by the ADC Audio recording Framework using the ADC periodic Framework driver for recording.

Interfaces used:

- ADC periodic Framework

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

7.5.2 Interface API

sf_audio_record_api_t

Function name	Description
.open	Initializes audio recording framework.
.start	Starts audio recording.
.stop	Stops audio recording.
.infoGet	Gets channel information(Mono/Stereo).
.close	Releases channel mutex and closes channel at HAL layer.
.versionGet	Gets version and stores it in provided pointer p_version.

7.5.3 Data structures

- sf_audio_record_info_t
- sf_audio_record_callback_args_t
- sf_audio_record_cfg_t
- sf_audio_record_instance_t

7.5.4 Enumerations

- sf_audio_record_channel_t
- sf_audio_record_data_size_t
- sf_audio_record_event_t

7.5.5 Typedefs

- sf_audio_record_ctrl_t

7.5.6 Defines

- #define SF_AUDIO_RECORD_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file
- #define SF_AUDIO_RECORD_API_VERSION_MINOR

  Initial value:`(1U)`

7.5.7 API Data

7.5.7.1 sf_audio_record_channel_t

`sf_audio_record_channel_t`

Detailed description

Definition of audio recording mode.

Enumerated values

Name	Description
SF_AUDIO_RECORD_CHANNEL_MONO	Support Mono Channel.

Name	Description
SF_AUDIO_RECORD_CHANNEL_STEREO	Support Stereo Channel.

7.5.7.2 sf_audio_record_data_size_t

```
sf_audio_record_data_size_t
```

Detailed description

Definition of audio recording data sample size.

Enumerated values

Name	Description
SF_AUDIO_RECORD_DATA_SIZE_8BIT	data width in the sample is 8bit
SF_AUDIO_RECORD_DATA_SIZE_16BIT	data width in the sample is 16bit

7.5.7.3 sf_audio_record_event_t

```
sf_audio_record_event_t
```

Detailed description

Definition of events for audio recording.

Enumerated values

Name	Description
SF_AUDIO_RECORD_EVENT_NEW_DATA	New data is available in the buffer.

7.5.7.4 sf_audio_record_ctrl_t

```
typedef void sf_audio_record_ctrl_t
```

Detailed description

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented as

- sf_audio_record_adc_instance_ctrl_t

7.5.8 API Structures

7.5.8.1 sf_audio_record_info_t

sf_audio_record_info_t

Detailed description

Variables

- sf_audio_record_channel_t channel_info

7.5.8.2 sf_audio_record_callback_args_t

sf_audio_record_callback_args_t

Detailed description

Variables

- sf_audio_record_event_t event

  Audio callback event.
- uint32_t buffer_index

  Index to the buffer where the new data is stored.
- void const * p_context

  Placeholder for user data.

7.5.8.3 sf_audio_record_cfg_t

sf_audio_record_cfg_t

Detailed description

Configuration for audio recording framework

Variables

- sf_audio_record_data_size_t capture_data_size

  Size of data in the sample 8 or 16 bit.
- uint32_t sampling_rate_hz

  Sampling rate for audio capture.
- void * p_capture_data_buffer

  Pointer to the buffer that will store the samples
- uint32_t capture_data_buffer_size

  total size of buffer configured by user to store samples
- uint32_t sample_count

  Samples per channel to be buffered before notifying the user via callback

- void(* p_callback)(sf_audio_record_callback_args_t *p_args)

  Callback function.
- void const * p_context

  Placeholder for user data.
- void const * p_extend

  Extension parameter for hardware specific settings.

7.5.8.4 sf_audio_record_api_t

sf_audio_record_api_t

Detailed description

Framework Audio Recording API structure. Implementations will use the following API.

7.5.8.5 open

```
ssp_err_t(* sf_audio_record_api_t::open) (sf_audio_record_ctrl_t *const p_ctrl,
sf_audio_record_cfg_t const *const p_cfg)
```

Brief description

Initializes audio recording framework.

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_Open

Table 21: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. Elements initialized here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_audio_record_ctrl_t*const p_ctrl`

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented assf_audio_record_adc_instance_ctrl_t

Parameter p_cfg

Definition: `sf_audio_record_cfg_t const *const p_cfg`

Configuration for audio recording framework

- sf_audio_record_cfg_t::sf_audio_record_data_size_t

  Size of data in the sample 8 or 16 bit.

  Enumerated as:

  – SF_AUDIO_RECORD_DATA_SIZE_8BIT

  – SF_AUDIO_RECORD_DATA_SIZE_16BIT
- sf_audio_record_cfg_t::sampling_rate_hz

  Sampling rate for audio capture.
- sf_audio_record_cfg_t::p_capture_data_buffer

  Pointer to the buffer that will store the samples
- sf_audio_record_cfg_t::capture_data_buffer_size

  total size of buffer configured by user to store samples
- sf_audio_record_cfg_t::sample_count

  Samples per channel to be buffered before notifying the user via callback
- sf_audio_record_cfg_t::p_callback

  Callback function.
- sf_audio_record_cfg_t::p_context

  Placeholder for user data.
- sf_audio_record_cfg_t::p_extend

  Extension parameter for hardware specific settings.

7.5.8.6 start

```
ssp_err_t(* sf_audio_record_api_t::start) (sf_audio_record_ctrl_t *const p_ctrl)
```

Brief description

Starts audio recording.

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_Start

Table 22: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set

Parameter p_ctrl

Definition: `sf_audio_record_ctrl_t*const p_ctrl`

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented assf_audio_record_adc_instance_ctrl_t

7.5.8.7 stop

```
ssp_err_t(* sf_audio_record_api_t::stop) (sf_audio_record_ctrl_t *const p_ctrl)
```

Brief description

Stops audio recording.

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_Stop

Table 23: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block .

Parameter p_ctrl

Definition: `sf_audio_record_ctrl_t*const p_ctrl`

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented assf_audio_record_adc_instance_ctrl_t

7.5.8.8 infoGet

```
ssp_err_t(* sf_audio_record_api_t::infoGet) (sf_audio_record_ctrl_t *const
p_ctrl, sf_audio_record_info_t *p_info)
```

Brief description

Gets channel information(Mono/Stereo).

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_InfoGet

Table 24: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block .

Parameter p_ctrl

Definition: `sf_audio_record_ctrl_t*const p_ctrl`

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented assf_audio_record_adc_instance_ctrl_t

7.5.8.9 close

```
ssp_err_t(* sf_audio_record_api_t::close) (sf_audio_record_ctrl_t *const p_ctrl)
```

Brief description

Releases channel mutex and closes channel at HAL layer.

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_Close

Table 25: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block .

Parameter p_ctrl

Definition: `sf_audio_record_ctrl_t*const p_ctrl`

Audio record framework control block. Allocate an instance specific control block to pass into the audio record framework API calls. Implemented assf_audio_record_adc_instance_ctrl_t

7.5.8.10 versionGet

```
ssp_err_t(* sf_audio_record_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Implemented as

- SF_AUDIO_RECORD_ADC_VersionGet

Table 26: Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.5.8.11 sf_audio_record_instance_t

sf_audio_record_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_audio_record_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_audio_record_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_audio_record_api_t const * p_api

  Pointer to the API structure for this instance.

7.6 BLE Framework API

7.6.1 SF BLE Framework Interface

RTOS-integrated SF BLE Framework Interface.

7.6.1.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Framework.

7.6.1.2 Interface API

sf_ble_api_t

Function name	Description
.open	Initializes the interface for data transfers. Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	De-initialize the interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.
.infoGet	Get BLE module information like chipset information and RSSI value.

Function name	Description
.provisionGet	Reads the current BLE Provisioning information.
.provisionSet	Provisions the BLE Driver and set bonding and security modes as provisioned.
.scan	Scans for available BLE devices and return the list to the caller.
.advertisementStart	Make the device discoverable by broadcasting device information to remote devices.
.advertisementStop	Stop the device from being discoverable.
.whitelistAdd	Add specified devices to whitelist.
.whitelistDel	Remove specified devices from whitelist.
.bondingStart	Initiate bonding process with remote BLE device and exchange security keys if enabled.
.bondingResponse	Respond to the bonding request from the remote BLE device. Send bonding response on reception of bonding indication.
.startEncryption	Start encryption over a connection.
.connect	Connect to a remote BLE device.
.listen	Listen for connection request from remote device.
.disconnect	Terminate connection with remote BLE device.
.gattAddCustomProfiles	Add Custom Profile to GATT Database. This function is called with list of service and characteristics which is to be added to GATT database. Services and characteristics which were previously added will be removed and newly passed services and characteristics will be added.If services and characteristics were previously added and now those services and characteristics are to be removed and no new services and characteristics are to be added , then call this API with service and characteristics length as zero
.gattServiceDiscovery	Perform service discovery used by GATT client.Perform service discovery on remote GATT server and get results.

Function name	Description
.gattCharDiscovery	Perform characteristics discovery used by GATT client.Perform characteristics discovery on remote GATT server and get results.
.gattCharDescDiscovery	Perform characteristics descriptor discovery used by GATT client.
.gattCharRead	Perform read characteristic used by GATT client.Read characteristic value from remote GATT server.
.gattCharWrite	Perform write characteristic used by GATT client.Write characteristic value on remote GATT server.
.gattCharExecuteWrite	Perform execute write on all pending write operations, used by GATT client.Execute or cancel all prepared writes of characteristics on remote GATT server.
.gattCharWriteLocal	Perform local characteristic write used by GATT server.Writes local characteristic value on GATT server.
.gattSendNotify	Send notification to GATT client, used by GATT server.
.gattSendIndicate	Send indication to GATT client, used by GATT server.Send indication to remote GATT client.
.gattWriteResponse	Send response to write operation received by GATT client, used by GATT server.Send response for write request received from remote GATT client.
.versionGet	Gets version and stores it in provided pointer p_version.

7.6.1.3 Data structures

- sf_ble_addr_t
- sf_ble_adv_data_t
- sf_ble_event_info_t
- sf_ble_sec_info_t
- sf_ble_provisioning_t
- sf_ble_scan_info_t
- sf_ble_bonding_start_t

- sf_ble_bonding_response_t
- sf_ble_sm_tk_info_t
- sf_ble_sm_tk_ind_t
- sf_ble_addr_verify_ind_t
- sf_ble_sm_key_ind_t
- sf_ble_sm_enc_info_t
- sf_ble_sec_enc_start_ind_t
- sf_ble_adv_info_t
- sf_ble_scan_t
- sf_ble_connection_t
- sf_ble_disconnect_t
- sf_ble_chipset_info_t
- sf_ble_info_t
- sf_ble_bonding_req_ind_t
- sf_ble_connect_info_t
- sf_ble_uuid_t
- sf_ble_svc_attribute_t
- sf_ble_char_attribute_t
- sf_ble_gatt_attr_event_t
- sf_ble_service_discovery_rsp_t
- sf_ble_service_discovery_req_t
- sf_ble_char_discovery_req_t
- sf_ble_char_discovery_rsp_t
- sf_ble_char_desc_discovery_rsp_t
- sf_ble_char_multiple_read_req_t
- sf_ble_char_read_req_t
- sf_ble_char_multiple_read_rsp_t
- sf_ble_char_read_by_uuid_rsp_t
- sf_ble_char_read_by_handle_rsp_t
- sf_ble_char_read_rsp_t
- sf_ble_char_write_req_t
- sf_ble_gatt_notif_ind_event_data_t
- sf_ble_write_cmd_event_data_t
- sf_ble_ctrl_t

- sf_ble_cfg_t
- sf_ble_instance_t

7.6.1.4 Enumerations

- sf_ble_event_t
- sf_ble_gap_role_t
- sf_ble_addr_type_t
- sf_ble_scan_mode_t
- sf_ble_bonding_mode_t
- sf_ble_sec_mode_t
- sf_ble_disc_type_t
- sf_ble_conn_type_t
- sf_ble_adv_filt_type_t
- sf_ble_init_filt_type_t
- sf_ble_adv_chnl_map_t
- sf_ble_iocap_t
- sf_ble_auth_type_t
- sf_ble_disconnect_reason_t
- sf_ble_key_dist_t
- sf_ble_char_property_t
- sf_ble_service_discovery_t
- sf_ble_char_discovery_t
- sf_ble_uuid_length_t
- sf_ble_char_read_t
- sf_ble_char_write_t
- sf_ble_execute_write_t
- sf_ble_write_response_t
- sf_ble_attribute_error_code_t
- sf_ble_char_attr_permissions_t

7.6.1.5 Typedefs

- sf_ble_callback_t
- sf_ble_conn_handle_t

7.6.1.6 Defines

- #define SF_BLE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file
- #define SF_BLE_API_VERSION_MINOR

  Initial value:`(1U)`

  Minor Version of the API defined in this file
- #define SF_BLE_MAX_BLE_ADV_DATA_LEN

  Initial value:`(100U)`

  BLE advertising data length.
- #define SF_BLE_MAX_NAME_LEN

  Initial value:`(66U)`

  Maximum length of name for BLE device.
- #define SF_BLE_MAX_GAP_NAME_LEN

  Initial value:`(23U)`

  Maximum length of GAP name.
- #define SF_BLE_ADDR_LEN

  Initial value:`(6U)`

  BLE address length.
- #define SF_BLE_SEC_KEY_LEN

  Initial value:`(0x10U)`

  BLE Security Key length.
- #define SF_BLE_128BITS_UUID_LENGTH

  Initial value:`(16U)`

  128 bit UUID length
- #define SF_BLE_RAND_NUM_LENGTH

  Initial value:`(8U)`

  Rand Number length.
- #define SF_BLE_TRUE

  Initial value:`(1U)`

  Boolean True Condition.

- #define SF_BLE_FALSE

  Initial value:`(0U)`

  Boolean False Condition.
- #define SF_BLE_MAX_MULTI_CHAR_READ_CNT

  Initial value:`(4U)`

  Maximum characteristics count for multiple read
- #define SF_BLE_MAX_CHAR_UUID_READ_CNT

  Initial value:`(10U)`

  Maximum characteristics count in result of characteristic read by UUID
- #define SF_BLE_GATT_LEN_UNDEF

  Initial value:`(0xFF)`

  Variable length characteristic value which can not be defined
- #define SF_BLE_ADV_DATA_LEN

  Initial value:`(0x1F)`

  BLE Advertising Data Length
- #define SF_BLE_GATT_PRI_SERVICE

  Initial value:`(0x2800U)`

  BLE GATT Service type Primary
- #define SF_BLE_GATT_INCLUDE_SERVICE

  Initial value:`(0x2802U)`

  BLE GATT Service type Included
- #define SF_BLE_GATT_CHAR_DECLARE

  Initial value:`(0x2803U)`

  BLE GATT Declare Characteristics

7.6.1.7 API Data

sf_ble_event_t

`sf_ble_event_t`

Detailed description

BLE events

Enumerated values

Name	Description
SF_BLE_EVENT_NONE	BLE user event none.
SF_BLE_EVENT_BONDING_INDICATION	BLE user event indicating reception of bonding request.
SF_BLE_EVENT_CONNECTION_COMP	BLE user event indicating connection completion.
SF_BLE_EVENT_DISCONNECT_COMP	BLE user event indicating disconnect.
SF_BLE_EVENT_SM_TK_REQ_IND	BLE user event indicating request for Temporary key.
SF_BLE_EVENT_SM_KEY_IND	BLE user event indicating received key from user.
SF_BLE_EVENT_SM_CHK_BD_ADDR_REQ	BLE user event indicating validation of remote address.
SF_BLE_EVENT_SM_ENC_START_IND	BLE user event indicating encryption started.
SF_BLE_EVENT_GATT_NOTIFICATION	BLE user event for Notification from peer.
SF_BLE_EVENT_GATT_INDICATION	BLE user event for Indication from peer.
SF_BLE_EVENT_GATT_WRITE_CMD_INDICATION	BLE user event for Write command from peer.
SF_BLE_EVENT_GATT_RESPONSE_TIMEOUT	BLE user event for GATT operation timeout.
SF_BLE_EVENT_GATT_ATTR_WRITE_REQ	BLE user event for Remote Write request on user Attributes.
SF_BLE_EVENT_GATT_ATTR_READ_REQ	BLE user event for Remote Read request on user Attributes.
SF_BLE_EVENT_GATT_ATTR_WRITE_COMPLETE	BLE user event Indicating Remote Write complete for user Attributes.
SF_BLE_EVENT_GATT_ATTR_WRITE_CCCD_REQ	BLE user event for Remote CCCD Write request on user Attributes.
SF_BLE_EVENT_GATT_ATTR_WRITE_CCCD_COMPLETE	BLE user event Indicating Remote CCCD Write complete for user Attributes.

sf_ble_gap_role_t

`sf_ble_gap_role_t`

Detailed description

GAP role

Enumerated values

Name	Description
SF_BLE_GAP_ROLE_MASTER	GAP role master.
SF_BLE_GAP_ROLE_SLAVE	GAP role slave.

sf_ble_addr_type_t

`sf_ble_addr_type_t`

Detailed description

BLE address type

Enumerated values

Name	Description
SF_BLE_ADDR_TYPE_PUBLIC	BLE address type public.
SF_BLE_ADDR_TYPE_RANDOM	BLE address type random.

sf_ble_scan_mode_t

`sf_ble_scan_mode_t`

Detailed description

BLE scan mode

Enumerated values

Name	Description
SF_BLE_SCAN_MODE_PASSIVE	BLE scan mode passive.
SF_BLE_SCAN_MODE_ACTIVE	BLE scan mode active.

sf_ble_bonding_mode_t

`sf_ble_bonding_mode_t`

Detailed description

BLE bonding mode

Enumerated values

Name	Description
SF_BLE_BONDING_MODE_NONBONDABLE	BLE bonding not supported.
SF_BLE_BONDING_MODE_BONDABLE	BLE bonding supported.

sf_ble_sec_mode_t

`sf_ble_sec_mode_t`

Detailed description

BLE security mode

Enumerated values

Name	Description
SF_BLE_SEC_MODE1_LVL1_NO_SEC	BLE no security.
SF_BLE_SEC_MODE1_LVL2_NOAUTH_PAIR_ENC	BLE no authentication pairing encryption.
SF_BLE_SEC_MODE1_LVL3_AUTH_PAIR_ENC	Authenticated pairing with encryption.
SF_BLE_SEC_MODE1_LVL4_AUTHLE_PAIR_ENC	Authenticated LE Secure Connections pairing with encryption.
SF_BLE_SEC_MODE2_LVL1_NOAUTH_DATA_SIGNED	Unauthenticated pairing with data signing.
SF_BLE_SEC_MODE2_LVL2_AUTH_DATA_SIGNED	Authentication pairing with data signing.

sf_ble_disc_type_t

`sf_ble_disc_type_t`

Detailed description

BLE discovery type

Enumerated values

Name	Description
SF_BLE_DISC_TYPE_NON_DISCOVERABLE	BLE non-discoverable.
SF_BLE_DISC_TYPE_GENERAL_DISCOVERABLE	BLE discoverable.
SF_BLE_DISC_TYPE_LIMITED_DISCOVERABLE	BLE limited discoverable.

sf_ble_conn_type_t

`sf_ble_conn_type_t`

Detailed description

BLE connection type

Enumerated values

Name	Description
SF_BLE_CONN_TYPE_BROADCASTER	Connection type broadcaster.
SF_BLE_CONN_TYPE_NON_CONNECTABLE	Connection type connection not allowed.
SF_BLE_CONN_TYPE_UNDIRECTED_CONNECTABLE	Connection type undirected.
SF_BLE_CONN_TYPE_DIRECTED_CONNECTABLE	Connection type directed.

sf_ble_adv_filt_type_t

`sf_ble_adv_filt_type_t`

Detailed description

BLE advertisement filter type

Enumerated values

Name	Description
SF_BLE_ADV_FILT_TYPE_ALLOW_ALL	Filter type allow all.
SF_BLE_ADV_FILT_TYPE_ALLOW_SCAN_WLIST_CON_ANY	Filter type scan only whitelist entries, connect to any.
SF_BLE_ADV_FILT_TYPE_ALLOW_SCAN_ANY_CON_WLIST	Filter type scan any, connect to whitelist entries only.
SF_BLE_ADV_FILT_TYPE_ALLOW_WLIST_ONLY	Filter type allow whitelist only for both scan and connection.

sf_ble_init_filt_type_t

`sf_ble_init_filt_type_t`

Detailed description

BLE initial filter type

Enumerated values

Name	Description
SF_BLE_INIT_FILT_TYPE_IGNORE_WLIST	Ignore whitelist.
SF_BLE_INIT_FILT_TYPE_USE_WLIST	Use whitelist.

sf_ble_adv_chnl_map_t

`sf_ble_adv_chnl_map_t`

Detailed description

BLE advertisement channel map

Enumerated values

Name	Description
SF_BLE_ADV_CHNL_37	Enable channel 37 use.
SF_BLE_ADV_CHNL_38	Enable channel 38 use.
SF_BLE_ADV_CHNL_39	Enable channel 39 use.
SF_BLE_ADV_CHNL_ALL_CHANNELS	all channels(37, 38 and 39) enabled

sf_ble_iocap_t

`sf_ble_iocap_t`

Detailed description

BLE System IO Capabilities

Enumerated values

Name	Description
SF_BLE_IO_CAP_DISPLAY_ONLY	Display Only.
SF_BLE_IO_CAP_DISPLAY_YES_NO	Display Yes No.
SF_BLE_IO_CAP_KB_ONLY	Keyboard Only.
SF_BLE_IO_CAP_NO_INPUT_NO_OUTPUT	No Input No Output.
SF_BLE_IO_CAP_KB_DISPLAY	Keyboard Display.

sf_ble_auth_type_t

`sf_ble_auth_type_t`

Detailed description

BLE security authorization requirement

Enumerated values

Name	Description
SF_BLE_AUTH_TYPE_NONE	No Security.
SF_BLE_AUTH_TYPE_UNAUTHENTICATED_NO_MITM	No MITM.
SF_BLE_AUTH_TYPE_AUTHENTICATED_MITM	MITM Protected.

sf_ble_disconnect_reason_t

`sf_ble_disconnect_reason_t`

Detailed description

BLE security authorization requirement

Enumerated values

Name	Description
SF_BLE_DISCONNECT_REASON_LOCAL_HOST	Disconnected by local machine.
SF_BLE_DISCONNECT_REASON_REMOTE_USER	Disconnected by remote device.
SF_BLE_DISCONNECT_REASON_ERR	Error in Connection so disconnected.

sf_ble_key_dist_t

`sf_ble_key_dist_t`

Detailed description

BLE security key distribution type

Enumerated values

Name	Description
SF_BLE_KEY_DIST_NONE	No Keys to distribute.
SF_BLE_KEY_DIST_ENCKEY	Encryption key in distribution.
SF_BLE_KEY_DIST_IDKEY	IRK (ID key)in distribution.

Name	Description
SF_BLE_KEY_DIST_SIGNKEY	CSRK(Signature key) in distribution.

sf_ble_char_property_t

`sf_ble_char_property_t`

Detailed description

Characteristic property

Enumerated values

Name	Description
SF_BLE_CHAR_PROPERTY_BCAST	Permits broadcasts of the Characteristic Value.
SF_BLE_CHAR_PROPERTY_RD	Permits reads of the Characteristic Value.
SF_BLE_CHAR_PROPERTY_WR_NO_RESP	Permits writes of the Characteristic Value without response.
SF_BLE_CHAR_PROPERTY_WR	Permits writes of the Characteristic Value with response.
SF_BLE_CHAR_PROPERTY_NTF	Permits notifications of the Characteristic Value without acknowledgment.
SF_BLE_CHAR_PROPERTY_IND	Permits indications of a Characteristic Value with acknowledgment.
SF_BLE_CHAR_PROPERTY_AUTH	Permits signed writes to the Characteristic Value.
SF_BLE_CHAR_PROPERTY_EXT_PROP	Additional characteristic properties.

sf_ble_service_discovery_t

`sf_ble_service_discovery_t`

Detailed description

Service discovery type

Enumerated values

Name	Description
SF_BLE_SERVICE_DISCOVERY_PRIMARY_ALL	Discover all primary services.
SF_BLE_SERVICE_DISCOVERY_PRIMARY_UUID	Discover primary service by UUID.
SF_BLE_SERVICE_DISCOVERY_INCLUDED	Discover includes services.

sf_ble_char_discovery_t

`sf_ble_char_discovery_t`

Detailed description

Characteristics discovery type

Enumerated values

Name	Description
SF_BLE_CHAR_DISCOVERY_ALL	Discover all characteristics.
SF_BLE_CHAR_DISCOVERY_UUID	Discover characteristics by UUID.

sf_ble_uuid_length_t

`sf_ble_uuid_length_t`

Detailed description

UUID length

Enumerated values

Name	Description
SF_BLE_UUID_LENGTH_16BITS	16 bit UUID
SF_BLE_UUID_LENGTH_32BITS	32 bit UUID
SF_BLE_UUID_LENGTH_128BITS	128 bit UUID

sf_ble_char_read_t

`sf_ble_char_read_t`

Detailed description

Characteristics read type

Enumerated values

Name	Description
SF_BLE_CHAR_READ_BY_HANDLE	Read single characteristic by handle.
SF_BLE_CHAR_READ_BY_UUID	Read single characteristic by UUID.
SF_BLE_CHAR_READ_LONG_BY_HANDLE	Read single long characteristic by handle.

Name	Description
SF_BLE_CHAR_DESC_READ_BY_HANDLE	Read single characteristic descriptor by handle.
SF_BLE_CHAR_DESC_READ_LONG_BY_HANDLE	Read single long characteristic descriptor by handle.
SF_BLE_CHAR_READ_MULTIPLE_BY_HANDLES	Read multiple characteristics by handles.

sf_ble_char_write_t

`sf_ble_char_write_t`

Detailed description

Characteristics write type

Enumerated values

Name	Description
SF_BLE_CHAR_WRITE_NO_RESPONSE	Write single characteristic with no response.
SF_BLE_CHAR_WRITE	Write single characteristic with response.
SF_BLE_CHAR_WRITE_LONG	Write single long characteristic with response.
SF_BLE_CHAR_DESC_WRITE	Write single characteristic descriptor with response.
SF_BLE_CHAR_DESC_WRITE_LONG	Write single long characteristic descriptor with response.

sf_ble_execute_write_t

`sf_ble_execute_write_t`

Detailed description

Characteristic execute write flag

Enumerated values

Name	Description
SF_BLE_EXECUTE_WRITE_FALSE	Execute write false i.e. cancel write operations.
SF_BLE_EXECUTE_WRITE_TRUE	Execute write true i.e. execute write operations.

sf_ble_write_response_t

`sf_ble_write_response_t`

Detailed description

Characteristic execute write flag

Enumerated values

Name	Description
SF_BLE_WRITE_RESPONSE_NOT_REQUIRED	Write response not required.
SF_BLE_WRITE_RESPONSE_REQUIRED	Write response required.

sf_ble_attribute_error_code_t

`sf_ble_attribute_error_code_t`

Detailed description

BLE Attribute write response code

Enumerated values

Name	Description
SF_BLE_ATTRIBUTE_ERR_SUCCESS	Success.
SF_BLE_ATTRIBUTE_ERR_INVALID_HANDLE	Invalid handle.
SF_BLE_ATTRIBUTE_ERR_WRITE_NOT_PERMITTED	Writing is not permitted.
SF_BLE_ATTRIBUTE_ERR_INSUFF_AUTHEN	Authentication required for the request.
SF_BLE_ATTRIBUTE_ERR_REQUEST_NOT_SUPPORTED	Unsupported request.
SF_BLE_ATTRIBUTE_ERR_INVALID_OFFSET	Invalid offset.
SF_BLE_ATTRIBUTE_ERR_INSUFF_AUTHOR	Authorization required for the request.
SF_BLE_ATTRIBUTE_ERR_ATTRIBUTE_NOT_FOUND	The attribute could not be found.
SF_BLE_ATTRIBUTE_ERR_ATTRIBUTE_NOT_LONG	The attribute is not long enough.
SF_BLE_ATTRIBUTE_ERR_INVALID_ATTRIBUTE_VAL_LEN	Invalid attribute value size.
SF_BLE_ATTRIBUTE_ERR_INSUFF_ENC	Encryption required for the request.
SF_BLE_ATTRIBUTE_ERR_OUT_OF_RANGE	Out of Range.

sf_ble_char_attr_permissions_t

`sf_ble_char_attr_permissions_t`

Detailed description

BLE GATT Attribute characteristics permission

Enumerated values

Name	Description
SF_BLE_ATT_PERMISSIONS_ALLACCESS	Full access.
SF_BLE_ATT_PERMISSIONS_READ_AUTHENTICATION	Read with authentication.
SF_BLE_ATT_PERMISSIONS_READ_ENCRYPTION	Read with encryption.
SF_BLE_ATT_PERMISSIONS_READ_AUTHORIZATION	Read with authorization.
SF_BLE_ATT_PERMISSIONS_READ_FORBIDDEN	Read forbidden.
SF_BLE_ATT_PERMISSIONS_WRITE_AUTHENTICATION	Write with authentication.
SF_BLE_ATT_PERMISSIONS_WRITE_ENCRYPTION	Write with encryption.
SF_BLE_ATT_PERMISSIONS_WRITE_AUTHORIZATION	Write with authorization.
SF_BLE_ATT_PERMISSIONS_WRITE_FORBIDDEN	Write forbidden.
SF_BLE_ATT_PERMISSIONS_NOACCESS	No access.

sf_ble_callback_t

```
typedef void(* sf_ble_callback_t) (sf_ble_event_info_t *ev)
```

Detailed description

BLE Callback Type

sf_ble_conn_handle_t

```
typedef uint16_t sf_ble_conn_handle_t
```

Detailed description

BLE Connection Handle

7.6.1.8 API Structures

sf_ble_addr_t

sf_ble_addr_t

Detailed description

BLE address

Variables

- uint8_t addr[SF_BLE_ADDR_LEN]

  6-byte array address value

sf_ble_adv_data_t

sf_ble_adv_data_t

Detailed description

BLE Advertising data structure

Variables

- uint8_t data[SF_BLE_ADV_DATA_LEN]

  Advertising data bytes array.

- uint8_t adv_data_length

  Advertising data length.

sf_ble_event_info_t

sf_ble_event_info_t

Detailed description

BLE event info

Variables

- void * p_data

  Data for BLE event.

- uint16_t event

  Event type.

sf_ble_sec_info_t

sf_ble_sec_info_t

Detailed description

Security Related Configuration

Variables

- sf_ble_sec_mode_t sec_mode

  security mode

- uint8_t id_key[SF_BLE_SEC_KEY_LEN]

  GAP Identity Resolving Security key.

- uint8_t csrk_key[SF_BLE_SEC_KEY_LEN]

  GAP Connection Signature Resolving Security key.

- uint8_t ltk_key[SF_BLE_SEC_KEY_LEN]

  GAP Connection Long term Security key.

- uint8_t rand_num[SF_BLE_RAND_NUM_LENGTH]

  GAP Connection Random number for Long term Security key.

- uint16_t ediv

  GAP Connection Encrypted Diversifier for Long term Security key.

sf_ble_provisioning_t

sf_ble_provisioning_t

Detailed description

BLE provisioning information

Variables

- uint8_t gap_name[SF_BLE_MAX_GAP_NAME_LEN]

  GAP name.

- uint8_t bcast_mode

  broadcast mode

- sf_ble_bonding_mode_t bonding_mode

  bonding mode

- sf_ble_sec_info_t sec_info

  Security information.

- sf_ble_gap_role_t gap_role

  GAP role (master/slave)

- sf_ble_callback_t p_ble_callback

  GAP user event callback.

sf_ble_scan_info_t

sf_ble_scan_info_t

Detailed description

BLE scan information structure

Variables

- uint16_t total_scan_duration

  BLE total scan duration in milliseconds.

- sf_ble_scan_mode_t scan_mode

  BLE scan mode.

- sf_ble_disc_type_t discovery_type

  Used only in active scan for specifying discovery type.

- sf_ble_addr_type_t address_type

  BLE address type.

- sf_ble_adv_filt_type_t filt_policy

  Scan Filter policy.

sf_ble_bonding_start_t

sf_ble_bonding_start_t

Detailed description

BLE Bonding Start information

Variables

- sf_ble_iocap_t iocap

  IO capabilities.

- uint8_t key_size

  Encryption key size.

- sf_ble_key_dist_t ikey_dist

  Initiator key distribution.

- sf_ble_key_dist_t rkey_dist

  Responder key distribution.

sf_ble_bonding_response_t

sf_ble_bonding_response_t

Detailed description

Bonding Response Parameter

Variables

- uint8_t accept

  accept or reject bonding

- sf_ble_iocap_t io_cap

  IO capabilities.

- uint8_t max_key_size

  Max key size.

- sf_ble_key_dist_t ikeys

  Initiator key distribution.

- sf_ble_key_dist_t rkeys

  Responder key distribution.

sf_ble_sm_tk_info_t

sf_ble_sm_tk_info_t

Detailed description

Security structures BLE Temporary Key information

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.
- uint8_t disp_en

  Whether to enable display.

sf_ble_sm_tk_ind_t

sf_ble_sm_tk_ind_t

Detailed description

BLE Temporary Key indication

Variables

- sf_ble_sm_tk_info_t tk_info

  input parameter,information to app about temporary key request
- uint8_t tk_req_status

  output parameter: request status, application has to set this in callback event whether request is valid or not
- uint8_t tk_key[SF_BLE_SEC_KEY_LEN]

  application has to set this in callback event temporary key for encryption

sf_ble_addr_verify_ind_t

sf_ble_addr_verify_ind_t

Detailed description

BLE Bluetooth address verification indication

Variables

- uint8_t bd_addr[SF_BLE_ADDR_LEN]

  input parameter specifying bluetooth address of remote device
- sf_ble_addr_type_t addr_type

  input parameter specifying Address type of remote BLE device
- uint8_t accept_addr

  output parameter: application has to set this parameter in callback if it accepts this address

sf_ble_sm_key_ind_t

sf_ble_sm_key_ind_t

Detailed description

BLE Received Key information

Variables

- uint8_t conn_idx

  connection index

- sf_ble_key_dist_t key_code

  type of security key
- uint16_t ediv

  BLE Security EDIV.
- uint8_t rand_num[SF_BLE_RAND_NUM_LENGTH]

  Random number for security.
- uint8_t ltk_key[SF_BLE_SEC_KEY_LEN]

  BLE long term security key.

sf_ble_sm_enc_info_t

sf_ble_sm_enc_info_t

Detailed description

BLE Start Encryption information

Variables

- sf_ble_conn_handle_t conn_idx

  connection index
- uint16_t ediv

  BLE Security EDIV.
- uint8_t rand_num[SF_BLE_RAND_NUM_LENGTH]

  Random number for security.
- uint8_t ltk_key[SF_BLE_SEC_KEY_LEN]

  BLE long term security key.

sf_ble_sec_enc_start_ind_t

sf_ble_sec_enc_start_ind_t

Detailed description

Encryption Start Indication Event for user

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle of the remote device.
- uint8_t status

  Result of encryption start, 0 = SUCCESS else error.
- uint8_t key_size

  Key Size.
- sf_ble_auth_type_t auth_type

  Security properties.

- uint8_t bonding_status

  Bonding Status, 0 = Unbonded, 1 = Bonded.

sf_ble_adv_info_t

sf_ble_adv_info_t

Detailed description

BLE advertisement information

Variables

- sf_ble_disc_type_t disc_mode

  Discovery mode.

- sf_ble_conn_type_t conn_mode

  Connection mode.

- uint16_t adv_intv_min

  Minimum interval for advertising.

- uint16_t adv_intv_max

  Maximum interval for advertising.

- sf_ble_addr_type_t own_addr_type

  Own address type: public=0x00 /random = 0x01.

- sf_ble_adv_chnl_map_t adv_chnl_map

  Advertising channel map.

- sf_ble_adv_filt_type_t adv_filt_policy

  Advertising filter policy.

- sf_ble_adv_data_t adv_data

  Advertising data.

sf_ble_scan_t

sf_ble_scan_t

Detailed description

BLE scan information

Variables

- sf_ble_addr_type_t addr_type

  Address type of remote BLE device.

- uint8_t bd_addr[SF_BLE_ADDR_LEN]

  Remote BLE address.

- uint16_t data_len

  Scan data length.

- uint8_t data[SF_BLE_MAX_BLE_ADV_DATA_LEN]

  Scan data.
- int16_t rssi

  RSSI value.

sf_ble_connection_t

sf_ble_connection_t

Detailed description

BLE connection information

Variables

- sf_ble_init_filt_type_t init_filt_type

  Connection filter type.
- sf_ble_addr_type_t addr_type

  BLE address type.
- uint8_t bd_addr[SF_BLE_ADDR_LEN]

  BLE address.

sf_ble_disconnect_t

sf_ble_disconnect_t

Detailed description

BLE Disconnect Event Information

Variables

- sf_ble_disconnect_reason_t reason

  Disconnection reason.
- uint8_t status

  Disconnect status if initiated by local host.
- sf_ble_conn_handle_t conhdl

  Connection handle of the remote device.

sf_ble_chipset_info_t

sf_ble_chipset_info_t

Detailed description

BLE chipset information

Variables

- uint32_t version

  chipset version

- uint8_t bd_addr[SF_BLE_ADDR_LEN]

  BLE address.

sf_ble_info_t

sf_ble_info_t

Detailed description

BLE module information

Variables

- sf_ble_chipset_info_t chipset

  Chipset information.

- uint16_t rssi

  RSSI value.

sf_ble_bonding_req_ind_t

sf_ble_bonding_req_ind_t

Detailed description

BLE bonding request indication information

Variables

- sf_ble_addr_t bd_addr

  BLE address.

- uint8_t index

  Connection index.

- uint8_t auth_req

  Authentication request type.

- uint8_t io_cap

  IO capability.

- uint8_t oob_data_flg

  Indicating if OOB data is present.

- uint8_t max_enc_size

  Maximum encryption key size.

- uint8_t ikey_dist

  Type of key distributed by the initiator.

- uint8_t rkey_dist

  Type of key distributed by the responder.

sf_ble_connect_info_t

sf_ble_connect_info_t

Detailed description

BLE connection information

Variables

- uint8_t status

  Confirmation status.

- uint8_t reserved

  Reserved.

- sf_ble_conn_handle_t conhdl

  Connection handle.

- uint8_t peer_addr_type

  Peer address type.

- sf_ble_addr_t peer_addr

  Peer BT address.

- uint8_t reserved2

  Reserved.

- uint16_t con_interval

  Connection interval.

- uint16_t con_latency

  Connection latency.

- uint16_t sup_to

  Link supervision time-out.

- uint8_t clk_accuracy

  Clock accuracy.

- uint8_t reserved3

  Reserved.

sf_ble_uuid_t

sf_ble_uuid_t

Detailed description

Union holding either 16-bit/32-bit/128-bit UUID

Variables

- uint16_t uuid16

  16 bit UUID

- uint32_t uuid32

  32 bit UUID

- uint8_t uuid128[SF_BLE_128BITS_UUID_LENGTH]

  128 bit UUID
- union{} union{}

  See source code for definition of this member.
- sf_ble_uuid_length_t uuid_length

  Service UUID length.

sf_ble_svc_attribute_t

sf_ble_svc_attribute_t

Detailed description

BLE GATT Service Attributes

Variables

- uint16_t attr_handle

  Service Handle.
- uint16_t attr_type

  Attribute type such as SF_BLE_GATT_PRI_SERVICE, SF_BLE_GATT_INCLUDE_SERVICE.
- uint16_t parent_svc_handle

  parent_svc_handle is service handle of parent service which is already registered

  If service is included service,
- uint8_t * p_attr_value

  Service UUID Pointer.
- sf_ble_uuid_length_t attr_value_len

  UUID Length.

sf_ble_char_attribute_t

sf_ble_char_attribute_t

Detailed description

BLE GATT Characteristics Attributes

Variables

- sf_ble_svc_attribute_t * p_service

  Service to which this characteristics belongs.
- sf_ble_uuid_t attr_uuid

  UUID of characteristics value.
- uint16_t attr_declare_handle

  Characteristics handle.

- uint16_t attr_declare_type

  Characteristics declare type SF_BLE_GATT_CHAR_DECLARE.
- uint16_t attr_value_handle

  Characteristics value handle.
- sf_ble_char_attr_permissions_t attr_perm

  Characteristics permission.
- sf_ble_char_property_t attr_properties

  Characteristics properties.
- uint8_t * p_attr_value

  Characteristics value data.
- uint8_t attr_value_len

  Characteristics value data length.

sf_ble_gatt_attr_event_t

sf_ble_gatt_attr_event_t

Detailed description

BLE GATT Characteristics Remote Event data, passed in provision callback for GATT Custom profile

Variables

- uint16_t attr_handle

  Attribute handle for which event is generated.
- uint16_t offset

  The offset at which the client is willing to write.
- uint8_t const * p_value

  The value at which the client is willing to write.
- uint16_t length

  The amount of bytes that the client is willing to write as from this offset.
- ssp_err_t response

  User will update this variable, Pass SSP_SUCCESS if user accepts the remote request.

sf_ble_service_discovery_rsp_t

sf_ble_service_discovery_rsp_t

Detailed description

Service discovery result

Variables

- uint16_t service_handle

  Service handle.

- sf_ble_uuid_t uuid

  Service UUID.
- uint16_t start_handle

  Start handle of sub-attributes handle range.
- uint16_t end_handle

  End handle of sub-attributes handle range.

sf_ble_service_discovery_req_t

sf_ble_service_discovery_req_t

Detailed description

Service discovery request

Variables

- sf_ble_uuid_t uuid

  Service UUID.
- uint16_t start_handle

  Discovery start handle.
- uint16_t end_handle

  Discovery end handle.
- sf_ble_service_discovery_t discovery_type

  Service discovery type.

sf_ble_char_discovery_req_t

sf_ble_char_discovery_req_t

Detailed description

Characteristic discovery request

Variables

- sf_ble_uuid_t uuid

  Characteristic UUID.
- uint16_t start_handle

  Discovery start handle.
- uint16_t end_handle

  Discovery end handle.
- sf_ble_char_discovery_t discovery_type

  Characteristic discovery type.

sf_ble_char_discovery_rsp_t

sf_ble_char_discovery_rsp_t

Detailed description

Characteristic discovery result

Variables

- uint16_t char_handle

  Characteristic handle.

- sf_ble_uuid_t uuid

  Characteristic UUID.

- uint16_t value_handle

  Characteristic value handle.

- sf_ble_char_property_t property

  Characteristic property.

sf_ble_char_desc_discovery_rsp_t

sf_ble_char_desc_discovery_rsp_t

Detailed description

Characteristic descriptor discovery result

Variables

- uint16_t desc_handle

  Characteristic descriptor handle.

- sf_ble_uuid_t uuid

  Characteristic descriptor UUID.

sf_ble_char_multiple_read_req_t

sf_ble_char_multiple_read_req_t

Detailed description

Multiple Characteristic descriptor read request

Variables

- uint16_t handles[SF_BLE_MAX_MULTI_CHAR_READ_CNT]

  Characteristic handles for multiple read.

- uint8_t expected_result_size[SF_BLE_MAX_MULTI_CHAR_READ_CNT]

  Expected result length in bytes.

sf_ble_char_read_req_t

sf_ble_char_read_req_t

Detailed description

Read Characteristic request

Variables

- sf_ble_char_read_t char_read_type

  Characteristic read type.
- uint16_t handle

  Characteristic value or descriptor handle.
- sf_ble_uuid_t uuid

  Characteristic UUID.
- uint16_t offset

  Offset for long Characteristic value.
- sf_ble_char_multiple_read_req_t * p_mul_read_req

  Pointer to multiple read Characteristic request.
- uint16_t handles_cnt

  Characteristic handles count for multiple read.

sf_ble_char_multiple_read_rsp_t

sf_ble_char_multiple_read_rsp_t

Detailed description

Read multiple Characteristic response

Variables

- uint8_t * p_data[SF_BLE_MAX_MULTI_CHAR_READ_CNT]

  Pointer to Characteristic value.
- uint16_t data_len[SF_BLE_MAX_MULTI_CHAR_READ_CNT]

  Characteristic value length.

sf_ble_char_read_by_uuid_rsp_t

sf_ble_char_read_by_uuid_rsp_t

Detailed description

Read Characteristic by UUID response

Variables

- uint16_t handle[SF_BLE_MAX_CHAR_UUID_READ_CNT]

  Characteristic handles.
- uint8_t * p_data[SF_BLE_MAX_CHAR_UUID_READ_CNT]

  Pointer to Characteristic value.
- uint16_t data_len[SF_BLE_MAX_CHAR_UUID_READ_CNT]

  Characteristic value length.

sf_ble_char_read_by_handle_rsp_t

sf_ble_char_read_by_handle_rsp_t

Detailed description

Read Characteristic by handle response

Variables

- uint8_t * p_data

  Pointer to Characteristic value.

- uint16_t data_len

  Characteristic value length.

sf_ble_char_read_rsp_t

Detailed description

Read Characteristic response

Variables

p_char_read_by_handle_rsp

p_char_read_by_uuid_rsp

p_char_multiple_read_rsp

p_char_read_by_handle_rsp

`sf_ble_char_read_by_handle_rsp_t::p_char_read_by_handle_rsp`

Brief description

Response for read Characteristic by handle.

p_char_read_by_uuid_rsp

`sf_ble_char_read_by_uuid_rsp_t::p_char_read_by_uuid_rsp`

Brief description

Response for read Characteristic by UUID.

p_char_multiple_read_rsp

`sf_ble_char_multiple_read_rsp_t::p_char_multiple_read_rsp`

Brief description

Response for read multiple Characteristics.

sf_ble_char_write_req_t

sf_ble_char_write_req_t

Detailed description

Write Characteristic request

Variables

- sf_ble_char_write_t char_write_type

  Characteristic write type.

- uint16_t handle

  Characteristic value or descriptor handle.

- uint8_t * p_data

  Pointer to data.
- uint16_t offset

  Offset for long Characteristic value.
- uint16_t data_length

  Data length.
- sf_ble_execute_write_t auto_execute

  Automatic execute write flag.

sf_ble_gatt_notif_ind_event_data_t

sf_ble_gatt_notif_ind_event_data_t

Detailed description

Notification/Indication event data

Variables

- uint16_t handle

  Characteristic value or descriptor handle.
- uint8_t * p_data

  Pointer to data.
- uint16_t data_length

  Data length.

sf_ble_write_cmd_event_data_t

sf_ble_write_cmd_event_data_t

Detailed description

Write command event data

Variables

- uint16_t handle

  Characteristic value or descriptor handle.
- uint16_t offset

  Offset for long Characteristic value.
- sf_ble_write_response_t response_required

  Write response required or not.
- uint8_t * p_data

  Pointer to data.
- uint16_t data_length

  Data length.

sf_ble_ctrl_t

sf_ble_ctrl_t

Detailed description

BLE Framework control structure

Variables

- void * p_driver_handle

  Storage for information needed for each BLE device driver in the system.

sf_ble_cfg_t

sf_ble_cfg_t

Detailed description

BLE configuration information

Variables

- uint8_t bd_addr[SF_BLE_ADDR_LEN]

  BLE address.

- sf_ble_addr_type_t own_addr_type

  self address type

- uint8_t max_slaves

  Maximum slaves allowed to be connected.

- uint8_t update_bd_addr

  Set this to true to update bluetooth address during SF_BLE_Open.

- uint16_t scan_interval

  BLE scan interval for receiving advertisement.

- uint16_t scan_window

  Period of time during which advertising data is received at the scan interval.

- uint16_t disc_time

  Duration for which the device remain discoverable.

- uint16_t con_interval

  Interval for transmitting and receiving data periodically after connection establishment.

- uint16_t slave_latency

  Period of time during which data is transmitted and received at the connection interval.

- uint16_t sup_timeout

  Link loss time-out.

- void const * p_extend

  Instance specific configuration.

sf_ble_api_t

sf_ble_api_t

Detailed description

Framework API structure. Implementations will use the following API.

open

```
ssp_err_t(* sf_ble_api_t::open) (sf_ble_ctrl_t *const p_ctrl, const sf_ble_cfg_t
*p_cfg)
```

Brief description

Initializes the interface for data transfers.

Detailed description

Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Table 27: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to BLE configuration structure.

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_cfg

Definition: `sf_ble_cfg_t sf_ble_cfg_t *p_cfg`

BLE configuration information

- `sf_ble_cfg_t`::`bd_addr`

  BLE address.

- `sf_ble_cfg_t`::`sf_ble_addr_type_t`

  self address type

  Enumerated as:

  – SF_BLE_ADDR_TYPE_PUBLIC

  – SF_BLE_ADDR_TYPE_RANDOM

- `sf_ble_cfg_t`::`max_slaves`

  Maximum slaves allowed to be connected.

- `sf_ble_cfg_t`::`update_bd_addr`

  Set this to true to update bluetooth address during SF_BLE_Open.

- sf_ble_cfg_t::scan_interval

  BLE scan interval for receiving advertisement.
- sf_ble_cfg_t::scan_window

  Period of time during which advertising data is received at the scan interval.
- sf_ble_cfg_t::disc_time

  Duration for which the device remain discoverable.
- sf_ble_cfg_t::con_interval

  Interval for transmitting and receiving data periodically after connection establishment.
- sf_ble_cfg_t::slave_latency

  Period of time during which data is transmitted and received at the connection interval.
- sf_ble_cfg_t::sup_timeout

  Link loss time-out.
- sf_ble_cfg_t::p_extend

  Instance specific configuration.

close

```
ssp_err_t(* sf_ble_api_t::close) (sf_ble_ctrl_t *const p_ctrl)
```

Brief description

De-initialize the interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Detailed description

Table 28: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

infoGet

```
ssp_err_t(* sf_ble_api_t::infoGet) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_info_t *p_ble_info)
```

Brief description

Get BLE module information like chipset information and RSSI value.

Detailed description

Table 29: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_handle	in	Pointer to connection handle
p_ble_info	out	Pointer to module information

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_ble_info

Definition: `sf_ble_info_t*p_ble_info`

BLE module information

- `sf_ble_info_t::chipset`

  Chipset information.

- `sf_ble_info_t::rssi`

  RSSI value.

provisionGet

```
ssp_err_t(* sf_ble_api_t::provisionGet) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_provisioning_t *p_ble_provisioning)
```

Brief description

Reads the current BLE Provisioning information.

Detailed description

Table 30: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_ble_provisioning	out	current provisioning information

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

Parameter p_ble_provisioning

Definition: sf_ble_provisioning_t*p_ble_provisioning

BLE provisioning information

- sf_ble_provisioning_t::gap_name

  GAP name.

- sf_ble_provisioning_t::bcast_mode

  broadcast mode

- sf_ble_provisioning_t::bonding_mode

  bonding mode

- sf_ble_provisioning_t::sec_info

  Security information.

- sf_ble_provisioning_t::gap_role

  GAP role (master/slave)

- sf_ble_provisioning_t::p_ble_callback

  GAP user event callback.

provisionSet

```
ssp_err_t(* sf_ble_api_t::provisionSet) (sf_ble_ctrl_t *const p_ctrl, const
sf_ble_provisioning_t *p_ble_provisioning)
```

Brief description

Provisions the BLE Driver and set bonding and security modes as provisioned.

Detailed description

Table 31: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_ble_provisioning	in	Pointer to BLE provisioning structure

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

Parameter p_ble_provisioning

Definition: sf_ble_provisioning_tsf_ble_provisioning_t *p_ble_provisioning

BLE provisioning information

- sf_ble_provisioning_t::gap_name

  GAP name.

- sf_ble_provisioning_t::bcast_mode

  broadcast mode

- sf_ble_provisioning_t::bonding_mode

  bonding mode

- sf_ble_provisioning_t::sec_info

  Security information.

- sf_ble_provisioning_t::gap_role

  GAP role (master/slave)

- sf_ble_provisioning_t::p_ble_callback

  GAP user event callback.

scan

```
ssp_err_t(* sf_ble_api_t::scan) (sf_ble_ctrl_t *const p_ctrl, sf_ble_scan_t
*p_scan, uint8_t *p_cnt, sf_ble_scan_info_t *p_scan_info)
```

Brief description

Scans for available BLE devices and return the list to the caller.

Detailed description

Table 32: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_scan	out	Pointer to scan information structure
p_cnt	inout	Pointer to number of BLE devices scanned
p_scan_info	in	Pointer to scan information structure

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

Parameter p_scan

Definition: sf_ble_scan_t*p_scan

BLE scan information

- sf_ble_scan_t::addr_type

  Address type of remote BLE device.

- sf_ble_scan_t::bd_addr

  Remote BLE address.

- sf_ble_scan_t::data_len

  Scan data length.

- sf_ble_scan_t::data

  Scan data.

- sf_ble_scan_t::rssi

  RSSI value.

Parameter p_cnt

uint8_t

Parameter p_scan_info

Definition: sf_ble_scan_info_t*p_scan_info

BLE scan information structure

- sf_ble_scan_info_t::total_scan_duration

  BLE total scan duration in milliseconds.

- sf_ble_scan_info_t::scan_mode

  BLE scan mode.

- sf_ble_scan_info_t::discovery_type

  Used only in active scan for specifying discovery type.

- sf_ble_scan_info_t::address_type

  BLE address type.

- sf_ble_scan_info_t::filt_policy

  Scan Filter policy.

advertisementStart

```
ssp_err_t(* sf_ble_api_t::advertisementStart) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_adv_info_t *const p_advt_info)
```

Brief description

Make the device discoverable by broadcasting device information to remote devices.

Detailed description

Table 33: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_advt_info	in	Pointer to advertisement information structure

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_advt_info

Definition: `sf_ble_adv_info_t*const p_advt_info`

BLE advertisement information

- `sf_ble_adv_info_t::disc_mode`

  Discovery mode.
- `sf_ble_adv_info_t::conn_mode`

  Connection mode.
- `sf_ble_adv_info_t::adv_intv_min`

  Minimum interval for advertising.
- `sf_ble_adv_info_t::adv_intv_max`

  Maximum interval for advertising.
- `sf_ble_adv_info_t::own_addr_type`

  Own address type: public=0x00 /random = 0x01.
- `sf_ble_adv_info_t::adv_chnl_map`

  Advertising channel map.
- `sf_ble_adv_info_t::adv_filt_policy`

  Advertising filter policy.
- `sf_ble_adv_info_t::adv_data`

  Advertising data.

advertisementStop

```
ssp_err_t(* sf_ble_api_t::advertisementStop) (sf_ble_ctrl_t *const p_ctrl)
```

Brief description

Stop the device from being discoverable.

Detailed description

Table 34: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

whitelistAdd

```
ssp_err_t(* sf_ble_api_t::whitelistAdd) (sf_ble_ctrl_t *const p_ctrl, const uint8_t *p_bd_addr)
```

Brief description

Add specified devices to whitelist.

Detailed description

Table 35: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_bd_addr	in	Pointer to BLE address

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_bd_addr

`uint8_t`

whitelistDel

```
ssp_err_t(* sf_ble_api_t::whitelistDel) (sf_ble_ctrl_t *const p_ctrl, const uint8_t *p_bd_addr)
```

Brief description

Remove specified devices from whitelist.

Detailed description

Table 36: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_bd_addr	in	Pointer to BLE address

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_bd_addr

`uint8_t`

bondingStart

```
ssp_err_t(* sf_ble_api_t::bondingStart) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, const uint8_t *p_bd_addr, sf_ble_bonding_start_t
*p_bonding_start)
```

Brief description

Initiate bonding process with remote BLE device and exchange security keys if enabled.

Detailed description

Table 37: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_handle	in	Pointer to connection handle.
p_bd_addr	in	Pointer to BLE address

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_bd_addr

`uint8_t`

bondingResponse

```
ssp_err_t(* sf_ble_api_t::bondingResponse) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, const uint8_t *p_bd_addr,
sf_ble_bonding_response_t *p_bonding_resp)
```

Brief description

Respond to the bonding request from the remote BLE device. Send bonding response on reception of bonding indication.

Detailed description

Table 38: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_handle	in	Pointer to connection handle
p_bd_addr	in	Pointer to BLE address
p_bonding_resp	in	Pointer to Bonding address

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_bd_addr

`uint8_t`

Parameter p_bonding_resp

Definition: `sf_ble_bonding_response_t*p_bonding_resp`

Bonding Response Parameter

- `sf_ble_bonding_response_t::accept`

  accept or reject bonding

- `sf_ble_bonding_response_t::io_cap`

  IO capabilities.

- `sf_ble_bonding_response_t::max_key_size`

  Max key size.

- `sf_ble_bonding_response_t::ikeys`

  Initiator key distribution.

- sf_ble_bonding_response_t::rkeys

  Responder key distribution.

startEncryption

```
ssp_err_t(* sf_ble_api_t::startEncryption) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_sm_enc_info_t const *p_enc_info)
```

Brief description

Start encryption over a connection.

Detailed description

Table 39: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_enc_info	in	information for starting encryption

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

Parameter p_enc_info

Definition: sf_ble_sm_enc_info_tconst *p_enc_info

BLE Start Encryption information

- sf_ble_sm_enc_info_t::conn_idx

  connection index

- sf_ble_sm_enc_info_t::ediv

  BLE Security EDIV.

- sf_ble_sm_enc_info_t::rand_num

  Random number for security.

- sf_ble_sm_enc_info_t::ltk_key

  BLE long term security key.

connect

```
ssp_err_t(* sf_ble_api_t::connect) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_connection_t const *const p_conn, sf_ble_conn_handle_t *p_handle)
```

Brief description

Connect to a remote BLE device.

Detailed description

Table 40: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_conn	in	Pointer to connection information
p_handle	out	Pointer to connection handle

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_conn

Definition: `sf_ble_connection_tconst *const p_conn`

BLE connection information

- `sf_ble_connection_t::init_filt_type`

  Connection filter type.

- `sf_ble_connection_t::addr_type`

  BLE address type.

- `sf_ble_connection_t::bd_addr`

  BLE address.

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

listen

```
ssp_err_t(* sf_ble_api_t::listen) (sf_ble_ctrl_t *const p_ctrl)
```

Brief description

Listen for connection request from remote device.

Detailed description

Table 41: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

disconnect

```
ssp_err_t(* sf_ble_api_t::disconnect) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle)
```

Brief description

Terminate connection with remote BLE device.

Detailed description

Table 42: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.
p_handle	out	Pointer to connection handle

Parameter p_ctrl

Definition: sf_ble_ctrl_t

BLE Framework control structure

Parameter p_handle

Definition: sf_ble_conn_handle_t*p_handle

BLE Connection Handle

gattAddCustomProfiles

```
ssp_err_t(* sf_ble_api_t::gattAddCustomProfiles) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_svc_attribute_t *p_svc_attr, uint32_t svc_attr_len,
sf_ble_char_attribute_t *p_char_attr, uint32_t char_attr_len)
```

Brief description

Add Custom Profile to GATT Database.

Detailed description

This function is called with list of service and characteristics which is to be added to GATT database. Services and characteristics which were previously added will be removed and newly passed services and characteristics will be added.

If services and characteristics were previously added and now those services and characteristics are to be removed and no new services and characteristics are to be added , then call this API with service and characteristics length as zero

Table 43: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_svc_attr	in	List of Services to add
svc_attr_len	in	No of elements in services list
p_char_attr	in	List of Characteristics to add
char_attr_len	in	No of elements in Characteristics list

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_svc_attr

Definition: `sf_ble_svc_attribute_t*p_svc_attr`

BLE GATT Service Attributes

- `sf_ble_svc_attribute_t::attr_handle`

  Service Handle.
- `sf_ble_svc_attribute_t::attr_type`

  Attribute type such as SF_BLE_GATT_PRI_SERVICE, SF_BLE_GATT_INCLUDE_SERVICE.
- `sf_ble_svc_attribute_t::parent_svc_handle`

  parent_svc_handle is service handle of parent service which is already registered
- `sf_ble_svc_attribute_t::p_attr_value`

  Service UUID Pointer.
- `sf_ble_svc_attribute_t::attr_value_len`

  UUID Length.

Parameter svc_attr_len

`uint32_t`

Parameter p_char_attr

Definition: `sf_ble_char_attribute_t*p_char_attr`

BLE GATT Characteristics Attributes

- `sf_ble_char_attribute_t::p_service`

  Service to which this characteristics belongs.

- sf_ble_char_attribute_t::attr_uuid

  UUID of characteristics value.
- sf_ble_char_attribute_t::attr_declare_handle

  Characteristics handle.
- sf_ble_char_attribute_t::attr_declare_type

  Characteristics declare type SF_BLE_GATT_CHAR_DECLARE.
- sf_ble_char_attribute_t::attr_value_handle

  Characteristics value handle.
- sf_ble_char_attribute_t::attr_perm

  Characteristics permission.
- sf_ble_char_attribute_t::attr_properties

  Characteristics properties.
- sf_ble_char_attribute_t::p_attr_value

  Characteristics value data.
- sf_ble_char_attribute_t::attr_value_len

  Characteristics value data length.

Parameter char_attr_len

`uint32_t`

gattServiceDiscovery

```
ssp_err_t(* sf_ble_api_t::gattServiceDiscovery) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_service_discovery_req_t const *const
p_sf_ble_svc_dscv_req, sf_ble_service_discovery_rsp_t *const
p_sf_ble_svc_dscv_rsp, uint32_t *const p_rsp_cnt)
```

Brief description

Perform service discovery used by GATT client.Perform service discovery on remote GATT server and get results.

Detailed description

Table 44: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
p_sf_ble_svc_dscv_req	in	Pointer to service discovery request
p_sf_ble_svc_dscv_rsp	out	Pointer to service discovery response

Table 44: Parameters (Continued)

Name	Direction	Description
p_rsp_cnt	inout	Input Size specifying maximum number of service discovery results which can be stored in response, output specifying number of service discovery results stored in response

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_sf_ble_svc_dscv_req

Definition: `sf_ble_service_discovery_req_tconst *const p_sf_ble_svc_dscv_req`

Service discovery request

- `sf_ble_service_discovery_req_t::uuid`

  Service UUID.
- `sf_ble_service_discovery_req_t::start_handle`

  Discovery start handle.
- `sf_ble_service_discovery_req_t::end_handle`

  Discovery end handle.
- `sf_ble_service_discovery_req_t::discovery_type`

  Service discovery type.

Parameter p_sf_ble_svc_dscv_rsp

Definition: `sf_ble_service_discovery_rsp_t*const p_sf_ble_svc_dscv_rsp`

Service discovery result

- `sf_ble_service_discovery_rsp_t::service_handle`

  Service handle.
- `sf_ble_service_discovery_rsp_t::uuid`

  Service UUID.
- `sf_ble_service_discovery_rsp_t::start_handle`

  Start handle of sub-attributes handle range.
- `sf_ble_service_discovery_rsp_t::end_handle`

  End handle of sub-attributes handle range.

Parameter p_rsp_cnt

`uint32_t`

gattCharDiscovery

```
ssp_err_t(* sf_ble_api_t::gattCharDiscovery) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_char_discovery_req_t const *const
p_sf_ble_char_dscv_req, sf_ble_char_discovery_rsp_t *const
p_sf_ble_char_dscv_rsp, uint32_t *const p_rsp_cnt)
```

Brief description

Perform characteristics discovery used by GATT client.Perform characteristics discovery on remote GATT server and get results.

Detailed description

Table 45: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
p_sf_ble_char_dscv_req	in	Pointer to characteristics discovery request
p_sf_ble_char_dscv_rsp	out	Pointer to characteristics discovery response
p_rsp_cnt	inout	Input Size specifying maximum number of characteristics discovery results which can be stored in response, output specifying number of characteristics discovery results stored in response

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_sf_ble_char_dscv_req

Definition: `sf_ble_char_discovery_req_tconst *const p_sf_ble_char_dscv_req`

Characteristic discovery request

- `sf_ble_char_discovery_req_t::uuid`

  Characteristic UUID.

- sf_ble_char_discovery_req_t::start_handle

  Discovery start handle.
- sf_ble_char_discovery_req_t::end_handle

  Discovery end handle.
- sf_ble_char_discovery_req_t::discovery_type

  Characteristic discovery type.

Parameter p_sf_ble_char_dscv_rsp

Definition: sf_ble_char_discovery_rsp_t*const p_sf_ble_char_dscv_rsp

Characteristic discovery result

- sf_ble_char_discovery_rsp_t::char_handle

  Characteristic handle.
- sf_ble_char_discovery_rsp_t::uuid

  Characteristic UUID.
- sf_ble_char_discovery_rsp_t::value_handle

  Characteristic value handle.
- sf_ble_char_discovery_rsp_t::property

  Characteristic property.

Parameter p_rsp_cnt

uint32_t

gattCharDescDiscovery

```
ssp_err_t(* sf_ble_api_t::gattCharDescDiscovery) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t start_handle, uint16_t end_handle,
sf_ble_char_desc_discovery_rsp_t *const p_sf_ble_chardesc_dscv_rsp, uint32_t
*const p_rsp_cnt)
```

Brief description

Perform characteristics descriptor discovery used by GATT client.

Detailed description

Table 46: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
start_handle	in	Start handle from set of handle ranges to be used in discovery

Table 46: Parameters (Continued)

Name	Direction	Description
end_handle	in	End handle from set of handle ranges to be used in discovery
p_sf_ble_chardesc_dscv_rsp	out	Pointer to characteristics descriptor discovery response
p_rsp_cnt	inout	Input Size specifying maximum number of characteristics descriptor discovery results which can be stored in response, output specifying number of characteristics descriptor discovery results stored in response

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter start_handle

`uint16_t`

Parameter end_handle

`uint16_t`

Parameter p_sf_ble_chardesc_dscv_rsp

Definition: `sf_ble_char_desc_discovery_rsp_t*const p_sf_ble_chardesc_dscv_rsp`

Characteristic descriptor discovery result

- `sf_ble_char_desc_discovery_rsp_t::desc_handle`

  Characteristic descriptor handle.

- `sf_ble_char_desc_discovery_rsp_t::uuid`

  Characteristic descriptor UUID.

Parameter p_rsp_cnt

`uint32_t`

gattCharRead

```
ssp_err_t(* sf_ble_api_t::gattCharRead) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_char_read_req_t const *const
p_char_read_req, sf_ble_char_read_rsp_t *const p_char_read_rsp)
```

Brief description

Perform read characteristic used by GATT client.Read characteristic value from remote GATT server.

Detailed description

Table 47: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
p_char_read_req	in	Pointer to characteristic read request
p_char_read_rsp	out	Pointer to characteristic read response

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_char_read_req

Definition: `sf_ble_char_read_req_tconst *const p_char_read_req`

Read Characteristic request

- `sf_ble_char_read_req_t::char_read_type`
  Characteristic read type.
- `sf_ble_char_read_req_t::handle`
  Characteristic value or descriptor handle.
- `sf_ble_char_read_req_t::uuid`
  Characteristic UUID.
- `sf_ble_char_read_req_t::offset`
  Offset for long Characteristic value.
- `sf_ble_char_read_req_t::p_mul_read_req`
  Pointer to multiple read Characteristic request.
- `sf_ble_char_read_req_t::handles_cnt`
  Characteristic handles count for multiple read.

Parameter p_char_read_rsp

Definition: `sf_ble_char_read_rsp_t*const p_char_read_rsp`

gattCharWrite

```
ssp_err_t(* sf_ble_api_t::gattCharWrite) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_char_write_req_t const *const
p_char_write_req)
```

Brief description

Perform write characteristic used by GATT client.Write characteristic value on remote GATT server.

Detailed description

Table 48: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
p_char_write_req	in	Pointer to characteristic write request

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter p_char_write_req

Definition: `sf_ble_char_write_req_tconst *const p_char_write_req`

Write Characteristic request

- `sf_ble_char_write_req_t::char_write_type`

  Characteristic write type.
- `sf_ble_char_write_req_t::handle`

  Characteristic value or descriptor handle.
- `sf_ble_char_write_req_t::p_data`

  Pointer to data.
- `sf_ble_char_write_req_t::offset`

  Offset for long Characteristic value.
- `sf_ble_char_write_req_t::data_length`

  Data length.
- `sf_ble_char_write_req_t::auto_execute`

  Automatic execute write flag.

gattCharExecuteWrite

```
ssp_err_t(* sf_ble_api_t::gattCharExecuteWrite) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_execute_write_t execute_flag)
```

Brief description

Perform execute write on all pending write operations, used by GATT client.Execute or cancel all prepared writes of characteristics on remote GATT server.

Detailed description

Table 49: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
execute_flag	in	Flag specifying whether to execute or cancel pending writes

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter execute_flag

Definition: `sf_ble_execute_write_texecute_flag`

Characteristic execute write flag

gattCharWriteLocal

```
ssp_err_t(* sf_ble_api_t::gattCharWriteLocal) (sf_ble_ctrl_t *const p_ctrl,
uint16_t char_handle, uint16_t data_length, uint8_t *const p_data)
```

Brief description

Perform local characteristic write used by GATT server.Writes local characteristic value on GATT server.

Detailed description

Table 50: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
char_handle	in	Characteristic handle

Table 50: Parameters (Continued)

Name	Direction	Description
data_length	in	Length of data to write
p_data	in	Pointer to data

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter char_handle

`uint16_t`

Parameter data_length

`uint16_t`

Parameter p_data

`uint8_t`

gattSendNotify

```
ssp_err_t(* sf_ble_api_t::gattSendNotify) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
```

Brief description

Send notification to GATT client, used by GATT server.

Detailed description

Table 51: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
char_handle	in	Characteristic handle whose value will be notified

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter char_handle

`uint16_t`

gattSendIndicate

```
ssp_err_t(* sf_ble_api_t::gattSendIndicate) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
```

Brief description

Send indication to GATT client, used by GATT server.Send indication to remote GATT client.

Detailed description

Table 52: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure
p_handle	in	Connection handle
char_handle	in	Characteristic handle whose value will be indicated

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter char_handle

`uint16_t`

gattWriteResponse

```
ssp_err_t(* sf_ble_api_t::gattWriteResponse) (sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t handle, sf_ble_attribute_error_code_t
error_code)
```

Brief description

Send response to write operation received by GATT client, used by GATT server.Send response for write request received from remote GATT client.

Detailed description

Table 53: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure

Table 53: Parameters (Continued)

Name	Direction	Description
p_handle	in	Connection handle
handle	in	Characteristic handle used for write operation
error_code	in	Characteristic write operation error code to be sent in response

Parameter p_ctrl

Definition: `sf_ble_ctrl_t`

BLE Framework control structure

Parameter p_handle

Definition: `sf_ble_conn_handle_t*p_handle`

BLE Connection Handle

Parameter handle

`uint16_t`

Parameter error_code

versionGet

```
ssp_err_t(* sf_ble_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 54: Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

sf_ble_instance_t

sf_ble_instance_t

Detailed description

BLE instance

Variables

- sf_ble_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_ble_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_ble_api_t const * p_api

  Pointer to the API structure for this instance.

7.6.2 SF BLE On-Board Profile Framework Interface

RTOS-integrated SF BLE On-Board Profile Framework Interface.

7.6.2.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE On-Board Profile Framework.

7.6.2.2 Interface API

sf_ble_onboard_profile_api_t

Function name	Description
.open	Initializes the interface for data transfers. Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	De-initialize the interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.
.onbpEnable	Enable On-Board Profile Enables On-Board profile on given connection handle with specified security type. Registers user callback for the profile.
.onbpDisable	Disable On-Board Profile Disables On-Board profile on given connection handle and unregisters user callback.
.onbpServerWriteData	Update Server Local Database Update Local GATT database of Profile Server.
.onbpServerSendNotification	Send notification from Server Sends Notification which will be data specific to On-Board Profile to Client.

Function name	Description
.onbpServerSendIndication	Send Indication from Server Sends Indication which contains profile specific data to client.
.onbpClientWriteCCCD	Writes CCCD configuration in Server This API writes CCCD configuration of Server and which enables or disables notification or indication from server.
.onbpClientWriteChar	Writes GATT characteristics with data passed Writes the GATT characteristics in the Server with the data passed.
.onbpClientReadChar	Reads GATT characteristics from Server Reads GATT characteristics from the Server for the profile specified.
.versionGet	Gets version and stores it in provided pointer p_version.

7.6.2.3 Data structures

- sf_ble_onboard_profile_cccd_changed_t
- sf_ble_attr_info_t
- sf_ble_long_attr_info_t
- sf_ble_onboard_profile_cfg_t
- sf_ble_onboard_profile_ctrl_t
- sf_ble_onboard_profile_instance_t

7.6.2.4 Enumerations

- sf_onbp_t
- sf_ble_prf_sec_t
- sf_ble_onbp_char_t
- sf_ble_cccd_val_t

7.6.2.5 Typedefs

- sf_ble_profile_callback_t

7.6.2.6 Defines

- #define SF_BLE_ONBOARD_PROFILE_API_VERSION_MAJOR

  Initial value:`(1U)`

  SSP BSP Include files Framework Include files Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file

- #define SF_BLE_ONBOARD_PROFILE_API_VERSION_MINOR

  Initial value:`(0U)`

  Minor Version of the API defined in this file

- #define SF_BLE_ATTM_MAX_VALUE

  Initial value:`(0x18)`

  GATT Attribute Length

7.6.2.7 API Data

sf_onbp_t

`sf_onbp_t`

Detailed description

BLE On-Board Profile types

Enumerated values

Name	Description
SF_ONBP_ANS_SERVER	Alert Notification Service Server.
SF_ONBP_ANS_CLIENT	Alert Notification Service Client.
SF_ONBP_BAS_SERVER	Battery Service Server.
SF_ONBP_BPS_SERVER	Blood Pressure Service Server.
SF_ONBP_BPS_CLIENT	Blood Pressure Service Client.
SF_ONBP_CTS_SERVER	Current Time Service Server.
SF_ONBP_DIS_SERVER	Device Information Service Server.
SF_ONBP_FMP_SERVER	Find Me Service Server.
SF_ONBP_FMP_CLIENT	Find Me Service Client.
SF_ONBP_HTP_SERVER	Health Thermometer Service Server.

Name	Description
SF_ONBP_HTP_CLIENT	Health Thermometer Service Client.
SF_ONBP_HRS_SERVER	Heart Rate Service Server.
SF_ONBP_HRS_CLIENT	Heart Rate Service Client.
SF_ONBP_HID_SERVER	HID Over GATT Service Server.
SF_ONBP_HID_BHOST_CLIENT	HID Over GATT Boot Host Service Client.
SF_ONBP_HID_RHOST_CLIENT	HID Over GATT Report Host Service Client.
SF_ONBP_IMA_SERVER	Immediate Alert Service Server.
SF_ONBP_LLS_SERVER	Link Loss Service Server.
SF_ONBP_LLS_CLIENT	Link Loss Service Client.
SF_ONBP_NDCS_SERVER	Next Daylight Savings Change Service Server.
SF_ONBP_PAPS_SERVER	Phone Alert Service Server.
SF_ONBP_PAPS_CLIENT	Phone Alert Service Client.
SF_ONBP_PXP_SERVER	Proximity Service Server.
SF_ONBP_PXP_CLIENT	Proximity Service Client.
SF_ONBP_RTUS_SERVER	Reference Time Update Service Server.
SF_ONBP_SCPS_SERVER	Scan Parameter Service Server.
SF_ONBP_SCPS_CLIENT	Scan Parameter Service Client.
SF_ONBP_TIP_SERVER	Time Information Service Server.
SF_ONBP_TIP_CLIENT	Time Information Service Client.
SF_ONBP_TXP_SERVER	Transmit Power Service Server.

sf_ble_prf_sec_t

`sf_ble_prf_sec_t`

Detailed description

BLE Profile Security type

Enumerated values

Name	Description
SF_BLE_PRF_SEC_NONE	No Security.
SF_BLE_PRF_SEC_UNAUTH	Unauthenticated Pairing.
SF_BLE_PRF_SEC_AUTH	Authenticated pairing.
SF_BLE_PRF_SEC_AUTZ	Requires Authorized.
SF_BLE_PRF_SEC_ENC	Encrypted communication.

sf_ble_onbp_char_t

`sf_ble_onbp_char_t`

Detailed description

On-Board Profile GATT Characteristics Code

Enumerated values

Name	Description
SF_BLE_ONBP_CHAR_HRP_HRCP	Heart Rate Control Point Characteristics (Property: Write), Refer sf_ble_prf_hrp_api_hrcp_t. Heart Rate Profile
SF_BLE_ONBP_CHAR_HRP_CCCD_NTF_HRMEAS	Heart Rate Measurement CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_hrp_api_hrmeas_t.
SF_BLE_ONBP_CHAR_HRP_BSL	Heart Rate Body Sensor Location Characteristics (Property: Read)
SF_BLE_ONBP_CHAR_ANP_SUP_NEW_ALERT	Supported New Alert Category Characteristics (Property: Read) Alert Notification Profile
SF_BLE_ONBP_CHAR_ANP_CCCD_NTF_NEW_ALERT	New Alert CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_anp_api_new_alert_ntf_t.
SF_BLE_ONBP_CHAR_ANP_SUP_UNREAD_ALERT	Supported Unread Alert Category Characteristics (Property: Read)
SF_BLE_ONBP_CHAR_ANP_CCCD_NTF_UNREAD_ALERT_STATUS	Unread Alert Status CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_anp_api_unread_alert_ntf_t.
SF_BLE_ONBP_CHAR_ANP_ALERT_NOTIFICATION_CTRL_POINT	Alert Notification Control Point (Property: Write), Refer sf_ble_anp_ancp_t.

Name	Description
SF_BLE_ONBP_CHAR_DIS_MANUF	Device Information Service Manufacturer Name String (Property: Read) Device Information Service Profile
SF_BLE_ONBP_CHAR_DIS_MODEL	Device Information Service Model Number String (Property: Read)
SF_BLE_ONBP_CHAR_DIS_SERNB	Device Information Service Serial number String (Property: Read)
SF_BLE_ONBP_CHAR_DIS_HWREV	Device Information Service HW Revision String (Property: Read)
SF_BLE_ONBP_CHAR_DIS_FWREV	Device Information Service Fw Revision String (Property: Read)
SF_BLE_ONBP_CHAR_DIS_SWREV	Device Information Service SW Revision String (Property: Read)
SF_BLE_ONBP_CHAR_DIS_SYSID	Device Information Service system ID (Property: Read)
SF_BLE_ONBP_CHAR_DIS_IEEE	Device Information Service IEEE Certification (Property: Read)
SF_BLE_ONBP_CHAR_DIS_PNPID	Device Information PNPID, Used in services like HOGP (Property: Read)
SF_BLE_ONBP_CHAR_IAS_ALERT_LEVEL	Alert Level (Property: Write), Refer sf_ble_prf_ias_alert_type_t. Immediate Alert Service Profile
SF_BLE_ONBP_CHAR_SCPS_CCCD_NTF_SCAN_REFRESH	Scan Refresh CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_prf_scps_scan_refresh_t. Scan Parameters Service Profile
SF_BLE_ONBP_CHAR_SCPS_SCAN_INTERVAL_WINDOW	Scan Interval Window (Property: Write), Refer sf_ble_prf_scps_scan_intv_t.
SF_BLE_ONBP_CHAR_CTS_CCCD_NTF_CURRENT_TIME	Current Time CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_cts_curr_time_ntf_t. Current Time Service Profile
SF_BLE_ONBP_CHAR_CTS_CCCD_NTF_CURRENT_TIME_VALUE	Used to read data for current time through read characteristics.
SF_BLE_ONBP_CHAR_CTS_LOCAL_TIME_INFORMATION	Local Time Information (Property: Read)
SF_BLE_ONBP_CHAR_CTS_REFERENCE_TIME_INFORMATION	Reference Time Information (Property: Read)

Name	Description
SF_BLE_ONBP_CHAR_NDCS_TIME_WITH_DST	Time with DST (Property: Read) Next DST Change Service Profile
SF_BLE_ONBP_CHAR_TIME_UPDATE_CONTROL_POINT	Time Update Control Point (Property: Write), Refer sf_ble_prf_rtus_time_updt_state_t. Reference Time Update Service Profile
SF_BLE_ONBP_CHAR_TIME_UPDATE_STATE	Time Update State (Property: Read)
SF_BLE_ONBP_CHAR_CCCD_NTF_CURRENT_TIME	Current Time CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_bas_battery_lvl_ntf_t. Time Information Service Profile
SF_BLE_ONBP_CHAR_CCCD_NTF_BATTERY_LEVEL	Battery Level (Property: NTF, Read, Write), Refer sf_ble_prf_bas_battery_lvl_t. Battery Service Profile
SF_BLE_ONBP_CHAR_PXPM_SET_ALERT	PXP Alert char. Alert Level Service Profile
SF_BLE_ONBP_CHAR_PXPM_GET_ALERT_LVL	PXP Get Alert Level.
SF_BLE_ONBP_CHAR_PXPM_GET_TX_POWER_LVL	PXP Get TX Power Level.
SF_BLE_ONBP_CHAR_HTP_TEMP_MEAS_IND	HTPC temp. measurement indication CCCD Characteristics (Property: IND, Read, Write), Refer. Health Thermometer Profile
SF_BLE_ONBP_CHAR_HTP_INTERM_TEMP_NTF	HTPC intermediate temp. notification CCCD Characteristics (Property: NTF, Read, Write), Refer.
SF_BLE_ONBP_CHAR_HTP_MEAS_INTV_IND	HTPC temp. measurement interval indication CCCD Characteristics (Property: IND, Read, Write), Refer.
SF_BLE_ONBP_CHAR_HTP_TEMP_TYPE	HTPC temp. temperature type(Property: Read)
SF_BLE_ONBP_CHAR_HTP_MEAS_INTV	HTPC temp. measurement interval.
SF_BLE_ONBP_CHAR_HTP_MEAS_INTV_RANGE	HTPC temp. measurement interval valid range.
SF_BLE_ONBP_CHAR_CCCD_NTF_REPORT_INPUT	Report Input CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_prf_hid_report_desc_t. HID Over GATT Service Profile
SF_BLE_ONBP_CHAR_CCCD_NTF_KB_INPUT_REPORT	Keyboard Input Report CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_prf_hid_report_desc_t.
SF_BLE_ONBP_CHAR_CCCD_NTF_KB_INPUT_REPORT_VALUE	Used to read Keyboard Input record value through read characteristics.

Name	Description
SF_BLE_ONBP_CHAR_CCCD_NTF_KB_OUTPUT_REPORT_VALUE	Used to read Keyboard Output record value through read characteristics.
SF_BLE_ONBP_CHAR_CCCD_NTF_MOUSE_INPUT_REPORT	Mouse Input Report CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_prf_hid_report_desc_t.
SF_BLE_ONBP_CHAR_CCCD_NTF_MOUSE_INPUT_REPORT_VALUE	Used to read Mouse Input record value through read characteristics.
SF_BLE_ONBP_CHAR_PROTOCOL_MODE	HID boot host protocol mode (Property: Read, Write), Refer sf_ble_prf_hid_protocol_mode_t.
SF_BLE_ONBP_CHAR_CONTROL_POINT	HID report host write control point (Property: Read, Write), Refer sf_ble_prf_hid_ctrl_point_val_t.
SF_BLE_ONBP_CHAR_BLP_CCCD_NTF_INTERM_CUFPRS	Read Cuff Pressure measurement CCCD Characteristics (Property: NTF, Read, Write), Refer sf_ble_blp_meas_info_t. Blood Pressure profile
SF_BLE_ONBP_CHAR_BLP_CCCD_IND_BLDPRS_MEAS	Blood Pressure Measurement CCCD Characteristics (Property: IND, Read, Write), Refer sf_ble_blp_meas_info_t.
SF_BLE_ONBP_CHAR_BLP_RD_BLS_BF	Read Blood Pressure Feature.
SF_BLE_ONBP_CHAR_CCCD_NTF_ALERT_STATUS	Alert Status (Property: NTF, Read, Write), Refer sf_ble_prf_alert_status. Phone Alert Status Service Profile
SF_BLE_ONBP_CHAR_CCCD_NTF_ALERT_STATUS_VALUE	Used to read Alert Status value through read characteristics.
SF_BLE_ONBP_CHAR_CCCD_NTF_RINGER_SETTING	Ringer Settings (Property: NTF, Read, Write), Refer sf_ble_prf_ringer_setting_t.
SF_BLE_ONBP_CHAR_CCCD_NTF_RINGER_SETTING_VALUE	Used to read Ringer Settings value through read characteristics.
SF_BLE_ONBP_CHAR_RINGER_CONTROL_POINT	PAPS Ringer control point write characteristics.

sf_ble_cccd_val_t

`sf_ble_cccd_val_t`

Detailed description

Value for BLE CCCD Configuration

Enumerated values

Name	Description
SF_BLE_CCCD_VAL_STOP_NTFIND	Stop Notification Indication.
SF_BLE_CCCD_VAL_START_NTF	Start Notification.
SF_BLE_CCCD_VAL_START_IND	Start Indication.

sf_ble_profile_callback_t

```
typedef void(* sf_ble_profile_callback_t) (sf_ble_event_info_t *ev)
```

Detailed description

User callback type for profile

7.6.2.8 API Structures

sf_ble_onboard_profile_cccd_changed_t

sf_ble_onboard_profile_cccd_changed_t

Detailed description

BLE Profile CCCD Changed indication data

Variables

- sf_ble_conn_handle_t conn_handle

  Connection handle.
- sf_ble_onbp_char_t char_code

  CCCD type that has been changed by the remote node.
- sf_ble_cccd_val_t cccd_val

  CCCD value.
- uint8_t inst_idx

  Instance index, Applicable for HOGP.

sf_ble_attr_info_t

sf_ble_attr_info_t

Detailed description

BLE Profile Read Char data

Variables

- uint8_t len

  data length
- uint8_t data[SF_BLE_ATTM_MAX_VALUE]

  data

sf_ble_long_attr_info_t

sf_ble_long_attr_info_t

Detailed description

BLE Profile Read Char data (Long type)

Variables

- uint8_t val_len

  size of the value data

- uint8_t reserved

  Reserved.

- uint16_t attr_hdl

  Attribute handle.

- uint8_t value[SF_BLE_ATTM_MAX_VALUE]

  actual value pairs

sf_ble_onboard_profile_cfg_t

sf_ble_onboard_profile_cfg_t

Detailed description

BLE On-Board Profile configuration information

Variables

- sf_ble_instance_t const * p_low_lvl_ble

  Low level BLE Interface.

- void * p_extend

  Extended configuration.

sf_ble_onboard_profile_ctrl_t

sf_ble_onboard_profile_ctrl_t

Detailed description

BLE On-Board Profile Framework control structure

Variables

- sf_ble_instance_t * p_low_lvl_ble

  Low level BLE Framework information needed by On-Board Profile.

sf_ble_onboard_profile_api_t

sf_ble_onboard_profile_api_t

Detailed description

BLE Configuration, Control and API structures Framework API structure. Implementations will use the following API.

open

```
ssp_err_t(* sf_ble_onboard_profile_api_t::open) (sf_ble_onboard_profile_ctrl_t
*const p_ctrl, const sf_ble_onboard_profile_cfg_t *p_cfg)
```

Brief description

Initializes the interface for data transfers.

Detailed description

Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Table 55: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to BLE configuration structure.

Parameter p_ctrl

Definition: `sf_ble_onboard_profile_ctrl_t`

BLE On-Board Profile Framework control structure

Parameter p_cfg

Definition: `sf_ble_onboard_profile_cfg_t sf_ble_onboard_profile_cfg_t *p_cfg`

BLE On-Board Profile configuration information

- `sf_ble_onboard_profile_cfg_t`::`sf_ble_instance_t`

  Low level BLE Interface.

- `sf_ble_onboard_profile_cfg_t`::`p_extend`

  Extended configuration.

close

```
ssp_err_t(* sf_ble_onboard_profile_api_t::close) (sf_ble_onboard_profile_ctrl_t
*const p_ctrl)
```

Brief description

De-initialize the interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Detailed description

Table 56: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the BLE module.

Parameter p_ctrl

Definition: sf_ble_onboard_profile_ctrl_t

BLE On-Board Profile Framework control structure

onbpEnable

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpEnable)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_profile_callback_t p_prf_cb, sf_ble_prf_sec_t sec)
```

Brief description

Enable On-Board Profile Enables On-Board profile on given connection handle with specified security type. Registers user callback for the profile.

Detailed description

Table 57: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type to enable
p_prf_cb	in	User callback for Profile
sec	in	Security type for profile

Parameter p_ctrl

Definition: sf_ble_onboard_profile_ctrl_t

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: sf_onbp_tprofile

BLE On-Board Profile types

Parameter p_prf_cb

Definition: sf_ble_profile_callback_tp_prf_cb

User callback type for profile

Parameter sec

onbpDisable

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpDisable)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile)
```

Brief description

Disable On-Board Profile Disables On-Board profile on given connection handle and unregisters user callback.

Detailed description

Table 58: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type to disable

Parameter p_ctrl

Definition: `sf_ble_onboard_profile_ctrl_t`

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: `sf_onbp_tprofile`

BLE On-Board Profile types

onbpServerWriteData

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerWriteData)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
```

Brief description

Update Server Local Database Update Local GATT database of Profile Server.

Detailed description

Table 59: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type
characteristics	in	Profile characteristics
p_data	in	Pointer to data

Parameter p_ctrl

Definition: sf_ble_onboard_profile_ctrl_t

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: sf_onbp_tprofile

BLE On-Board Profile types

Parameter characteristics

Definition: sf_ble_onbp_char_tcharacteristics

On-Board Profile GATT Characteristics Code

Parameter p_data

const

onbpServerSendNotification

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerSendNotification)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
```

Brief description

Send notification from Server Sends Notification which will be data specific to On-Board Profile to Client.

Detailed description

Table 60: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type
characteristics	in	Profile characteristics
p_data	in	Pointer to data

Parameter p_ctrl

Definition: sf_ble_onboard_profile_ctrl_t

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: sf_onbp_tprofile

BLE On-Board Profile types

Parameter characteristics

Definition: sf_ble_onbp_char_tcharacteristics

On-Board Profile GATT Characteristics Code

Parameter p_data

const

onbpServerSendIndication

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerSendIndication)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
```

Brief description

Send Indication from Server Sends Indication which contains profile specific data to client.

Detailed description

Table 61: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type
characteristics	in	Profile characteristics
p_data	in	Pointer to data

Parameter p_ctrl

Definition: sf_ble_onboard_profile_ctrl_t

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: sf_onbp_tprofile

BLE On-Board Profile types

Parameter characteristics

Definition: sf_ble_onbp_char_tcharacteristics

On-Board Profile GATT Characteristics Code

Parameter p_data

const

onbpClientWriteCCCD

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientWriteCCCD)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t cccd_char, sf_ble_cccd_val_t cccd_val)
```

Brief description

Writes CCCD configuration in Server This API writes CCCD configuration of Server and which enables or disables notification or indication from server.

Detailed description

Table 62: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Parameter_Description
cccd_char	in	CCCD Code
cccd_val	in	Configuration data of CCCD

Parameter p_ctrl

Definition: `sf_ble_onboard_profile_ctrl_t`

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: `sf_onbp_tprofile`

BLE On-Board Profile types

Parameter cccd_char

Definition: `sf_ble_onbp_char_tcccd_char`

On-Board Profile GATT Characteristics Code

Parameter cccd_val

onbpClientWriteChar

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientWriteChar)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
```

Brief description

Writes GATT characteristics with data passed Writes the GATT characteristics in the Server with the data passed.

Detailed description

Table 63: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle
profile	in	Profile type
characteristics	in	GATT characteristics code
p_data	in	Pointer to data

Parameter p_ctrl

Definition: `sf_ble_onboard_profile_ctrl_t`

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: `sf_onbp_tprofile`

BLE On-Board Profile types

Parameter characteristics

Definition: `sf_ble_onbp_char_tcharacteristics`

On-Board Profile GATT Characteristics Code

Parameter p_data

`const`

onbpClientReadChar

```
ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientReadChar)
(sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle,
sf_onbp_t profile, sf_ble_onbp_char_t characteristics)
```

Brief description

Reads GATT characteristics from Server Reads GATT characteristics from the Server for the profile specified.

Detailed description

Table 64: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure for BLE
p_handle	in	Pointer to connection handle

Table 64: Parameters (Continued)

Name	Direction	Description
profile	in	Profile type
characteristics	in	Profile characteristics

Parameter p_ctrl

Definition: `sf_ble_onboard_profile_ctrl_t`

BLE On-Board Profile Framework control structure

Parameter p_handle

Parameter profile

Definition: `sf_onbp_tprofile`

BLE On-Board Profile types

Parameter characteristics

Definition: `sf_ble_onbp_char_tcharacteristics`

On-Board Profile GATT Characteristics Code

versionGet

```
ssp_err_t(* sf_ble_onboard_profile_api_t::versionGet) (ssp_version_t *const
p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 65: Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

sf_ble_onboard_profile_instance_t

sf_ble_onboard_profile_instance_t

Detailed description

Instance structure

Variables

- sf_ble_onboard_profile_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_ble_onboard_profile_cfg_t const * p_cfg
  Pointer to the configuration structure for this instance.
- sf_ble_onboard_profile_api_t const * p_api
  Pointer to the API structure for this instance.

7.6.3 SF BLE Alert Notification Profile Framework Interface

RTOS-integrated SF BLE Alert Notification Profile Framework Interface.

7.6.3.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Alert Notification Profile Framework.

7.6.3.2 Interface API

sf_ble_anp_api_new_alert_t

Function name	Description
.category_id	Category ID.
.alert_num	Number of New Alert.
.text_size	Text Size.
.text	Actual Text.

sf_ble_anp_api_new_alert_ntf_t

Function name	Description
.conhdl	Connection handle.
.new_alert	New Alert data.

sf_ble_anp_api_unread_alert_t

Function name	Description
.category_id	Category ID.

Function name	Description
.unread_count	Number of Unread Alert.

sf_ble_anp_api_unread_alert_ntf_t

Function name	Description
.conhdl	Connection handle.
.alert	Unread Alert data.

7.6.3.3 Data structures

- sf_ble_anp_ancp_t
- sf_ble_anp_ancp_change_t
- sf_ble_anp_api_new_alert_t
- sf_ble_anp_api_new_alert_ntf_t
- sf_ble_anp_api_unread_alert_t
- sf_ble_anp_api_unread_alert_ntf_t

7.6.3.4 Enumerations

- sf_ble_prf_anpc_event_t
- sf_ble_prf_anps_event_t
- sf_ble_prf_anp_category_id
- sf_ble_prf_anp_cmd_id_t

7.6.3.5 Defines

- #define SF_BLE_ANP_ALT_TEXT_MAX

  Initial value:`(18U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  Buffer size of data for Alert Notification

7.6.3.6 API Data

sf_ble_prf_anpc_event_t

`sf_ble_prf_anpc_event_t`

Detailed description

Profile Client user events

Enumerated values

Name	Description
SF_BLE_PRF_ANPC_EVENT_NONE	Event not supported.
SF_BLE_PRF_ANPC_EVENT_NEW_ALT_NTF	New Alert Data received event, Refer sf_ble_anp_api_new_alert_ntf_t.
SF_BLE_PRF_ANPC_EVENT_UNREAD_ALT_NTF	Unread Alert Data received event, Refer sf_ble_anp_api_unread_alert_ntf_t.
SF_BLE_PRF_ANPC_EVENT_READ_CHAR_RES	Read Char Complete Event.

sf_ble_prf_anps_event_t

`sf_ble_prf_anps_event_t`

Detailed description

Alert notification server profile user events

Enumerated values

Name	Description
SF_BLE_PRF_ANPS_EVENT_NONE	Event not supported.
SF_BLE_PRF_ANPS_EVENT_ALT_NF_CP_IND	Alert Notification Control Point Changed indication, Refer sf_ble_anp_ancp_change_t.
SF_BLE_PRF_ANPS_EVENT_CCCD_NTF_IND	CCCD Notification Setting change event.

sf_ble_prf_anp_category_id

`sf_ble_prf_anp_category_id`

Detailed description

Alert Notification Control Point Category ID

Enumerated values

Name	Description
SF_BLE_PRF_ANP_CATEGORY_ID_SIMPLE_ALERT	Simple Alert: General text alert or non-text alert.

Name	Description
SF_BLE_PRF_ANP_CATEGORY_ID_EMAIL	Email Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_NEWS	News feeds Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_CALL	Incoming Call Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_MISSED_CALL	Missed Call Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_SMS_MMS	SMS/MMS Message Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_VOICE_MAIL	Voice Mail Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_SCHEDULE	Alert occurred on calendar, planner.
SF_BLE_PRF_ANP_CATEGORY_ID_HIGH_PRIORITY_ALERT	High Prioritized Alert.
SF_BLE_PRF_ANP_CATEGORY_ID_INSTANT_MESSAGE	Incoming Instant Messages.
SF_BLE_PRF_ANP_CATEGORY_ID_ALL	All Supported Categories.

sf_ble_prf_anp_cmd_id_t

`sf_ble_prf_anp_cmd_id_t`

Detailed description

Alert Notification Control Point Command ID

Enumerated values

Name	Description
SF_BLE_PRF_ANP_CMD_ID_NEW_ALERT_ENABLE	Enable New Incoming Alert Notification.
SF_BLE_PRF_ANP_CMD_ID_UNREAD_ALERT_ENABLE	Enable Unread Category Status Notification.
SF_BLE_PRF_ANP_CMD_ID_NEW_ALERT_DISABLE	Disable New Incoming Alert Notification.
SF_BLE_PRF_ANP_CMD_ID_UNREAD_ALERT_DISABLE	Disable Unread Category Status Notification.
SF_BLE_PRF_ANP_CMD_ID_NEW_ALERT_NTF_REQ	Notify New Incoming Alert immediately.
SF_BLE_PRF_ANP_CMD_ID_UNREAD_ALERT_NTF_REQ	Notify Unread Category Status immediately.

7.6.3.7 API Structures

sf_ble_anp_ancp_t

sf_ble_anp_ancp_t

Detailed description

Write Characteristics data for Alert Notification Control Point

Variables

- sf_ble_prf_anp_cmd_id_t command_id

  Command type.

- sf_ble_prf_anp_category_id category_id

  Category on which to act.

sf_ble_anp_ancp_change_t

sf_ble_anp_ancp_change_t

Detailed description

Alert Notification Control Point Change Indication for Sensor

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_anp_ancp_t control_point_value

  Control point value.

sf_ble_anp_api_new_alert_t

sf_ble_anp_api_new_alert_t

Detailed description

Notification Data for New Alert, also used for sending notification from server

category_id

`sf_ble_prf_anp_category_id::category_id`

Brief description

Category ID.

alert_num

`uint8_t sf_ble_anp_api_new_alert_t::alert_num`

Brief description

Number of New Alert.

text_size

`uint8_t sf_ble_anp_api_new_alert_t::text_size`

Brief description

Text Size.

text

`uint8_t sf_ble_anp_api_new_alert_t::text[SF_BLE_ANP_ALT_TEXT_MAX]`

Brief description

Actual Text.

sf_ble_anp_api_new_alert_ntf_t

sf_ble_anp_api_new_alert_ntf_t

Detailed description

Notification Data received for New Alert

conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

new_alert

`sf_ble_anp_api_new_alert_t::new_alert`

Brief description

New Alert data.

sf_ble_anp_api_unread_alert_t

sf_ble_anp_api_unread_alert_t

Detailed description

Notification Data for Unread Alert, also used for sending notification from server

category_id

`sf_ble_prf_anp_category_id::category_id`

Brief description

Category ID.

unread_count

`uint8_t sf_ble_anp_api_unread_alert_t::unread_count`

Brief description

Number of Unread Alert.

sf_ble_anp_api_unread_alert_ntf_t

sf_ble_anp_api_unread_alert_ntf_t

Detailed description

Notification Data received for Unread Alert

conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

alert

sf_ble_anp_api_unread_alert_t::alert

Brief description

Unread Alert data.

7.6.4 SF BLE Battery Service Profile Framework Interface

RTOS-integrated SF BLE Battery Service Profile Framework Interface.

7.6.4.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Battery Service Profile Framework.

7.6.4.2 Data structures

- sf_ble_bas_battery_lvl_ntf_t

7.6.4.3 Variables

- sf_ble_prf_bas_battery_lvl_t

7.6.4.4 sf_ble_prf_bas_battery_lvl_t

SSP_HEADER typedef uint8_t sf_ble_prf_bas_battery_lvl_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Battery Level

7.6.4.5 API Structures

sf_ble_bas_battery_lvl_ntf_t

sf_ble_bas_battery_lvl_ntf_t

Detailed description

Control Point Change Indication for Sensor

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_bas_battery_lvl_t batt_lvl

  Battery Level.

7.6.5 SF BLE Blood Pressure Profile Framework Interface

RTOS-integrated SF BLE Blood Pressure Profile Framework Interface.

7.6.5.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Blood Pressure Profile Framework.

7.6.5.2 Data structures

- sf_ble_blp_meas_info_t
- sf_ble_blp_meas_recv_data_t

7.6.5.3 Enumerations

- sf_ble_prf_blpc_event_t
- sf_ble_prf_blps_event_t

7.6.5.4 API Data

sf_ble_prf_blpc_event_t

`sf_ble_prf_blpc_event_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Profile user client events

Enumerated values

Name	Description
SF_BLE_PRF_BLPC_EVENT_NONE	Event not supported.
SF_BLE_PRF_BLPC_EVENT_MEAS_NTF_IND	BLE user event indicating BLPC profile measurement notification.
SF_BLE_PRF_BLPC_EVENT_WRITE_CHAR_RES	BLE user event indicating BLPC profile read char response.
SF_BLE_PRF_BLPC_EVENT_READ_CHAR_RES	BLE user event indicating BLPC write char response.

sf_ble_prf_blps_event_t

`sf_ble_prf_blps_event_t`

Detailed description

Profile user server events

Enumerated values

Name	Description
SF_BLE_PRF_BLPS_EVENT_NONE	Event not supported.
SF_BLE_PRF_BLPS_EVENT_NTFIND_IND	BLE user event indicating notification indication.

7.6.5.5 API Structures

sf_ble_blp_meas_info_t

sf_ble_blp_meas_info_t

Detailed description

Blood Pressure Measurements Info

Variables

- uint8_t flag_stable_meas

  Stable or intermediary type of measurements. Set to 0 if intermediate measurement.
- uint8_t flags

  flags
- int16_t press_val1

  blood pressure value - Systolic or cuff pressure. Cuff pressure has to be filled here
- int16_t press_val2

  blood pressure value - Diastolic or subfield1
- int16_t press_val3

  blood pressure value - MAP or subfield2
- sf_ble_prf_cts_date_time_t stamp

  time stamp
- int16_t rate

  pulse rate
- uint8_t id

  user ID
- uint8_t reserved

  Reserved.
- uint16_t meas_sts

  measurement status

sf_ble_blp_meas_recv_data_t

sf_ble_blp_meas_recv_data_t

Detailed description

Blood Pressure Measurements Data Received from Server

Variables

- uint16_t conhdl

  Connection handle.

- sf_ble_onbp_char_t char_code

  Characteristic code.

- sf_ble_blp_meas_info_t meas_info

  BLP measurement data.

7.6.6 SF BLE Current Time Service Profile Framework Interface

RTOS-integrated SF BLE Current Time Service Profile Framework Interface.

7.6.6.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Current Time Service Profile Framework.

7.6.6.2 Data structures

- sf_ble_prf_cts_date_time_t
- sf_ble_prf_cts_curr_time_t
- sf_ble_cts_local_time_t
- sf_ble_cts_ref_time_t
- sf_ble_cts_curr_time_ntf_t

7.6.6.3 API Structures

sf_ble_prf_cts_date_time_t

sf_ble_prf_cts_date_time_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Current Time Service Date Time information

Variables

- uint16_t year

  Year value.

- uint8_t month

  Month value.
- uint8_t day

  Day value.
- uint8_t hour

  Hour value.
- uint8_t min

  Minute value.
- uint8_t sec

  Second value.
- uint8_t reserved

  Reserved.

sf_ble_prf_cts_curr_time_t

sf_ble_prf_cts_curr_time_t

Detailed description

Current time information with Date Time

Variables

- sf_ble_prf_cts_date_time_t stamp

  Date time info.
- uint8_t day_of_week

  Day of week.
- uint8_t fractions256

  Fraction value.
- uint8_t adjust_reason

  Adjust reason.
- uint8_t reserved

  Reserved.

sf_ble_cts_local_time_t

sf_ble_cts_local_time_t

Detailed description

Local Time Information structure

Variables

- int8_t time_zone

  Time Zone.

- uint8_t dst_offset

  DST Offset.

sf_ble_cts_ref_time_t

sf_ble_cts_ref_time_t

Detailed description

Reference Time Information structure

Variables

- uint8_t time_source

  Source of time.

- uint8_t accuracy

  Estimated accuracy of time compared to original time source.

- uint8_t days_since_update

  Days that passed since time was updated successfully from time source.

- uint8_t hours_since_update

  Time that passed since time was updated successfully from time source.

sf_ble_cts_curr_time_ntf_t

sf_ble_cts_curr_time_ntf_t

Detailed description

Notification Data of Current Time received from server

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_cts_curr_time_t current_time

  heart rate measurement data

7.6.7 SF BLE Find Me Profile Framework Interface

RTOS-integrated SF BLE Find Me Profile Framework Interface.

7.6.7.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Find Me Profile Framework.

7.6.7.2 API Data

sf_ble_prf_fmpt_event_t

`sf_ble_prf_fmpt_event_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Profile Server user events

Enumerated values

Name	Description
SF_BLE_PRF_FMPT_EVENT_NONE	BLE event unsupported.
SF_BLE_PRF_FMPT_EVENT_ALERT_LVL_CHANGED	BLE user event Alert level changed from locator, Refer sf_ble_ias_alert_lvl_change_t.

7.6.8 SF BLE HID Over GATT Profile Framework Interface

RTOS-integrated SF BLE HID Over GATT Profile Framework Interface.

7.6.8.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE HID Over GATT Profile Framework.

7.6.8.2 Data structures

- sf_ble_prf_hid_report_desc_t
- sf_ble_prf_hid_report_ind_t
- sf_ble_prf_hid_change_event_t
- sf_ble_prf_dis_pnpID_t

7.6.8.3 Enumerations

- sf_ble_prf_hidd_event_t
- sf_ble_prf_hid_event_t
- sf_ble_hidd_device_type_t

7.6.8.4 Typedefs

- sf_ble_prf_hid_protocol_mode_t
- sf_ble_prf_hid_ctrl_point_val_t

7.6.8.5 Defines

- #define SF_BLE_PRF_HIDS_REPORT_MAX

  Initial value:`(32U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  Maximum Number of reports in HID

7.6.8.6 API Data

sf_ble_prf_hidd_event_t

`sf_ble_prf_hidd_event_t`

Detailed description

Profile Server user events

Enumerated values

Name	Description
SF_BLE_PRF_HIDD_EVENT_NONE	Event not supported.
SF_BLE_PRF_HIDD_EVENT_REPORT_IND	Report value updated from Boot Host or Report Host, Refer sf_ble_prf_hid_report_ind_t.
SF_BLE_PRF_HIDD_EVENT_CFG_IND	CCCD change indication from Boot Host or Report Host.
SF_BLE_PRF_HIDD_EVENT_PROTO_MODE_CHG_EVT	protocol mode change event from Boot Host or Report Host, Refer sf_ble_prf_hid_change_event_t
SF_BLE_PRF_HIDD_EVENT_REPORT_EVT	report value update event from Boot Host or Report Host, Refer sf_ble_prf_hid_report_ind_t
SF_BLE_PRF_HIDD_EVENT_CP_CHANGED_EVT	suspend event from Report Host, Refer sf_ble_prf_hid_change_event_t

sf_ble_prf_hid_event_t

`sf_ble_prf_hid_event_t`

Detailed description

Profile Client user events

Enumerated values

Name	Description
SF_BLE_PRF_HID_EVENT_NONE	Event not supported.

Name	Description
SF_BLE_HID_BHOST_EVENT_REPORT_NTF	Report value received from HID device, Refer sf_ble_prf_hid_report_ind_t. HID BHOST
SF_BLE_HID_BHOST_EVENT_READ_CHAR_RESP	read char response received from HID device
SF_BLE_HID_RHOST_EVENT_REPORT_NTF	Report value received from HID device, Refer sf_ble_prf_hid_report_ind_t. HID RHOST
SF_BLE_HID_RHOST_EVENT_BATTERY_LVL_NTF	Battery level received from HID device, Refer sf_ble_bas_battery_lvl_ntf_t.
SF_BLE_HID_RHOST_EVENT_READ_CHAR_RESP	Read char response received from HID device.
SF_BLE_HID_RHOST_EVENT_READ_LONG_CHAR_RESP	Long char read response received from HID device.

sf_ble_hidd_device_type_t

```
sf_ble_hidd_device_type_t
```

Detailed description

HID Device types

Enumerated values

Name	Description
SF_BLE_HIDD_HID_DEVICE	HID Device type.
SF_BLE_HIDD_BOOT_KEYBOARD	Boot Keyboard type.
SF_BLE_HIDD_BOOT_MOUSE	Boot Mouse type.

sf_ble_prf_hid_protocol_mode_t

```
typedef uint8_t sf_ble_prf_hid_protocol_mode_t
```

Detailed description

Protocol Mode Characteristics

sf_ble_prf_hid_ctrl_point_val_t

```
typedef uint8_t sf_ble_prf_hid_ctrl_point_val_t
```

Detailed description

HID Control Point Characteristics

7.6.8.7 API Structures

sf_ble_prf_hid_report_desc_t

sf_ble_prf_hid_report_desc_t

Detailed description

HID Report structure

Variables

- sf_ble_hidd_device_type_t device_type

  Device type.

- uint8_t report_type

  Report type.

- uint8_t value[SF_BLE_PRF_HIDS_REPORT_MAX]

  Report values.

- uint16_t value_size

  Report size.

sf_ble_prf_hid_report_ind_t

sf_ble_prf_hid_report_ind_t

Detailed description

HID Report Indication structure

Variables

- uint16_t conhdl

  Connection handle.

- uint8_t inst_idx

  Instance Index.

- uint8_t reserved

  Reserved.

- sf_ble_prf_hid_report_desc_t report

  Report received from either BHOST or RHOST.

sf_ble_prf_hid_change_event_t

sf_ble_prf_hid_change_event_t

Detailed description

HID Profile Values Change structure

Variables

- uint16_t conhdl

  Connection handle.

- uint8_t inst_idx

  Instance Index.
- sf_ble_prf_hid_protocol_mode_t protocol_mode_val

  Protocol Mode.
- sf_ble_prf_hid_ctrl_point_val_t control_point_val

  HID Control Point.
- union{} ble_prf_value

  Value which is changed by client.

  See source code for definition of this member.

sf_ble_prf_dis_pnpID_t

sf_ble_prf_dis_pnpID_t

Detailed description

Structure to set the current Device Information PnP Id characteristic value

Variables

- uint8_t vendorIdSource

  Vendor ID source.
- uint16_t vendorId

  Vendor ID.
- uint16_t productId

  Product ID.
- uint16_t productVersion

  Version of Product.

7.6.9 SF BLE Heart Rate Profile Framework Interface

RTOS-integrated SF BLE Heart Rate Profile Framework Interface.

7.6.9.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Heart Rate Profile Framework.

7.6.9.2 Interface API

sf_ble_hrp_api_hrmeas_t

Function name	Description
.flags	HRP bit Flags.
.rr_interval_num	number of RR Interval
.heart_rate_measure	Heart Rate measurement value.
.energy_expended	Energy Expended in Kilo Joules from last time it was reset.
.rr_interval	RR interval(s)

sf_ble_hrp_api_meas_ntf_t

Function name	Description
.conhdl	Connection handle.
.measurements_info	heart rate measurement data

7.6.9.3 Data structures

- sf_ble_hrp_api_hrmeas_t
- sf_ble_hrp_api_meas_ntf_t
- sf_ble_hrp_cp_change_t

7.6.9.4 Enumerations

- sf_ble_prf_hrpc_event_t
- sf_ble_prf_hrps_event_t

7.6.9.5 Typedefs

- sf_ble_prf_hrp_api_hrcp_t

7.6.9.6 Defines

- #define SF_BLE_PRF_HRP_API_RR_INTERVAL_BUFF_LEN

  Initial value:`(0x9U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  Heart Rate RR Interval Buffer Length

7.6.9.7 API Data

sf_ble_prf_hrpc_event_t

`sf_ble_prf_hrpc_event_t`

Detailed description

Profile Client user events

Enumerated values

Name	Description
SF_BLE_PRF_HRPC_EVENT_NONE	Event not supported.
SF_BLE_PRF_HRPC_EVENT_MEAS_NTF	Heart Rate Measurement data received event, Refer sf_ble_hrp_api_meas_ntf_t.
SF_BLE_PRF_HRPC_EVENT_READ_CHAR_RES	Read Char Complete Event.

sf_ble_prf_hrps_event_t

`sf_ble_prf_hrps_event_t`

Detailed description

Profile Server user events

Enumerated values

Name	Description
SF_BLE_PRF_HRPS_EVENT_NONE	Event not supported.
SF_BLE_PRF_HRPS_EVENT_NTF_CHG_IND	CCCD Notification Setting Change Event.
SF_BLE_PRF_HRPS_EVENT_HRCP_CHG_IND	Heart Rate Control Point Changed Event, Refer sf_ble_hrp_cp_change_t.

sf_ble_prf_hrp_api_hrcp_t

```
typedef uint16_t sf_ble_prf_hrp_api_hrcp_t
```

Detailed description

Write Characteristics data for Heart Rate Control Point

7.6.9.8 API Structures

sf_ble_hrp_api_hrmeas_t

sf_ble_hrp_api_hrmeas_t

Detailed description

Notification Data for Heart Rate Measurement, also used for sending notification from server

flags

`uint8_t sf_ble_hrp_api_hrmeas_t::flags`

Brief description

HRP bit Flags.

rr_interval_num

`uint8_t sf_ble_hrp_api_hrmeas_t::rr_interval_num`

Brief description

number of RR Interval

heart_rate_measure

`uint16_t sf_ble_hrp_api_hrmeas_t::heart_rate_measure`

Brief description

Heart Rate measurement value.

energy_expended

`uint16_t sf_ble_hrp_api_hrmeas_t::energy_expended`

Brief description

Energy Expended in Kilo Joules from last time it was reset.

rr_interval

`uint16_t sf_ble_hrp_api_hrmeas_t::rr_interval[SF_BLE_PRF_HRP_API_RR_INTERVAL_BUFF_LEN]`

Brief description

RR interval(s)

sf_ble_hrp_api_meas_ntf_t

sf_ble_hrp_api_meas_ntf_t

Detailed description

Notification Data of Heart Rate measurement received from sensor

conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

measurements_info

`sf_ble_hrp_api_hrmeas_t::measurements_info`

Brief description

heart rate measurement data

sf_ble_hrp_cp_change_t

sf_ble_hrp_cp_change_t

Detailed description

Control Point Change Indication for Sensor

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_hrp_api_hrcp_t control_point_value

  Control point value.

7.6.10 SF BLE Health Thermometer Profile Framework Interface

RTOS-integrated SF BLE Health Thermometer Profile Framework Interface.

7.6.10.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Health Thermometer Profile Framework.

7.6.10.2 Data structures

- sf_ble_prf_htp_temp_info_t
- sf_ble_prf_htp_temp_info_ind_t
- st_sf_ble_prf_htp_meas_intv_val_t

7.6.10.3 Enumerations

- sf_ble_event_prf_htpt_t
- sf_ble_event_prf_htpc_t
- sf_ble_prf_htp_meas_state_t
- sf_ble_prf_htp_temp_type_t

7.6.10.4 API Data

sf_ble_event_prf_htpt_t

`sf_ble_event_prf_htpt_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. HTP thermometer (server) user events

Enumerated values

Name	Description
SF_BLE_EVENT_PRF_HTPT_NONE	Event not supported.
SF_BLE_EVENT_PRF_HTPT_MEAS_INTV_CHG_IND	Thermometer measurement interval characteristic change indication.
SF_BLE_EVENT_PRF_HTPT_CFG_IND	BLE user event indicating HTPT profile CCCD change indication.

sf_ble_event_prf_htpc_t

`sf_ble_event_prf_htpc_t`

Detailed description

HTP Collector (client) user events

Enumerated values

Name	Description
SF_BLE_EVENT_PRF_HTPC_NONE	Event not supported.
SF_BLE_EVENT_PRF_HTPC_TEMP_IND	BLE user event indicating HTPC profile temperature value indication.
SF_BLE_EVENT_PRF_HTPC_MEAS_INTV_IND	BLE user event indicating HTPC profile measurement interval value indication.
SF_BLE_PRF_HTPC_EVENT_READ_CHAR_RES	BLE user event for read char response.

sf_ble_prf_htp_meas_state_t

`sf_ble_prf_htp_meas_state_t`

Detailed description

HTP Collector (client) user events

Enumerated values

Name	Description
SF_BLE_PRF_HTP_MEAS_STATE_IN_PROGRESS	Measurement in progress.
SF_BLE_PRF_HTP_MEAS_STATE_COMPLETE	Measurement is complete.

sf_ble_prf_htp_temp_type_t

`sf_ble_prf_htp_temp_type_t`

Detailed description

HTP Temperature Types

Enumerated values

Name	Description
SF_BLE_PRF_HTP_TYPE_ARMPIT	Temperature type Armpit.
SF_BLE_PRF_HTP_TYPE_BODY	Temperature type Body.
SF_BLE_PRF_HTP_TYPE_EAR	Temperature type Ear.
SF_BLE_PRF_HTP_TYPE_FINGER	Temperature type Finger.
SF_BLE_PRF_HTP_TYPE_GASTROINTESTINAL	Temperature type Gastrointestinal.
SF_BLE_PRF_HTP_TYPE_MOUTH	Temperature type Mouth.
SF_BLE_PRF_HTP_TYPE_RECTUM	Temperature type Rectum.
SF_BLE_PRF_HTP_TYPE_TOE	Temperature type Toe.
SF_BLE_PRF_HTP_TYPE_TYMPANUM	Temperature type Tympanum.

7.6.10.5 API Structures

sf_ble_prf_htp_temp_info_t

sf_ble_prf_htp_temp_info_t

Detailed description

Health Thermometer Info

Variables

- uint8_t flag_stable_meas

  Stable or intermediary type of temperature.

- uint8_t flags
  flags
- int32_t temp_val
  temp value
- sf_ble_prf_cts_date_time_t stamp
  time stamp
- sf_ble_prf_htp_temp_type_t type
  type
- uint8_t reserved
  Reserved.

sf_ble_prf_htp_temp_info_ind_t

sf_ble_prf_htp_temp_info_ind_t

Detailed description

Thermometer Information Indication

Variables

- uint16_t conhdl
  Connection handle.
- sf_ble_prf_htp_temp_info_t temp_info
  Thermometer info.

st_sf_ble_prf_htp_meas_intv_val_t

st_sf_ble_prf_htp_meas_intv_val_t

Detailed description

Measurement Interval value

Variables

- uint16_t conhdl
  Connection handle.
- uint16_t intv
  Interval value.

7.6.11 SF BLE Immediate Alert Profile Framework Interface

RTOS-integrated SF BLE Immediate Alert Profile Framework Interface.

7.6.11.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Immediate Alert Profile Framework.

7.6.11.2 Data structures

- sb_ble_prf_ias_set_alert_t
- sf_ble_prf_ias_alert_lvl_change_t

7.6.11.3 Enumerations

- sf_ble_prf_ias_alert_type_t
- sf_ble_prf_ias_svc_code_t

7.6.11.4 API Data

sf_ble_prf_ias_alert_type_t

`sf_ble_prf_ias_alert_type_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Alert Level Values

Enumerated values

Name	Description
SF_BLE_PRF_ALERT_TYPE_NONE	No Alert.
SF_BLE_PRF_ALERT_TYPE_MILD	Mild Alert.
SF_BLE_PRF_ALERT_TYPE_HIGH	High Alert.

sf_ble_prf_ias_svc_code_t

`sf_ble_prf_ias_svc_code_t`

Detailed description

SVC codes

Enumerated values

Name	Description
SF_BLE_SVC_SET_LK_LOSS_ALERT	Code for LLS Alert Level Char.
SF_BLE_SVC_SET_IMMDT_ALERT	Code for IAS Alert Level Char.

7.6.11.5 API Structures

sb_ble_prf_ias_set_alert_t

sb_ble_prf_ias_set_alert_t

Detailed description

Alert level and type

Variables

- sf_ble_prf_ias_svc_code_t svc_code

  SVC code.

- sf_ble_prf_ias_alert_type_t lvl

  Alert level.

sf_ble_prf_ias_alert_lvl_change_t

sf_ble_prf_ias_alert_lvl_change_t

Detailed description

Alert Level Change Indication for Server

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_ias_alert_type_t alert_lvl

  Control point value.

7.6.12 SF BLE Next DST Change Service Profile Framework Interface

RTOS-integrated SF BLE Next DST Change Service Profile Framework Interface.

7.6.12.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Next DST Change Service Profile Framework.

7.6.12.2 Data structures

- sf_ble_prf_ndcs_time_dst_t

7.6.12.3 API Structures

sf_ble_prf_ndcs_time_dst_t

sf_ble_prf_ndcs_time_dst_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Next Time with DST Information structure

Variables

- sf_ble_prf_cts_date_time_t stamp

  Current time stamp.

- uint8_t dst_offset

  DST Offset.

- uint8_t reserved

  Reserved.

7.6.13 SF BLE Phone Alert Status Profile Framework Interface

RTOS-integrated SF BLE Phone Alert Status Profile Framework Interface.

7.6.13.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Phone Alert Status Profile Framework.

7.6.13.2 Data structures

- sf_ble_prf_ringer_cp_change_t
- sf_ble_prf_ringer_setting_ntf_t
- sf_ble_prf_alert_status_ntf_t

7.6.13.3 Enumerations

- sf_ble_prf_papsc_event_t
- sf_ble_prf_papss_event_t
- sf_ble_prf_ringer_cp_t
- sf_ble_prf_ringer_setting_t
- sf_ble_prf_alert_setting_t

7.6.13.4 Typedefs

- sf_ble_prf_alert_status

7.6.13.5 API Data

sf_ble_prf_papsc_event_t

```
sf_ble_prf_papsc_event_t
```

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Profile Client user events

Enumerated values

Name	Description
SF_BLE_PRF_PAPSC_EVENT_NONE	Event not supported.
SF_BLE_PRF_PAPSC_EVENT_READ_CHAR_RES	BLE user event indicating PAPS target alert indication received from locator.
SF_BLE_PRF_PAPSC_EVENT_RINGER_SETTING_IND	BLE user event indicating PAPS ringer setting indication, Refer sf_ble_prf_ringer_setting_ntf_t.
SF_BLE_PRF_PAPSC_EVENT_ALERT_STATUS_RES	BLE user event indicating PAPS alert status response, Refer sf_ble_prf_alert_status_ntf_t.

sf_ble_prf_papss_event_t

`sf_ble_prf_papss_event_t`

Detailed description

Profile Server user events

Enumerated values

Name	Description
SF_BLE_PRF_PAPSS_EVENT_NONE	Event not supported.
SF_BLE_PRF_PAPSS_EVENT_RINGER_CP_IND	BLE user event indicating ringer control point indication, Refer sf_ble_prf_ringer_cp_change_t.
SF_BLE_PRF_PAPSS_EVENT_CFG_NTF_IND	BLE user event indicating notification indication.

sf_ble_prf_ringer_cp_t

`sf_ble_prf_ringer_cp_t`

Detailed description

Ringer Control Point value

Enumerated values

Name	Description
SF_BLE_PRF_RINGER_CP_SILENT_MODE	Silent Mode.
SF_BLE_PRF_RINGER_CP_MUTE_ONCE	Mute Once.
SF_BLE_PRF_RINGER_CP_CANCEL_SILENT_MODE	Cancel Silent Mode.

sf_ble_prf_ringer_setting_t

```
sf_ble_prf_ringer_setting_t
```

Detailed description

Ringer Setting Value

Enumerated values

Name	Description
SF_BLE_PRF_RINGER_SETTING_SILENT	Ringer Silent.
SF_BLE_PRF_RINGER_SETTING_NORMAL	Ringer Normal.

sf_ble_prf_alert_setting_t

```
sf_ble_prf_alert_setting_t
```

Detailed description

PASP Alert status

Enumerated values

Name	Description
SF_BLE_PRF_ALERT_NONE	No active alert.
SF_BLE_PRF_ALERT_RINGER	Ringer State is active.
SF_BLE_PRF_ALERT_VIBRATOR	Vibrate State is active.
SF_BLE_PRF_ALERT_DISPLAY	Display Alert Status State is active.

sf_ble_prf_alert_status

```
typedef uint8_t sf_ble_prf_alert_status
```

Detailed description

PASP Alert Status

7.6.13.6 API Structures

sf_ble_prf_ringer_cp_change_t

sf_ble_prf_ringer_cp_change_t

Detailed description

Ringer Control Point value changed indication

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_ringer_cp_t ringer_cp

  Ringer control point value.

sf_ble_prf_ringer_setting_ntf_t

sf_ble_prf_ringer_setting_ntf_t

Detailed description

Ringer Setting notification received data

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_ringer_setting_t ringer_setting

  Ringer control point value.

sf_ble_prf_alert_status_ntf_t

sf_ble_prf_alert_status_ntf_t

Detailed description

Alert Status notification received data

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_alert_status alert_status

  Ringer control point value.

7.6.14 SF BLE Proximity Profile Framework Interface

RTOS-integrated SF BLE Proximity Profile Framework Interface.

7.6.14.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Proximity Profile Framework.

7.6.14.2 API Data

sf_ble_prf_pxpr_event_t

`sf_ble_prf_pxpr_event_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. PXP reporter (server) user events

Enumerated values

Name	Description
SF_BLE_PRF_PXPR_EVENT_NONE	unsupported event
SF_BLE_PRF_PXPR_EVENT_ALT_IND	BLE user event indicating PXPR profile alert indication.
SF_BLE_PRF_PXPR_EVENT_CMD_DISALLOWED_IND	command disallowed error received

sf_ble_prf_pxpm_event_t

`sf_ble_prf_pxpm_event_t`

Detailed description

PXP monitor (client) user events

Enumerated values

Name	Description
SF_BLE_PRF_PXPM_EVENT_NONE	unsupported event
SF_BLE_PRF_PXPM_EVENT_READ_CHAR_RESP	BLE user event indicating PXPR profile alert indication.
SF_BLE_PRF_PXPM_EVENT_CMD_DISALLOWED_IND	command disallowed error received
SF_BLE_PRF_PXPM_EVENT_ERROR_IND	error received for command issued

7.6.15 SF BLE Reference Time Update Service Profile Framework Interface

RTOS-integrated SF BLE Reference Time Update Service Profile Framework Interface.

7.6.15.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Reference Time Update Service Profile Framework.

7.6.15.2 Data structures

- sf_ble_prf_rtus_time_updt_state_t
- sf_ble_tip_cp_change_t

7.6.15.3 Variables

- sf_ble_prf_tip_time_ctrl_point_t

7.6.15.4 sf_ble_prf_tip_time_ctrl_point_t

```
SSP_HEADER typedef uint8_t sf_ble_prf_tip_time_
```

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Time Update Control point

7.6.15.5 API Structures

sf_ble_prf_rtus_time_updt_state_t

sf_ble_prf_rtus_time_updt_state_t

Detailed description

Reference Time Update State structure

Variables

- uint8_t current_state

  Current state of Reference time.

- uint8_t update_result

  Result of update.

sf_ble_tip_cp_change_t

sf_ble_tip_cp_change_t

Detailed description

Time Update Control Point Change Indication for Server

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_tip_time_ctrl_point_t control_point_value

  Time Update Control point value.

7.6.16 SF BLE Scan Parameters Service Profile Framework Interface

RTOS-integrated SF BLE Scan Parameters Service Profile Framework Interface.

7.6.16.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Scan Parameters Service Profile Framework.

7.6.16.2 Data structures

- sf_ble_prf_scps_scan_intv_t
- sf_ble_scps_scan_intv_change_t

7.6.16.3 Enumerations

- sf_ble_prf_scps_event_t

7.6.16.4 Typedefs

- sf_ble_prf_scps_scan_refresh_t

7.6.16.5 API Data

sf_ble_prf_scps_event_t

```
sf_ble_prf_scps_event_t
```

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Profile Server user events

Enumerated values

Name	Description
SF_BLE_PRF_SCPS_EVENT_NONE	Event not supported.
SF_BLE_PRF_SCPS_EVENT_INDNTF_IND	BLE Notification Setting Change Indication.
SF_BLE_PRF_SCPS_EVENT_CHG_EVT	BLE Scan Interval Changed Indication, Refer sf_ble_scps_scan_intv_change_t.

sf_ble_prf_scps_scan_refresh_t

```
typedef uint8_t sf_ble_prf_scps_scan_refresh_t
```

Detailed description

Scan Refresh Data value

7.6.16.6 API Structures

sf_ble_prf_scps_scan_intv_t

sf_ble_prf_scps_scan_intv_t

Detailed description

Scan interval window characteristic data

Variables

- uint16_t le_scan_interval

  scan interval value

- uint16_t le_scan_window

  scan window value

sf_ble_scps_scan_intv_change_t

sf_ble_scps_scan_intv_change_t

Detailed description

Scan interval window Change Indication for Sensor

Variables

- sf_ble_conn_handle_t conhdl

  Connection handle.

- sf_ble_prf_scps_scan_intv_t scan_interval_window_val

  Scan Interval window.

7.6.17 SF BLE Time Information Profile Framework Interface

RTOS-integrated SF BLE Time Information Profile Framework Interface.

7.6.17.1 Summary

This SSP Interface provides access to the ThreadX-aware SF BLE Time Information Profile Framework.

7.6.17.2 Data structures

- sf_ble_prf_tip_write_data_t

7.6.17.3 Enumerations

- sf_ble_prf_tipc_event_t
- sf_ble_prf_tips_event_t

7.6.17.4 API Data

sf_ble_prf_tipc_event_t

`sf_ble_prf_tipc_event_t`

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Time Profile Client events

Enumerated values

Name	Description
SF_BLE_PRF_TIPC_EVENT_NONE	Event not supported.
SF_BLE_PRF_TIPC_EVENT_READ_CHAR_RES	Read Char Complete Event.
SF_BLE_PRF_TIPC_EVENT_CURR_TIME_NTF	Current Time information received, Refer sf_ble_cts_curr_time_ntf_t.

sf_ble_prf_tips_event_t

`sf_ble_prf_tips_event_t`

Detailed description

Time Profile Server events

Enumerated values

Name	Description
SF_BLE_PRF_TIPS_EVENT_NONE	Event not supported.
SF_BLE_PRF_TIPS_EVENT_NTF_IND	CCCD Notification Received event.
SF_BLE_PRF_TIPS_EVENT_CP_IND	Time Update control point changed, Refer sf_ble_prf_tip_time_ctrl_point_t.

7.6.17.5 API Structures

sf_ble_prf_tip_write_data_t

Detailed description

Write Local Database Information

Variables

current_time

local_time

ref_time

next_dst

update_state

current_time

`sf_ble_prf_cts_curr_time_t::current_time`

Brief description

Current Time Information.

local_time

`sf_ble_cts_local_time_t::local_time`

Brief description

Local Time Information.

ref_time

`sf_ble_cts_ref_time_t::ref_time`

Brief description

Reference Time Information.

next_dst

`sf_ble_prf_ndcs_time_dst_t::next_dst`

Brief description

Next DST Information.

update_state

`sf_ble_prf_rtus_time_updt_state_t::update_state`

Brief description

Update Status Information.

7.7 Block Media Framework Interface

RTOS-integrated File system Interface to access Synergy block media devices.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

The interface provides an adaption layer from the FileX I/O to block media devices.

7.7.1 Summary

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

See also FileX Interface description: FileX Port Block Media Framework

7.7.2 Interface API

sf_block_media_api_t

Function name	Description
.open	Open a device channel for read/write and control.
.read	Read data from a media channel.
.write	Write data to a media channel.
.ioctl	Send control commands to and receives the status from the media port.
.close	Close the open media channel.
.versionGet	Return the version of the driver.

7.7.3 Data structures

- sf_block_media_cfg_t
- sf_block_media_instance_t

7.7.4 Typedefs

- sf_block_media_ctrl_t

7.7.5 Defines

- #define BLOCK_MEDIA_API_VERSION_MAJOR
  Initial value:`(1U)`
- #define BLOCK_MEDIA_API_VERSION_MINOR
  Initial value:`(2U)`

7.7.6 API Data

7.7.6.1 sf_block_media_ctrl_t

```
typedef void sf_block_media_ctrl_t
```

Detailed description

Block media framework control block. Allocate an instance specific control block to pass into the block media framework API calls. Implemented as

- sf_block_media_sdmmc_instance_ctrl_t

7.7.7 API Structures

7.7.7.1 sf_block_media_cfg_t

sf_block_media_cfg_t

Detailed description

Interface Configuration

Variables

- uint32_t block_size

  Block size in bytes.

- void const * p_extend

  Instance dependent configuration.

7.7.7.2 sf_block_media_api_t

sf_block_media_api_t

Detailed description

Shared Interface definition for Block Media

7.7.7.3 open

```
(* sf_block_media_api_t::open) (sf_block_media_ctrl_t *p_ctrl,
sf_block_media_cfg_t const *const p_cfg)
```

Detailed description

Open a device channel for read/write and control. Implemented as

- SF_Block_Media_SDMMC_Open

Table 66: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.

Parameter p_cfg

Definition: `sf_block_media_cfg_t const *const p_cfg`

Interface Configuration

- `sf_block_media_cfg_t`::`block_size`

  Block size in bytes.

- `sf_block_media_cfg_t`::`p_extend`

  Instance dependent configuration.

7.7.7.4 read

```
(* sf_block_media_api_t::read) (sf_block_media_ctrl_t *p_ctrl, uint8_t *const
p_dest, uint32_t const start_sector, uint32_t const sector_count)
```

Detailed description

Read data from a media channel. Implemented as

- SF_Block_Media_SDMMC_Read

Table 67: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.
p_dest	in	Destination address to read data out.
start_sector	in	Beginning sector address to read.
sector_count	in	Number of sectors to read.

Parameter p_cfg

Parameter p_dest

`uint8_t`

Parameter start_sector

`uint32_t`

Parameter sector_count

`uint32_t`

7.7.7.5 write

```
(* sf_block_media_api_t::write) (sf_block_media_ctrl_t *p_ctrl, uint8_t const
*const p_src, uint32_t const start_sector, uint32_t const sector_count)
```

Detailed description

Write data to a media channel. Implemented as

- SF_Block_Media_SDMMC_Write

Table 68: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.
p_src	in	Source address of data for writing.
start_sector	in	Beginning sector address to write to.
sector_count	in	Number of sectors to write.

Parameter p_cfg

Parameter p_src

`uint8_t`

Parameter start_sector

`uint32_t`

Parameter sector_count

`uint32_t`

7.7.7.6 ioctl

```
(* sf_block_media_api_t::ioctl) (sf_block_media_ctrl_t *p_ctrl, const command,
void *p_data)
```

Detailed description

Send control commands to and receives the status from the media port. Implemented as

- SF_Block_Media_SDMMC_Control

Table 69: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.
command	in	Command to execute.
p_data	inout	Void pointer to data in or out.

Parameter p_cfg

Parameter command

Parameter p_data

`const`

7.7.7.7 close

```
(* sf_block_media_api_t::close) (sf_block_media_ctrl_t *p_ctrl)
```

Detailed description

Close the open media channel. Implemented as

- SF_Block_Media_SDMMC_Close

Table 70: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.

Parameter p_cfg

7.7.7.8 versionGet

```
(* sf_block_media_api_t::versionGet) ( *const p_version)
```

Detailed description

Return the version of the driver. Implemented as

- SF_Block_Media_SDMMC_VersionGet

Table 71: Parameters

Name	Direction	Description
p_cfg	in	Pointer to the media configuration structure for a channel.
p_version	out	Memory address to return version information to.

Parameter p_cfg

Parameter p_version

7.7.7.9 sf_block_media_instance_t

sf_block_media_instance_t

Detailed description

Interface Instance

Variables

- sf_block_media_ctrl_t * p_ctrl

  Block media pointer to device driver control structure.
- sf_block_media_cfg_t const * p_cfg

  Block media pointer to device driver configuration structure.
- sf_block_media_api_t const * p_api

  Block media pointer to device driver api structure.

7.8 SF CELLULAR Framework Interface

RTOS-integrated SF CELUULAR Framework Interface.

7.8.1 Summary

This SSP Interface provides access to the ThreadX-aware SF CELLULAR Framework.

7.8.2 Interface API

sf_cellular_api_t

Function name	Description
.open	Initializes and enables the Cellular module.
.close	Disables the Cellular module.
.provisioningGet	Pointer to function to Get the Cellular module provisioning information.
.provisioningSet	Pointer to function to Set the Cellular module's provisioning information.
.infoGet	Reads the Cellular module's information.
.statisticsGet	Returns statistics information of Cellular module.
.transmit	Passes packet buffer to PPP stack for transmission.
.versionGet	Gets version and stores it in provided pointer p_version.
.networkConnect	Initiates the Data connection.
.networkDisconnect	Terminates the Data connection.
.networkStatusGet	Get Network Status information.
.simPinSet	Set SIM Pin.
.simLock	Locks SIM.
.simUnlock	Unlocks SIM.
.simIDGet	Gets the SIM ID.
.fotaCheck	Checks for Available Firmware upgrade.
.fotaStart	Starts the Firmware upgrade.
.fotaStop	Stops the Firmware upgrade.
.reset	Reset cellular module. This reset() API will only work when module is opened.

7.8.3 Data structures

- sf_cellular_provisioning_t

- sf_cellular_ctrl_t
- sf_cellular_stats_t
- sf_cellular_info_t
- sf_cellular_network_status_t
- sf_cellular_callback_args_t
- sf_cellular_op_t
- sf_cellular_at_cmd_set_t
- sf_cellular_cfg_t
- sf_cellular_instance_t

7.8.4 Enumerations

- sf_cellular_op_select_mode_t
- sf_cellular_timezone_update_mode_t
- sf_cellular_event_t
- sf_cellular_reset_type_t
- sf_cellular_airplane_mode_t
- sf_cellular_pdp_type_t
- sf_cellular_op_name_format_t
- sf_cellular_auth_type_t
- sf_cellular_at_cmd_index_t

7.8.5 Defines

- #define SF_CELLULAR_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file
- #define SF_CELLULAR_API_VERSION_MINOR

  Initial value:`(1U)`

  Minor Version of the API defined in this file
- #define SF_CELLULAR_MAX_OPERATOR_NAME_LEN

  Initial value:`(32U)`

  Maximum length for Operator name.

- #define SF_CELLULAR_MAX_PREFFERED_OPERATOR_COUNT

  Initial value:`(5U)`

  Maximum number of preferred operator.
- #define SF_CELLULAR_IMEI_LEN

  Initial value:`(16U)`

  Maximum length of IMEI number.
- #define SF_CELLULAR_FWVERSION_LEN

  Initial value:`(32U)`

  Maximum length of Firmware Version.
- #define SF_CELLULAR_MAX_STRING_LEN

  Initial value:`(32U)`

  Maximum string length.
- #define SF_CELLULAR_CHIPSET_LEN

  Initial value:`(16U)`

  Maximum length of Chipset info.
- #define SF_CELLULAR_MFG_NAME_LEN

  Initial value:`(16U)`

  Maximum length of manufacturer.
- #define SF_CELLULAR_CID_LEN

  Initial value:`(16U)`

  Maximum length of CID.
- #define SF_CELLULAR_IMSI_LEN

  Initial value:`(24U)`

  Maximum length of IMSI.
- #define SF_CELLULAR_IP_ADDR_LEN

  Initial value:`(128U)`

  Maximum lenfth of IP address.
- #define SF_CELLULAR_TRUE

  Initial value:`(1U)`

  Logical Value TRUE for Cellular
- #define SF_CELLULAR_FALSE

  Initial value:`(0U)`

  Logical Value FALSE for Cellular

7.8.6 API Data

7.8.6.1 sf_cellular_op_select_mode_t

`sf_cellular_op_select_mode_t`

Detailed description

Operator selection mode

Enumerated values

Name	Description
SF_CELLULAR_OP_SELECT_MODE_AUTO	Automatic Operator selection.
SF_CELLULAR_OP_SELECT_MODE_MANUAL	Manual Operator selection.
SF_CELLULAR_OP_SELECT_MODE_DEREGISTER	De-register from the network.
SF_CELLULAR_OP_SELECT_MODE_MANUAL_FALLBACK	Manual with fallback to automatic.

7.8.6.2 sf_cellular_timezone_update_mode_t

`sf_cellular_timezone_update_mode_t`

Detailed description

Timezone update mode

Enumerated values

Name	Description
SF_CELLULAR_TIMEZONE_UPDATE_AUTO_DISABLE	Disable automatic time zone update.
SF_CELLULAR_TIMEZONE_UPDATE_AUTO_ENABLE	Enable automatic time zone update.

7.8.6.3 sf_cellular_event_t

`sf_cellular_event_t`

Detailed description

Cellular Framework event codes

Enumerated values

Name	Description
SF_CELLULAR_EVENT_RX	Packet received event.
SF_CELLULAR_EVENT_PROVISIONSET	Provisioning Set event.

7.8.6.4 sf_cellular_reset_type_t

`sf_cellular_reset_type_t`

Detailed description

Cellular Module reset type

Enumerated values

Name	Description
SF_CELLULAR_RESET_TYPE_SOFT	Soft reset module using AT command.
SF_CELLULAR_RESET_TYPE_HARD	Hard reset module by toggling Reset Pin.

7.8.6.5 sf_cellular_airplane_mode_t

`sf_cellular_airplane_mode_t`

Detailed description

Airplane mode

Enumerated values

Name	Description
SF_CELLULAR_AIRPLANE_MODE_OFF	Airplane mode disabled.
SF_CELLULAR_AIRPLANE_MODE_ON	Airplane mode enabled.

7.8.6.6 sf_cellular_pdp_type_t

`sf_cellular_pdp_type_t`

Detailed description

PDP type

Enumerated values

Name	Description
SF_CELLULAR_PDP_TYPE_IP	Internet protocol.
SF_CELLULAR_PDP_TYPE_PPP	Point to point protocol.
SF_CELLULAR_PDP_TYPE_IPV6	Internet protocol, version 6.
SF_CELLULAR_PDP_TYPE_IPV4V6	Virtual introduced to handle dual stack UE capability.

7.8.6.7 sf_cellular_op_name_format_t

`sf_cellular_op_name_format_t`

Detailed description

Cellular operator name format

Enumerated values

Name	Description
SF_CELLULAR_OP_NAME_FORMAT_LONG	Long alphanumeric.
SF_CELLULAR_OP_NAME_FORMAT_SHORT	Short alphanumeric.
SF_CELLULAR_OP_NAME_FORMAT_NUMERIC	Numeric.

7.8.6.8 sf_cellular_auth_type_t

`sf_cellular_auth_type_t`

Detailed description

Cellular authentication type

Enumerated values

Name	Description
SF_CELLULAR_AUTH_TYPE_NONE	No authentication.
SF_CELLULAR_AUTH_TYPE_PAP	PAP.
SF_CELLULAR_AUTH_TYPE_CHAP	CHAP.

7.8.6.9 sf_cellular_at_cmd_index_t

`sf_cellular_at_cmd_index_t`

Detailed description

Enumeration for AT command index

Enumerated values

Name	Description
SF_CELLULAR_AT_CMD_INDEX_AT	Index for Command AT.
SF_CELLULAR_AT_CMD_INDEX_ATZ0	Index for Command ATZ0.
SF_CELLULAR_AT_CMD_INDEX_AT_CREG_SET_0	Index for Command to set AT+CREG.
SF_CELLULAR_AT_CMD_INDEX_AT_CMEE_SET_0	Index for Command to set AT+CMEE.
SF_CELLULAR_AT_CMD_INDEX_AT_ECHO	Index for Command ATE.
SF_CELLULAR_AT_CMD_INDEX_AT_SAVE	Index for Command AT&W.
SF_CELLULAR_AT_CMD_INDEX_AT_CREG	Index for Command AT+CREG.
SF_CELLULAR_AT_CMD_INDEX_AT_CPIN_STATUS_GET	Index for Command to get status of SIM lock.
SF_CELLULAR_AT_CMD_INDEX_AT_ENTER_CPIN	Index for Command to unlock SIM.
SF_CELLULAR_AT_CMD_INDEX_AT_CPIN_SET	Index for Command to set SIM PIN.
SF_CELLULAR_AT_CMD_INDEX_AT_CGDCONT_SET	Index for Command to set AT+CGDCOND.
SF_CELLULAR_AT_CMD_INDEX_AT_CPOL_SET	Index for Command to set AT+CPOL.
SF_CELLULAR_AT_CMD_INDEX_AT_COPS_AUTO_SET	Index for Command to set AUTO AT+COPS.
SF_CELLULAR_AT_CMD_INDEX_AT_COPS_MANUAL_SET	Index for Command to set Manual AT+COPS.
SF_CELLULAR_AT_CMD_INDEX_AT_AIRPLANE_OFF	Index for Command to set Airplane mode OFF.
SF_CELLULAR_AT_CMD_INDEX_AT_AIRPLANE_ON	Index for Command to set Airplane mode ON.
SF_CELLULAR_AT_CMD_INDEX_AT_CONTEXT_ACTIVE	Index for Command to activate context.
SF_CELLULAR_AT_CMD_INDEX_AT_CONTEXT_DEACTIVE	Index for Command to deactivate context.

Name	Description
SF_CELLULAR_AT_CMD_INDEX_AT_CGDATA_ACTIVE	Index for Command to activate Data mode.
SF_CELLULAR_AT_CMD_INDEX_AT_CGDATA_DEACTIVE	Index for Command to deactivate Data mode.
SF_CELLULAR_AT_CMD_INDEX_AT_CSQ_GET	Index for Command to get signal quality.
SF_CELLULAR_AT_CMD_INDEX_AT_VER_GET	Index for Command to get Modem stack Version.
SF_CELLULAR_AT_CMD_INDEX_AT_CHIPSET_GET	Index for Command to get chipset details.
SF_CELLULAR_AT_CMD_INDEX_AT_IMEI_GET	Index for Command to get IMEI number.
SF_CELLULAR_AT_CMD_INDEX_AT_MANF_NAME_GET	Index for Command to get manufacturer name.
SF_CELLULAR_AT_CMD_INDEX_AT_SIMID_GET	Index for Command to get SIM Card ID.
SF_CELLULAR_AT_CMD_INDEX_AT_NET_TYPE_STATUS_GET	Index for Command to get network type information.
SF_CELLULAR_AT_CMD_INDEX_AT_NET_STATUS_GET	Index for Command to get network status information.
SF_CELLULAR_AT_CMD_INDEX_AT_LOCK_SIM	Index for Command to lock SIM card.
SF_CELLULAR_AT_CMD_INDEX_AT_UNLOCK_SIM	Index for Command to unlock SIM card.
SF_CELLULAR_AT_CMD_INDEX_AT_ENTER_DATA_MODE	Index for Command to enter data mode.
SF_CELLULAR_AT_CMD_INDEX_AT_EXIT_DATA_MODE	Index for Command to exit data mode.
SF_CELLULAR_AT_CMD_INDEX_AT_USERNAME_SET	Index for Command to set username.
SF_CELLULAR_AT_CMD_INDEX_AT_PASSWORD_SET	Index for Command to set password.
SF_CELLULAR_AT_CMD_INDEX_AT_AUTH_TYPE_SET	Index for Command to set authorisation type.
SF_CELLULAR_AT_CMD_INDEX_AT_AUTO_TIME_UPDATE_ENABLE_SET	Index for Command to enable auto time update.
SF_CELLULAR_AT_CMD_INDEX_AT_AUTO_TIME_UPDATE_DISABLE_SET	Index for Command to disable auto time update.
SF_CELLULAR_AT_CMD_INDEX_AT_EMPTY_APN_SET	Index for Command to set empty APN.
SF_CELLULAR_AT_CMD_INDEX_AT_GET_IP_ADDR	Index for Command to get IP address.

7.8.7 API Structures

7.8.7.1 sf_cellular_provisioning_t

sf_cellular_provisioning_t

Detailed description

Cellular Provisioning information structure

Variables

- uint8_t apn[SF_CELLULAR_MAX_STRING_LEN]

  Access Point Name.
- sf_cellular_auth_type_t auth_type

  Authentication type: PAP/CHAP.
- uint8_t username[SF_CELLULAR_MAX_STRING_LEN]

  User name used for authentication.
- uint8_t password[SF_CELLULAR_MAX_STRING_LEN]

  Password used for authentication.
- sf_cellular_airplane_mode_t airplane_mode

  Airplane mode.
- uint8_t context_id

  Context ID to be used for connection.
- sf_cellular_pdp_type_t pdp_type

  PDP Type for Context.

7.8.7.2 sf_cellular_ctrl_t

sf_cellular_ctrl_t

Detailed description

Cellular Framework control structure

Variables

- void * p_driver_handle

  Stores information required by underlying Cellular device driver.

7.8.7.3 sf_cellular_stats_t

sf_cellular_stats_t

Detailed description

The statistic and error counters for this instance

Variables

- uint32_t rx_bytes

  Bytes received successfully.
- uint32_t tx_bytes

  Bytes transmitted successfully.
- uint32_t rx_err

  Bytes receive errors.
- uint32_t tx_err

  Bytes transmit errors.

7.8.7.4 sf_cellular_info_t

sf_cellular_info_t

Detailed description

Cellular Driver Information

Variables

- uint8_t mfg_name[SF_CELLULAR_MFG_NAME_LEN]

  Manufacturer name.
- uint8_t chipset[SF_CELLULAR_CHIPSET_LEN]

  Pointer to string showing Cellular chipset/driver information.
- uint8_t fw_version[SF_CELLULAR_FWVERSION_LEN]

  Cellular firmware version.
- uint8_t imei[SF_CELLULAR_IMEI_LEN]

  IMEI number.
- uint16_t rssi

  Received signal strength indication.
- uint16_t ber

  Bit rate error.
- uint8_t ip_addr[SF_CELLULAR_IP_ADDR_LEN]

  IP address.

7.8.7.5 sf_cellular_network_status_t

sf_cellular_network_status_t

Detailed description

Network Status information

Variables

- uint16_t country_code

  Country code.
- uint16_t operator_code

  Operator code.
- int16_t rssi

  RSSI.
- uint8_t cid[SF_CELLULAR_CID_LEN]

  Cell ID.
- uint8_t imsi[SF_CELLULAR_IMSI_LEN]

  IMSI.
- uint8_t op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]

  Operator name.
- uint8_t service_domain

  Service Domain.
- uint8_t active_band

  Active Band.

7.8.7.6 sf_cellular_callback_args_t

sf_cellular_callback_args_t

Detailed description

Callback structure for Cellular driver to get the data receive notification

Variables

- sf_cellular_event_t event

  Event Code.
- uint8_t * p_data

  Pointer to received data.
- uint32_t length

  Receive Data Length.
- void const * p_context

  Context Provided to user during callback.

7.8.7.7 sf_cellular_op_t

sf_cellular_op_t

Detailed description

Preferred operator selection structure

Variables

- sf_cellular_op_name_format_t op_name_format

  Cellular operator name format.

- uint8_t op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]

  Cellular operator name.

7.8.7.8 sf_cellular_at_cmd_set_t

sf_cellular_at_cmd_set_t

Detailed description

Structure defining AT commands parameters

Variables

- uint8_t * p_cmd

  AT Command.

- uint8_t * p_success_resp

  Success response string.

- uint16_t max_resp_length

  Maximum length of expected response.

- uint8_t retry

  Retry count.

- uint16_t retry_delay

  Delay between AT command retry.

7.8.7.9 sf_cellular_cfg_t

sf_cellular_cfg_t

Detailed description

Define the Cellular configuration parameters

Variables

- sf_cellular_op_select_mode_t op_select_mode

  Cellular Operator selection mode.

- sf_cellular_op_t op

  Cellular operator. Valid when operator selection mode is manual.

- uint16_t num_pref_ops

  Number of preferred operators in the pref_ops array.

- sf_cellular_op_t pref_ops[SF_CELLULAR_MAX_PREFFERED_OPERATOR_COUNT]

  Array of structures describing preferred operators.

- sf_cellular_timezone_update_mode_t tz_upd_mode

  TimeZone update mode policy.

- uint8_t * p_sim_pin

  SIM Pin.

- uint8_t * p_puk_pin

  PUK Pin.

- ssp_err_t(* p_prov_callback)(sf_cellular_callback_args_t *p_args)

  Pointer to provisioning callback function, used in NSAL

- void(* p_recv_callback)(sf_cellular_callback_args_t *p_args)

  This is the receive callback function used by NetX which will take a data packet from the Cellular module and hand it over to NetX for further processing.

- void const * p_context

  User defined context passed into callback function.

- void const * p_extend

  Instance specific configuration.

- sf_cellular_at_cmd_set_t const * p_cmd_set

  Instance specific command set.

7.8.7.10 sf_cellular_api_t

sf_cellular_api_t

Detailed description

Cellular Framework API structure.

7.8.7.11 open

```
ssp_err_t(* sf_cellular_api_t::open) (sf_cellular_ctrl_t *p_ctrl,
sf_cellular_cfg_t const *const p_cfg)
```

Brief description

Initializes and enables the Cellular module.

Detailed description

Table 72: Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to Cellular configuration structure.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_cfg

Definition: `sf_cellular_cfg_t const *const p_cfg`

Define the Cellular configuration parameters

- `sf_cellular_cfg_t`::`sf_cellular_op_select_mode_t`

  Cellular Operator selection mode.

  Enumerated as:

  – SF_CELLULAR_OP_SELECT_MODE_AUTO

  – SF_CELLULAR_OP_SELECT_MODE_MANUAL

  – SF_CELLULAR_OP_SELECT_MODE_DEREGISTER

  – SF_CELLULAR_OP_SELECT_MODE_MANUAL_FALLBACK
- `sf_cellular_cfg_t`::`sf_cellular_op_t`

  Cellular operator. Valid when operator selection mode is manual.
- `sf_cellular_cfg_t`::`num_pref_ops`

  Number of preferred operators in the pref_ops array.
- `sf_cellular_cfg_t`::`sf_cellular_op_t`

  Array of structures describing preferred operators.
- `sf_cellular_cfg_t`::`sf_cellular_timezone_update_mode_t`

  TimeZone update mode policy.

  Enumerated as:

  – SF_CELLULAR_TIMEZONE_UPDATE_AUTO_DISABLE

  – SF_CELLULAR_TIMEZONE_UPDATE_AUTO_ENABLE

- sf_cellular_cfg_t::p_sim_pin

  SIM Pin.

- sf_cellular_cfg_t::p_puk_pin

  PUK Pin.

- sf_cellular_cfg_t::ssp_err_t

  Pointer to provisioning callback function, used in NSAL

- sf_cellular_cfg_t::p_recv_callback

  This is the receive callback function used by NetX which will take a data packet from the Cellular module and hand it over to NetX for further processing.

- sf_cellular_cfg_t::p_context

  User defined context passed into callback function.

- sf_cellular_cfg_t::p_extend

  Instance specific configuration.

- sf_cellular_cfg_t::sf_cellular_at_cmd_set_t

  Instance specific command set.

7.8.7.12 close

```
ssp_err_t(* sf_cellular_api_t::close) (sf_cellular_ctrl_t *const p_ctrl)
```

Brief description

Disables the Cellular module.

Detailed description

Table 73: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module. .

Parameter p_ctrl

Definition: sf_cellular_ctrl_t

Cellular Framework control structure

7.8.7.13 provisioningGet

```
ssp_err_t(* sf_cellular_api_t::provisioningGet) (sf_cellular_ctrl_t *const
p_ctrl, sf_cellular_provisioning_t *const p_cellular_provisioning)
```

Brief description

Pointer to function to Get the Cellular module provisioning information.

Detailed description

Table 74: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_cellular_provisioning	out	Pointer to Cellular provisioning structure.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_cellular_provisioning

Definition: `sf_cellular_provisioning_t*const p_cellular_provisioning`

Cellular Provisioning information structure

- `sf_cellular_provisioning_t::apn`

  Access Point Name.
- `sf_cellular_provisioning_t::auth_type`

  Authentication type: PAP/CHAP.
- `sf_cellular_provisioning_t::username`

  User name used for authentication.
- `sf_cellular_provisioning_t::password`

  Password used for authentication.
- `sf_cellular_provisioning_t::airplane_mode`

  Airplane mode.
- `sf_cellular_provisioning_t::context_id`

  Context ID to be used for connection.
- `sf_cellular_provisioning_t::pdp_type`

  PDP Type for Context.

7.8.7.14 provisioningSet

```
ssp_err_t(* sf_cellular_api_t::provisioningSet) (sf_cellular_ctrl_t *const
p_ctrl, sf_cellular_provisioning_t const *const p_cellular_provisioning)
```

Brief description

Pointer to function to Set the Cellular module's provisioning information.

Detailed description

Table 75: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_cellular_provisioning	in	Pointer to Cellular provisioning structure.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_cellular_provisioning

Definition: `sf_cellular_provisioning_tconst *const p_cellular_provisioning`

Cellular Provisioning information structure

- `sf_cellular_provisioning_t::apn`

  Access Point Name.
- `sf_cellular_provisioning_t::auth_type`

  Authentication type: PAP/CHAP.
- `sf_cellular_provisioning_t::username`

  User name used for authentication.
- `sf_cellular_provisioning_t::password`

  Password used for authentication.
- `sf_cellular_provisioning_t::airplane_mode`

  Airplane mode.
- `sf_cellular_provisioning_t::context_id`

  Context ID to be used for connection.
- `sf_cellular_provisioning_t::pdp_type`

  PDP Type for Context.

7.8.7.15 infoGet

```
ssp_err_t(* sf_cellular_api_t::infoGet) (sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_info_t *const p_cellular_info)
```

Brief description

Reads the Cellular module's information.

Detailed description

Table 76: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_cellular_info	out	Pointer to Cellular info structure.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_cellular_info

Definition: `sf_cellular_info_t*const p_cellular_info`

Cellular Driver Information

- `sf_cellular_info_t::mfg_name`

  Manufacturer name.
- `sf_cellular_info_t::chipset`

  Pointer to string showing Cellular chipset/driver information.
- `sf_cellular_info_t::fw_version`

  Cellular firmware version.
- `sf_cellular_info_t::imei`

  IMEI number.
- `sf_cellular_info_t::rssi`

  Received signal strength indication.
- `sf_cellular_info_t::ber`

  Bit rate error.
- `sf_cellular_info_t::ip_addr`

  IP address.

7.8.7.16 statisticsGet

```
ssp_err_t(* sf_cellular_api_t::statisticsGet) (sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_stats_t *const p_cellular_device_stats)
```

Brief description

Returns statistics information of Cellular module.

Detailed description

Table 77: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_cellular_device_stats	out	Pointer to Cellular statistics information structure.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_cellular_device_stats

Definition: `sf_cellular_stats_t*const p_cellular_device_stats`

The statistic and error counters for this instance

- `sf_cellular_stats_t::rx_bytes`

  Bytes received successfully.
- `sf_cellular_stats_t::tx_bytes`

  Bytes transmitted successfully.
- `sf_cellular_stats_t::rx_err`

  Bytes receive errors.
- `sf_cellular_stats_t::tx_err`

  Bytes transmit errors.

7.8.7.17 transmit

```
ssp_err_t(* sf_cellular_api_t::transmit) (sf_cellular_ctrl_t *const p_ctrl,
uint8_t *const p_buf, uint32_t length)
```

Brief description

Passes packet buffer to PPP stack for transmission.

Detailed description

Table 78: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_buf	in	Pointer to packet buffer to transmit
length	in	Length of packet buffer

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_buf

`uint8_t`

Parameter length

`uint32_t`

7.8.7.18 versionGet

```
ssp_err_t(* sf_cellular_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 79: Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

7.8.7.19 networkConnect

```
ssp_err_t(* sf_cellular_api_t::networkConnect) (sf_cellular_ctrl_t *const p_ctrl)
```

Brief description

Initiates the Data connection.

Detailed description

Table 80: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

7.8.7.20 networkDisconnect

```
ssp_err_t(* sf_cellular_api_t::networkDisconnect) (sf_cellular_ctrl_t *const
p_ctrl)
```

Brief description

Terminates the Data connection.

Detailed description

Table 81: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

7.8.7.21 networkStatusGet

```
ssp_err_t(* sf_cellular_api_t::networkStatusGet) (sf_cellular_ctrl_t *const
p_ctrl, sf_cellular_network_status_t *p_network_status)
```

Brief description

Get Network Status information.

Detailed description

Table 82: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_network_status	out	Pointer to Network Status structure

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_network_status

Definition: `sf_cellular_network_status_t*p_network_status`

Network Status information

- `sf_cellular_network_status_t::country_code`
  Country code.
- `sf_cellular_network_status_t::operator_code`
  Operator code.
- `sf_cellular_network_status_t::rssi`
  RSSI.
- `sf_cellular_network_status_t::cid`
  Cell ID.
- `sf_cellular_network_status_t::imsi`
  IMSI.
- `sf_cellular_network_status_t::op_name`
  Operator name.
- `sf_cellular_network_status_t::service_domain`
  Service Domain.
- `sf_cellular_network_status_t::active_band`
  Active Band.

7.8.7.22 simPinSet

```
ssp_err_t(* sf_cellular_api_t::simPinSet) (sf_cellular_ctrl_t *const p_ctrl,
uint8_t *const p_old_pin, uint8_t *const p_new_pin)
```

Brief description

Set SIM Pin.

Detailed description

Table 83: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_old_pin	in	Pointer to char array containing current 4 digit pin.
p_new_pin	in	Pointer to char array containing new 4 digit pin.

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_old_pin

`uint8_t`

Parameter p_new_pin

`uint8_t`

7.8.7.23 simLock

```
ssp_err_t(* sf_cellular_api_t::simLock) (sf_cellular_ctrl_t *const p_ctrl,
uint8_t *const p_pin)
```

Brief description

Locks SIM.

Detailed description

Table 84: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_pin	in	PIN number to lock the SIM

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_pin

`uint8_t`

7.8.7.24 simUnlock

```
ssp_err_t(* sf_cellular_api_t::simUnlock) (sf_cellular_ctrl_t *const p_ctrl,
uint8_t *const p_pin)
```

Brief description

Unlocks SIM.

Detailed description

Table 85: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_pin	in	PIN number to unlock the SIM

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_pin

`uint8_t`

7.8.7.25 simIDGet

```
ssp_err_t(* sf_cellular_api_t::simIDGet) (sf_cellular_ctrl_t *const p_ctrl,
uint8_t *p_sim_id)
```

Brief description

Gets the SIM ID.

Detailed description

Table 86: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_sim_id	out	SIM ID

Parameter p_ctrl

Definition: sf_cellular_ctrl_t

Cellular Framework control structure

Parameter p_sim_id

uint8_t

7.8.7.26 fotaCheck

```
ssp_err_t(* sf_cellular_api_t::fotaCheck) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotacheck)
```

Brief description

Checks for Available Firmware upgrade.

Detailed description

Table 87: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_fotacheck	in	Pointer to fota check specific data structure

Parameter p_ctrl

Definition: sf_cellular_ctrl_t

Cellular Framework control structure

Parameter p_fotacheck

const

7.8.7.27 fotaStart

```
ssp_err_t(* sf_cellular_api_t::fotaStart) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotastart)
```

Brief description

Starts the Firmware upgrade.

Detailed description

Table 88: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_fotastart	in	Pointer to fota start specific data structure

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_fotastart

`const`

7.8.7.28 fotaStop

```
ssp_err_t(* sf_cellular_api_t::fotaStop) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotastop)
```

Brief description

Stops the Firmware upgrade.

Detailed description

Table 89: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
p_fotastop	in	Pointer to fota stop specific data structure

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter p_fotastop

`const`

7.8.7.29 reset

```
ssp_err_t(* sf_cellular_api_t::reset) (sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_reset_type_t reset_type)
```

Brief description

Reset cellular module. This reset API will only work when module is opened.

Detailed description

Table 90: Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Cellular module.
reset_type	in	Reset type

Parameter p_ctrl

Definition: `sf_cellular_ctrl_t`

Cellular Framework control structure

Parameter reset_type

7.8.7.30 sf_cellular_instance_t

sf_cellular_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_cellular_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_cellular_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_cellular_api_t const * p_api

  Pointer to the API structure for this instance.

7.9 SF CELLULAR NSAL Framework Interface

RTOS-integrated SF CELUULAR NSAL Framework Interface.

7.9.1 Summary

This SSP Interface provides access to the ThreadX-aware SF CELLULAR NSAL Framework.

7.9.2 Functions

- sf_cellular_deinit

7.9.3 Data structures

- sf_cellular_nsal_cfg_t

7.9.4 sf_cellular_deinit

```
ssp_err_t sf_cellular_deinit ( NX_IP_DRIVER  * driver_req_ptr ,
 sf_cellular_instance_t  const * p_cellular_instance ,  sf_cellular_nsal_cfg_t
* p_cellular_nsal_cfg )
```

7.9.4.1 Detailed description

Function Prototypes

7.9.4.2 Function steps

- Disconnect Network
- Close Cellular Driver
- Return an error to NetX.

7.9.5 API Structures

7.9.5.1 sf_cellular_nsal_cfg_t

sf_cellular_nsal_cfg_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Define the NSAL configuration parameters

Variables

- uint8_t * p_ppp_stack

  PPP Stack.

- uint32_t ppp_stack_size

  PPP Stack size.

- UINT priority

  PPP Thread Priority.
- NX_PPP * p_ppp

  NetX PPP Interface.
- NX_IP * p_ip

  NetX IP Interface.
- NX_PACKET_POOL * p_ppp_packet_pool

  NetX Packet pool for internal.
- void(* p_ppp_invalid_packet_cb)(NX_PACKET *p_packet_ptr)

  Callback handler for Invalid packet.
- void(* p_ppp_send_byte)(UCHAR byte)

  PPP Send byte callback function.
- void(* p_link_down_cb)(NX_PPP *p_ppp_ptr)

  PPP Link down notification callback.

  Link Notification callback function
- void(* p_link_up_cb)(NX_PPP *p_ppp_ptr)

  PPP Link up notification callback.
- sf_cellular_auth_type_t auth_type

  Authentication Type.
- UINT(* p_chap_get_challenge_cb)(CHAR *p_rand_value, CHAR *p_id, CHAR *p_name)

  Get challenge notification callback.

  CHAP Callback Function
- UINT(* p_chap_get_responder_cb)(CHAR *p_system, CHAR *p_name, CHAR *p_secret)

  Get Responder notification callback.
- UINT(* p_chap_get_verify_cb)(CHAR *p_system, CHAR *p_name, CHAR *p_secret)

  Get Chap verification callback.
- UINT(* p_pap_generate_login)(CHAR *p_name, CHAR *p_password)

  PAP Authentication generate login callback.

  PAP Callback Function
- UINT(* p_pap_verify_login)(CHAR *p_name, CHAR *p_password)

  PAP authentication verification callback.
- uint32_t local_ip

  Local IP Address.

- uint32_t peer_ip

  Peer IP Address.
- void const * p_extend

  Instance specific configuration.

7.10 SF_CELLULAR_COMMON

SF_Cellular Framework API Common Code.
Cellular Framework header files Framework header files for this package HAL Layer communication interface header file

7.10.1 Functions

- SF_CELLULAR_COMMON_Open
- SF_CELLULAR_COMMON_Close
- SF_CELLULAR_COMMON_InfoGet
- SF_CELLULAR_COMMON_StatisticsGet
- SF_CELLULAR_COMMON_Transmit
- SF_CELLULAR_COMMON_ProvisioningGet
- SF_CELLULAR_COMMON_ProvisioningSet
- SF_CELLULAR_COMMON_NetworkConnect
- SF_CELLULAR_COMMON_NetworkDisconnect
- SF_CELLULAR_COMMON_NetworkStatusGet
- SF_CELLULAR_COMMON_SimPinSet
- SF_CELLULAR_COMMON_SimLock
- SF_CELLULAR_COMMON_SimUnlock
- SF_CELLULAR_COMMON_SimIDGet
- SF_CELLULAR_COMMON_FotaCheck
- SF_CELLULAR_COMMON_FotaStart
- SF_CELLULAR_COMMON_FotaStop

7.10.2 SF_CELLULAR_COMMON_Open

```
ssp_err_t SF_CELLULAR_COMMON_Open ( sf_cellular_ctrl_t  * p_ctrl ,
 sf_cellular_cfg_t  const *const p_cfg )
```

7.10.2.1 Brief description

Initialize Cellular NimbeLink Cellular driver.

7.10.2.2 Detailed description

Implements open Initializes NimbeLink Cellular Driver and Configure the parameters as per the p_cfg Update global variables for future use.

Table 91: Parameters

Name	Direction	Description
p_ctrl	out	Cellular control block
p_cfg	in	Cellular configuration structure

Table 92: Return values

Name	Description
SSP_SUCCESS	Driver initialization successfully.
SSP_ERR_ALREADY_OPEN	Cellular NimbeLink CELR Driver is already opened.
SSP_ERR_CELLULAR_CONFIG_FAILED	Cellular NimbeLink CELR module Configuration failed
SSP_ERR_CELLULAR_INIT_FAILED	Cellular NimbeLink CELR module initialization failed
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Driver initialization failed
SSP_ERR_IN_USE	Device already in used

7.10.2.3 Function steps

- Get Mutex Lock
- Set Cellular configuration

7.10.3 SF_CELLULAR_COMMON_Close

```
ssp_err_t SF_CELLULAR_COMMON_Close ( sf_cellular_ctrl_t  *const p_ctrl )
```

7.10.3.1 Brief description

Stop Cellular NimbeLink Cellular driver functionality.

7.10.3.2 Detailed description

Implements close This function deactivates the PDP context and Update global variables for future use

Table 93: Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block

Table 94: Return values

Name	Description
SSP_SUCCESS	Cellular Driver stop successfully.
SSP_ERR_NOT_OPEN	Device is not opened.
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Driver un-initialization failed
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.3.3 Function steps

- Get cellular information
- Get Mutex Lock
- De-Activate the PDP context

7.10.4 SF_CELLULAR_COMMON_InfoGet

```
ssp_err_t SF_CELLULAR_COMMON_InfoGet ( sf_cellular_ctrl_t  *const p_ctrl ,
 sf_cellular_info_t  *const p_cellular_info )
```

7.10.4.1 Brief description

Get Cellular module information.

7.10.4.2 Detailed description

Implements infoGet Get Cellular module information like chipset/driver information, RSSI, noise level, link quality

Table 95: Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_cellular_info	in	Cellular information structure

Table 96: Return values

Name	Description
SSP_SUCCESS	Successfully get the Cellular information
SSP_ERR_NOT_OPEN	Driver not opened.
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Failed reading Cellular information
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.4.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.5 SF_CELLULAR_COMMON_StatisticsGet

```
ssp_err_t SF_CELLULAR_COMMON_StatisticsGet ( sf_cellular_ctrl_t  *const p_ctrl ,
 sf_cellular_stats_t  *const p_cellular_device_stats )
```

7.10.5.1 Brief description

Get the interface statistics.

7.10.5.2 Detailed description

Implements statisticsGet Collect the statistics information of Cellular interface

Table 97: Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_cellular_device_stats	in	Cellular statistic structure

Table 98: Return values

Name	Description
SSP_SUCCESS	Successfully get the Statistics information.
SSP_ERR_UNSUPPORTED	Functionality is not supported.
SSP_ERR_NOT_OPEN	Device not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Failed Reading statistics information
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.5.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.6 SF_CELLULAR_COMMON_Transmit

```
ssp_err_t SF_CELLULAR_COMMON_Transmit ( sf_cellular_ctrl_t  *const p_ctrl ,
  uint8_t *const p_buf ,   uint32_t length )
```

7.10.6.1 Brief description

Passes packet buffer to PPP stack for transmission.

7.10.6.2 Detailed description

Implements transmit Send packet buffer to PPP stack

Table 99: Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_buf	in	transmit byte buffer pointer
length	in	Length of data

Table 100:Return values

Name	Description
SSP_SUCCESS	Successfully send the packet buffer.
SSP_ERR_NOT_OPEN	Device not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Transmitting Data failed
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.6.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.7 SF_CELLULAR_COMMON_ProvisioningGet

```
ssp_err_t SF_CELLULAR_COMMON_ProvisioningGet ( sf_cellular_ctrl_t
*const p_ctrl ,  sf_cellular_provisioning_t  *const p_cellular_provisioning )
```

7.10.7.1 Brief description

Reads the provisioning information.

7.10.7.2 Detailed description

Implements provisioningGet Reads Cellular's provisioning information

Table 101:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_cellular_provisioning	out	Cellular provisioning structure

Table 102:Return values

Name	Description
SSP_SUCCESS	Successfully read the provisioning information.
SSP_ERR_NOT_OPEN	Device not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Reading provisioning information failed
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.7.3 Function steps

- Get cellular information
- Get Mutex Lock
- Copy provisioning info to driver structure

7.10.8 SF_CELLULAR_COMMON_ProvisioningSet

```
ssp_err_t SF_CELLULAR_COMMON_ProvisioningSet ( sf_cellular_ctrl_t
*const p_ctrl ,  sf_cellular_provisioning_t  const
*const p_cellular_provisioning )
```

7.10.8.1 Brief description

Sets the provisioning information.

7.10.8.2 Detailed description

Implements provisioningSet Sets Cellular's provisioning information

Table 103:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_cellular_provisioning	in	Cellular provisioning structure

Table 104:Return values

Name	Description
SSP_SUCCESS	Successfully set the provisioning information.
SSP_ERR_NOT_OPEN	Device not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Provisioning configuration failed
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.8.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.9 SF_CELLULAR_COMMON_NetworkConnect

```
ssp_err_t SF_CELLULAR_COMMON_NetworkConnect ( sf_cellular_ctrl_t
*const p_ctrl )
```

7.10.9.1 Brief description

Establishes the Data connection.

7.10.9.2 Detailed description

Implements networkConnect Establishes the Network Connection

Table 105:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block

Table 106:Return values

Name	Description
SSP_SUCCESS	Successfully establishes the Network connection
SSP_ERR_NOT_OPEN	Device not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_CELLULAR_FAILED	Failed to establish the Network Connection
SSP_ERR_IN_USE	Device already in used
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.9.3 Function steps

- Get cellular information
- Get Mutex Lock
- Enter Data Mode

7.10.10 SF_CELLULAR_COMMON_NetworkDisconnect

```
ssp_err_t SF_CELLULAR_COMMON_NetworkDisconnect ( sf_cellular_ctrl_t
*const p_ctrl )
```

7.10.10.1 Brief description

Terminates the connection.

7.10.10.2 Detailed description

Implements networkDisconnect

Table 107:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block

Table 108:Return values

Name	Description
SSP_SUCCESS	Successfully disconnect the connection
SSP_ERR_NOT_OPEN	Cellular driver is not opened
SSP_ERR_CELLULAR_FAILED	Failed to terminate the network connection
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.10.3 Function steps

- Get cellular information
- Get Mutex Lock
- Exit Data Mode

7.10.11 SF_CELLULAR_COMMON_NetworkStatusGet

```
ssp_err_t SF_CELLULAR_COMMON_NetworkStatusGet ( sf_cellular_ctrl_t
*const p_ctrl ,  sf_cellular_network_status_t  * p_network_status )
```

7.10.11.1 Brief description

Get Network Status information.

7.10.11.2 Detailed description

Implements networkStatusGet

Table 109:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_network_status	out	Cellular network structure

Table 110:Return values

Name	Description
SSP_SUCCESS	Successfully read the Network status information
SSP_ERR_NOT_OPEN	Cellular driver is not opened
SSP_ERR_CELLULAR_FAILED	Failed reading Network Status information.
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.11.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.12 SF_CELLULAR_COMMON_SimPinSet

```
ssp_err_t SF_CELLULAR_COMMON_SimPinSet ( sf_cellular_ctrl_t  *const p_ctrl ,
  uint8_t *const p_old_pin ,   uint8_t *const p_new_pin )
```

7.10.12.1 Brief description

Set the SIM PIN.

7.10.12.2 Detailed description

Implements sf_cellular_api_t::simSetPin

Table 111:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_old_pin	in	Old SIM Pin
p_new_pin	in	New SIM Pin

Table 112:Return values

Name	Description
SSP_SUCCESS	Successfully set SIM Pin
SSP_ERR_NOT_OPEN	Cellular driver is not opened
SSP_ERR_CELLULAR_FAILED	Failed to set the SIM Pin
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.12.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.13 SF_CELLULAR_COMMON_SimLock

```
ssp_err_t SF_CELLULAR_COMMON_SimLock ( sf_cellular_ctrl_t  *const p_ctrl ,
  uint8_t *const p_pin )
```

7.10.13.1 Brief description

Lock the SIM.

7.10.13.2 Detailed description

Implements simLock

Table 113:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_pin	in	SIM Pin

Table 114:Return values

Name	Description
SSP_SUCCESS	Successfully lock the SIM
SSP_ERR_NOT_OPEN	Cellular driver is not opened
SSP_ERR_CELLULAR_FAILED	Failed to Lock the SIM
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.13.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.14 SF_CELLULAR_COMMON_SimUnlock

```
ssp_err_t SF_CELLULAR_COMMON_SimUnlock ( sf_cellular_ctrl_t  *const p_ctrl ,
  uint8_t *const p_pin )
```

7.10.14.1 Brief description

Unlock the SIM.

7.10.14.2 Detailed description

Implements simUnlock

Table 115:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_pin	in	SIM Pin

Table 116:Return values

Name	Description
SSP_SUCCESS	Successfully unlock the SIM
SSP_ERR_NOT_OPEN	Cellular driver is not opened
SSP_ERR_CELLULAR_FAILED	Failed to unlock the SIM
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.14.3 Function steps

- Get cellular information
- Get Mutex Lock

7.10.15 SF_CELLULAR_COMMON_SimIDGet

```
ssp_err_t SF_CELLULAR_COMMON_SimIDGet ( sf_cellular_ctrl_t  *const p_ctrl ,
  uint8_t * p_sim_id )
```

7.10.15.1 Brief description

Get SIM ID.

7.10.15.2 Detailed description

Implements sf_cellular_api_t::simGetID Get SIM ID

Table 117:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_sim_id	out	SIM ID

Table 118:Return values

Name	Description
SSP_SUCCESS	Successfully get the SIM ID.
SSP_ERR_CELLULAR_FAILED	Failed to get the SIM ID
SSP_ERR_NOT_OPEN	Device is not opened
SSP_ERR_ASSERTION	Argument NULL is passed
SSP_ERR_IN_USE	Device already in use
SSP_ERR_CELLULAR_INVALID_STATE	Module in Data mode can't send AT command

7.10.15.3 Function steps

- Get cellular information

7.10.16 SF_CELLULAR_COMMON_FotaCheck

```
ssp_err_t SF_CELLULAR_COMMON_FotaCheck ( sf_cellular_ctrl_t  *const p_ctrl ,
  void * p_fotacheck )
```

7.10.16.1 Brief description

Checks for available firmware upgrade.

7.10.16.2 Detailed description

Implements fotaCheck Checks for available firmware upgrade.

Table 119:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_fotacheck	in	Fota check specific structure

Table 120:Return values

Name	Description
SSP_ERR_UNSUPPORTED	Functionality not supported

7.10.17 SF_CELLULAR_COMMON_FotaStart

```
ssp_err_t SF_CELLULAR_COMMON_FotaStart ( sf_cellular_ctrl_t  *const p_ctrl ,
  void * p_fotastart )
```

7.10.17.1 Brief description

Perform the firmware upgrade.

7.10.17.2 Detailed description

Implements fotaStart Perform the firmware upgrade.

Table 121:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_fotastart	in	Fota start specific structure

Table 122:Return values

Name	Description
SSP_ERR_UNSUPPORTED	Functionality not supported

7.10.18 SF_CELLULAR_COMMON_FotaStop

```
ssp_err_t SF_CELLULAR_COMMON_FotaStop ( sf_cellular_ctrl_t  *const p_ctrl ,
  void * p_fotastop )
```

7.10.18.1 Brief description

Stop firmware upgrade.

7.10.18.2 Detailed description

Implements fotaStop Stop firmware upgrade

Table 123:Parameters

Name	Direction	Description
p_ctrl	in	Cellular control block
p_fotastop	in	Fota stop specific structure

Table 124:Return values

Name	Description
SSP_ERR_UNSUPPORTED	Functionality not supported

7.11 Communications Framework Interface

RTOS-integrated communications Framework Interface.

Implemented by:

- UART Framework Instance - UART implementation
- USB Communication Framework - USBX CDC ACM device implementation
- Telnet Communication Framework - NetX telnet server implementation

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

See also Framework Communications Interface description: UART Communications Framework

7.11.1 Interface API

sf_comms_api_t

Function name	Description
.open	Initialize communications driver.
.close	Clean up communications driver.
.read	Read data from communications driver. This call will return after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
.write	Write data to communications driver. This call will return after all bytes are written or if a timeout occurs while waiting for access to the driver.
.lock	Lock the communications driver. Reserve exclusive access to the communications driver.
.unlock	Unlock the communications driver. Release exclusive access to the communications driver.
.versionGet	Store the driver version in the provided p_version.

7.11.2 Data structures

- sf_comms_cfg_t
- sf_comms_instance_t

7.11.3 Enumerations

- sf_comms_lock_t

7.11.4 Typedefs

- sf_comms_ctrl_t

7.11.5 Defines

- #define SF_COMMS_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file

- #define SF_COMMS_API_VERSION_MINOR

  Initial value:`(4U)`

7.11.6 API Data

7.11.6.1 sf_comms_lock_t

`sf_comms_lock_t`

Detailed description

Communications locks

Enumerated values

Name	Description
SF_COMMS_LOCK_TX	Lock Transmit.
SF_COMMS_LOCK_RX	Lock Receive.
SF_COMMS_LOCK_ALL	Lock Transmit and Receive.

7.11.6.2 sf_comms_ctrl_t

```
typedef void sf_comms_ctrl_t
```

Detailed description

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented as

- sf_console_instance_ctrl_t

7.11.7 API Structures

7.11.7.1 sf_comms_cfg_t

sf_comms_cfg_t

Detailed description

Configuration for RTOS integrated communications driver

Variables

- void const * p_extend

  Pointer to lower level communications control structure.

7.11.7.2 sf_comms_api_t

sf_comms_api_t

Detailed description

Framework communications API structure. Implementations will use the following API.

7.11.7.3 open

```
ssp_err_t(* sf_comms_api_t::open) (sf_comms_ctrl_t *const p_ctrl, sf_comms_cfg_t
const *const p_cfg)
```

Detailed description

Initialize communications driver.

Table 125:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a control structure allocated by user. The control structure is initialized in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_cfg

Definition: `sf_comms_cfg_t const *const p_cfg`

Configuration for RTOS integrated communications driver

- `sf_comms_cfg_t`::`p_extend`

  Pointer to lower level communications control structure.

7.11.7.4 close

```
ssp_err_t(* sf_comms_api_t::close) (sf_comms_ctrl_t *const p_ctrl)
```

Detailed description

Clean up communications driver.

Table 126:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

7.11.7.5 read

```
ssp_err_t(* sf_comms_api_t::read) (sf_comms_ctrl_t *const p_ctrl, uint8_t *const
p_dest, uint32_t const bytes, UINT const timeout)
```

Detailed description

Read data from communications driver. This call will return after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.

Table 127:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.
p_dest	in	Destination address to read data out
bytes	in	Read data length
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint32_t`

Parameter timeout

`const`

7.11.7.6 write

```
ssp_err_t(* sf_comms_api_t::write) (sf_comms_ctrl_t *const p_ctrl, uint8_t const
*const p_src, uint32_t const bytes, UINT const timeout)
```

Detailed description

Write data to communications driver. This call will return after all bytes are written or if a timeout occurs while waiting for access to the driver.

Table 128:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.
p_src	in	Source address to read data out from
bytes	in	Write data length
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter bytes

`uint32_t`

Parameter timeout

`const`

7.11.7.7 lock

```
ssp_err_t(* sf_comms_api_t::lock) (sf_comms_ctrl_t *const p_ctrl, sf_comms_lock_t
lock_type, UINT timeout)
```

Detailed description

Lock the communications driver. Reserve exclusive access to the communications driver.

Table 129:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.
lock_type	in	Locking type, transmission channel or reception channel
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

Parameter lock_type

Definition: `sf_comms_lock_tlock_type`

Communications locks

Parameter timeout

`const`

7.11.7.8 unlock

```
ssp_err_t(* sf_comms_api_t::unlock) (sf_comms_ctrl_t *const p_ctrl,
sf_comms_lock_t lock_type)
```

Detailed description

Unlock the communications driver. Release exclusive access to the communications driver.

Table 130:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.
lock_type	in	Locking type, transmission channel or reception channel

Parameter p_ctrl

Definition: `sf_comms_ctrl_t*const p_ctrl`

Communications framework control block. Allocate an instance specific control block to pass into the communications framework API calls. Implemented assf_console_instance_ctrl_t

Parameter lock_type

Definition: `sf_comms_lock_tlock_type`

Communications locks

7.11.7.9 versionGet

```
ssp_err_t(* sf_comms_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Store the driver version in the provided p_version.

Table 131:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for communications driver.
p_version	in	Pointer to memory version to be stored.

Parameter p_ctrl

Parameter p_version

7.11.7.10 sf_comms_instance_t

sf_comms_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_comms_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_comms_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_comms_api_t const * p_api

  Pointer to the API structure for this instance.

7.12 Console Framework Interface

RTOS-integrated Console Framework Interface.

7.12.1 Summary

This module is a ThreadX-aware Console Framework.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

See also Console Interface description Console Framework

7.12.2 Interface API

sf_console_api_t

Function name	Description
.open	This function configures the console. This function must be called before any other console functions.
.close	The close API handles cleans up internal driver data.
.prompt	Prints prompt string from menu, waits for input, parses input based on menu, and calls callback function if a command is identified.

Function name	Description
.parse	Looks for input string in menu, and calls callback function if found.
.read	Reads data into the destination byte by byte and echos input to the console. Backspace, delete, and left/right arrow keys supported. Read completes when a line ending CR, CR+LF, or CR+NULL is received, or when the input exceeds the number of bytes input. If the buffer overflows SF_CONSOLE_MAX_INPUT_LENGTH, read will return an error code.
.write	The write API gets mutex object and handles UART data transmission at UART HAL layer. gets event flag to synchronize to completion of data transfer.
.argumentFind	Finds a command line argument in an input string and returns the index of the character immediately following the argument and any string numbers converted to integers.
.versionGet	Stores version information in provided pointer.

7.12.3 Data structures

- sf_console_callback_args_t
- sf_console_command_t
- sf_console_menu_t
- sf_console_cfg_t
- sf_console_instance_t

7.12.4 Typedefs

- sf_console_ctrl_t
- sf_console_cb_args_t

7.12.5 Defines

- #define SF_CONSOLE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file

- #define SF_CONSOLE_API_VERSION_MINOR

  Initial value:`(4U)`
- #define SF_CONSOLE_HELP_COMMAND

  Initial value:`((uint8_t *) "?")`

  Command to print each command and help in menu
- #define SF_CONSOLE_MENU_PREVIOUS_COMMAND

  Initial value:`((uint8_t *) "^")`

  Previous command
- #define SF_CONSOLE_ROOT_MENU_COMMAND

  Initial value:`((uint8_t *) "~")`

  Root menu command
- #define SF_CONSOLE_MAX_INPUT_LENGTH

  Initial value:`(128U)`

  Input length
- #define SF_CONSOLE_CALLBACK_NEXT_FUNCTION

  Initial value:`((void(*)(sf_console_callback_args_t * p_args)) 0x70000000)`

  Use this macro to access the next menu layer from this command.

7.12.6 API Data

7.12.6.1 sf_console_ctrl_t

```
typedef void sf_console_ctrl_t
```

Detailed description

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented as

- sf_console_instance_ctrl_t

7.12.6.2 sf_console_cb_args_t

```
typedef sf_console_callback_args_t sf_console_cb_args_t
```

Detailed description

DEPRECATED definition, please use sf_console_callback_args_t instead.

7.12.7 API Structures

7.12.7.1 sf_console_callback_args_t

sf_console_callback_args_t

Detailed description

Console callback arguments

Variables

- sf_console_ctrl_t * p_ctrl

  Pointer to console that received the command that caused this callback.

- uint8_t const * p_remaining_string

  String remaining after parsing command.

- uint8_t const * context

  Pointer to user provided data.

- uint32_t bytes

  The number of bytes remaining in the input string.

7.12.7.2 sf_console_command_t

sf_console_command_t

Detailed description

Console command structure, used to create a console menu with associated callbacks.

Variables

- uint8_t * command

  Command string.

- uint8_t * help

  Description of command.

- void(* callback)(sf_console_callback_args_t *p_args)

  Callback to call when command is selected.

- void const * context

  User provided context passed into callback.

7.12.7.3 sf_console_menu_t

sf_console_menu_t

Detailed description

Console menu structure.

Variables

- struct st_sf_console_menu const * menu_prev

  Previous menu.

- uint8_t const * menu_name

  Menu name, used as a prompt.

- uint32_t num_commands

  Number of commands in this menu.

- sf_console_command_t const * command_list

  Pointer to an array of commands of length num_commands.

7.12.7.4 sf_console_cfg_t

sf_console_cfg_t

Detailed description

Configuration for RTOS integrated console framework.

Variables

- sf_comms_instance_t const * p_comms

  Pointer to communications driver instance.

- sf_console_menu_t const * p_initial_menu

  First menu to print during Open.

- bool echo

  Whether to echo input commands to transmitter.

- bool autostart

  If true, prompt will occur with p_initial_menu after initialization.

7.12.7.5 sf_console_api_t

sf_console_api_t

Detailed description

Console framework API structure. Console implementations will use the following API.

7.12.7.6 open

```
(* sf_console_api_t::open) (sf_console_ctrl_t *const p_ctrl, sf_console_cfg_t
const *const p_cfg)
```

Brief description

This function configures the console. This function must be called before any other console functions.

Detailed description

Implemented as

- SF_CONSOLE_Open

Table 132:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a device structure allocated by user. The device control structure is initialized in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_cfg

Definition: `sf_console_cfg_t const *const p_cfg`

Configuration for RTOS integrated console framework.

- `sf_console_cfg_t`::`sf_comms_instance_t`

  Pointer to communications driver instance.

- `sf_console_cfg_t`::`sf_console_menu_t`

  First menu to print during Open.

- `sf_console_cfg_t`::`echo`

  Whether to echo input commands to transmitter.

- `sf_console_cfg_t`::`autostart`

  If true, prompt will occur with p_initial_menu after initialization.

7.12.7.7 close

```
(* sf_console_api_t::close) (sf_console_ctrl_t *const p_ctrl)
```

Brief description

The close API handles cleans up internal driver data.

Detailed description

Implemented as

- SF_CONSOLE_Close

Table 133:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for UART driver.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

7.12.7.8 prompt

```
(* sf_console_api_t::prompt) (sf_console_ctrl_t *const p_ctrl, sf_console_menu_t
const *const p_menu, UINT const timeout)
```

Brief description

Prints prompt string from menu, waits for input, parses input based on menu, and calls callback function if a command is identified.

Detailed description

Implemented as

- SF_CONSOLE_Prompt

Table 134:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for UART driver.
p_menu	in	Set to NULL to stay on current menu maintained by the console framework. To change menus, pass a pointer to the new menu.
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_menu

Definition: `sf_console_menu_tconst *const p_menu`

Console menu structure.

- `sf_console_menu_t::menu_prev`

  Previous menu.

- `sf_console_menu_t::menu_name`

  Menu name, used as a prompt.

- `sf_console_menu_t::num_commands`

  Number of commands in this menu.

- `sf_console_menu_t::command_list`

  Pointer to an array of commands of length num_commands.

Parameter timeout

`const`

7.12.7.9 parse

```
(* sf_console_api_t::parse) (sf_console_ctrl_t *const p_ctrl, sf_console_menu_t
const *const p_cmd_list, uint8_t const *const p_input, uint32_t const bytes)
```

Brief description

Looks for input string in menu, and calls callback function if found.

Detailed description

Implemented as

- SF_CONSOLE_Parse

Table 135:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for UART driver.
p_cmd_list	in	Pointer to a menu of valid input commands for this prompt
p_input	in	Pointer to a null terminated string to search for in the command list
bytes	in	Length of the input string.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_cmd_list

Definition: `sf_console_menu_tconst *const p_cmd_list`

Console menu structure.

- `sf_console_menu_t::menu_prev`

  Previous menu.

- `sf_console_menu_t::menu_name`

  Menu name, used as a prompt.

- `sf_console_menu_t::num_commands`

  Number of commands in this menu.

- `sf_console_menu_t::command_list`

  Pointer to an array of commands of length num_commands.

Parameter p_input

`uint8_t`

Parameter bytes

`uint32_t`

7.12.7.10 read

```
(* sf_console_api_t::read) (sf_console_ctrl_t *const p_ctrl, uint8_t *const
p_dest, uint32_t const bytes, uint32_t const timeout)
```

Brief description

Reads data into the destination byte by byte and echos input to the console. Backspace, delete, and left/right arrow keys supported. Read completes when a line ending CR, CR+LF, or CR+NULL is received, or when the input exceeds the number of bytes input. If the buffer overflows SF_CONSOLE_MAX_INPUT_LENGTH, read will return an error code.

Detailed description

Implemented as

- SF_CONSOLE_Read

Table 136:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for UART driver.

Table 136:Parameters (Continued)

Name	Direction	Description
p_dest	in	Destination address to read data out
bytes	in	Read data length
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint32_t`

Parameter timeout

`uint32_t`

7.12.7.11 write

```
(* sf_console_api_t::write) (sf_console_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint32_t const timeout)
```

Brief description

The write API gets mutex object and handles UART data transmission at UART HAL layer. gets event flag to synchronize to completion of data transfer.

Detailed description

Implemented as

- SF_CONSOLE_Write

Table 137:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for UART driver.

Table 137:Parameters (Continued)

Name	Direction	Description
p_src	in	Pointer to a NULL terminated string. Length must be less than SF_CONSOLE_MAX_WRITE_LENGTH.
timeout	in	ThreadX timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_console_ctrl_t*const p_ctrl`

Console framework control block. Allocate an instance specific control block to pass into the console framework API calls. Implemented assf_console_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter timeout

`uint32_t`

7.12.7.12 argumentFind

```
(* sf_console_api_t::argumentFind) (uint8_t const *const p_arg, uint8_t const
*const p_str, int32_t *const p_index, int32_t *const p_data)
```

Brief description

Finds a command line argument in an input string and returns the index of the character immediately following the argument and any string numbers converted to integers.

Detailed description

Implemented as

- SF_CONSOLE_ArgumentFind

Table 138:Parameters

Name	Direction	Description
p_arg	in	Pointer to argument to find.
p_src	in	Pointer to source string to find the argument in.

Table 138:Parameters (Continued)

Name	Direction	Description
p_index	out	Pointer to location to store index. Set to -1 if argument is not found in input string. Pass NULL if index is not requested.
p_data	out	Pointer to location to store data following the argument. Set to -1 if argument is not found in input string. Pass NULL if data is not requested.

Parameter p_arg

`uint8_t`

Parameter p_src

Parameter p_index

Parameter p_data

7.12.7.13 versionGet

```
(* sf_console_api_t::versionGet) ( *const p_version)
```

Brief description

Stores version information in provided pointer.

Detailed description

Implemented as

- SF_CONSOLE_VersionGet

Table 139:Parameters

Name	Direction	Description
p_version	out	Code and API version used stored here.

Parameter p_version

7.12.7.14 sf_console_instance_t

sf_console_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_console_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_console_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_console_api_t const * p_api

  Pointer to the API structure for this instance.

7.13 SSP Crypto Framework Common Module Interface

Interface definition for Synergy Crypto Framework module.
Interface definition for Synergy Crypto Framework Common Module.

7.13.1 Summary

This is the Interface of SF_CRYPTO Framework module.
SF_CRYPTO Interface description: Crypto Framework

7.13.2 Summary

This is the Interface of SF_CRYPTO Framework module.
SF_CRYPTO Interface description: Crypto Framework

7.13.3 Functions

- SF_CRYPTO_Open
- SF_CRYPTO_Close
- SF_CRYPTO_Lock
- SF_CRYPTO_Unlock
- SF_CRYPTO_StatusGet
- SF_CRYPTO_VersionGet

7.13.4 Typedefs

- sf_crypto_ctrl_t

7.13.5 Defines

- #define SF_CRYPTO_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. The API version of SSP Crypto Framework Common Module

- #define SF_CRYPTO_API_VERSION_MINOR

  Initial value:`(0U)`

- #define SF_CRYPTO_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. The API version of SSP Crypto Framework Common Module

- #define SF_CRYPTO_CODE_VERSION_MINOR

  Initial value:`(0U)`

7.13.6 SF_CRYPTO_Open

```
ssp_err_t SF_CRYPTO_Open ( sf_crypto_ctrl_t  *const p_api_ctrl ,
 sf_crypto_cfg_t  const *const p_cfg )
```

7.13.6.1 Brief description

SSP Crypto Framework Common Open operation.

7.13.6.2 Detailed description

Table 140:Parameters

Name	Direction	Description
p_api_ctrl	inout	Pointer to a Crypto framework control block
p_cfg	in	Pointer to a Crypto framework configuration structure

Table 141:Return values

Name	Description
SSP_SUCCESS	Crypto framework was successfully opened.
SSP_ERR_ASSERTION	NULL pointer is passed.
SSP_ERR_INTERNAL	RTOS service returned a unexpected error.
SSP_ERR_CRYPTO_HAL_ERROR	Crypto HAL driver retuned an error.

7.13.7 SF_CRYPTO_Close

```
ssp_err_t SF_CRYPTO_Close ( sf_crypto_ctrl_t  *const p_api_ctrl )
```

7.13.7.1 Brief description

SSP Crypto Framework Common Close operation.

7.13.7.2 Detailed description

Table 142:Parameters

Name	Direction	Description
p_api_ctrl	inout	Pointer to a Crypto framework control block

Table 143:Return values

Name	Description
SSP_SUCCESS	Module was successfully closed.
SSP_ERR_NOT_OPEN	Module has not opened.
SSP_ERR_ASSERTION	NULL pointer is passed.
SSP_ERR_INTERNAL	RTOS service returned a unexpected error.

7.13.8 SF_CRYPTO_Lock

```
ssp_err_t SF_CRYPTO_Lock ( sf_crypto_ctrl_t  *const p_api_ctrl )
```

7.13.8.1 Brief description

Locks the module. This API is utilized for locking shared resources.

7.13.8.2 Detailed description

Table 144:Parameters

Name	Direction	Description
p_api_ctrl	inout	Pointer to a Crypto framework control block

Table 145:Return values

Name	Description
SSP_SUCCESS	Module resources are successfully locked.
SSP_ERR_TIMEOUT	Unable to get ownership of the mutex within the specified time.
SSP_ERR_INTERNAL	Thread suspension was aborted. Critical error.
SSP_ERR_ASSERTION	NULL pointer is passed. to wait.

7.13.9 SF_CRYPTO_Unlock

```
ssp_err_t SF_CRYPTO_Unlock ( sf_crypto_ctrl_t  *const p_api_ctrl )
```

7.13.9.1 Brief description

Unlocks the module. This API is utilized for unlocking shared resources.

7.13.9.2 Detailed description

Table 146:Parameters

Name	Direction	Description
p_api_ctrl	inout	Pointer to a Crypto framework control block

Table 147:Return values

Name	Description
SSP_SUCCESS	Module resources are successfully unlocked.
SSP_ERR_ASSERTION	NULL pointer is passed.
SSP_ERR_INTERNAL	Mutex is not owned by a caller thread.

7.13.10 SF_CRYPTO_StatusGet

```
ssp_err_t SF_CRYPTO_StatusGet ( sf_crypto_ctrl_t  *const p_api_ctrl ,
 sf_crypto_state_t  * p_status )
```

7.13.10.1 Brief description

Gets the Crypto Common Framework module status.

7.13.10.2 Detailed description

Table 148:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to a Crypto framework control block
p_status	out	Memory location to store module status.

Table 149:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

7.13.11 SF_CRYPTO_VersionGet

```
ssp_err_t SF_CRYPTO_VersionGet ( ssp_version_t  *const p_version )
```

7.13.11.1 Brief description

Gets the version of Crypto Common Framework module.

7.13.11.2 Detailed description

Table 150:Parameters

Name	Direction	Description
p_version	out	Pointer to the memory to store the version information.

Table 151:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

7.13.12 API Data

7.13.12.1 sf_crypto_key_type_t

`sf_crypto_key_type_t`

Detailed description

Supported key types

Enumerated values

Name	Description
SF_CRYPTO_KEY_TYPE_RSA_PLAIN_TEXT	RSA Key pair in standard format and plain text.
SF_CRYPTO_KEY_TYPE_RSA_CRT_PLAIN_TEXT	RSA Key pair in CRT format and plain text.
SF_CRYPTO_KEY_TYPE_RSA_WRAPPED	RSA Key pair public key in plain text and wrapped standard format private key.
SF_CRYPTO_KEY_TYPE_AES_WRAPPED	Wrapped AES key.

7.13.12.2 sf_crypto_key_size_t

`sf_crypto_key_size_t`

Detailed description

Supported key sizes

Enumerated values

Name	Description
SF_CRYPTO_KEY_SIZE_RSA_1024	RSA 1024-bit key.
SF_CRYPTO_KEY_SIZE_RSA_2048	RSA 2048-bit key.
SF_CRYPTO_KEY_SIZE_AES_128	AES 128-bit key for CBC, CTR, ECB, GCM chaining modes.
SF_CRYPTO_KEY_SIZE_AES_XTS_128	AES 128-bit key for XTS chaining mode only.
SF_CRYPTO_KEY_SIZE_AES_192	AES 192-bit key for CBC, CTR, ECB, GCM chaining modes.
SF_CRYPTO_KEY_SIZE_AES_256	AES 256-bit key for CBC, CTR, ECB, GCM chaining modes.
SF_CRYPTO_KEY_SIZE_AES_XTS_256	AES 256-bit key for XTS chaining mode only.

7.13.12.3 sf_crypto_state_t

`sf_crypto_state_t`

Detailed description

State codes for the SSP Crypto Framework Common Module

Enumerated values

Name	Description
SF_CRYPTO_CLOSED	The module is closed.
SF_CRYPTO_OPENED	The module is opened.

7.13.12.4 sf_crypto_event_t

```
sf_crypto_event_t
```

Detailed description

Event code for the SSP Crypto Framework Common Module. This event code is all reserved for the future use.

Enumerated values

Name	Description
SF_CRYPTO_EVENT_PROCEDURE_DONE	Crypto hardware procedure done.
SF_CRYPTO_EVENT_ERROR	Error occurred.

7.13.12.5 sf_crypto_close_option_t

```
sf_crypto_close_option_t
```

Detailed description

SF_CRYPTO Close option. The module executes close operation if any SF_CRYPTO_XXX modules have already closed if SF_CRYPTO_CLOSE_OPTION_DEFAULT option is specified. The module performs close operation regardless of any SF_CRYPTO_XXX module status if SF_CRYPTO_CLOSE_OPTION_FORCE_CLOSE is specified.

Enumerated values

Name	Description
SF_CRYPTO_CLOSE_OPTION_DEFAULT	Close the module if no any SF_CRYPTO_XXX modules opened.
SF_CRYPTO_CLOSE_OPTION_FORCE_CLOSE	Close the module regardless of SF_CRYPTO_XXX modules status.

7.13.12.6 sf_crypto_ctrl_t

```
typedef void sf_crypto_ctrl_t
```

Detailed description

SSP Crypto Framework Common Module control block. Allocate an instance specific control block to pass into the SSP Crypto Framework Common Module API calls. Implemented as

- sf_crypto_instance_ctrl_t

7.13.13 Extensions

7.13.13.1 sf_crypto_data_handle_t

sf_crypto_data_handle_t

Detailed description

A structure to handle data among Crypto Framework modules

Variables

- uint8_t * p_data

  Pointer to data.

- uint32_t data_length

  The length of data pointed by p_data.

7.13.13.2 sf_crypto_callback_args_t

sf_crypto_callback_args_t

Detailed description

Callback arguments for the SSP Crypto Framework Common Module

Variables

- sf_crypto_event_t event

  Event code of the low level hardware.

- ssp_err_t error

  Error code if SF_CRYPTO_EVENT_ERROR.

7.13.13.3 sf_crypto_cfg_t

sf_crypto_cfg_t

Detailed description

Configuration structure for the SSP Crypto Framework Common Module

Variables

- uint32_t wait_option

  Wait option for RTOS service calls.

- crypto_instance_t * p_lower_lvl_crypto

  Pointer to a low-level Crypto engine HAL driver instance.
- void const * p_extend

  Extension parameter for hardware specific settings.
- void const * p_context

  Placeholder for user data.
- void * p_memory_pool

  Byte pool address.
- uint32_t memory_pool_size

  Byte pool size.
- sf_crypto_close_option_t close_option

  Close option.

7.13.13.4 sf_crypto_api_t

sf_crypto_api_t

Detailed description

Shared Interface definition for the SSP Crypto Framework Common Module

7.13.13.5 sf_crypto_instance_t

sf_crypto_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_crypto_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_crypto_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_crypto_api_t const * p_api

  Pointer to the API structure for this instance.

7.13.13.6 sf_crypto_instance_ctrl_t

sf_crypto_instance_ctrl_t

Detailed description

SSP Crypto Framework Common Module instance control block

Variables

- sf_crypto_state_t status

  Module status.
- TX_MUTEX mutex

  Mutex used in the Crypto Framework.
- TX_SEMAPHORE semaphore

  Semaphore used in the Crypto Framework (Reserve)
- TX_BYTE_POOL byte_pool

  Byte pool used in the Crypto Framework.
- uint32_t wait_option

  Wait time option used for RTOS service calls.
- uint32_t open_counter

  Counter to keep the number of SF_CRYPTO_XXX opened.
- void * p_lower_lvl_crypto

  Pointer to a low-level Crypto engine HAL driver instance.
- void(* p_callback)(*p_args)

  Pointer to callback function.
- void * p_context

  Pointer to a context.
- sf_crypto_close_option_t close_option

  Close option.

7.14 SSP Crypto HASH Framework Interface

Interface definition for Synergy Crypto HASH Framework module.

7.14.1 Summary

This is the Interface of SF_CRYPTO_HASH Framework module.
SF_CRYPTO_HASH Interface description:

7.14.2 Summary

This is the Interface of SF_CRYPTO KEY Framework module.
SF_CRYPTO_HASH Interface description: Crypto Framework

7.14.3 Functions

- SF_CRYPTO_HASH_Open
- SF_CRYPTO_HASH_Close
- SF_CRYPTO_HASH_MessageDigestInit
- SF_CRYPTO_HASH_MessageDigestUpdate
- SF_CRYPTO_HASH_MessageDigestFinal
- SF_CRYPTO_HASH_VersionGet
- sf_crypto_hash_instance_initialize
- sf_crypto_hash_instance_deinitialize
- sf_crypto_hash_digest_size_get
- sf_crypto_hash_message_block_size_get
- sf_crypto_hash_memory_allocate
- sf_crypto_hash_message_digest_initialize
- sf_crypto_hash_interface_get
- sf_crypto_hash_module_open_status_check
- sf_crypto_hash_message_digest_update_status_check
- sf_crypto_hash_message_digest_final_status_check
- sf_crypto_hash_data_blocks
- sf_crypto_hash_get_message_digest

7.14.4 Typedefs

- sf_crypto_hash_t
- sf_crypto_hash_ctrl_t

7.14.5 Defines

- #define SF_CRYPTO_HASH_API_VERSION_MAJOR

  Initial value: `(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  The API version of SSP Crypto HASH Framework
- #define SF_CRYPTO_HASH_API_VERSION_MINOR

  Initial value: `(0U)`

- #define SF_CRYPTO_HASH_MESSAGE_DIGEST_SIZE_SHA1

  Initial value:`(20U)`

  Message Digest size for SHA1. Message Digest size for each HASH algorithm in bytes
- #define SF_CRYPTO_HASH_MESSAGE_DIGEST_SIZE_SHA224

  Initial value:`(28U)`

  Message Digest size for SHA224.
- #define SF_CRYPTO_HASH_MESSAGE_DIGEST_SIZE_SHA256

  Initial value:`(32U)`

  Message Digest size for SHA256.
- #define SF_CRYPTO_HASH_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. The API version of SSP Crypto Framework
- #define SF_CRYPTO_HASH_CODE_VERSION_MINOR

  Initial value:`(0U)`

7.14.6 SF_CRYPTO_HASH_Open

```
ssp_err_t SF_CRYPTO_HASH_Open ( sf_crypto_hash_ctrl_t  *const p_api_ctrl ,
 sf_crypto_hash_cfg_t  const *const p_cfg )
```

7.14.6.1 Brief description

SSP Crypto HASH Framework Open operation.

7.14.6.2 Detailed description

Table 152:Return values

Name	Description
SSP_SUCCESS	The module was successfully opened.
SSP_ERR_ASSERTION	NULL is passed through an argument.
SSP_ERR_CRYPTO_COMMON_NOT_OPENED	Crypto Framework Common Module has yet been opened.
SSP_ERR_ALREADY_OPEN	The module has been already opened.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.

Table 152:Return values (Continued)

Name	Description
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.14.7 SF_CRYPTO_HASH_Close

```
ssp_err_t SF_CRYPTO_HASH_Close ( sf_crypto_hash_ctrl_t  *const p_api_ctrl )
```

7.14.7.1 Brief description

SSP Crypto HASH Framework Close operation.

7.14.7.2 Detailed description

Table 153:Return values

Name	Description
SSP_SUCCESS	The module was successfully closed.
SSP_ERR_ASSERTION	NULL is passed through the argument.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.14.8 SF_CRYPTO_HASH_MessageDigestInit

```
ssp_err_t SF_CRYPTO_HASH_MessageDigestInit ( sf_crypto_hash_ctrl_t
*const p_api_ctrl )
```

7.14.8.1 Brief description

SSP Crypto HASH Framework - Generates the initial message digest in an internal context buffer. Can be called once messageDigestFinal() is called to initialize a new digest operation. Unless a different HASH type is used, users do not need to close the module for new digest operation. This is a blocking call.

7.14.8.2 Detailed description

Table 154:Return values

Name	Description
SSP_SUCCESS	The module updated a message Digest successfully.
SSP_ERR_ASSERTION	NULL is passed through an argument.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

NOTE: When this function is called if SF_CRYPTO_HASH_DIGEST_INITIALIZED or SF_CRYPTO_HASH_DIGEST_UPDATED, the message digest will be initialized.

7.14.9 SF_CRYPTO_HASH_MessageDigestUpdate

```
ssp_err_t SF_CRYPTO_HASH_MessageDigestUpdate ( sf_crypto_hash_ctrl_t
*const p_api_ctrl ,  sf_crypto_data_handle_t  const *const p_data_in )
```

7.14.9.1 Brief description

SSP Crypto HASH Framework - Hashes input data and saves it in an internal context buffer. Can be called multiple times for additional blocks of data. This is a blocking call.

7.14.9.2 Detailed description

Table 155:Return values

Name	Description
SSP_SUCCESS	The module updated a message Digest successfully.
SSP_ERR_ASSERTION	NULL is passed through an argument.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.
SSP_ERR_INVALID_CALL	Function call was made if the module state had not yet transitioned to SF_CRYPTO_HASH_DIGEST_INITIALIZED.

Table 155:Return values (Continued)

Name	Description
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.14.10 SF_CRYPTO_HASH_MessageDigestFinal

```
ssp_err_t SF_CRYPTO_HASH_MessageDigestFinal ( sf_crypto_hash_ctrl_t
*const p_api_ctrl ,  sf_crypto_data_handle_t  *const p_msg_digest ,   uint32_t
* p_size )
```

7.14.10.1 Brief description

SSP Crypto HASH Framework - Hashes the last block of data and returns the message digest in the output buffer. Once this function is called, no additional function calls can be made but open function call can be made to initialize a new digest operation. The Output buffer will contain the message digest on success and the buffer length will be updated to reflect the size of the digest. On error, only the length is set to 0.This is a blocking call.

7.14.10.2 Detailed description

Table 156:Return values

Name	Description
SSP_SUCCESS	The module updated a message Digest successfully.
SSP_ERR_ASSERTION	NULL is passed through an argument.
SSP_ERR_INVALID_SIZE	The memory size to store a message digest is not sufficient for the digest type.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.
SSP_ERR_INVALID_CALL	Function call was made if the module state had not yet transitioned to SF_CRYPTO_HASH_DIGEST_UPDATED.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

NOTE: p_msg_digest->p_data must be WORD aligned. The memory allocation to store a message digest is user's responsibility.

7.14.11 SF_CRYPTO_HASH_VersionGet

```
ssp_err_t SF_CRYPTO_HASH_VersionGet ( ssp_version_t  *const p_version )
```

7.14.11.1 Brief description

Gets driver version based on compile time macros.

7.14.11.2 Detailed description

Table 157:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

7.14.12 sf_crypto_hash_instance_initialize

```
ssp_err_t sf_crypto_hash_instance_initialize ( sf_crypto_hash_instance_ctrl_t
* p_ctrl ,   uint32_t digest_size )
```

7.14.12.1 Brief description

Sub-routine for Crypto HASH Framework to initialize software and hardware instances. This function is called by SF_CRYPTO_HASH_Open.

7.14.12.2 Detailed description

Table 158:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a HASH framework control block.
digest_size	in	Digest size.

Table 159:Return values

Name	Description
SSP_SUCCESS	Software and hardware instances were successfully initialized.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.13 sf_crypto_hash_instance_deinitialize

```
ssp_err_t sf_crypto_hash_instance_deinitialize ( sf_crypto_hash_instance_ctrl_t
* p_ctrl ,   uint32_t digest_size )
```

7.14.13.1 Brief description

Sub-routine for Crypto HASH Framework to de-initialize software and hardware instances. This function is called by SF_CRYPTO_HASH_Close.

7.14.13.2 Detailed description

Table 160:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a HASH framework control block.
digest_size	in	Digest size.

Table 161:Return values

Name	Description
SSP_SUCCESS	Software and hardware instances were successfully de-initialized.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.14 sf_crypto_hash_digest_size_get

```
sf_crypto_hash_digest_size_get ( sf_crypto_hash_type_t  hash_type ,
 bool final )
```

7.14.14.1 Brief description

Sub-routine for Crypto HASH Framework to get size of message digest. This function is called by SF_CRYPTO_HASH_Open. SF_CRYPTO_HASH_Close, SF_CRYPTO_HASH_MessageDigestFinal, sf_crypto_hash_message_digest_initialize() and sf_crypto_hash_get_message_digest().

7.14.14.2 Detailed description

Table 162:Parameters

Name	Direction	Description
hash_type	in	HASH type. SHA1, SHA224 and SHA256 are supported.
final	in	Set True to get the final message digest size for specified HASH type. Set False to get intermediate message digest size. The digest size would be different for SHA224.

Table 163:Return values

Name	Description
size	Size of message digest in bytes.

7.14.15 sf_crypto_hash_message_block_size_get

```
sf_crypto_hash_message_block_size_get ( sf_crypto_hash_type_t  hash_type )
```

7.14.15.1 Brief description

Sub-routine for Crypto HASH Framework to get size of message block. This function is called by SF_CRYPTO_HASH_MessageDigestUpdate.

7.14.15.2 Detailed description

Table 164:Parameters

Name	Direction	Description
hash_type	in	HASH type. SHA1, SHA224 and SHA256 are supported.

Table 165:Return values

Name	Description
size	Size of message block in bytes.

7.14.16 sf_crypto_hash_memory_allocate

```
ssp_err_t sf_crypto_hash_memory_allocate ( sf_crypto_hash_instance_ctrl_t
*const p_ctrl ,   uint32_t digest_size )
```

7.14.16.1 Brief description

Sub-routine for Crypto HASH Framework to allocate required memory in the byte memory pool. This function is called by SF_CRYPTO_HASH_Open. Allocates additional 3 bytes when allocating memory space from the byte pool memory since the start address might not be aligned to word memory boundary.

7.14.16.2 Detailed description

Table 166:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a HASH framework control block.
digest_size	in	Digest size.

7.14.17 sf_crypto_hash_message_digest_initialize

```
sf_crypto_hash_message_digest_initialize ( sf_crypto_hash_type_t  hash_type ,
sf_crypto_hash_context_t  * p_hash_context )
```

7.14.17.1 Brief description

Sub-routine for Crypto HASH Framework to set initial message digest. This function is called by SF_CRYPTO_HASH_Open.

7.14.17.2 Detailed description

Table 167:Parameters

Name	Direction	Description
hash_type	in	HASH type. SHA1, SHA224 and SHA256 are supported.
p_hash_context	out	Pointer to a HASH context data.

7.14.18 sf_crypto_hash_interface_get

```
 sf_crypto_hash_interface_get ( sf_crypto_hash_instance_ctrl_t  const
*const p_ctrl )
```

7.14.18.1 Brief description

Subroutine to get a HASH HAL API instance. This function is called by sf_crypto_hash_instance_initialize() and sf_crypto_hash_instance_deinitialize(), sf_crypto_hash_data_blocks() and sf_crypto_hash_get_message_digest().

7.14.18.2 Detailed description

Table 168:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a HASH framework control block.

Table 169:Return values

Name	Description
non-NULL	Address to a HASH API instance.
NULL	No any HASH API instance available.

7.14.19 sf_crypto_hash_module_open_status_check

```
sf_crypto_hash_module_open_status_check ( sf_crypto_hash_state_t  status )
```

7.14.19.1 Brief description

Subroutine to check if the module has been opened.

7.14.19.2 Detailed description

Table 170:Parameters

Name	Direction	Description
status	in	Current module status.

Table 171:Return values

Name	Description
Bool	True if module is opened. False is module is closed.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.20 sf_crypto_hash_message_digest_update_status_check

```
ssp_err_t sf_crypto_hash_message_digest_update_status_check ( sf_crypto_hash_sta
te_t  status )
```

7.14.20.1 Brief description

Subroutine to check if the module has been opened and the message digest has been initialized.

7.14.20.2 Detailed description

Table 172:Parameters

Name	Direction	Description
status	in	Current module status.

Table 173:Return values

Name	Description
SSP_SUCCESS	Module has been opened and the message digest has been ever updated.
SSP_ERR_NOT_OPEN	Module is not opened.
SSP_ERR_INVALID_CALL	Message digest has not yet updated.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.21 sf_crypto_hash_message_digest_final_status_check

```
ssp_err_t sf_crypto_hash_message_digest_final_status_check ( sf_crypto_hash_state_t  status )
```

7.14.21.1 Brief description

Subroutine to check if the module has been opened and the message digest has been ever updated.

7.14.21.2 Detailed description

Table 174:Parameters

Name	Direction	Description
status	in	Current module status.

Table 175:Return values

Name	Description
SSP_SUCCESS	Module has been opened and the message digest has been ever updated.
SSP_ERR_NOT_OPEN	Module is not opened.
SSP_ERR_INVALID_CALL	Message digest has not yet updated.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.22 sf_crypto_hash_data_blocks

```
ssp_err_t sf_crypto_hash_data_blocks ( sf_crypto_hash_instance_ctrl_t
* p_ctrl ,  sf_crypto_data_handle_t  const *const p_data_in )
```

7.14.22.1 Brief description

Subroutine to hash data blocks and update the message digest.

7.14.22.2 Detailed description

Table 176:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a HASH framework control block.
p_data_in	in	Input data.

Table 177:Return values

Name	Description
SSP_SUCCESS	Module has been opened and the message digest has been ever updated.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.
Others	Crypto HAL module returned an error. See Common Error Codes for other possible return codes.

NOTE: This function does not evaluate the argument. It is the caller's responsibility.

7.14.23 sf_crypto_hash_get_message_digest

```
ssp_err_t sf_crypto_hash_get_message_digest ( sf_crypto_hash_instance_ctrl_t
* p_ctrl ,  sf_crypto_data_handle_t  *const p_msg_digest ,   uint32_t * p_size )
```

7.14.23.1 Brief description

Subroutine to hash final data block(s) and get the message digest.

7.14.23.2 Detailed description

Table 178:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a HASH framework control block.
p_msg_digest	in	Pointer to an output data buffer and the buffer size. Message digest will be generated in the buffer.

Table 179:Return values

Name	Description
SSP_SUCCESS	The module updated a message Digest successfully.
SSP_ERR_UNSUPPORTED	HASH algorithms are not supported for the MCU part.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.14.24 API Data

7.14.24.1 sf_crypto_hash_state_t

`sf_crypto_hash_state_t`

Detailed description

State codes for the SSP SSP Crypto HASH Framework

Enumerated values

Name	Description
SF_CRYPTO_HASH_CLOSED	The module is closed.
SF_CRYPTO_HASH_OPENED	The module is opened. The initial message digest is not yet generated.
SF_CRYPTO_HASH_DIGEST_INITIALIZED	Message digest is initialized.

Name	Description
SF_CRYPTO_HASH_DIGEST_UPDATED	Message digest is updated.

7.14.24.2 sf_crypto_hash_type_t

```
sf_crypto_hash_type_t
```

Detailed description

HASH algorithm types for the SSP SSP Crypto HASH Framework

Enumerated values

Name	Description
SF_CRYPTO_HASH_ALGORITHM_SHA1	SHA-1 algorithm type.
SF_CRYPTO_HASH_ALGORITHM_SHA224	SHA-224 algorithm type.
SF_CRYPTO_HASH_ALGORITHM_SHA256	SHA-256 algorithm type.

7.14.24.3 sf_crypto_hash_t

```
typedef sf_crypto_data_handle_t sf_crypto_hash_t
```

7.14.24.4 sf_crypto_hash_ctrl_t

```
typedef void sf_crypto_hash_ctrl_t
```

Detailed description

SSP Crypto framework control block. Allocate an instance specific control block to pass into the SSP Crypto framework API calls. Implemented as

- sf_crypto_instance_ctrl_t

7.14.25 Extensions

7.14.25.1 sf_crypto_hash_context_t

sf_crypto_hash_context_t

Detailed description

HASH internal context structure for a message digest

Variables

- uint8_t * p_message_digest

  Intermediate digest stored buffer.
- uint8_t * p_message_buffer

  Intermediate message data stored buffer.
- uint64_t message_bytes

  Number of bytes from user data processed.
- uint32_t message_bytes_buffered

  Number of bytes buffered in the message data stored buffer.

7.14.25.2 sf_crypto_hash_callback_args_t

sf_crypto_hash_callback_args_t

Detailed description

Callback arguments for the SSP Crypto HASH framework

Variables

- ssp_err_t error

  Error code if SF_CRYPTO_EVENT_ERROR.

7.14.25.3 sf_crypto_hash_cfg_t

sf_crypto_hash_cfg_t

Detailed description

Configuration structure for the SSP SSP Crypto HASH framework

Variables

- sf_crypto_hash_type_t hash_type

  HASH algorithm type.
- sf_crypto_instance_t * p_lower_lvl_crypto_common

  Pointer to a Crypto Framework common instance.
- hash_instance_t * p_lower_lvl_instance

  pointer to HASH lower-level module instance
- void * p_extend

  Pointer to an optional configuration for Crypto HAL module.

7.14.25.4 sf_crypto_hash_api_t

sf_crypto_hash_api_t

Detailed description

Shared Interface definition for the SSP SSP Crypto framework

7.14.25.5 sf_crypto_hash_instance_t

sf_crypto_hash_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_crypto_hash_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_crypto_hash_cfg_t * p_cfg

  Pointer to the configuration structure for this instance.

- sf_crypto_hash_api_t const * p_api

  Pointer to the API structure for this instance.

7.14.25.6 sf_crypto_hash_instance_ctrl_t

sf_crypto_hash_instance_ctrl_t

Detailed description

SSP Crypto HASH Framework instance control block

Variables

- sf_crypto_hash_state_t status

  Module status.

- sf_crypto_hash_type_t hash_type

  HASH algorithm type.

- sf_crypto_hash_context_t hash_context

  Context for calculating message digest.

- sf_crypto_instance_t * p_lower_lvl_crypto_common

  Pointer to a Crypto Framework common instance.

- hash_instance_t * p_lower_lvl_instance

  pointer to lower-level crypto module control structure

7.15 SSP Crypto TRNG Framework Interface

Interface definition for Synergy Crypto TRNG Framework module.

Interface declaration for Synergy Crypto TRNG Framework module.

7.15.1 Summary

This is the Interface of SF_CRYPTO_TRNG Framework module.
SF_CRYPTO_TRNG Interface description: Crypto Framework

7.15.2 Summary

This is the Interface declarations of SF_CRYPTO_TRNG Framework module.
SF_CRYPTO_TRNG Interface declaration: Crypto Framework

7.15.3 Functions

- SF_CRYPTO_TRNG_Open
- SF_CRYPTO_TRNG_Close
- SF_CRYPTO_TRNG_RandomNumberGenerate
- SF_CRYPTO_TRNG_VersionGet

7.15.4 Variables

- sf_crypto_trng_version
- g_sf_crypto_trng_api

7.15.5 Typedefs

- sf_crypto_trng_ctrl_t

7.15.6 Defines

- #define SF_CRYPTO_TRNG_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  The API version of SSP Crypto Framework
- #define SF_CRYPTO_TRNG_API_VERSION_MINOR

  Initial value:`(0U)`
- #define WORDS_IN_SINGLE_HW_PROC_CALL

  Initial value:`(4)`
- #define BYTES_IN_WORD

  Initial value:`(4)`

- #define BYTES_IN_SINGLE_HW_PROC_CALL

  Initial value:`((BYTES_IN_WORD) * (WORDS_IN_SINGLE_HW_PROC_CALL))`
- #define EXTRA_BUFFER_SIZE_WORDS

  Initial value:`((BYTES_IN_SINGLE_HW_PROC_CALL) / (BYTES_IN_WORD))`

7.15.7 SF_CRYPTO_TRNG_Open

```
ssp_err_t SF_CRYPTO_TRNG_Open ( sf_crypto_trng_ctrl_t  *const p_api_ctrl ,
 sf_crypto_trng_cfg_t  const *const p_cfg )
```

7.15.7.1 Brief description

SSP Crypto TRNG Framework Open operation.

7.15.7.2 Detailed description

Table 180:Return values

Name	Description
SSP_SUCCESS	The module was successfully opened
SSP_ERR_ASSERTION	One or more input parameters are NULL.
SSP_ERR_CRYPTO_COMMON_NOT_OPENED	Crypto framework common module has not been opened before calling this API.
SSP_ERR_ALREADY_OPEN	The module has already been opened.
Others	Crypto HAL module returned an error or resource lock/unlock failed. See Common Error Codes for other possible return codes.

7.15.7.3 Function steps

- Define and enter critical region

7.15.8 SF_CRYPTO_TRNG_Close

```
ssp_err_t SF_CRYPTO_TRNG_Close ( sf_crypto_trng_ctrl_t  *const p_api_ctrl )
```

7.15.8.1 Brief description

SSP Crypto TRNG Framework Close operation.

7.15.8.2 Detailed description

Table 181:Return values

Name	Description
SSP_SUCCESS	The module was successfully closed
SSP_ERR_ASSERTION	One or more input parameters are NULL.
SSP_ERR_CRYPTO_COMMON_NOT_OPENED	Crypto Common module has not been opened.
SSP_ERR_NOT_OPEN	The module has not been opened before calling this API.
Others	Crypto HAL module returned an error or resource lock/unlock failed. See Common Error Codes for other possible return codes.

7.15.9 SF_CRYPTO_TRNG_RandomNumberGenerate

```
ssp_err_t SF_CRYPTO_TRNG_RandomNumberGenerate ( sf_crypto_trng_ctrl_t
*const p_api_ctrl ,  sf_crypto_data_handle_t  *const p_random_number_buff )
```

7.15.9.1 Brief description

SSP Crypto TRNG Framework True Random Generation operation.

7.15.9.2 Detailed description

Table 182:Return values

Name	Description
SSP_SUCCESS	The module was successfully closed.
SSP_ERR_NOT_OPEN	The module has not been opened before calling this API.
SSP_ERR_ASSERTION	One or more input parameters are NULL.

Table 182:Return values (Continued)

Name	Description
Others	Crypto HAL module returned an error or resource lock/unlock failed. See Common Error Codes for other possible return codes.

7.15.10 SF_CRYPTO_TRNG_VersionGet

```
ssp_err_t SF_CRYPTO_TRNG_VersionGet ( ssp_version_t  *const p_version )
```

7.15.10.1 Brief description

Sets TRNG Framework Code and API version based on compile time macros.

7.15.10.2 Detailed description

Table 183:Return values

Name	Description
SSP_SUCCESS	Successful close
SSP_ERR_ASSERTION	The parameter p_version is NULL.

7.15.11 g_sf_crypto_trng_api

```
sf_crypto_trng_api_t::g_sf_crypto_trng_api
```

7.15.11.1 Detailed description

Filled in Interface API structure for this Instance.

7.15.11.2 Initialized as

```
g_sf_crypto_trng_api=
{
    .open                       = SF_CRYPTO_TRNG_Open,
    .close                      = SF_CRYPTO_TRNG_Close,
    .randomNumberGenerate       = SF_CRYPTO_TRNG_RandomNumberGenerate,
    .versionGet                 = SF_CRYPTO_TRNG_VersionGet
}
```

7.15.12 API Data

7.15.12.1 sf_crypto_trng_state_t

```
sf_crypto_trng_state_t
```

Detailed description

State codes for the SSP Crypto TRNG framework

Enumerated values

Name	Description
SF_CRYPTO_TRNG_CLOSED	Crypto TRNG Framework Module module is closed.
SF_CRYPTO_TRNG_OPENED	Crypto TRNG Framework Module module is opened.

7.15.12.2 sf_crypto_trng_ctrl_t

```
typedef void sf_crypto_trng_ctrl_t
```

Detailed description

SSP Crypto TRNG framework control block. Implemented as

- sf_crypto_trng_ctrl_t

7.15.13 Extensions

7.15.13.1 sf_crypto_trng_cfg_t

sf_crypto_trng_cfg_t

Detailed description

Configuration structure for the SSP Crypto TRNG framework

Variables

- sf_crypto_instance_t * p_lower_lvl_common

  Pointer to a Crypto Framework common instance.

- trng_instance_t * p_lower_lvl_instance

  Pointer to Crypto TRNG HAL instance.

- void * p_extend

  Pointer to an optional configuration for HW specific settings.

7.15.13.2 sf_crypto_trng_api_t

sf_crypto_trng_api_t

Detailed description

Shared Interface definition for the SSP Crypto framework

7.15.13.3 sf_crypto_trng_instance_t

sf_crypto_trng_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_crypto_trng_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_crypto_trng_cfg_t * p_cfg

  Pointer to the configuration structure for this instance.
- sf_crypto_trng_api_t const * p_api

  Pointer to the API structure for this instance.

7.16 SSP Crypto Key Framework Interface

Interface definition for Synergy Crypto Key Framework module.

Interface declaration for Synergy Crypto Key Framework module.

SF_CRYPTO_KEY Interface description: Crypto Framework

7.16.1 Summary

This is the Interface of SF_CRYPTO KEY Framework module.

SF_CRYPTO_KEY Interface description: Crypto Framework

7.16.2 Functions

- sf_crypto_key_hal_open
- sf_crypto_key_hal_close
- sf_crypto_key_hal_generate
- SF_CRYPTO_KEY_Open
- SF_CRYPTO_KEY_Close

- SF_CRYPTO_KEY_Generate
- SF_CRYPTO_KEY_VersionGet

7.16.3 Variables

- sf_crypto_key_version
- g_sf_crypto_key_api

7.16.4 Typedefs

- sf_crypto_key_t
- sf_crypto_key_ctrl_t

7.16.5 Defines

- #define SF_CRYPTO_KEY_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. The API version of SSP Crypto Framework
- #define SF_CRYPTO_KEY_API_VERSION_MINOR

  Initial value:`(0U)`
- #define SF_CRYPTO_KEY_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. The API version of SSP Crypto Framework
- #define SF_CRYPTO_KEY_CODE_VERSION_MINOR

  Initial value:`(0U)`
- #define OPEN

  Initial value:`(0x43525054U)`

  "CRPT" in ASCII, used to identify Crypto Framework opened.

7.16.6 sf_crypto_key_hal_open

```
ssp_err_t sf_crypto_key_hal_open ( sf_crypto_key_instance_ctrl_t
*const p_ctrl ,  sf_crypto_key_cfg_t  const *const p_cfg )
```

7.16.6.1 Brief description

Sub-routine for Crypto KeyFramework to open a Crypto HAL module. This function is called by SF_CRYPTO_KEY_Open.

7.16.6.2 Detailed description

Table 184:Return values

Name	Description
SSP_SUCCESS	The module was successfully opened.
SSP_ERR_UNSUPPORTED	The module does not support the key type specified by user.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.7 sf_crypto_key_hal_close

```
ssp_err_t sf_crypto_key_hal_close ( sf_crypto_key_instance_ctrl_t
*const p_ctrl )
```

7.16.7.1 Brief description

Sub-routine for Crypto KeyFramework to close a Crypto HAL module. This function is called by SF_CRYPTO_KEY_Close.

7.16.7.2 Detailed description

Table 185:Return values

Name	Description
SSP_SUCCESS	The module was successfully closed.
SSP_ERR_UNSUPPORTED	The module does not support the key type specified by user.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.8 sf_crypto_key_hal_generate

```
ssp_err_t sf_crypto_key_hal_generate ( sf_crypto_key_instance_ctrl_t
*const p_ctrl ,  sf_crypto_key_t  *const p_secret_key ,  sf_crypto_key_t
*const p_public_key )
```

7.16.8.1 Brief description

Sub-routine for Crypto KeyFramework to let a Crypto HAL module generate a key. This function is called by SF_CRYPTO_KEY_Close.

7.16.8.2 Detailed description

Table 186:Return values

Name	Description
SSP_SUCCESS	The module generated a key successfully.
SSP_ERR_UNSUPPORTED	The module does not support the key type specified by user.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.9 SF_CRYPTO_KEY_Open

```
ssp_err_t SF_CRYPTO_KEY_Open ( sf_crypto_key_ctrl_t  *const p_api_ctrl ,
 sf_crypto_key_cfg_t  const *const p_cfg )
```

7.16.9.1 Brief description

SSP Crypto KeyFramework Open operation.

7.16.9.2 Detailed description

Table 187:Return values

Name	Description
SSP_SUCCESS	The module was successfully opened.
SSP_ERR_ASSERTION	NULL is passed through an argument.

Table 187:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_COMMON_NOT_OPENED	Crypto Framework Common Module has yet been opened.
SSP_ERR_ALREADY_OPEN	The module has been already opened.
SSP_ERR_UNSUPPORTED	The module does not support the key type specified by user.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.10 SF_CRYPTO_KEY_Close

```
ssp_err_t SF_CRYPTO_KEY_Close ( sf_crypto_key_ctrl_t  *const p_api_ctrl )
```

7.16.10.1 Brief description

SSP Crypto Key Framework Close operation.

7.16.10.2 Detailed description

Table 188:Return values

Name	Description
SSP_SUCCESS	The module was successfully closed.
SSP_ERR_ASSERTION	NULL is passed through the argument.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.11 SF_CRYPTO_KEY_Generate

```
ssp_err_t SF_CRYPTO_KEY_Generate ( sf_crypto_key_ctrl_t  *const p_api_ctrl ,
 sf_crypto_key_t  *const p_secret_key ,  sf_crypto_key_t  *const p_public_key )
```

7.16.11.1 Brief description

SSP Crypto Framework Key Generate operation.

7.16.11.2 Detailed description

Table 189:Return values

Name	Description
SSP_SUCCESS	The module created a key successfully.
SSP_ERR_ASSERTION	NULL is passed through an argument.
SSP_ERR_NOT_OPEN	The module has yet been opened. Call Open API first.
SSP_ERR_UNSUPPORTED	The module does not support the key type specified by user.
Others	Crypto HAL module returned an error or resource lock/unlock was failed. See Common Error Codes for other possible return codes.

7.16.12 SF_CRYPTO_KEY_VersionGet

```
ssp_err_t SF_CRYPTO_KEY_VersionGet ( ssp_version_t  *const p_version )
```

7.16.12.1 Brief description

Gets driver version based on compile time macros.

7.16.12.2 Detailed description

Table 190:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

7.16.13 g_sf_crypto_key_api

```
sf_crypto_key_api_t::g_sf_crypto_key_api
```

7.16.13.1 Initialized as

```
g_sf_crypto_key_api=
{
    .open           = SF_CRYPTO_KEY_Open,
    .close          = SF_CRYPTO_KEY_Close,
    .keyGenerate    = SF_CRYPTO_KEY_Generate,
    .versionGet     = SF_CRYPTO_KEY_VersionGet
}
```

7.16.14 API Data

7.16.14.1 sf_crypto_key_state_t

```
sf_crypto_key_state_t
```

Detailed description

State codes for the SSP Crypto Key framework module. Once the module is opened successfully, then the state is transition to OPENED state. After Key operations, the Key framework module must be closed with CLOSED state.

Enumerated values

Name	Description
SF_CRYPTO_KEY_CLOSED	The Key module is closed.
SF_CRYPTO_KEY_OPENED	The Key module is opened The code means 'OPEN'.

7.16.14.2 sf_crypto_key_t

```
typedef sf_crypto_data_handle_t sf_crypto_key_t
```

7.16.14.3 sf_crypto_key_ctrl_t

```
typedef void sf_crypto_key_ctrl_t
```

Detailed description

SSP Crypto framework control block. Allocate an instance specific control block to pass into the SSP Crypto framework API calls. Implemented as

- sf_crypto_instance_ctrl_t

7.16.15 Extensions

7.16.15.1 sf_crypto_key_cfg_t

sf_crypto_key_cfg_t

Detailed description

Configuration structure for the SSP SSP Crypto Key framework

Variables

- sf_crypto_key_type_t key_type

  Key type to be generated.

- sf_crypto_key_size_t key_size

  Key size to be generated.

- sf_crypto_data_handle_t domain_params

  Pointer to domain parameters for the requested key type. Structure contains, pointer to the data (contains the domain data) and data length. Should set to NULL for RSA and AES.

- sf_crypto_instance_t * p_lower_lvl_crypto_common

  Pointer to a Crypto Framework common instance.

- void const * p_extend

  Extension parameter for hardware specific settings (Future purpose).

7.16.15.2 sf_crypto_key_api_t

sf_crypto_key_api_t

Detailed description

Shared Interface definition for the SSP SSP Crypto framework

7.16.15.3 sf_crypto_key_instance_t

sf_crypto_key_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_crypto_key_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_crypto_key_cfg_t * p_cfg

  Pointer to the configuration structure for this instance.

- sf_crypto_key_api_t const * p_api

  Pointer to the API structure for this instance.

7.16.15.4 sf_crypto_key_instance_ctrl_t

sf_crypto_key_instance_ctrl_t

Detailed description

SSP Crypto Key Framework instance control block

Variables

- sf_crypto_key_state_t status

  Module status.

- sf_crypto_key_type_t key_type

  Key type.

- sf_crypto_key_size_t key_size

  Key size.

- sf_crypto_data_handle_t domain_params

  Domain parameters structure with pointer to data and length.

- sf_crypto_instance_ctrl_t * p_fwk_common_ctrl

  Pointer to the Crypto Framework Common instance.

- sf_crypto_api_t * p_fwk_common_api

  Pointer to the Crypto Framework Common instance.

- void * p_hal_ctrl

  pointer to Crypto module control structure

- void * p_hal_api

  pointer to Crypto module API structure

7.17 GUIX Interface

Interface definition for Adapting Express Logic GUIX for Synergy graphics drivers.

7.17.1 Summary

This is the Interface of SF_EL_GX Framework module which ties Synergy graphics device drivers to GUIX. The interface provides following driver adaptation for GUIX:

- DISPLAY driver for displaying image on LCD(e.g. GLCDC)

- Dave/2d driver for image rendering by 2DG engine
- JPEG driver for the image rendering by JPEG engine
- Software image rendering with no hardware acceleration

Implemented by: GUIX Synergy Port

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

GUIX Interface description: GUIX Synergy Port Framework Module

7.17.2 Interface API

sf_el_gx_api_t

Function name	Description
.open	Open SSP GUIX adaptation framework.
.close	Close SSP GUIX adaptation framework.
.versionGet	Get version.
.setup	Express Logic GUIX Driver setup entry.
.canvasInit	Canvas initialization. Set the memory address of the initial canvas.

7.17.3 Data structures

- sf_el_gx_callback_args_t
- sf_el_gx_cfg_t
- sf_el_gx_instance_t

7.17.4 Enumerations

- sf_el_gx_state_t
- sf_el_gx_device_t
- sf_el_gx_event_t

7.17.5 Typedefs

- sf_el_gx_ctrl_t

7.17.6 Defines

- #define SF_EL_GX_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  The API version of GUIX integrated driver framework
- #define SF_EL_GX_API_VERSION_MINOR

  Initial value:`(3U)`

7.17.7 API Data

7.17.7.1 sf_el_gx_state_t

`sf_el_gx_state_t`

Detailed description

State codes for the SSP GUIX adaptation framework

Enumerated values

Name	Description
SF_EL_GX_CLOSED	
SF_EL_GX_OPENED	
SF_EL_GX_CONFIGURED	

7.17.7.2 sf_el_gx_device_t

`sf_el_gx_device_t`

Detailed description

Low level device code for the GUIX

Enumerated values

Name	Description
SF_EL_GX_DEVICE_NONE	Non hardware.
SF_EL_GX_DEVICE_DISPLAY	Display device.
SF_EL_GX_DEVICE_DRW	2D Graphics Engine
SF_EL_GX_DEVICE_JPEG	JPEG Decoder.

7.17.7.3 sf_el_gx_event_t

```
sf_el_gx_event_t
```

Detailed description

Display event codes

Enumerated values

Name	Description
SF_EL_GX_EVENT_ERROR	Low level driver error occurs.
SF_EL_GX_EVENT_DISPLAY_VSYNC	Display interface VSYNC.
SF_EL_GX_EVENT_UNDERFLOW	Display interface underflow.

7.17.7.4 sf_el_gx_ctrl_t

```
typedef void sf_el_gx_ctrl_t
```

Detailed description

GUIX adaptation framework control block. Allocate an instance specific control block to pass into the GUIX adaptation framework API calls. Implemented as

- sf_el_gx_instance_ctrl_t

7.17.8 API Structures

7.17.8.1 sf_el_gx_callback_args_t

sf_el_gx_callback_args_t

Detailed description

Callback arguments for the SSP GUIX adaptation framework

Variables

- sf_el_gx_device_t device

  Device code.
- sf_el_gx_event_t event

  Event code of the low level hardware.
- uint32_t error

  Error code if SF_EL_GX_EVENT_ERROR.

7.17.8.2 sf_el_gx_cfg_t

sf_el_gx_cfg_t

Detailed description

Configuration structure for the SSP GUIX adaptation framework

Variables

- display_instance_t * p_display_instance

  Pointer to a display instance.
- display_runtime_cfg_t * p_display_runtime_cfg

  Pointer to a runtime display configuration.
- void * p_canvas

  Pointer to a canvas(reserved)
- void * p_framebuffer_a

  Pointer to a frame buffer(A)
- void * p_framebuffer_b

  Pointer to a frame buffer(B)
- void(* p_callback)(sf_el_gx_callback_args_t *p_args)

  Pointer to callback function.
- void * p_context

  Pointer to a context.
- void * p_jpegbuffer

  Pointer to a JPEG work buffer.
- uint32_t jpegbuffer_size

  Size of a JPEG work buffer.
- uint16_t rotation_angle

  Screen rotation angle(0/90/270)

- void * p_sf_jpeg_decode_instance

  Pointer to a JPEG framework instance.

7.17.8.3 sf_el_gx_api_t

sf_el_gx_api_t

Detailed description

Shared Interface definition for the SSP GUIX adaptation framework

7.17.8.4 open

```
ssp_err_t(* sf_el_gx_api_t::open) (sf_el_gx_ctrl_t *const p_ctrl, sf_el_gx_cfg_t
const *const p_cfg)
```

Detailed description

Open SSP GUIX adaptation framework. Implemented as

- SF_EL_GX_Open

Table 191:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to SF_EL_GX control block structure. Must be declared by user. Value set here.
p_cfg	in	Pointer to SF_EL_GX configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `sf_el_gx_ctrl_t*const p_ctrl`

GUIX adaptation framework control block. Allocate an instance specific control block to pass into the GUIX adaptation framework API calls. Implemented assf_el_gx_instance_ctrl_t

Parameter p_cfg

Definition: `sf_el_gx_cfg_t const *const p_cfg`

Configuration structure for the SSP GUIX adaptation framework

- `sf_el_gx_cfg_t`::`display_instance_t`

  Pointer to a display instance.

- `sf_el_gx_cfg_t`::`display_runtime_cfg_t`

  Pointer to a runtime display configuration.

- sf_el_gx_cfg_t::p_canvas

  Pointer to a canvas(reserved)
- sf_el_gx_cfg_t::p_framebuffer_a

  Pointer to a frame buffer(A)
- sf_el_gx_cfg_t::p_framebuffer_b

  Pointer to a frame buffer(B)
- sf_el_gx_cfg_t::p_callback

  Pointer to callback function.
- sf_el_gx_cfg_t::p_context

  Pointer to a context.
- sf_el_gx_cfg_t::p_jpegbuffer

  Pointer to a JPEG work buffer.
- sf_el_gx_cfg_t::jpegbuffer_size

  Size of a JPEG work buffer.
- sf_el_gx_cfg_t::rotation_angle

  Screen rotation angle(0/90/270)
- sf_el_gx_cfg_t::p_sf_jpeg_decode_instance

  Pointer to a JPEG framework instance.

7.17.8.5 close

```
ssp_err_t(* sf_el_gx_api_t::close) (sf_el_gx_ctrl_t *const p_ctrl)
```

Detailed description

Close SSP GUIX adaptation framework. Implemented as

- SF_EL_GX_Close

Table 192:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to SF_EL_GX control block structure.

Parameter p_ctrl

Definition: `sf_el_gx_ctrl_t*const p_ctrl`

GUIX adaptation framework control block. Allocate an instance specific control block to pass into the GUIX adaptation framework API calls. Implemented assf_el_gx_instance_ctrl_t

7.17.8.6 versionGet

```
ssp_err_t(* sf_el_gx_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get version. Implemented as

- SF_EL_GX_VersionGet

Table 193:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory to store the version information.

Parameter p_version

7.17.8.7 setup

```
UINT(* sf_el_gx_api_t::setup) (GX_DISPLAY *p_display)
```

Detailed description

Express Logic GUIX Driver setup entry. Implemented as

- SF_EL_GX_Setup

Table 194:Parameters

Name	Direction	Description
p_display	in	Pointer to GUIX display driver setup function.

Parameter p_display

`const`

7.17.8.8 canvasInit

```
ssp_err_t(* sf_el_gx_api_t::canvasInit) (sf_el_gx_ctrl_t *const p_ctrl,
GX_WINDOW_ROOT *p_window_root)
```

Detailed description

Canvas initialization. Set the memory address of the initial canvas. Implemented as

- SF_EL_GX_CanvasInit

Table 195:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to SF_EL_GX control block structure.
p_window_root	in	Pointer to GUIX root window context.

Parameter p_ctrl

Definition: `sf_el_gx_ctrl_t*const p_ctrl`

GUIX adaptation framework control block. Allocate an instance specific control block to pass into the GUIX adaptation framework API calls. Implemented assf_el_gx_instance_ctrl_t

Parameter p_window_root

`const`

7.17.8.9 sf_el_gx_instance_t

sf_el_gx_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_el_gx_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_el_gx_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_el_gx_api_t const * p_api

  Pointer to the API structure for this instance.

7.18 SF_EL_UX API Framework

RTOS-integrated SF_EL_UX Framework.

7.18.1 Summary

This module is a ThreadX-aware SF_EL_UX Framework.

7.18.2 Functions

- ux_system_initialize
- ux_system_uninitialize
- ux_hcd_ehci_initialize
- ux_hcd_isp1161_initialize
- ux_hcd_ohci_initialize
- ux_hcd_sim_host_initialize
- ux_host_stack_class_get
- ux_host_stack_class_instance_create
- ux_host_stack_class_instance_destroy
- ux_host_stack_class_instance_get
- ux_host_stack_class_register
- ux_host_stack_configuration_interface_get
- ux_host_stack_device_configuration_get
- ux_host_stack_device_configuration_select
- ux_host_stack_device_get
- ux_host_stack_endpoint_transfer_abort
- ux_host_stack_hcd_register
- ux_host_stack_initialize
- ux_host_stack_interface_endpoint_get
- ux_host_stack_interface_setting_select
- ux_host_stack_transfer_request
- ux_host_stack_transfer_request_abort
- ux_host_stack_hnp_polling_thread_entry
- ux_host_stack_role_swap
- ux_dcd_at91_initialize
- ux_dcd_isp1181_initialize
- ux_dcd_ml6965_initialize
- ux_dcd_sim_slave_initialize
- ux_device_class_storage_entry
- ux_device_class_storage_thread
- ux_device_stack_alternate_setting_get
- ux_device_stack_alternate_setting_set
- ux_device_stack_class_register

- ux_device_stack_class_unregister
- ux_device_stack_configuration_get
- ux_device_stack_configuration_set
- ux_device_stack_descriptor_send
- ux_device_stack_disconnect
- ux_device_stack_endpoint_stall
- ux_device_stack_host_wakeup
- ux_device_stack_initialize
- ux_device_stack_uninitialize
- ux_device_stack_interface_delete
- ux_device_stack_interface_get
- ux_device_stack_interface_set
- ux_device_stack_interface_start
- ux_device_stack_transfer_request
- ux_device_stack_transfer_request_abort

7.18.3 ux_system_initialize

```
ux_system_initialize (  VOID * non_cached_memory_pool_start ,
 ULONG non_cached_memory_size ,   VOID * cached_memory_pool_start ,
 ULONG cached_memory_size )
```

7.18.4 ux_system_uninitialize

```
ux_system_uninitialize (  VOID  )
```

7.18.5 ux_hcd_ehci_initialize

```
ux_hcd_ehci_initialize ( UX_HCD  * hcd )
```

7.18.6 ux_hcd_isp1161_initialize

```
ux_hcd_isp1161_initialize ( UX_HCD  * hcd )
```

7.18.7 ux_hcd_ohci_initialize

```
ux_hcd_ohci_initialize ( UX_HCD  * hcd )
```

7.18.8 ux_hcd_sim_host_initialize

```
ux_hcd_sim_host_initialize ( UX_HCD  * hcd )
```

7.18.9 ux_host_stack_class_get

```
ux_host_stack_class_get (  UCHAR * class_name ,  UX_HOST_CLASS  ** host_class )
```

7.18.10 ux_host_stack_class_instance_create

```
ux_host_stack_class_instance_create ( UX_HOST_CLASS  * host_class ,   VOID
* class_instance )
```

7.18.11 ux_host_stack_class_instance_destroy

```
ux_host_stack_class_instance_destroy ( UX_HOST_CLASS  * host_class ,   VOID
* class_instance )
```

7.18.12 ux_host_stack_class_instance_get

```
ux_host_stack_class_instance_get ( UX_HOST_CLASS  * host_class ,
 UINT class_index ,   VOID ** class_instance )
```

7.18.13 ux_host_stack_class_register

```
ux_host_stack_class_register (  UCHAR * class_name ,   UINT(*)(struct
UX_HOST_CLASS_COMMAND_STRUCT *) class_entry_function )
```

7.18.14 ux_host_stack_configuration_interface_get

```
ux_host_stack_configuration_interface_get ( UX_CONFIGURATION  * configuration ,
 UINT interface_index ,   UINT alternate_setting_index ,  UX_INTERFACE
** interface )
```

7.18.15 ux_host_stack_device_configuration_get

```
ux_host_stack_device_configuration_get ( UX_DEVICE  * device ,
 UINT configuration_index ,  UX_CONFIGURATION  ** configuration )
```

7.18.16 ux_host_stack_device_configuration_select

```
ux_host_stack_device_configuration_select ( UX_CONFIGURATION  * configuration )
```

7.18.17 ux_host_stack_device_get

```
ux_host_stack_device_get (  ULONG device_index ,  UX_DEVICE  ** device )
```

7.18.18 ux_host_stack_endpoint_transfer_abort

```
ux_host_stack_endpoint_transfer_abort ( UX_ENDPOINT  * endpoint )
```

7.18.19 ux_host_stack_hcd_register

```
ux_host_stack_hcd_register (  UCHAR * hcd_name ,   UINT(*)(struct UX_HCD_STRUCT
*) hcd_initialize_function ,   ULONG hcd_param1 ,   ULONG hcd_param2 )
```

7.18.20 ux_host_stack_initialize

```
ux_host_stack_initialize ( UX_HOST_CLASS UINT(*)(ULONG, *, VOID
*) ux_system_host_change_function )
```

7.18.21 ux_host_stack_interface_endpoint_get

```
ux_host_stack_interface_endpoint_get ( UX_INTERFACE  * interface ,
 UINT endpoint_index ,  UX_ENDPOINT  ** endpoint )
```

7.18.22 ux_host_stack_interface_setting_select

```
ux_host_stack_interface_setting_select ( UX_INTERFACE  * interface )
```

7.18.23 ux_host_stack_transfer_request

```
ux_host_stack_transfer_request ( UX_TRANSFER  * transfer_request )
```

7.18.24 ux_host_stack_transfer_request_abort

```
ux_host_stack_transfer_request_abort ( UX_TRANSFER  * transfer_request )
```

7.18.25 ux_host_stack_hnp_polling_thread_entry

```
ux_host_stack_hnp_polling_thread_entry (  ULONG id )
```

7.18.26 ux_host_stack_role_swap

```
ux_host_stack_role_swap ( UX_DEVICE  * device )
```

7.18.27 ux_dcd_at91_initialize

```
ux_dcd_at91_initialize (  ULONG dcd_io )
```

7.18.28 ux_dcd_isp1181_initialize

```
ux_dcd_isp1181_initialize (  ULONG dcd_io ,   ULONG dcd_irq ,
 ULONG dcd_vbus_address )
```

7.18.29 ux_dcd_ml6965_initialize

```
ux_dcd_ml6965_initialize (  ULONG dcd_io ,   ULONG dcd_irq ,
 ULONG dcd_vbus_address )
```

7.18.30 ux_dcd_sim_slave_initialize

```
ux_dcd_sim_slave_initialize (  VOID  )
```

7.18.31 ux_device_class_storage_entry

```
ux_device_class_storage_entry ( UX_SLAVE_CLASS_COMMAND  * command )
```

7.18.32 ux_device_class_storage_thread

```
ux_device_class_storage_thread (  ULONG  )
```

7.18.33 ux_device_stack_alternate_setting_get

```
ux_device_stack_alternate_setting_get (  ULONG interface_value )
```

7.18.34 ux_device_stack_alternate_setting_set

```
ux_device_stack_alternate_setting_set (  ULONG interface_value ,
 ULONG alternate_setting_value )
```

7.18.35 ux_device_stack_class_register

```
ux_device_stack_class_register (  UCHAR * class_name ,   UINT(*)(struct
UX_SLAVE_CLASS_COMMAND_STRUCT *) class_entry_function ,
  ULONG configuration_number ,   ULONG interface_number ,   VOID * parameter )
```

7.18.36 ux_device_stack_class_unregister

```
 ux_device_stack_class_unregister (  UCHAR * class_name ,   UINT(*)(struct
UX_SLAVE_CLASS_COMMAND_STRUCT *) class_entry_function )
```

7.18.37 ux_device_stack_configuration_get

```
 ux_device_stack_configuration_get (  VOID  )
```

7.18.38 ux_device_stack_configuration_set

```
 ux_device_stack_configuration_set (  ULONG configuration_value )
```

7.18.39 ux_device_stack_descriptor_send

```
 ux_device_stack_descriptor_send (  ULONG descriptor_type ,
  ULONG request_index ,   ULONG host_length )
```

7.18.40 ux_device_stack_disconnect

```
 ux_device_stack_disconnect (  VOID  )
```

7.18.41 ux_device_stack_endpoint_stall

```
 ux_device_stack_endpoint_stall ( UX_SLAVE_ENDPOINT  * endpoint )
```

7.18.42 ux_device_stack_host_wakeup

```
 ux_device_stack_host_wakeup (  VOID  )
```

7.18.43 ux_device_stack_initialize

```
 ux_device_stack_initialize (  UCHAR * device_framework_high_speed ,
  ULONG device_framework_length_high_speed ,   UCHAR
* device_framework_full_speed ,   ULONG device_framework_length_full_speed ,
  UCHAR * string_framework ,   ULONG string_framework_length ,   UCHAR
* language_id_framework ,   ULONG language_id_framework_length ,
  UINT(*)(ULONG) ux_system_slave_change_function )
```

7.18.44 ux_device_stack_uninitialize

```
 ux_device_stack_uninitialize (  VOID  )
```

7.18.45 ux_device_stack_interface_delete

```
ux_device_stack_interface_delete ( UX_SLAVE_INTERFACE  * interface )
```

7.18.46 ux_device_stack_interface_get

```
ux_device_stack_interface_get (  UINT interface_value )
```

7.18.47 ux_device_stack_interface_set

```
ux_device_stack_interface_set (  UCHAR * device_framework ,
 ULONG device_framework_length ,   ULONG alternate_setting_value )
```

7.18.48 ux_device_stack_interface_start

```
ux_device_stack_interface_start ( UX_SLAVE_INTERFACE  * interface )
```

7.18.49 ux_device_stack_transfer_request

```
ux_device_stack_transfer_request ( UX_SLAVE_TRANSFER  * transfer_request ,
 ULONG slave_length ,   ULONG host_length )
```

7.18.50 ux_device_stack_transfer_request_abort

```
 ux_device_stack_transfer_request_abort ( UX_SLAVE_TRANSFER
* transfer_request ,   ULONG completion_code )
```

7.19 External IRQ Framework Interface

RTOS-integrated External IRQ Framework Interface.

7.19.1 Summary

This module is a ThreadX-aware external IRQ Framework Interface for external inputs such as switches or other binary signals. The Interface is implemented by External IRQ Framework.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

External IRQ Framework Interface description: External IRQ Framework

7.19.2 Interface API

sf_external_irq_api_t

Function name	Description
.open	Acquire mutex, then handle driver initialization at the HAL layer.
.wait	Wait for the next external interrupt expiration, then return.
.versionGet	Get version and store it in provided pointer p_version.
.close	Release channel mutex and close channel at HAL layer.

7.19.3 Data structures

- sf_external_irq_cfg_t
- sf_external_irq_instance_t

7.19.4 Enumerations

- sf_external_irq_event_t

7.19.5 Typedefs

- sf_external_irq_ctrl_t

7.19.6 Defines

- #define SF_EXTERNAL_IRQ_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_EXTERNAL_IRQ_API_VERSION_MINOR

  Initial value:`(2U)`

7.19.7 API Data

7.19.7.1 sf_external_irq_event_t

`sf_external_irq_event_t`

Detailed description

Options for what should happen when the external interrupt expires.

Enumerated values

Name	Description
SF_EXTERNAL_IRQ_EVENT_NONE	Nothing happens during expiration. Can be used for data transfer.
SF_EXTERNAL_IRQ_EVENT_SEMAPHORE_PUT	Posts to internal semaphore. Select this if using SF_EXTERNAL_IRQ_Wait.

7.19.7.2 sf_external_irq_ctrl_t

```
typedef void sf_external_irq_ctrl_t
```

Detailed description

External interrupt control block. Allocate an instance specific control block to pass into the external interrupt framework API calls. Implemented as

- sf_external_irq_instance_ctrl_t

7.19.8 API Structures

7.19.8.1 sf_external_irq_cfg_t

sf_external_irq_cfg_t

Detailed description

Configuration for RTOS integrated external interrupt driver

Variables

- external_irq_instance_t const * p_lower_lvl_irq

  All info needed to work with lower layer

- sf_external_irq_event_t event

  Select what happens when the external IRQ is triggered.

7.19.8.2 sf_external_irq_api_t

sf_external_irq_api_t

Detailed description

External IRQ framework API structure. External IRQ implementations use the following API.

7.19.8.3 open

```
ssp_err_t(* sf_external_irq_api_t::open) (sf_external_irq_ctrl_t *const p_ctrl,
sf_external_irq_cfg_t const *const p_cfg)
```

Detailed description

Acquire mutex, then handle driver initialization at the HAL layer. Implemented as

- SF_EXTERNAL_IRQ_Open

Table 196:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. The device control structure is initialized in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_external_irq_ctrl_t*const p_ctrl`

External interrupt control block. Allocate an instance specific control block to pass into the external interrupt framework API calls. Implemented assf_external_irq_instance_ctrl_t

Parameter p_cfg

Definition: `sf_external_irq_cfg_t const *const p_cfg`

Configuration for RTOS integrated external interrupt driver

- `sf_external_irq_cfg_t`::`external_irq_instance_t`

  All info needed to work with lower layer

- `sf_external_irq_cfg_t`::`sf_external_irq_event_t`

  Select what happens when the external IRQ is triggered.

  Enumerated as:

  – SF_EXTERNAL_IRQ_EVENT_NONE

  – SF_EXTERNAL_IRQ_EVENT_SEMAPHORE_PUT

7.19.8.4 wait

```
ssp_err_t(* sf_external_irq_api_t::wait) (sf_external_irq_ctrl_t *const p_ctrl,
ULONG const timeout)
```

Detailed description

Wait for the next external interrupt expiration, then return.

NOTE: Call SF_EXTERNAL_IRQ_Open to configure the external IRQ before using this function. During SF_EXTERNAL_IRQ_Open, set sf_external_irq_cfg_t::sf_external_irq_event_t to SF_EXTERNAL_IRQ_EVENT_SEMAPHORE_PUT.

Implemented as

- SF_EXTERNAL_IRQ_Wait

Table 197:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in SF_EXTERNAL_IRQ_Open.
timeout	in	ThreadX timeout. Select TX_NO_WAIT, a value in system clock counts between 1 and 0xFFFFFFFF, or TX_WAIT_FOREVER.

Parameter p_ctrl

Definition: `sf_external_irq_ctrl_t*const p_ctrl`

External interrupt control block. Allocate an instance specific control block to pass into the external interrupt framework API calls. Implemented assf_external_irq_instance_ctrl_t

Parameter timeout

`const`

7.19.8.5 versionGet

```
ssp_err_t(* sf_external_irq_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- SF_EXTERNAL_IRQ_VersionGet

Table 198:Parameters

Name	Direction	Description
p_version	out	Code and API version used stored here.

Parameter p_version

7.19.8.6 close

```
ssp_err_t(* sf_external_irq_api_t::close) (sf_external_irq_ctrl_t *const p_ctrl)
```

Detailed description

Release channel mutex and close channel at HAL layer. Implemented as

- SF_EXTERNAL_IRQ_Close

Table 199:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to device control block initialized in Open call for this external interrupt.

Parameter p_ctrl

Definition: `sf_external_irq_ctrl_t*const p_ctrl`

External interrupt control block. Allocate an instance specific control block to pass into the external interrupt framework API calls. Implemented assf_external_irq_instance_ctrl_t

7.19.8.7 sf_external_irq_instance_t

sf_external_irq_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_external_irq_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_external_irq_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_external_irq_api_t const * p_api

  Pointer to the API structure for this instance.

7.20 JPEG Decode Framework Interface

RTOS-integrated JPEG Decode Framework Interface.

7.20.1 Summary

This is a ThreadX aware generic JPEG decoding framework for run-time JPEG decode applications. It can be implemented by either hardware or software. For Synergy parts, the interface is implemented by the on-chip JPEG decoding engine. The connection to the HAL layer is established by passing in a driver structure in SF_JPEG_Decode_Open.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Framework JPEG Decode Interface description: JPEG Decode Framework

7.20.2 Interface API

sf_jpeg_decode_api_t

Function name	Description
.open	Acquire mutex, then handle driver initialization at the HAL layer. This function releases mutex before it returns to the caller.
.inputBufferSet	Feed data into JPEG codec module.
.outputBufferSet	Read processed data from JPEG codec module.
.linesDecodedGet	Obtain number of lines JPEG codec decoded.
.horizontalStrideSet	Configure the horizontal stride value.
.imageSubsampleSet	Configure the horizontal and vertical subsample values. This allows an application to reduce the size of the decoded image.
.wait	Wait for current JPEG codec operation to finish

Function name	Description
.statusGet	Obtain JPEG codec status
.imageSizeGet	Obtain the size of the image. This function is only useful for decoding a JPEG image.
.pixelFormatGet	Obtain the pixel format of the image. This function is only useful for decoding a JPEG image.
.close	Closes JPEG codec device. Un-finished codec operation is interrupted, and output data are discarded.
.versionGet	Gets version and stores it in provided pointer p_version.

7.20.3 Data structures

- sf_jpeg_decode_cfg_t
- sf_jpeg_decode_instance_t

7.20.4 Typedefs

- sf_jpeg_decode_ctrl_t

7.20.5 Defines

- #define SF_JPEG_DECODE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file

- #define SF_JPEG_DECODE_API_VERSION_MINOR

  Initial value:`(2U)`

7.20.6 API Data

7.20.6.1 sf_jpeg_decode_ctrl_t

```
typedef void sf_jpeg_decode_ctrl_t
```

Detailed description

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented as

- sf_jpeg_decode_instance_ctrl_t

7.20.7 API Structures

7.20.7.1 sf_jpeg_decode_cfg_t

sf_jpeg_decode_cfg_t

Detailed description

Configuration for RTOS integrated JPEG driver

Variables

- jpeg_decode_instance_t const * p_lower_lvl_jpeg_decode

  Pointer to a driver structure that implements this interface. Pre-configured driver structures are located in r_jpeg_-decode.c and extern'ed in r_jpeg_decode.h.

7.20.7.2 sf_jpeg_decode_api_t

sf_jpeg_decode_api_t

Detailed description

JPEG Decode API structure. Implementations will use the following API.

7.20.7.3 open

```
(* sf_jpeg_decode_api_t::open) (sf_jpeg_decode_ctrl_t *const p_ctrl,
sf_jpeg_decode_cfg_t const *const p_cfg)
```

Detailed description

Acquire mutex, then handle driver initialization at the HAL layer. This function releases mutex before it returns to the caller.

Table 200:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. Elements initialized here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: sf_jpeg_decode_ctrl_t*const p_ctrl

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_cfg

Definition: sf_jpeg_decode_cfg_t const *const p_cfg

Configuration for RTOS integrated JPEG driver

- sf_jpeg_decode_cfg_t::jpeg_decode_instance_t

  Pointer to a driver structure that implements this interface. Pre-configured driver structures are located in r_jpeg_-decode.c and extern'ed in r_jpeg_decode.h.

7.20.7.4 inputBufferSet

```
(* sf_jpeg_decode_api_t::inputBufferSet) (sf_jpeg_decode_ctrl_t *const p_ctrl,
void *const p_buffer, uint32_t const buffer_size)
```

Detailed description

Feed data into JPEG codec module.

Table 201:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_buffer	in	Buffer contains data to be processed by the JPEG codec module. The buffer starting address must be 8-byte aligned
buffer_size	in	Size of the data buffer, must be multiple of 8 bytes

Parameter p_ctrl

Definition: sf_jpeg_decode_ctrl_t*const p_ctrl

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_buffer

const

Parameter buffer_size

uint32_t

7.20.7.5 outputBufferSet

```
(* sf_jpeg_decode_api_t::outputBufferSet) (sf_jpeg_decode_ctrl_t *const p_ctrl,
void *p_buffer, uint32_t buffer_size)
```

Detailed description

Read processed data from JPEG codec module.

Table 202:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_buffer	in	User-supplied buffer space to hold output from JPEG codec module. The buffer starting address must be 8-byte aligned
buffer_size	in	Size of the output data buffer

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_buffer

`const`

Parameter buffer_size

`uint32_t`

7.20.7.6 linesDecodedGet

```
(* sf_jpeg_decode_api_t::linesDecodedGet) (sf_jpeg_decode_ctrl_t *const p_ctrl,
uint32_t *const p_lines)
```

Detailed description

Obtain number of lines JPEG codec decoded.

Table 203:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.

Table 203:Parameters (Continued)

Name	Direction	Description
p_lines	out	Number of lines decoded into the output buffer.

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_lines

`uint32_t`

7.20.7.7 horizontalStrideSet

```
(* sf_jpeg_decode_api_t::horizontalStrideSet) (sf_jpeg_decode_ctrl_t *const
p_ctrl, uint32_t horizontal_stride)
```

Detailed description

Configure the horizontal stride value.

Table 204:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
horizontal_stride	out	Set the horizontal stride value, in pixels

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter horizontal_stride

`uint32_t`

7.20.7.8 imageSubsampleSet

```
(* sf_jpeg_decode_api_t::imageSubsampleSet) (sf_jpeg_decode_ctrl_t *const p_ctrl,
horizontal_subsample, vertical_subsample)
```

Detailed description

Configure the horizontal and vertical subsample values. This allows an application to reduce the size of the decoded image.

Table 205:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
horizontal_subsample	in	Set the horizontal subsample value
vertical_subsample	in	Set the vertical subsample value

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter horizontal_subsample

Parameter vertical_subsample

7.20.7.9 wait

```
(* sf_jpeg_decode_api_t::wait) (sf_jpeg_decode_ctrl_t *const p_ctrl, *const
p_status, uint32_t timeout)
```

Detailed description

Wait for current JPEG codec operation to finish

Table 206:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_status	out	Status of current JPEG codec module
timeout	out	Amount of time (in ThreadX ticks) to wait

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_status

Parameter timeout

uint32_t

7.20.7.10 statusGet

(* sf_jpeg_decode_api_t::statusGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, *const p_status)

Detailed description

Obtain JPEG codec status

Table 207:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_status	out	Status of current JPEG codec module

Parameter p_ctrl

Definition: sf_jpeg_decode_ctrl_t*const p_ctrl

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_status

7.20.7.11 imageSizeGet

(* sf_jpeg_decode_api_t::imageSizeGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, uint16_t *p_horizontal_size, uint16_t *p_vertical_size)

Detailed description

Obtain the size of the image. This function is only useful for decoding a JPEG image.

Table 208:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_horizontal_size	out	Width of the image, in pixels
p_vertical_size	out	Height of the image, in pixels

Parameter p_ctrl

Definition: sf_jpeg_decode_ctrl_t*const p_ctrl

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_horizontal_size

`uint16_t`

Parameter p_vertical_size

`uint16_t`

7.20.7.12 pixelFormatGet

```
(* sf_jpeg_decode_api_t::pixelFormatGet) (sf_jpeg_decode_ctrl_t *const p_ctrl,
*const p_color_space)
```

Detailed description

Obtain the pixel format of the image. This function is only useful for decoding a JPEG image.

Table 209:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block initialized in SF_JPEG_Decode_Open.
p_color_space	out	Color space of the image

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

Parameter p_color_space

7.20.7.13 close

```
(* sf_jpeg_decode_api_t::close) (sf_jpeg_decode_ctrl_t *const p_ctrl)
```

Detailed description

Closes JPEG codec device. Un-finished codec operation is interrupted, and output data are discarded.

Table 210:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block set in SF_JPEG_Decode_Open.

Parameter p_ctrl

Definition: `sf_jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode framework control block. Allocate an instance specific control block to pass into the JPEG decode framework API calls. Implemented assf_jpeg_decode_instance_ctrl_t

7.20.7.14 versionGet

`(* sf_jpeg_decode_api_t::versionGet) ( *const p_version)`

Detailed description

Gets version and stores it in provided pointer p_version.

Table 211:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.20.7.15 sf_jpeg_decode_instance_t

sf_jpeg_decode_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_jpeg_decode_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_jpeg_decode_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_jpeg_decode_api_t const * p_api

  Pointer to the API structure for this instance.

7.21 essaging Framework Interface

RTOS-integrated Messaging Framework Interface.

7.21.1 Summary

This module is a ThreadX-aware Messaging Framework. This Interface is implemented by Messaging Framework.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Messaging Interface description: Messaging Framework

7.21.2 Interface API

sf_message_api_t

Function name	Description
.open	Initialize message framework. Initiate the messaging framework control block, configure the work memory corresponding to the configuration parameters.
.close	Finalize message framework.
.bufferAcquire	Acquire buffer for message passing from the block.
.bufferRelease	Release buffer obtained from SF_MESSAGE_BufferAcquire.
.post	Post message to the subscribers.
.pend	Pend message.
.versionGet	Get the version of the messaging framework.

7.21.3 Data structures

- sf_message_header_t
- sf_message_instance_range_t
- sf_message_subscriber_t
- sf_message_subscriber_list_t
- sf_message_callback_args_t
- sf_message_post_err_t
- sf_message_buffer_ctrl_t
- sf_message_cfg_t
- sf_message_acquire_cfg_t
- sf_message_post_cfg_t
- sf_message_instance_t

7.21.4 Enumerations

- sf_message_state_t
- sf_message_callback_event_t
- sf_message_priority_t
- sf_message_release_option_t

7.21.5 Typedefs

- sf_message_ctrl_t

7.21.6 Defines

- #define SF_MESSAGE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file
- #define SF_MESSAGE_API_VERSION_MINOR

  Initial value:`(2U)`

7.21.7 API Data

7.21.7.1 sf_message_state_t

`sf_message_state_t`

Detailed description

Messaging framework state

Enumerated values

Name	Description
SF_MESSAGE_STATE_CLOSED	Messaging framework is closed.
SF_MESSAGE_STATE_OPENED	Messaging framework is opened.

7.21.7.2 sf_message_callback_event_t

`sf_message_callback_event_t`

Detailed description

Messaging callback response

Enumerated values

Name	Description
SF_MESSAGE_CALLBACK_EVENT_ACK	ACK response.
SF_MESSAGE_CALLBACK_EVENT_NAK	NAK response.

7.21.7.3 sf_message_priority_t

`sf_message_priority_t`

Detailed description

Messaging framework state

Enumerated values

Name	Description
SF_MESSAGE_PRIORITY_NORMAL	Gives a message to be sent normal priority.
SF_MESSAGE_PRIORITY_HIGH	Gives a message to be sent higher priority than previous ones.

7.21.7.4 sf_message_release_option_t

`sf_message_release_option_t`

Detailed description

Messaging option

Enumerated values

Name	Description
SF_MESSAGE_RELEASE_OPTION_NONE	No option.
SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE	Buffer forced release option.
SF_MESSAGE_RELEASE_OPTION_ACK	ACK response (note if both ACK and NAK are set at same time, NAK.
SF_MESSAGE_RELEASE_OPTION_NAK	NAK response.

7.21.7.5 sf_message_ctrl_t

```
typedef void sf_message_ctrl_t
```

Detailed description

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented as

- sf_message_instance_ctrl_t

7.21.8 API Structures

7.21.8.1 sf_message_header_t

sf_message_header_t

Detailed description

Message header definition

Variables

- uint32_t event
- uint32_t class_code

  Event class code.
- uint32_t class_instance

  Event class instance number.
- uint32_t code

  Event code.
- struct{} event_b

  See source code for definition of this member.
- union{} union{}

  See source code for definition of this member.

7.21.8.2 sf_message_instance_range_t

sf_message_instance_range_t

Detailed description

Subscriber lists definitions

Variables

- uint8_t start

  Start of the event class instance range.

- uint8_t end

  End of the event class instance range.

7.21.8.3 sf_message_subscriber_t

sf_message_subscriber_t

Detailed description

Message subscriber

Variables

- TX_QUEUE * p_queue

  Pointer to the message queue for subscriber thread.

- sf_message_instance_range_t instance_range

  Range of the event class instance to receive message.

7.21.8.4 sf_message_subscriber_list_t

sf_message_subscriber_list_t

Detailed description

Message subscriber list

Variables

- sf_message_event_class_t event_class

  Event class code.

- uint16_t number_of_nodes

  Number of nodes in the subscriber group.

- sf_message_subscriber_t ** pp_subscriber_group

  Subscriber group for the event class.

7.21.8.5 sf_message_callback_args_t

sf_message_callback_args_t

Detailed description

Message framework callback parameters

Variables

- sf_message_callback_event_t event

  Event code.

- void const * p_context

  Context provided to user during callback.

7.21.8.6 sf_message_post_err_t

sf_message_post_err_t

Detailed description

Post error information structure

Variables

- TX_QUEUE * p_queue

  Queue.

7.21.8.7 sf_message_buffer_ctrl_t

sf_message_buffer_ctrl_t

Detailed description

Buffer control block structure

Variables

- void(* p_callback)(sf_message_callback_args_t *)

  Optional user callback function.

- void const * p_context

  Context provided to user during callback.

- sf_message_buffer_ctrl_t::st_buffer_ctrl_flagstruct flag_b

7.21.8.8 sf_message_buffer_ctrl_t::st_buffer_ctrl_flag

sf_message_buffer_ctrl_t::st_buffer_ctrl_flag

Brief description

< Flags

Detailed description

Variables

- uint32_t semaphore

  Counting semaphore to prevent a buffer from being released.

- uint32_t buffer_keep

  Buffer keep request.

- uint32_t nak_response

  NAK (ORed logic for multiple subscribers)

- uint32_t reserved

  Reserved bits.

- uint32_t in_use

  Buffer in-use.

7.21.8.9 sf_message_cfg_t

sf_message_cfg_t

Detailed description

Messaging framework configuration structure definition

Variables

- void * p_work_memory_start

  Start address of the memory area.

- uint32_t work_memory_size_bytes

  Size of working memory area in bytes.

- uint32_t buffer_size

  Bytes of the message block.

- sf_message_subscriber_list_t ** pp_subscriber_lists

  Pointer array to the subscriber lists.

- uint8_t * p_block_pool_name

  Pointer to the block pool name.

7.21.8.10 sf_message_acquire_cfg_t

sf_message_acquire_cfg_t

Detailed description

Messaging framework Post API function configuration structure definition

Variables

- bool buffer_keep

  Buffer keep option.

7.21.8.11 sf_message_post_cfg_t

sf_message_post_cfg_t

Detailed description

Messaging framework Acquire API function configuration structure definition

Variables

- sf_message_priority_t priority

  Message priority.

- void(* p_callback)(sf_message_callback_args_t *)

  User callback function.

- void const * p_context

  Context provided to user during callback.

7.21.8.12 sf_message_api_t

sf_message_api_t

Detailed description

Messaging Framework API structure. Implementations will use the following API.

7.21.8.13 open

```
ssp_err_t(* sf_message_api_t::open) (sf_message_ctrl_t *const p_ctrl,
sf_message_cfg_t const *const p_cfg)
```

Detailed description

Initialize message framework. Initiate the messaging framework control block, configure the work memory corresponding to the configuration parameters. Implemented as

- SF_MESSAGE_Open

Table 212:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the messaging control block
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `sf_message_ctrl_t*const p_ctrl`

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

Parameter p_cfg

Definition: `sf_message_cfg_t const *const p_cfg`

Messaging framework configuration structure definition

- `sf_message_cfg_t`::`p_work_memory_start`

  Start address of the memory area.

- `sf_message_cfg_t`::`work_memory_size_bytes`

  Size of working memory area in bytes.

- sf_message_cfg_t::buffer_size

  Bytes of the message block.

- sf_message_cfg_t::sf_message_subscriber_list_t

  Pointer array to the subscriber lists.

- sf_message_cfg_t::p_block_pool_name

  Pointer to the block pool name.

7.21.8.14 close

```
ssp_err_t(* sf_message_api_t::close) (sf_message_ctrl_t *const p_ctrl)
```

Detailed description

Finalize message framework. Implemented as

- SF_MESSAGE_Close

Table 213:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the messaging control block

Parameter p_ctrl

Definition: sf_message_ctrl_t*const p_ctrl

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

7.21.8.15 bufferAcquire

```
ssp_err_t(* sf_message_api_t::bufferAcquire) (sf_message_ctrl_t const *const
p_ctrl, sf_message_header_t **pp_buffer, sf_message_acquire_cfg_t const *const
p_acquire_cfg, uint32_t const wait_option)
```

Detailed description

Acquire buffer for message passing from the block. Implemented as

- SF_MESSAGE_BufferAcquire

Table 214:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the messaging control block

Table 214:Parameters (Continued)

Name	Direction	Description
pp_buffer	inout	Pointer to the pointer to the allocated buffer memory
p_acquire_cfg	in	Pointer to the buffer acquisition configuration
wait_option	in	Wait option (TX_NO_WAIT, TX_WAIT_FOREVER or numerical values)

Parameter p_ctrl

Definition: `sf_message_ctrl_tconst *const p_ctrl`

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

Parameter pp_buffer

Definition: `sf_message_header_t**pp_buffer`

Message header definition

- `sf_message_header_t::event`
- `sf_message_header_t::class_code`

  Event class code.
- `sf_message_header_t::class_instance`

  Event class instance number.
- `sf_message_header_t::code`

  Event code.
- `sf_message_header_t::event_b`
- `sf_message_header_t::struct{}`

Parameter p_acquire_cfg

Definition: `sf_message_acquire_cfg_t const *const p_acquire_cfg`

Messaging framework Post API function configuration structure definition

- `sf_message_acquire_cfg_t`::`buffer_keep`

  Buffer keep option.

Parameter wait_option

uint32_t

7.21.8.16 bufferRelease

```
ssp_err_t(* sf_message_api_t::bufferRelease) (sf_message_ctrl_t *const p_ctrl,
sf_message_header_t *const p_buffer, sf_message_release_option_t const option)
```

Detailed description

Release buffer obtained from SF_MESSAGE_BufferAcquire. Implemented as

- SF_MESSAGE_BufferRelease

Table 215:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the messaging control block
p_buffer	in	Pointer to the buffer allocated by SF_MESSAGE_BufferAcquire
option	in	Buffer release option (SF_MESSAGE_RELEASE_OPTION_NONE, SF_MESSAGE_RELEASE_OPTION_ACK, SF_MESSAGE_RELEASE_OPTION_NAK, SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE)

Parameter p_ctrl

Definition: `sf_message_ctrl_t*const p_ctrl`

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

Parameter p_buffer

Definition: `sf_message_header_t*const p_buffer`

Message header definition

- `sf_message_header_t::event`
- `sf_message_header_t::class_code`

  Event class code.
- `sf_message_header_t::class_instance`

  Event class instance number.

- sf_message_header_t::code

  Event code.

- sf_message_header_t::event_b

- sf_message_header_t::struct{}

Parameter option

7.21.8.17 post

```
ssp_err_t(* sf_message_api_t::post) (sf_message_ctrl_t *const p_ctrl,
sf_message_header_t const *const p_buffer, sf_message_post_cfg_t const *const
p_post_cfg, sf_message_post_err_t *const p_post_err, uint32_t const wait_option)
```

Detailed description

Post message to the subscribers. Implemented as

- SF_MESSAGE_Post

Table 216:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the messaging control block
p_buffer	in	Pointer to the buffer allocated by SF_MESSAGE_BufferAcquire
p_post_cfg	in	Pointer to the message post configuration
wait_option	in	Wait option (TX_NO_WAIT, TX_WAIT_FOREVER or numerical values)

Parameter p_ctrl

Definition: sf_message_ctrl_t*const p_ctrl

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

Parameter p_buffer

Definition: sf_message_header_tconst *const p_buffer

Message header definition

- sf_message_header_t::event

- sf_message_header_t::class_code

  Event class code.
- sf_message_header_t::class_instance

  Event class instance number.
- sf_message_header_t::code

  Event code.
- sf_message_header_t::event_b

- sf_message_header_t::struct{}

Parameter p_post_cfg

Definition: sf_message_post_cfg_t const *const p_post_cfg

Messaging framework Acquire API function configuration structure definition

- sf_message_post_cfg_t::sf_message_priority_t

  Message priority.

  Enumerated as:

  – SF_MESSAGE_PRIORITY_NORMAL

  – SF_MESSAGE_PRIORITY_HIGH
- sf_message_post_cfg_t::p_callback

  User callback function.
- sf_message_post_cfg_t::p_context

  Context provided to user during callback.

Parameter wait_option

uint32_t

7.21.8.18 pend

```
ssp_err_t(* sf_message_api_t::pend) (sf_message_ctrl_t const *const p_ctrl,
TX_QUEUE const *const p_queue, sf_message_header_t **pp_buffer, uint32_t const
wait_option)
```

Detailed description

Pend message. Implemented as

- SF_MESSAGE_Pend

Table 217:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the messaging control block
p_queue	in	Pointer to a threadX message queue object
pp_buffer	inout	Pointer to the pointer to the buffer where message is stored.
wait_option	in	Wait option (TX_NO_WAIT, TX_WAIT_FOREVER or numerical values)

Parameter p_ctrl

Definition: `sf_message_ctrl_tconst *const p_ctrl`

Message framework control block. Allocate an instance specific control block to pass into the message framework API calls. Implemented assf_message_instance_ctrl_t

Parameter p_queue

`const`

Parameter pp_buffer

Definition: `sf_message_header_t**pp_buffer`

Message header definition

- `sf_message_header_t::event`
- `sf_message_header_t::class_code`

  Event class code.
- `sf_message_header_t::class_instance`

  Event class instance number.
- `sf_message_header_t::code`

  Event code.
- `sf_message_header_t::event_b`
- `sf_message_header_t::struct{}`

Parameter wait_option

`uint32_t`

7.21.8.19 versionGet

```
ssp_err_t(* sf_message_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get the version of the messaging framework. Implemented as

- SF_MESSAGE_VersionGet

Table 218:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory where to store the version number

Parameter p_version

7.21.8.20 sf_message_instance_t

sf_message_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_message_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_message_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_message_api_t const * p_api

  Pointer to the API structure for this instance.

7.22 Power Profiles Framework Interface

Power Profiles Framework Interface.

7.22.1 Summary

This framework allows an application to place itself in one of several available Low Power configurations. The application may make API calls that will place it into a low power sleep mode from which an external interrupt, or periodic RTC interrupt may awaken it.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Framework Power Profiles Interface description: Power Profiles Framework

7.22.2 Interface API

sf_power_profiles_api_t

Function name	Description
.open	Acquires mutex, then initialize the lower layer drivers at the HAL layer
.sleep	Places the MCU in Software Standby mode.
.close	Releases channel mutex and closes channel at HAL layer.
.versionGet	Gets version and stores it in provided pointer p_version.

7.22.3 Data structures

- sf_power_profiles_callback_args_t
- sf_power_profiles_ctrl_t
- sf_power_profiles_cfg_t
- sf_power_profiles_instance_t

7.22.4 Enumerations

- sf_power_profiles_mode_t
- sf_power_profiles_event_t

7.22.5 Defines

- #define SF_POWER_PROFILES_API_VERSION_MAJOR

  Initial value: `(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file

- #define SF_POWER_PROFILES_API_VERSION_MINOR
  Initial value:(2U)

7.22.6 API Data

7.22.6.1 sf_power_profiles_mode_t

`sf_power_profiles_mode_t`

Detailed description

Options for the callback events.

Enumerated values

Name	Description
SF_POWER_PROFILES_MODE_RUN	Normal startup mode (No sleep)
SF_POWER_PROFILES_MODE_RTC	Wakeup using RTC.
SF_POWER_PROFILES_MODE_EXTERNAL	Wakeup from an external source.

7.22.6.2 sf_power_profiles_event_t

`sf_power_profiles_event_t`

Detailed description

Options for the callback events.

Enumerated values

Name	Description
SF_POWER_PROFILES_EVENT_PRE_SLEEP	Callback just before entering sleep mode.
SF_POWER_PROFILES_EVENT_POST_SLEEP	Callback just after waking up.

7.22.7 API Structures

7.22.7.1 sf_power_profiles_callback_args_t

sf_power_profiles_callback_args_t

Detailed description

Power profiles callback arguments definitions

Variables

- sf_power_profiles_event_t event

  Power profiles callback event.

- void * p_context

  Placeholder for user data.

7.22.7.2 sf_power_profiles_ctrl_t

sf_power_profiles_ctrl_t

Detailed description

Channel control block. DO NOT INITIALIZE. Initialization occurs when SF_POWER_PROFILES_Open is called

Variables

- uint32_t open

  Used by driver to check if pointer to control block is valid.

- ioport_cfg_t const * p_wake_ioport_pin_tbl

  Pointer to ioport settings for wakeup.

- ioport_cfg_t const * p_sleep_ioport_pin_tbl

  Pointer to ioport settings for sleep.

- sf_power_profiles_mode_t operating_mode

  Power profile mode to use.

- lpm_api_t const * p_api_lpm

  Pointer to lower level Low Power driver function pointers.

- rtc_api_t const * p_api_rtc

  Pointer to lower level RTC driver function pointers.

- rtc_ctrl_t * p_ctrl_rtc

  Pointer to lower level RTC driver control block.

- void(* p_callback)(sf_power_profiles_callback_args_t *p_args)

  Callback function.

- void * p_context

  Placeholder for user data.

7.22.7.3 sf_power_profiles_cfg_t

sf_power_profiles_cfg_t

Detailed description

Configuration for RTOS integrated Power Profiles driver

Variables

- ioport_cfg_t const *const p_wake_ioport_pin_tbl

  Pointer to ioport settings for wakeup.

- ioport_cfg_t const *const p_sleep_ioport_pin_tbl

  Pointer to ioport settings for sleep.

- sf_power_profiles_mode_t operating_mode

  Power profile mode to use.

- bool retain_output_signals

  (Future implementation:) True (Default) ==> address bus and bus control signals retain the output state False ==> signals are set to high-impedance state

- lpm_instance_t const *const p_lower_lvl_lpm

  Pointer to the LPM instance.

- rtc_instance_t const *const p_lower_lvl_rtc

  Pointer to the RTC instance (if any)

- void(* p_callback)(sf_power_profiles_callback_args_t *p_args)

  Callback function.

- void * p_context

  Placeholder for user data.

7.22.7.4 sf_power_profiles_api_t

sf_power_profiles_api_t

Detailed description

Framework Power Profiles API structure. Implementations will use the following API.

7.22.7.5 open

```
ssp_err_t(* sf_power_profiles_api_t::open) (sf_power_profiles_ctrl_t *const
p_ctrl, sf_power_profiles_cfg_t const *const p_cfg)
```

Detailed description

Acquires mutex, then initialize the lower layer drivers at the HAL layer Implemented as

- SF_POWER_PROFILES_Open

Table 219:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. Elements initialized here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_power_profiles_ctrl_t`

Channel control block. DO NOT INITIALIZE. Initialization occurs when SF_POWER_PROFILES_Open is called

Parameter p_cfg

Definition: `sf_power_profiles_cfg_t const *const p_cfg`

Configuration for RTOS integrated Power Profiles driver

- `sf_power_profiles_cfg_t`::`ioport_cfg_t`

  Pointer to ioport settings for wakeup.

- `sf_power_profiles_cfg_t`::`ioport_cfg_t`

  Pointer to ioport settings for sleep.

- `sf_power_profiles_cfg_t`::`sf_power_profiles_mode_t`

  Power profile mode to use.

  Enumerated as:

  – SF_POWER_PROFILES_MODE_RUN

  – SF_POWER_PROFILES_MODE_RTC

  – SF_POWER_PROFILES_MODE_EXTERNAL

- `sf_power_profiles_cfg_t`::`retain_output_signals`

  (Future implementation:) True (Default) ==> address bus and bus control signals retain the output state False ==> signals are set to high-impedance state

- `sf_power_profiles_cfg_t`::`lpm_instance_t`

  Pointer to the LPM instance.

- `sf_power_profiles_cfg_t`::`rtc_instance_t`

  Pointer to the RTC instance (if any)

- `sf_power_profiles_cfg_t`::`p_callback`

  Callback function.

- sf_power_profiles_cfg_t::p_context

  Placeholder for user data.

7.22.7.6 sleep

```
ssp_err_t(* sf_power_profiles_api_t::sleep) (sf_power_profiles_ctrl_t *const
p_ctrl)
```

Detailed description

Places the MCU in Software Standby mode. Implemented as

- SF_POWER_PROFILES_Sleep

NOTE: If the Power Profiles operating mode has been set to SF_POWER_PROFILES_MODE_RTC, then the MCU is configured to enter a low power mode that also allows the RTC and it's associated clock to run. It is up to the application to configure the RTC such that it interrupts at a specified interval.

Table 220:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in SF_POWER_PROFILES_Open.

Parameter p_ctrl

Definition: sf_power_profiles_ctrl_t

Channel control block. DO NOT INITIALIZE. Initialization occurs when SF_POWER_PROFILES_Open is called

7.22.7.7 close

```
ssp_err_t(* sf_power_profiles_api_t::close) (sf_power_profiles_ctrl_t *const
p_ctrl)
```

Detailed description

Releases channel mutex and closes channel at HAL layer. Implemented as

- SF_POWER_PROFILES_Close

Table 221:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in SF_POWER_PROFILES_Open.

Parameter p_ctrl

Definition: sf_power_profiles_ctrl_t

Channel control block. DO NOT INITIALIZE. Initialization occurs when SF_POWER_PROFILES_Open is called

7.22.7.8 versionGet

```
ssp_err_t(* sf_power_profiles_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 222:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.22.7.9 sf_power_profiles_instance_t

sf_power_profiles_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_power_profiles_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_power_profiles_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_power_profiles_api_t const * p_api

  Pointer to the API structure for this instance.

7.23 SF Socket WIFI Framework Interface

RTOS-integrated SF Socket WIFI Framework Interface.

7.23.1 Summary

This SSP Interface provides access OnChip stack BSD Socket API.

7.23.2 Functions

- socket
- close
- bind
- listen
- connect
- accept
- send
- recv
- sendto
- recvfrom
- setsockopt
- getsockopt
- select

7.23.3 Interface API

sf_socket_api_t

Function name	Description
.open	Pointer to function which initializes the network interface for data transfers Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.
.versionGet	Gets version and stores it in provided pointer p_version.

7.23.4 Data structures

- in_addr
- sockaddr
- sockaddr_in

- sf_socket_ctrl_t
- sf_socket_cfg_t
- sf_socket_instance_t

7.23.5 Typedefs

- socklen_t

7.23.6 Defines

- #define SF_SOCKET_WIFI_API_VER_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file
- #define SF_SOCKET_WIFI_API_VER_MINOR

  Initial value:`(0U)`

  Minor Version of the API defined in this file

7.23.7 socket

```
socket (  int domain ,   int type ,   int protocol )
```

7.23.7.1 Detailed description

API which creates socket

Table 223:Parameters

Name	Direction	Description
domain	in	Socket family
type	in	Socket type
protocol	in	Protocol type

7.23.8 close

```
close (  int socket_fd )
```

7.23.8.1 Detailed description

API which closes socket

Table 224:Parameters

Name	Direction	Description
socket_fd	in	Local socket

7.23.9 bind

```
bind (  int socket_fd ,  sockaddr const struct * p_local_sock_addr ,
socklen_t  addrlen )
```

7.23.9.1 Detailed description

Bind socket to interface which is identified by IP address

Table 225:Parameters

Name	Direction	Description
socket_fd	in	Local socket
p_local_sock_addr	in	Pointer to local socket address
addrlen	in	Size of sock address structure

7.23.10 listen

```
listen (  int sockfd ,   int backlog )
```

7.23.10.1 Detailed description

Listen for tcp connection. Set socket in listen mode for tcp connection.

Table 226:Parameters

Name	Direction	Description
sockfd	in	Local socket
backlog	in	Max number of connection queue.

7.23.11 connect

```
connect (  int sockfd ,  sockaddr const struct * p_serv_addr ,
socklen_t  addrlen )
```

7.23.11.1 Detailed description

Establish TCP connection with remote socket

Table 227:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_serv_addr	in	Pointer to remote socket address
addrlen	in	Size of sock address structure

7.23.12 accept

```
accept (  int sockfd ,  sockaddr struct * p_cliaddr ,  socklen_t  * p_addrlen )
```

7.23.12.1 Detailed description

Accept connection request from remote.

Table 228:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_cliaddr	out	Pointer to remote socket address which trying to connect
p_addrlen	out	Pointer to address length of client socket address

7.23.13 send

```
send (  int sockfd ,   const void * p_buf ,   size_t length ,   int flags )
```

7.23.13.1 Detailed description

Send data to remote socket.

Table 229:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	in	Pointer to data buffer
length	in	Data buffer length
flags	in	Socket flags

7.23.14 recv

```
recv (  int sockfd ,   void * p_buf ,   size_t length ,   int flags )
```

7.23.14.1 Detailed description

Receive data from remote socket.

Table 230:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	out	Pointer to data buffer where data will be received
length	in	Maximum length of data which can be received
flags	in	Socket flags

7.23.15 sendto

```
sendto (  int sockfd ,   const void * p_buf ,   size_t length ,   int flags ,
sockaddr const struct * p_dest_addr ,  socklen_t  addrlen )
```

7.23.15.1 Detailed description

Send data to remote socket.

Table 231:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	in	Pointer to data buffer to sent
length	in	Data buffer length
flags	in	Socket flag
p_dest_addr	in	Pointer to remote socket address where to send data
addrlen	in	Length of socket address structure

7.23.16 recvfrom

```
recvfrom (  int sockfd ,   void * p_buf ,   size_t length ,   int flags ,
sockaddr struct * p_src_addr ,  socklen_t  * p_addrlen )
```

7.23.16.1 Detailed description

Receive data from remote socket.

Table 232:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	out	Pointer to data buffer where data will be received
length	in	Maximum length of data which can be received
flags	in	Socket flag
p_src_addr	out	Pointer to remote socket address which has sent data

Table 232:Parameters (Continued)

Name	Direction	Description
p_addrlen	out	Length of socket address structure

7.23.17 setsockopt

```
 setsockopt (  int sockfd ,   int level ,   int optname ,   const void
* p_optval ,  socklen_t  optlen )
```

7.23.17.1 Detailed description

Set Socket options.

Table 233:Parameters

Name	Direction	Description
sockfd	in	Local socket
level	in	Sockets API level
optname	in	Option to be set
p_optval	in	Option value to be set
optlen	in	Length of option value

7.23.18 getsockopt

```
getsockopt (  int sockfd ,   int level ,   int optname ,   void * p_optval ,
socklen_t  * p_optlen )
```

7.23.18.1 Detailed description

Get Socket options.

Table 234:Parameters

Name	Direction	Description
sockfd	in	Local socket
level	in	Sockets API level

Table 234:Parameters (Continued)

Name	Direction	Description
optname	in	Option to be get
p_optval	out	Option value to be get
p_optlen	in	Length of option value

7.23.19 select

```
 select (  int nfds ,  fd_set  * p_readfds ,  fd_set  * p_writefds ,  fd_set
* p_exceptfds ,  timeval struct * p_timeout )
```

7.23.19.1 Detailed description

Wait on a given socket for specified amount of time. In case of any activity e.g. arrival of packet it comes out of wait.

Table 235:Parameters

Name	Direction	Description
nfds	in	Max fd
p_readfds	in	Pointer to fd_set to check whether data is available for read
p_writefds	in	Pointer to fd_set to check whether data is available for write
p_exceptfds	in	Pointer to fd_set to check whether exceptional condition occurred
p_timeout	in	Wait time in milliseconds

7.23.20 API Data

7.23.20.1 socklen_t

```
typedef int32_t socklen_t
```

Detailed description

Socket Structure Length

7.23.21 API Structures

7.23.21.1 in_addr

in_addr

Detailed description

IP address used by sockaddr

Socket Internet Address structure

Variables

- unsigned long s_addr

  load with inet_aton()

  Load with inet_aton()
- ULONG s_addr

7.23.21.2 sockaddr

sockaddr

Detailed description

Socket Address information

Socket Internet Address structure with port and family

Variables

- short sin_family

  Address family.
- unsigned short sin_port

  Port number.
- in_addrstruct sin_addr

  IP Address.
- char sin_zero[8]

  zero this if you want to

  Zero this if you want to.
- USHORT sa_family
- UCHAR sa_data[14]

7.23.21.3 sockaddr_in

sockaddr_in

Detailed description

Socket address, Internet style.

Variables

- uint16_t sin_family

  Internet Protocol (AF_INET)

- uint16_t sin_port

  Address port (16 bits)

- in_addrstruct sin_addr

  Internet address (32 bits)

- int8_t sin_zero[8]

  Not used.

  Not used structure member.

- USHORT sin_family
- USHORT sin_port
- CHAR sin_zero[8]

7.23.21.4 sf_socket_ctrl_t

sf_socket_ctrl_t

Detailed description

Socket Interface control structure

Variables

- sf_wifi_onchip_stack_instance_t * p_lower_lvl_onchip_wifi

  low level wifi interface

7.23.21.5 sf_socket_cfg_t

sf_socket_cfg_t

Detailed description

Socket Interface configuration structure

Variables

- sf_wifi_onchip_stack_instance_t * p_lower_lvl_onchip_wifi

  Pointer to SF on-chip stack instance.

- void * p_extend

  Extended configuration.

7.23.21.6 sf_socket_api_t

sf_socket_api_t

Detailed description

Socket Interface API

7.23.21.7 open

```
(* sf_socket_api_t::open) (sf_socket_ctrl_t *p_ctrl, sf_socket_cfg_t const *const p_cfg)
```

Brief description

Pointer to function which initializes the network interface for data transfers Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Detailed description

Table 236:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to configuration structure.

Parameter p_ctrl

Definition: `sf_socket_ctrl_t`

Socket Interface control structure

Parameter p_cfg

Definition: `sf_socket_cfg_t const *const p_cfg`

Socket Interface configuration structure

- `sf_socket_cfg_t`::`sf_wifi_onchip_stack_instance_t`

  Pointer to SF on-chip stack instance.

- `sf_socket_cfg_t`::`p_extend`

  Extended configuration.

7.23.21.8 close

```
(* sf_socket_api_t::close) (sf_socket_ctrl_t *const p_ctrl)
```

Brief description

Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Detailed description

Table 237:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the control block

Parameter p_ctrl

Definition: `sf_socket_ctrl_t`

Socket Interface control structure

7.23.21.9 versionGet

```
(* sf_socket_api_t::versionGet) ( *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 238:Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

7.23.21.10 sf_socket_instance_t

sf_socket_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_socket_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_socket_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_socket_api_t const * p_api

  Pointer to the API structure for this instance.

7.24 SF Socket CELLULAR Framework Interface

RTOS-integrated SF Socket Cellular Framework Interface.

7.24.1 Summary

This SSP Interface provides access OnChip stack BSD Socket API.

7.24.2 Functions

- socket
- close
- bind
- listen
- connect
- accept
- send
- recv
- sendto
- recvfrom
- setsockopt
- getsockopt
- select

7.24.3 Interface API

sf_cellular_socket_api_t

Function name	Description
.open	Pointer to function which initializes the network interface for data transfers Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Function name	Description
.versionGet	Gets version and stores it in provided pointer p_version.

7.24.4 Data structures

- in_addr
- sockaddr
- sockaddr_in
- sf_cellular_socket_ctrl_t
- sf_cellular_socket_cfg_t
- sf_cellular_socket_instance_t

7.24.5 Typedefs

- socklen_t

7.24.6 Defines

- #define SF_CELLULAR_SOCKET_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. SF Cellular Socket APIs Major Version
- #define SF_CELLULAR_SOCKET_API_VERSION_MINOR

  Initial value:`(0U)`

  SF Cellular Socket APIs Minor Version

7.24.7 socket

```
socket (  int domain ,   int type ,   int protocol )
```

7.24.7.1 Detailed description

API which creates socket

Table 239:Parameters

Name	Direction	Description
domain	in	Socket family
type	in	Socket type
protocol	in	Protocol type

7.24.8 close

```
close (  int socket_fd )
```

7.24.8.1 Detailed description

API which closes socket

Table 240:Parameters

Name	Direction	Description
socket_fd	in	Local socket

7.24.9 bind

```
bind (  int socket_fd ,  sockaddr const struct * p_local_sock_addr ,
socklen_t  addrlen )
```

7.24.9.1 Detailed description

Bind socket to interface which is identified by IP address

Table 241:Parameters

Name	Direction	Description
socket_fd	in	Local socket
p_local_sock_addr	in	Pointer to local socket address
addrlen	in	Size of sock address structure

7.24.10 listen

```
listen (  int sockfd ,   int backlog )
```

7.24.10.1 Detailed description

Listen for tcp connection. Set socket in listen mode for tcp connection.

Table 242:Parameters

Name	Direction	Description
sockfd	in	Local socket
backlog	in	Max number of connection queue.

7.24.11 connect

```
connect (  int sockfd ,  sockaddr const struct * p_serv_addr ,
socklen_t  addrlen )
```

7.24.11.1 Detailed description

Establish TCP connection with remote socket

Table 243:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_serv_addr	in	Pointer to remote socket address
addrlen	in	Size of sock address structure

7.24.12 accept

```
accept (  int sockfd ,  sockaddr struct * p_cliaddr ,  socklen_t  * p_addrlen )
```

7.24.12.1 Detailed description

Accept connection request from remote.

Table 244:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_cliaddr	out	Pointer to remote socket address which trying to connect
p_addrlen	out	Pointer to address length of client socket address

7.24.13 send

```
send (  int sockfd ,   const void * p_buf ,   size_t length ,   int flags )
```

7.24.13.1 Detailed description

Send data to remote socket.

Table 245:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	in	Pointer to data buffer
length	in	Data buffer length
flags	in	Socket flags

7.24.14 recv

```
recv (  int sockfd ,   void * p_buf ,   size_t length ,   int flags )
```

7.24.14.1 Detailed description

Receive data from remote socket.

Table 246:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	out	Pointer to data buffer where data will be received
length	in	Maximum length of data which can be received
flags	in	Socket flags

7.24.15 sendto

```
sendto (  int sockfd ,   const void * p_buf ,   size_t length ,   int flags ,
sockaddr const struct * p_dest_addr ,  socklen_t  addrlen )
```

7.24.15.1 Detailed description

Send data to remote socket.

Table 247:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	in	Pointer to data buffer to sent
length	in	Data buffer length
flags	in	Socket flag
p_dest_addr	in	Pointer to remote socket address where to send data
addrlen	in	Length of socket address structure

7.24.16 recvfrom

```
recvfrom (  int sockfd ,   void * p_buf ,   size_t length ,   int flags ,
sockaddr struct * p_src_addr ,  socklen_t  * p_addrlen )
```

7.24.16.1 Detailed description

Receive data from remote socket.

Table 248:Parameters

Name	Direction	Description
sockfd	in	Local socket
p_buf	out	Pointer to data buffer where data will be received
length	in	Maximum length of data which can be received
flags	in	Socket flag
p_src_addr	out	Pointer to remote socket address which has sent data
p_addrlen	out	Length of socket address structure

7.24.17 setsockopt

```
 setsockopt (  int sockfd ,   int level ,   int optname ,   const void
* p_optval ,  socklen_t  optlen )
```

7.24.17.1 Detailed description

Set Socket options.

Table 249:Parameters

Name	Direction	Description
sockfd	in	Local socket
level	in	Sockets API level
optname	in	Option to be set
p_optval	in	Option value to be set
optlen	in	Length of option value

7.24.18 getsockopt

```
getsockopt (  int sockfd ,   int level ,   int optname ,   void * p_optval ,
socklen_t  * p_optlen )
```

7.24.18.1 Detailed description

Get Socket options.

Table 250:Parameters

Name	Direction	Description
sockfd	in	Local socket
level	in	Sockets API level
optname	in	Option to be get
p_optval	out	Option value to be get
p_optlen	in	Length of option value

7.24.19 select

```
 select (  int nfds ,  fd_set  * p_readfds ,  fd_set  * p_writefds ,  fd_set
* p_exceptfds ,  timeval struct * p_timeout )
```

7.24.19.1 Detailed description

Wait on a given socket for specified amount of time. In case of any activity e.g. arrival of packet it comes out of wait.

Table 251:Parameters

Name	Direction	Description
nfds	in	Max fd
p_readfds	in	Pointer to fd_set to check whether data is available for read
p_writefds	in	Pointer to fd_set to check whether data is available for write
p_exceptfds	in	Pointer to fd_set to check whether exceptional condition occurred

Table 251:Parameters (Continued)

Name	Direction	Description
p_timeout	in	Wait time in milliseconds

7.24.20 API Data

7.24.20.1 socklen_t

```
typedef int32_t socklen_t
```

Detailed description

Socket address Length

7.24.21 API Structures

7.24.21.1 in_addr

in_addr

Detailed description

IP address used by sockaddr

Socket Internet Address structure

Variables

- unsigned long s_addr

  load with inet_aton()

  Load with inet_aton()
- ULONG s_addr

7.24.21.2 sockaddr

sockaddr

Detailed description

Socket Address information

Socket Internet Address structure with port and family

Variables

- short sin_family

  Address family.

- unsigned short sin_port

  Port number.
- in_addrstruct sin_addr

  IP Address.
- char sin_zero[8]

  zero this if you want to

  Zero this if you want to.
- USHORT sa_family
- UCHAR sa_data[14]

7.24.21.3 sockaddr_in

sockaddr_in

Detailed description

Socket address, Internet style.

Variables

- uint16_t sin_family

  Internet Protocol (AF_INET)
- uint16_t sin_port

  Address port (16 bits)
- in_addrstruct sin_addr

  Internet address (32 bits)
- int8_t sin_zero[8]

  Not used.

  Not used structure member.
- USHORT sin_family
- USHORT sin_port
- CHAR sin_zero[8]

7.24.21.4 sf_cellular_socket_ctrl_t

sf_cellular_socket_ctrl_t

Detailed description

Socket Interface control structure

Variables

- sf_cellular_instance_t * p_lower_lvl_cellular

  low level cellular interface

7.24.21.5 sf_cellular_socket_cfg_t

sf_cellular_socket_cfg_t

Detailed description

Socket Interface configuration structure

Variables

- sf_cellular_instance_t * p_lower_lvl_cellular

  Pointer to SF on-chip stack instance.

- void * p_extend

  Extended configuration.

7.24.21.6 sf_cellular_socket_api_t

sf_cellular_socket_api_t

Detailed description

Socket Interface API

7.24.21.7 open

```
(* sf_cellular_socket_api_t::open) (sf_cellular_socket_ctrl_t *p_ctrl,
sf_cellular_socket_cfg_t const *const p_cfg)
```

Brief description

Pointer to function which initializes the network interface for data transfers Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Detailed description

Table 252:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the control block for the Cellular module Socket interface.
p_cfg	in	Pointer to Cellular Socket interface configuration structure.

Parameter p_ctrl

Definition: sf_cellular_socket_ctrl_t

Socket Interface control structure

Parameter p_cfg

Definition: `sf_cellular_socket_cfg_t const *const p_cfg`

Socket Interface configuration structure

- `sf_cellular_socket_cfg_t`::`sf_cellular_instance_t`

  Pointer to SF on-chip stack instance.

- `sf_cellular_socket_cfg_t`::`p_extend`

  Extended configuration.

7.24.21.8 close

```
(* sf_cellular_socket_api_t::close) (sf_cellular_socket_ctrl_t *const p_ctrl)
```

Brief description

Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Detailed description

Table 253:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the control block for the Cellular module Socket interface.

Parameter p_ctrl

Definition: `sf_cellular_socket_ctrl_t`

Socket Interface control structure

7.24.21.9 versionGet

```
(* sf_cellular_socket_api_t::versionGet) ( *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 254:Parameters

Name	Direction	Description
p_version	out	Pointer to SSP Version structure

Parameter p_version

7.24.21.10 sf_cellular_socket_instance_t

sf_cellular_socket_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_cellular_socket_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_cellular_socket_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_cellular_socket_api_t const * p_api

  Pointer to the API structure for this instance.

7.25 SPI Framework Interface

RTOS-integrated SPI Framework Interface.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

7.25.1 Summary

This SSP Interface provides access to the ThreadX-aware SPI Framework. The Interface is implemented by the SPI Framework.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

SPI Framework Interface description: SPI Framework

7.25.2 Interface API

sf_spi_api_t

Function name	Description
.open	Open a designated SPI device on a bus.

Function name	Description
.read	Receive data from SPI device.
.write	Transmit data to SPI device.
.writeRead	Simultaneously transmit data to an SPI device while receiving data from an SPI device (full duplex). The writeread API gets mutex object, handles SPI data transmission at SPI HAL layer and receive data from the SPI HAL layer. The API uses the event flag wait to synchronize to completion of data transfer .
.close	Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.
.lock	Lock the bus for a device. The locking allows devices to reserve a bus to themselves for a given period of time (i.e. between lock and unlock). This allows devices to complete several reads and writes on the bus without interrupt.
.unlock	Unlock the bus for a particular device and make the bus usable for other devices.
.version	Get the version information of the underlying driver.

7.25.3 Data structures

- sf_spi_bus_t
- sf_spi_cfg_t
- sf_spi_instance_t

7.25.4 Enumerations

- sf_spi_dev_state_t

7.25.5 Typedefs

- sf_spi_ctrl_t

7.25.6 Defines

- #define SF_SPI_API_VERSION_MAJOR

  Initial value:`(1U)`
- #define SF_SPI_API_VERSION_MINOR

  Initial value:`(3U)`

7.25.7 API Data

7.25.7.1 sf_spi_dev_state_t

`sf_spi_dev_state_t`

Detailed description

SF SPI device state

Enumerated values

Name	Description
SF_SPI_DEV_STATE_CLOSED	SPI device is closed.
SF_SPI_DEV_STATE_OPENED	SPI device is opened.

7.25.7.2 sf_spi_ctrl_t

```
typedef void sf_spi_ctrl_t
```

Detailed description

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented as

- sf_spi_instance_ctrl_t

7.25.8 API Structures

7.25.8.1 sf_spi_bus_t

sf_spi_bus_t

Detailed description

Data structure defining an SPI bus.

Variables

- uint8_t channel

  Channel.
- uint32_t freq_hz_min

  Bus min frequency supported.
- TX_MUTEX * p_lock_mutex

  Lock mutex handle for this channel.
- TX_EVENT_FLAGS_GROUP * p_sync_eventflag

  Pointer to the event flag object for SPI data transfer.
- sf_spi_ctrl_t ** pp_curr_ctrl

  Current device using the bus.
- uint8_t * p_bus_name

  peripheral name SCI_SPI/RSPI
- spi_api_t const * p_lower_lvl_api

  Pointer to SPI HAL interface to be used in the framework.
- uint8_t device_count

  Number of devices on the bus, initialize to 0.

7.25.8.2 sf_spi_cfg_t

sf_spi_cfg_t

Detailed description

Configuration for Framework SPI driver.

Variables

- sf_spi_bus_t * p_bus

  Bus used by the device.
- ioport_port_pin_t chip_select

  Chip select for this device.
- ioport_level_t chip_select_level_active

  Polarity of CS, active High or Low.
- spi_cfg_t const * p_lower_lvl_cfg

  Pointer to SPI HAL configuration.

7.25.8.3 sf_spi_api_t

sf_spi_api_t

Detailed description

Definition of the SPI framework interface.

7.25.8.4 open

```
ssp_err_t(* sf_spi_api_t::open) (sf_spi_ctrl_t *p_ctrl, sf_spi_cfg_t const *const
p_cfg)
```

Brief description

Open a designated SPI device on a bus.

Detailed description

Table 255:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to SPI Framework configuration structure.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

Parameter p_cfg

Definition: `sf_spi_cfg_t const *const p_cfg`

Configuration for Framework SPI driver.

- `sf_spi_cfg_t`::`sf_spi_bus_t`

  Bus used by the device.

- `sf_spi_cfg_t`::`ioport_port_pin_t`

  Chip select for this device.

- `sf_spi_cfg_t`::`ioport_level_t`

  Polarity of CS, active High or Low.

- `sf_spi_cfg_t`::`spi_cfg_t`

  Pointer to SPI HAL configuration.

7.25.8.5 read

```
ssp_err_t(* sf_spi_api_t::read) (sf_spi_ctrl_t *const p_ctrl, void *const p_dest,
uint32_t const length, spi_bit_width_t const bit_width, uint32_t const timeout)
```

Brief description

Receive data from SPI device.

Detailed description

NOTE: Call open to configure the SPI device before using this function.

Table 256:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the device.
p_dest	out	Pointer to destination buffer into which data will be copied that is received from a SPI device. It is the responsibility of the caller to ensure that adequate space is available to hold the requested data count.
length	in	Indicates the number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Indicates data bit width to be transferred.
timeout	in	Timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

Parameter p_dest

`const`

Parameter length

`uint32_t`

Parameter bit_width

Parameter timeout

`uint32_t`

7.25.8.6 write

```
ssp_err_t(* sf_spi_api_t::write) (sf_spi_ctrl_t *const p_ctrl, void *const p_src,
uint32_t const length, spi_bit_width_t const bit_width, uint32_t const timeout)
```

Brief description

Transmit data to SPI device.

Detailed description

NOTE: Call open to configure the SPI device before using this function.

Table 257:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the device.
p_src	in	Pointer to a source data buffer from which data will be transmitted to a SPI device. The argument must not be NULL.
length	in	Indicates the number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Indicates data bit width to be transferred.
timeout	in	Timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

Parameter p_src

`const`

Parameter length

`uint32_t`

Parameter bit_width

Parameter timeout

`uint32_t`

7.25.8.7 writeRead

```
ssp_err_t(* sf_spi_api_t::writeRead) (sf_spi_ctrl_t *const p_ctrl, void *const
p_src, void *const p_dest, uint32_t const length, spi_bit_width_t const bit_width,
uint32_t const timeout)
```

Brief description

Simultaneously transmit data to an SPI device while receiving data from an SPI device (full duplex).

Detailed description

The writeread API gets mutex object, handles SPI data transmission at SPI HAL layer and receive data from the SPI HAL layer. The API uses the event flag wait to synchronize to completion of data transfer .

NOTE: Call open to configure the SPI before using this function.

Table 258:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the channel.
p_src	in	Pointer to a source data buffer from which data will be transmitted to a SPI device. The argument must not be NULL.
p_dest	out	Pointer to destination buffer into which data will be copied that is received from a SPI device. It is the responsibility of the caller to ensure that adequate space is available to hold the requested data count.
length	in	Indicates the number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Indicates data bit width to be transferred.

Table 258:Parameters (Continued)

Name	Direction	Description
timeout	in	Timeout. Options include TX_NO_WAIT (0x00000000), TX_WAIT_FOREVER (0xFFFFFFFF), and timeout value (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

Parameter p_src

`const`

Parameter p_dest

`const`

Parameter length

`uint32_t`

Parameter bit_width

Parameter timeout

`uint32_t`

7.25.8.8 close

```
ssp_err_t(* sf_spi_api_t::close) (sf_spi_ctrl_t *const p_ctrl)
```

Brief description

Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.

Detailed description

Table 259:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the device.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

7.25.8.9 lock

```
ssp_err_t(* sf_spi_api_t::lock) (sf_spi_ctrl_t *const p_ctrl)
```

Brief description

Lock the bus for a device. The locking allows devices to reserve a bus to themselves for a given period of time (i.e. between lock and unlock). This allows devices to complete several reads and writes on the bus without interrupt.

Detailed description

Table 260:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the device.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

7.25.8.10 unlock

```
ssp_err_t(* sf_spi_api_t::unlock) (sf_spi_ctrl_t *const p_ctrl)
```

Brief description

Unlock the bus for a particular device and make the bus usable for other devices.

Detailed description

Table 261:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the device.

Parameter p_ctrl

Definition: `sf_spi_ctrl_t*const p_ctrl`

SPI framework control block. Allocate an instance specific control block to pass into the SPI framework API calls. Implemented assf_spi_instance_ctrl_t

7.25.8.11 version

```
ssp_err_t(* sf_spi_api_t::version) (ssp_version_t *const p_version)
```

Brief description

Get the version information of the underlying driver.

Detailed description

Table 262:Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

7.25.8.12 sf_spi_instance_t

sf_spi_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_spi_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_spi_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_spi_api_t const * p_api

  Pointer to the API structure for this instance.

7.26 Thread Monitor Framework Interface

RTOS-integrated Framework Interface for monitoring of threads.

Any misbehaving threads cause a reset of the device. Both the WDT and IWDT HAL modules are supported by this framework module.

7.26.1 Summary

This is a ThreadX aware Watchdog Timer Thread Monitor Framework for monitoring threads in an application in which threads are executing at an expected rate. Threads to be monitored register themselves through SF_THREAD_MONITOR_ThreadRegister and increment a count by calling

SF_THREAD_MONITOR_CountIncrement each time they execute. Each monitored thread also provides expected maximum and minimum count values for normal execution.

The Thread Monitor runs periodically and checks the count value of each monitored thread. If the count value falls outside of the expected range of values, the Watchdog Timer is allowed to reset the device. If all thread counts are within their expected ranges, then the Watchdog Timer is refreshed.

The WDT and IWDT modules are supported by the Thread Monitor.

The Framework Layer can be used to protect the entire software project. This is achieved through a high priority thread (Framework Layer) which runs periodically within the refresh permitted window of the Watchdog Timer selected (IWDT is safest as has its own clock source and is started automatically after reset). This thread monitors the state of every other thread in the system. If any of these threads are not running as expected, then the Watchdog Timer is not refreshed and is not allowed to reset the system. If the threads are running as expected, then the Watchdog Timer is refreshed.

Monitoring the other threads is achieved as follows: Each monitored thread increments a count variable each time it runs. The Thread Monitor thread checks the count variable of each thread to make sure it is within an expected range. If any of the variables are out of range a reset is allowed. Otherwise all variables are cleared to zero and the watchdog is refreshed. A profiling mode is used to establish the expected ranges.

This approach is described in the following article:

Jack Ganssle, "Great Watchdog Timers for Embedded Systems," www.ganssle.com/watchdogs.htm

This method requires the instrumenting of each thread to increment its count variable, but this is little overhead for the massive gain in protection.

Interface used: WDT Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Thread Monitor Interface description: Thread Monitor Framework

7.26.2 Interface API

sf_thread_monitor_api_t

Function name	Description
.open	Configures the WDT or IWDT module. From the configuration data the timeout period of the WDT/IWDT is determined and a thread created monitoring registered threads.
.close	Suspends the thread monitoring thread. Watchdog peripheral no longer refreshed.
.threadRegister	Registers a thread for monitoring.
.threadUnregister	Removes a thread from being monitored.

Function name	Description
.countIncrement	Safely increments a monitored thread's count value.
.versionGet	Get version and store it in provided pointer p_version.

7.26.3 Data structures

- sf_thread_monitor_cfg_t
- sf_thread_monitor_thread_counter_t
- sf_thread_monitor_counter_min_max_t
- sf_thread_monitor_instance_t

7.26.4 Typedefs

- sf_thread_monitor_ctrl_t

7.26.5 Defines

- #define THREAD_MONITOR_THREAD_STACK_SIZE

  Initial value:`400u`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_THREAD_MONITOR_API_VERSION_MAJOR

  Initial value:`(1U)`

  Version of the API defined in this file
- #define SF_THREAD_MONITOR_API_VERSION_MINOR

  Initial value:`(3U)`

7.26.6 API Data

7.26.6.1 sf_thread_monitor_ctrl_t

```
typedef void sf_thread_monitor_ctrl_t
```

Detailed description

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented as

- sf_thread_monitor_instance_ctrl_t

7.26.7 API Structures

7.26.7.1 sf_thread_monitor_cfg_t

sf_thread_monitor_cfg_t

Detailed description

Configuration for RTOS Thread Monitor driver

Variables

- wdt_instance_t const * p_lower_lvl_wdt

  Pointer to lower level watchdog instance.

- bool profiling_mode_enabled

  Enables or disables profiling mode.

- UINT priority

  Priority of thread monitor thread.

7.26.7.2 sf_thread_monitor_thread_counter_t

sf_thread_monitor_thread_counter_t

Detailed description

Counter block for each monitored thread.

Variables

- uint32_t current_count

  Current count value for a thread.

- uint32_t minimum_count

  Minimum expected count value. If the current count is less than this value the watchdog will reset.

- uint32_t maximum_count

  Maximum expected count value. If the current count is more than this value the watchdog will reset

- bool active

  Indicates to the monitoring thread whether this count data is currently active. When a thread is registered this value will be set to false as the count is likely to be a partial count and so should not be monitored. This value will be set to true by the thread monitor thread when it clears all counts to zero.

- TX_THREAD * p_thread

  Pointer to thread for this counter data.

7.26.7.3 sf_thread_monitor_counter_min_max_t

sf_thread_monitor_counter_min_max_t

Detailed description

Counter block for each monitored thread.

Variables

- uint32_t minimum_count

  Minimum expected count value. If the current count is less than this value the watchdog will reset.

- uint32_t maximum_count

  Maximum expected count value. If the current count is more than this value the watchdog will reset

7.26.7.4 sf_thread_monitor_api_t

sf_thread_monitor_api_t

Detailed description

Thread monitor API structure. Thread Monitor implementations use the following API.

7.26.7.5 open

```
(* sf_thread_monitor_api_t::open) (sf_thread_monitor_ctrl_t *const p_ctrl,
sf_thread_monitor_cfg_t const *const p_cfg)
```

Brief description

Configures the WDT or IWDT module. From the configuration data the timeout period of the WDT/IWDT is determined and a thread created monitoring registered threads.

Detailed description

Implemented as

- SF_THREAD_MONITOR_Open

Table 263:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. Elements initialized here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_thread_monitor_ctrl_t*const p_ctrl`

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented assf_thread_monitor_instance_ctrl_t

Parameter p_cfg

Definition: `sf_thread_monitor_cfg_t const *const p_cfg`

Configuration for RTOS Thread Monitor driver

- `sf_thread_monitor_cfg_t`::`wdt_instance_t`

  Pointer to lower level watchdog instance.

- `sf_thread_monitor_cfg_t`::`profiling_mode_enabled`

  Enables or disables profiling mode.

- `sf_thread_monitor_cfg_t`::`priority`

  Priority of thread monitor thread.

7.26.7.6 close

```
(* sf_thread_monitor_api_t::close) (sf_thread_monitor_ctrl_t *const p_ctrl)
```

Brief description

Suspends the thread monitoring thread. Watchdog peripheral no longer refreshed.

Detailed description

Implemented as

- SF_THREAD_MONITOR_Close

Table 264:Parameters

Name	Direction	Description
p_ctrl	inout	Control structure set in SF_THREAD_MONITOR_Open.

Parameter p_ctrl

Definition: `sf_thread_monitor_ctrl_t*const p_ctrl`

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented assf_thread_monitor_instance_ctrl_t

7.26.7.7 threadRegister

```
(* sf_thread_monitor_api_t::threadRegister) (sf_thread_monitor_ctrl_t *const
p_ctrl, sf_thread_monitor_counter_min_max_t const *p_counter_min_max)
```

Brief description

Registers a thread for monitoring.

Detailed description

Implemented as

- SF_THREAD_MONITOR_ThreadRegister

Table 265:Parameters

Name	Direction	Description
p_ctrl	inout	Control structure set in SF_THREAD_MONITOR_Open.
p_counter_min_max	in	Pointer to structure containing min and max values for thread to be registered values.

Parameter p_ctrl

Definition: `sf_thread_monitor_ctrl_t*const p_ctrl`

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented assf_thread_monitor_instance_ctrl_t

Parameter p_counter_min_max

Definition: `sf_thread_monitor_counter_min_max_tconst *p_counter_min_max`

Counter block for each monitored thread.

- `sf_thread_monitor_counter_min_max_t::minimum_count`

  Minimum expected count value. If the current count is less than this value the watchdog will reset.

- `sf_thread_monitor_counter_min_max_t::maximum_count`

  Maximum expected count value. If the current count is more than this value the watchdog will reset

7.26.7.8 threadUnregister

```
(* sf_thread_monitor_api_t::threadUnregister) (sf_thread_monitor_ctrl_t *const
p_ctrl)
```

Brief description

Removes a thread from being monitored.

Detailed description

Implemented as

- SF_THREAD_MONITOR_ThreadUnregister

Table 266:Parameters

Name	Direction	Description
p_ctrl	inout	Control structure set in SF_THREAD_MONITOR_Open.

Parameter p_ctrl

Definition: `sf_thread_monitor_ctrl_t*const p_ctrl`

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented assf_thread_monitor_instance_ctrl_t

7.26.7.9 countIncrement

```
(* sf_thread_monitor_api_t::countIncrement) (sf_thread_monitor_ctrl_t *const p_ctrl)
```

Brief description

Safely increments a monitored thread's count value.

Detailed description

Implemented as

- SF_THREAD_MONITOR_CountIncrement

Table 267:Parameters

Name	Direction	Description
p_ctrl	inout	Control structure set in SF_THREAD_MONITOR_Open.

Parameter p_ctrl

Definition: `sf_thread_monitor_ctrl_t*const p_ctrl`

Thread monitor control block. Allocate an instance specific control block to pass into the thread monitor API calls. Implemented assf_thread_monitor_instance_ctrl_t

7.26.7.10 versionGet

```
(* sf_thread_monitor_api_t::versionGet) ( *const p_version)
```

Brief description

Get version and store it in provided pointer p_version.

Detailed description

Implemented as

- SF_THREAD_MONITOR_VersionGet

Table 268:Parameters

Name	Direction	Description
p_version	inout	Pointer to structure storing API and code versions.

Parameter p_version

7.26.7.11 sf_thread_monitor_instance_t

sf_thread_monitor_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_thread_monitor_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_thread_monitor_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_thread_monitor_api_t const * p_api

  Pointer to the API structure for this instance.

7.27 CTSU Framework Interface

RTOS-integrated CTSU Framework Interface.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

7.27.1 Summary

This is a ThreadX-aware CTSU interface which is used to drive the CTSU HAL driver. It can be used to run the CTSU hardware, and read back the results of the scans.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CTSU Framework Interface description: Capacitive Touch Framework

7.27.2 Interface API

sf_touch_ctsu_api_t

Function name	Description
.open	Initialize the Touch CTSU Framework; configures the lower level hardware also and run scans at the configured Update Frequency.
.read	Read the results from the CTSU including raw data, binary data and other derived data according to the selected options.
.close	Close the Framework by ending any scans in progress and destroying internal threads.
.versionGet	Get version and store it in provided pointer p_version.

7.27.3 Data structures

- sf_touch_ctsu_callback_args_t
- sf_touch_ctsu_cfg_t
- sf_touch_ctsu_instance_t

7.27.4 Typedefs

- sf_touch_ctsu_ctrl_t

7.27.5 Defines

- #define SF_TOUCH_CTSU_VERSION_MAJOR
  Initial value:`(1U)`
- #define SF_TOUCH_CTSU_VERSION_MINOR
  Initial value:`(1U)`

7.27.6 API Data

7.27.6.1 sf_touch_ctsu_ctrl_t

```
typedef void sf_touch_ctsu_ctrl_t
```

Detailed description

Cap touch framework control block. Allocate an instance specific control block to pass into the cap touch framework API calls. Implemented as

- sf_touch_ctsu_instance_ctrl_t

7.27.7 API Structures

7.27.7.1 sf_touch_ctsu_callback_args_t

sf_touch_ctsu_callback_args_t

Detailed description

Callback argument structure

Variables

- void * p_context

7.27.7.2 sf_touch_ctsu_cfg_t

sf_touch_ctsu_cfg_t

Detailed description

Configuration for RTOS integrated CTSU Touch framework.

Variables

- UINT priority

  Priority of the touch panel thread.
- uint16_t update_hz

  The frequency to run the scans. This is set by the latest open() call.
- void(* p_callback)(sf_touch_ctsu_callback_args_t *p_args)

  Callback function;.
- void * p_context

  Arguments to the callback.
- ctsu_instance_t * p_ctsu_instance

  CTSU instance.

7.27.7.3 sf_touch_ctsu_api_t

sf_touch_ctsu_api_t

Detailed description

CTSU Framework API structure. Implementations will use the following API.

7.27.7.4 open

```
(* sf_touch_ctsu_api_t::open) (sf_touch_ctsu_ctrl_t *const p_ctrl,
sf_touch_ctsu_cfg_t const *const p_cfg)
```

Detailed description

Initialize the Touch CTSU Framework; configures the lower level hardware also and run scans at the configured Update Frequency. Implemented as

- SF_TOUCH_CTSU_Open

Table 269:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_ctrl_t*const p_ctrl`

Cap touch framework control block. Allocate an instance specific control block to pass into the cap touch framework API calls. Implemented assf_touch_ctsu_instance_ctrl_t

Parameter p_cfg

Definition: `sf_touch_ctsu_cfg_t const *const p_cfg`

Configuration for RTOS integrated CTSU Touch framework.

- `sf_touch_ctsu_cfg_t`::`priority`

  Priority of the touch panel thread.

- `sf_touch_ctsu_cfg_t`::`update_hz`

  The frequency to run the scans. This is set by the latest open() call.

- `sf_touch_ctsu_cfg_t`::`p_callback`

  Callback function;.

- `sf_touch_ctsu_cfg_t`::`p_context`

  Arguments to the callback.

- `sf_touch_ctsu_cfg_t`::`ctsu_instance_t`

  CTSU instance.

7.27.7.5 read

```
(* sf_touch_ctsu_api_t::read) (sf_touch_ctsu_ctrl_t *const p_ctrl, void *p_dest,
opts, const *channels, const uint16_t count)
```

Detailed description

Read the results from the CTSU including raw data, binary data and other derived data according to the selected options. Implemented as

- SF_TOUCH_CTSU_Read

Table 270:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_dest	out	Pointer to the destination location for the read data
opts	in	Read options
channels	in	Specify the channel/channel pairs to read data for
channels	in	Specify the number of channel/channel pairs to read data for

Parameter p_ctrl

Definition: `sf_touch_ctsu_ctrl_t*const p_ctrl`

Cap touch framework control block. Allocate an instance specific control block to pass into the cap touch framework API calls. Implemented assf_touch_ctsu_instance_ctrl_t

Parameter p_dest

`const`

Parameter opts

Parameter channels

Parameter channels

7.27.7.6 close

```
(* sf_touch_ctsu_api_t::close) (sf_touch_ctsu_ctrl_t *const p_ctrl)
```

Detailed description

Close the Framework by ending any scans in progress and destroying internal threads. Implemented as

- SF_TOUCH_CTSU_Close

Table 271:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_ctrl_t*const p_ctrl`

Cap touch framework control block. Allocate an instance specific control block to pass into the cap touch framework API calls. Implemented assf_touch_ctsu_instance_ctrl_t

7.27.7.7 versionGet

```
(* sf_touch_ctsu_api_t::versionGet) ( *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- SF_TOUCH_CTSU_VersionGet

Table 272:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.27.7.8 sf_touch_ctsu_instance_t

sf_touch_ctsu_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_touch_ctsu_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_touch_ctsu_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_touch_ctsu_api_t const * p_api

  Pointer to the API structure for this instance.

7.28 CTSU Button Framework Interface

RTOS-integrated CTSU Button Framework Interface.

This framework layer uses the CTSU Framework Layer to implement a button interface. Using this Button Framework the user can configure and use multiple buttons using the configuration structure generated from WorkBench. An action on each button (press, release etc) will result in a callback with an argument indicating the button id and event type.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CTSU Button Framework Interface description: Capacitive Touch Button Framework

7.28.1 Interface API

sf_touch_ctsu_button_api_t

Function name	Description
.open	Initialize the Touch CTSU Button Framework; configures the lower level hardware and registers callback functions for all the buttons.
.enable	Enable callback notification for a configured button.
.disable	Disable callback notification for a configured button.
.close	Close the Touch CTSU Button Framework; Closes the button framework and lower layers if no other modules are using it.
.versionGet	Get version and store it in provided pointer p_version.

7.28.2 Data structures

- sf_touch_ctsu_button_callback_args_t
- sf_touch_ctsu_button_individual_t
- sf_touch_ctsu_button_cfg_t
- sf_touch_ctsu_button_hdl
- sf_touch_ctsu_button_instance_t

7.28.3 Enumerations

- sf_touch_button_state_t

7.28.4 Typedefs

- sf_touch_ctsu_button_id

- sf_touch_ctsu_button_ctrl_t

7.28.5 Defines

- #define SF_TOUCH_CTSU_BUTTON_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version Number of API.

- #define SF_TOUCH_CTSU_BUTTON_VERSION_MINOR

  Initial value:`(1U)`

7.28.6 API Data

7.28.6.1 sf_touch_button_state_t

`sf_touch_button_state_t`

Detailed description

Button States.

Enumerated values

Name	Description
TOUCH_BUTTON_STATE_RELEASED	Button is in the released state.
TOUCH_BUTTON_STATE_PRESSED	Button is in the pressed state.
TOUCH_BUTTON_STATE_LONG_HOLD	Button is pressed down for a long time (duration in sf_touch_ctsu_button_config.h)
TOUCH_BUTTON_STATE_SHORT_HOLD	Button is pressed down for a short time (duration in sf_touch_ctsu_button_config.h.
TOUCH_BUTTON_STATE_STUCK	Button is stuck.
TOUCH_BUTTON_STATE_INITIAL	Button has been initialized successfully.
TOUCH_BUTTON_STATE_CLOSING	Button has been disabled and will no longer generate events.
TOUCH_BUTTON_STATE_MULTI_TOUCH	More than one touch element is being touched.
TOUCH_BUTTON_STATE_DISABLED	Button is disabled from being updated.

7.28.6.2 sf_touch_ctsu_button_id

```
typedef uint32_t sf_touch_ctsu_button_id
```

Detailed description

Unique identifier used for each button

7.28.6.3 sf_touch_ctsu_button_ctrl_t

```
typedef void sf_touch_ctsu_button_ctrl_t
```

Detailed description

Cap touch button framework control block. Allocate an instance specific control block to pass into the cap touch button framework API calls. Implemented as

- sf_touch_ctsu_button_instance_ctrl_t

7.28.7 API Structures

7.28.7.1 sf_touch_ctsu_button_callback_args_t

sf_touch_ctsu_button_callback_args_t

Detailed description

Button callback arguments definitions

Variables

- sf_touch_ctsu_button_id id

  Unique id for button.

- sf_touch_button_state_t event

  Button callback event.

- void const * p_context

  Placeholder for user data.

7.28.7.2 sf_touch_ctsu_button_individual_t

sf_touch_ctsu_button_individual_t

Detailed description

Definition of a button and it's behavior in software.

Variables

- ctsu_channel_pair_t button_channel

  Define the channel/channel pairs which make up a button.

- uint8_t release_enable

  enable release events
- uint8_t press_enable

  enable press events
- uint8_t repeat_enable

  enable repeat events
- uint8_t shorthold_enable

  enable short hold events
- uint8_t longhold_enable

  enable long hold events
- uint8_t byte

  Byte representation of events enabled.
- union{} event_enable

  See source code for definition of this member.
- uint16_t debounce_threshold

  Number of consecutive times a button is determined as touched before state changes.
- sf_touch_ctsu_button_id id

  Unique id for button.

7.28.7.3 sf_touch_ctsu_button_cfg_t

sf_touch_ctsu_button_cfg_t

Detailed description

Button Configuration Structure

Variables

- sf_touch_ctsu_instance_t const *const p_lower_lvl_touch_framework

  Pointer to the Touch Framework instance.
- uint32_t button_count

  Number of buttons in configuration.
- sf_touch_ctsu_button_individual_t ** pp_button_cfgs

  Pointer to button configurations.
- void(* p_callback)(sf_touch_ctsu_button_callback_args_t *p_args)

  Callback function.
- void const * p_context

  Placeholder for user data.

- void const * p_extend

  Extension parameter for instance specific settings.

7.28.7.4 sf_touch_ctsu_button_hdl

sf_touch_ctsu_button_hdl

Detailed description

Handle for each button

Variables

- sf_touch_ctsu_button_individual_t button_cfg

  Individual button configuration.

- sf_touch_button_state_t state

  Represents the current state of the button.

- int16_t offset

  Offset in the results array and bit offset in the binary.

- sf_touch_button_state_t prev_state

  Holds previous state of the button.

- uint32_t debounce_counter

  Debounce counter.

- uint32_t active_event_counter

  Amount of time for which button is spending between states.

- uint32_t press_down_counter

  Time for which button is held down.

- uint32_t open

  Indicate that button has been opened.

7.28.7.5 sf_touch_ctsu_button_api_t

sf_touch_ctsu_button_api_t

Detailed description

Touch CTSU Button Framework API structure. Implementations will use the following API.

7.28.7.6 open

```
ssp_err_t(* sf_touch_ctsu_button_api_t::open) (sf_touch_ctsu_button_ctrl_t *const
p_ctrl, sf_touch_ctsu_button_cfg_t const *const p_cfg)
```

Detailed description

Initialize the Touch CTSU Button Framework; configures the lower level hardware and registers callback functions for all the buttons. Implemented as

- SF_TOUCH_CTSU_Button_Open

Table 273:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_button_ctrl_t*const p_ctrl`

Cap touch button framework control block. Allocate an instance specific control block to pass into the cap touch button framework API calls. Implemented assf_touch_ctsu_button_instance_ctrl_t

Parameter p_cfg

Definition: `sf_touch_ctsu_button_cfg_t const *const p_cfg`

Button Configuration Structure

- `sf_touch_ctsu_button_cfg_t`::`sf_touch_ctsu_instance_t`

  Pointer to the Touch Framework instance.

- `sf_touch_ctsu_button_cfg_t`::`button_count`

  Number of buttons in configuration.

- `sf_touch_ctsu_button_cfg_t`::`sf_touch_ctsu_button_individual_t`

  Pointer to button configurations.

- `sf_touch_ctsu_button_cfg_t`::`p_callback`

  Callback function.

- `sf_touch_ctsu_button_cfg_t`::`p_context`

  Placeholder for user data.

- `sf_touch_ctsu_button_cfg_t`::`p_extend`

  Extension parameter for instance specific settings.

7.28.7.7 enable

```
ssp_err_t(* sf_touch_ctsu_button_api_t::enable) (sf_touch_ctsu_button_ctrl_t
*const p_ctrl, sf_touch_ctsu_button_id const button_id)
```

Detailed description

Enable callback notification for a configured button. Implemented as

- SF_TOUCH_CTSU_Button_Enable

Table 274:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
button_id	in	Unique identifier for a button

Parameter p_ctrl

Definition: `sf_touch_ctsu_button_ctrl_t*const p_ctrl`

Cap touch button framework control block. Allocate an instance specific control block to pass into the cap touch button framework API calls. Implemented assf_touch_ctsu_button_instance_ctrl_t

Parameter button_id

`const`

7.28.7.8 disable

```
ssp_err_t(* sf_touch_ctsu_button_api_t::disable) (sf_touch_ctsu_button_ctrl_t
*const p_ctrl, sf_touch_ctsu_button_id const button_id)
```

Detailed description

Disable callback notification for a configured button. Implemented as

- SF_TOUCH_CTSU_Button_Disable

Table 275:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
button_id	in	Unique identifier for a button

Parameter p_ctrl

Definition: `sf_touch_ctsu_button_ctrl_t*const p_ctrl`

Cap touch button framework control block. Allocate an instance specific control block to pass into the cap touch button framework API calls. Implemented assf_touch_ctsu_button_instance_ctrl_t

Parameter button_id

`const`

7.28.7.9 close

```
ssp_err_t(* sf_touch_ctsu_button_api_t::close) (sf_touch_ctsu_button_ctrl_t
*const p_ctrl)
```

Detailed description

Close the Touch CTSU Button Framework; Closes the button framework and lower layers if no other modules are using it. Implemented as

- SF_TOUCH_CTSU_Button_Close

Table 276:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_button_ctrl_t*const p_ctrl`

Cap touch button framework control block. Allocate an instance specific control block to pass into the cap touch button framework API calls. Implemented assf_touch_ctsu_button_instance_ctrl_t

7.28.7.10 versionGet

```
ssp_err_t(* sf_touch_ctsu_button_api_t::versionGet) (ssp_version_t *const
p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- SF_TOUCH_CTSU_Button_VersionGet

Table 277:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.28.7.11 sf_touch_ctsu_button_instance_t

sf_touch_ctsu_button_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_touch_ctsu_button_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_touch_ctsu_button_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_touch_ctsu_button_api_t const * p_api

  Pointer to the API structure for this instance.

7.29 CTSU Slider Framework Interface

RTOS-integrated CTSU Slider Framework Interface.

This framework layer uses the CTSU Framework Layer to implement a slider or wheel interface. Using this Slider Framework the user can configure and use multiple sliders/wheels using the configuration structure generated from WorkBench. An action on each slider/wheel (press, release etc) will result in a callback for that slider with an argument indicating the event type.

Related SSP architecture topics:

- What is an SSP Interface? SSP Interfaces
- What is an SSP Layer? SSP Predefined Layers
- How to use SSP Interfaces and Modules? Using SSP Modules

CTSU Slider Framework Interface description: Capacitive Touch Slider Framework

7.29.1 Interface API

sf_touch_ctsu_slider_api_t

Function name	Description
.open	Initialize the Touch CTSU Slider Framework, configures the lower level hardware and registers callback functions for all the sliders.
.enable	Enable callback notification for a configured slider.
.disable	Disable callback notification for a configured slider.
.close	Close the Touch CTSU Slider Framework; Closes the slider framework and lower layers if no other modules are using it.
.versionGet	Get version and store it in provided pointer p_version.

7.29.2 Data structures

- sf_touch_ctsu_slider_callback_args_t
- sf_touch_ctsu_slider_cfg_t
- sf_touch_ctsu_slider_instance_t

7.29.3 Enumerations

- sf_touch_ctsu_slider_state_t
- sf_slider_type_t

7.29.4 Typedefs

- sf_touch_ctsu_slider_id_t
- sf_touch_ctsu_slider_ctrl_t

7.29.5 Defines

- #define SF_TOUCH_CTSU_SLIDER_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version Number of API.
- #define SF_TOUCH_CTSU_SLIDER_VERSION_MINOR

  Initial value:`(1U)`

7.29.6 API Data

7.29.6.1 sf_touch_ctsu_slider_state_t

`sf_touch_ctsu_slider_state_t`

Detailed description

Slider States

Enumerated values

Name	Description
SF_TOUCH_CTSU_SLIDER_STATE_NO_CHANGE	Slider is in the released state
SF_TOUCH_CTSU_SLIDER_STATE_INITIALIZED	Slider is in the pressed state

Name	Description
SF_TOUCH_CTSU_SLIDER_STATE_TOUCHED	Slider is pressed
SF_TOUCH_CTSU_SLIDER_STATE_RELEASED	Slider is released
SF_TOUCH_CTSU_SLIDER_STATE_CLOSED	Slider has been disabled and will no longer generate events.
SF_TOUCH_CTSU_SLIDER_STATE_MULTI_TOUCH	More than one touch element is being touched
SF_TOUCH_CTSU_SLIDER_STATE_DISABLED	Slider is disabled from being updated.
SF_TOUCH_CTSU_SLIDER_STATE_HELD	Slider is held. (Continued press)

7.29.6.2 sf_slider_type_t

```
sf_slider_type_t
```

Detailed description

Slider type definition

Enumerated values

Name	Description
SF_SLIDER_TYPE_LINEAR	The slider is of a linear type
SF_SLIDER_TYPE_CIRCULAR	The slider is of a circular type (wheel)

7.29.6.3 sf_touch_ctsu_slider_id_t

```
typedef uint32_t sf_touch_ctsu_slider_id_t
```

Detailed description

Unique identifier used for each slider

7.29.6.4 sf_touch_ctsu_slider_ctrl_t

```
typedef void sf_touch_ctsu_slider_ctrl_t
```

Detailed description

Cap touch slider framework control block. Allocate an instance specific control block to pass into the cap touch slider framework API calls. Implemented as

- sf_touch_ctsu_slider_instance_ctrl_t

7.29.7 API Structures

7.29.7.1 sf_touch_ctsu_slider_callback_args_t

sf_touch_ctsu_slider_callback_args_t

Detailed description

Slider callback arguments definitions

Variables

- sf_touch_ctsu_slider_id_t id

  Unique slider identifier.

- uint32_t event

  Slider callback event.

- uint32_t current_position

  Current slider position.

- void const * p_context

  Placeholder for user data.

7.29.7.2 sf_touch_ctsu_slider_cfg_t

sf_touch_ctsu_slider_cfg_t

Detailed description

Slider Configuration Structure

Variables

- sf_touch_ctsu_instance_t * p_lower_lvl_touch_framework

  Pointer to the Touch Framework instance

- uint32_t slider_count

  Number of sliders in configuration

- void(* p_callback)(sf_touch_ctsu_slider_callback_args_t *p_args)

  Callback function

- void const * p_context

  Placeholder for user data

- void const * p_extend

  Extension parameter for instance specific settings.

7.29.7.3 sf_touch_ctsu_slider_api_t

sf_touch_ctsu_slider_api_t

Detailed description

Touch CTSU Slider Framework API structure. Implementations will use the following API.

7.29.7.4 open

```
ssp_err_t(* sf_touch_ctsu_slider_api_t::open) (sf_touch_ctsu_slider_ctrl_t *const
p_ctrl, sf_touch_ctsu_slider_cfg_t const *const p_cfg)
```

Detailed description

Initialize the Touch CTSU Slider Framework, configures the lower level hardware and registers callback functions for all the sliders. Implemented as

- SF_TOUCH_CTSU_Slider_Open

Table 278:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_slider_ctrl_t*const p_ctrl`

Cap touch slider framework control block. Allocate an instance specific control block to pass into the cap touch slider framework API calls. Implemented assf_touch_ctsu_slider_instance_ctrl_t

Parameter p_cfg

Definition: `sf_touch_ctsu_slider_cfg_t const *const p_cfg`

Slider Configuration Structure

- `sf_touch_ctsu_slider_cfg_t`::`sf_touch_ctsu_instance_t`

  Pointer to the Touch Framework instance
- `sf_touch_ctsu_slider_cfg_t`::`slider_count`

  Number of sliders in configuration
- `sf_touch_ctsu_slider_cfg_t`::`p_callback`

  Callback function
- `sf_touch_ctsu_slider_cfg_t`::`p_context`

  Placeholder for user data
- `sf_touch_ctsu_slider_cfg_t`::`p_extend`

  Extension parameter for instance specific settings.

7.29.7.5 enable

```
ssp_err_t(* sf_touch_ctsu_slider_api_t::enable) (sf_touch_ctsu_slider_ctrl_t
*const p_ctrl, sf_touch_ctsu_slider_id_t const slider_id)
```

Detailed description

Enable callback notification for a configured slider. Implemented as

- SF_TOUCH_CTSU_Slider_Enable

Table 279:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
slider_id	in	Unique identifier for a slider

Parameter p_ctrl

Definition: `sf_touch_ctsu_slider_ctrl_t*const p_ctrl`

Cap touch slider framework control block. Allocate an instance specific control block to pass into the cap touch slider framework API calls. Implemented assf_touch_ctsu_slider_instance_ctrl_t

Parameter slider_id

7.29.7.6 disable

```
ssp_err_t(* sf_touch_ctsu_slider_api_t::disable) (sf_touch_ctsu_slider_ctrl_t
*const p_ctrl, sf_touch_ctsu_slider_id_t const slider_id)
```

Detailed description

Disable callback notification for a configured slider. Implemented as

- SF_TOUCH_CTSU_Slider_Disable

Table 280:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
slider_id	in	Unique identifier for a slider

Parameter p_ctrl

Definition: `sf_touch_ctsu_slider_ctrl_t*const p_ctrl`

Cap touch slider framework control block. Allocate an instance specific control block to pass into the cap touch slider framework API calls. Implemented assf_touch_ctsu_slider_instance_ctrl_t

Parameter slider_id

7.29.7.7 close

```
ssp_err_t(* sf_touch_ctsu_slider_api_t::close) (sf_touch_ctsu_slider_ctrl_t
*const p_ctrl)
```

Detailed description

Close the Touch CTSU Slider Framework; Closes the slider framework and lower layers if no other modules are using it. Implemented as

- SF_TOUCH_CTSU_Slider_Close

Table 281:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `sf_touch_ctsu_slider_ctrl_t*const p_ctrl`

Cap touch slider framework control block. Allocate an instance specific control block to pass into the cap touch slider framework API calls. Implemented assf_touch_ctsu_slider_instance_ctrl_t

7.29.7.8 versionGet

```
ssp_err_t(* sf_touch_ctsu_slider_api_t::versionGet) (ssp_version_t *const
p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- SF_TOUCH_CTSU_Slider_VersionGet

Table 282:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

7.29.7.9 sf_touch_ctsu_slider_instance_t

sf_touch_ctsu_slider_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_touch_ctsu_slider_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- sf_touch_ctsu_slider_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- sf_touch_ctsu_slider_api_t const * p_api

  Pointer to the API structure for this instance.

7.30 Touch Panel Framework Interface

RTOS-integrated Touch Panel Framework Interface.

7.30.1 Summary

This module is a ThreadX-aware Touch Panel Framework which scans for touch events and posts them to the Messaging Framework for distribution to touch event subscribers. This Interface is implemented by I2C Touch Panel Framework.

Interfaces used:

- External IRQ Framework Interface
- I2C Interface
- essaging Framework Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Touch Panel Framework Interface description: Touch Panel Framework

7.30.2 Interface API

sf_touch_panel_api_t

Function name	Description
.open	Create required RTOS objects, call lower level module for hardware specific initialization, and create a thread to post touch data to a message queue.

Function name	Description
.calibrate	Begin calibration routine based on provided expected coordinates. Returns SSP_SUCCESS only if the tolerance is longer than the distance from the expected touch point to the actual touch point (using the formula below): p_calibrate->tolerance_pixels ^ 2 > (p_calibrate->x - x_measured) ^ 2 + (p_calibrate->y - y_measured) ^ 2
.start	Start scanning for touch events.
.stop	Stop scanning for touch events.
.reset	Reset touch controller if reset pin is provided, and resets the I2C bus.
.close	Terminate touch thread and close channel at HAL layer.
.versionGet	Gets version and stores it in provided pointer p_version.

7.30.3 Data structures

- sf_touch_panel_payload_t
- sf_touch_panel_cfg_t
- sf_touch_panel_calibrate_t
- sf_touch_panel_instance_t

7.30.4 Enumerations

- sf_touch_panel_event_t

7.30.5 Typedefs

- sf_touch_panel_ctrl_t

7.30.6 Defines

- #define SF_TOUCH_PANEL_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define SF_TOUCH_PANEL_API_VERSION_MINOR

  Initial value:`(2U)`
- #define SF_TOUCH_PANEL_MESSAGE_WORDS

  Initial value:`((sizeof(sf_touch_panel_payload_t) + 3) / 4)`

  Touch panel message size in 4 byte words, rounded up.
- #define SF_TOUCH_PANEL_MAX_MESSAGES

  Initial value:`(4)`

  Maximum number of messages in touch panel message queue.
- #define SF_TOUCH_BYTES_PER_WORD

  Initial value:`(4)`

  Macro defining the number of bytes per word.
- #define SF_TOUCH_PANEL_MESSAGE_MEM_BYTES

  Initial value:`(SF_TOUCH_PANEL_MESSAGE_WORDS * SF_TOUCH_BYTES_PER_WORD \ * SF_TOUCH_PANEL_MAX_MESSAGES)`

  Touch message queue memory size.

7.30.7 API Data

7.30.7.1 sf_touch_panel_event_t

`sf_touch_panel_event_t`

Detailed description

Touch event list.

Enumerated values

Name	Description
SF_TOUCH_PANEL_EVENT_INVALID	Invalid touch data.
SF_TOUCH_PANEL_EVENT_HOLD	Touch has not moved since last touch event.
SF_TOUCH_PANEL_EVENT_MOVE	Touch has moved since last touch event.
SF_TOUCH_PANEL_EVENT_DOWN	New touch event reported.
SF_TOUCH_PANEL_EVENT_UP	Touch released.
SF_TOUCH_PANEL_EVENT_NONE	No valid touch event happened.

7.30.7.2 sf_touch_panel_ctrl_t

```
typedef void sf_touch_panel_ctrl_t
```

Detailed description

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented as

- sf_touch_panel_i2c_instance_ctrl_t

7.30.8 API Structures

7.30.8.1 sf_touch_panel_payload_t

sf_touch_panel_payload_t

Detailed description

Touch data payload posted to the message queue.

Variables

- sf_message_header_t header

  Required header for messaging framework.
- int16_t x

  X coordinate.
- int16_t y

  Y coordinate.
- sf_touch_panel_event_t event_type

  Touch event type.

7.30.8.2 sf_touch_panel_cfg_t

sf_touch_panel_cfg_t

Detailed description

Configuration for RTOS integrated touch panel framework.

Variables

- uint16_t hsize_pixels

  Horizontal size of screen in pixels.
- uint16_t vsize_pixels

  Vertical size of screen in pixels.
- UINT priority

  Priority of the touch panel thread.

- sf_message_instance_t const * p_message

  Pointer to messaging framework control block.

- uint8_t event_class_instance

  Event class instance number for posting touch event class messages.

- uint16_t update_hz

  The frequency to report repeat (SF_TOUCH_PANEL_EVENT_DOWN or SF_TOUCH_PANEL_EVENT_HOLD) touch events in Hertz. This will be converted to RTOS ticks in the driver and rounded up to the nearest integer value of RTOS ticks.

- uint16_t rotation_angle

  Touch coordinate rotation angle(0/90/180/270)

- void const * p_extend

  Pointer to hardware specific extension. See sf_touch_panel_<instance>.h.

7.30.8.3 sf_touch_panel_calibrate_t

sf_touch_panel_calibrate_t

Detailed description

Calibration data passed to SF_TOUCH_PANEL_Calibrate.

Variables

- uint16_t x

  Expected x coordinate.

- uint16_t y

  Expected y coordinate.

- uint16_t tolerance_pixels

  Acceptable linear deviation from the expected coordinates in pixels.

- void const * p_extend

  Pointer to hardware specific extension. See sf_touch_panel_<instance>_cfg_t in sf_touch_panel_<instance>.h.

7.30.8.4 sf_touch_panel_api_t

sf_touch_panel_api_t

Detailed description

Touch panel API structure. Touch panel implementations use the following API.

7.30.8.5 open

```
ssp_err_t(* sf_touch_panel_api_t::open) (sf_touch_panel_ctrl_t *const p_ctrl,
sf_touch_panel_cfg_t const *const p_cfg)
```

Detailed description

Create required RTOS objects, call lower level module for hardware specific initialization, and create a thread to post touch data to a message queue. Implemented as

- SF_TOUCH_PANEL_I2C_Open

Table 283:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. This control structure is initialized in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `sf_touch_panel_ctrl_t*const p_ctrl`

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

Parameter p_cfg

Definition: `sf_touch_panel_cfg_t const *const p_cfg`

Configuration for RTOS integrated touch panel framework.

- `sf_touch_panel_cfg_t`::`hsize_pixels`

  Horizontal size of screen in pixels.

- `sf_touch_panel_cfg_t`::`vsize_pixels`

  Vertical size of screen in pixels.

- `sf_touch_panel_cfg_t`::`priority`

  Priority of the touch panel thread.

- `sf_touch_panel_cfg_t`::`sf_message_instance_t`

  Pointer to messaging framework control block.

- `sf_touch_panel_cfg_t`::`event_class_instance`

  Event class instance number for posting touch event class messages.

- `sf_touch_panel_cfg_t`::`update_hz`

  The frequency to report repeat (SF_TOUCH_PANEL_EVENT_DOWN or SF_TOUCH_PANEL_EVENT_HOLD) touch events in Hertz. This will be converted to RTOS ticks in the driver and rounded up to the nearest integer value of RTOS ticks.

- sf_touch_panel_cfg_t::rotation_angle

  Touch coordinate rotation angle(0/90/180/270)

- sf_touch_panel_cfg_t::p_extend

  Pointer to hardware specific extension. See sf_touch_panel_<instance>.h.

7.30.8.6 calibrate

```
ssp_err_t(* sf_touch_panel_api_t::calibrate) (sf_touch_panel_ctrl_t *const
p_ctrl, sf_touch_panel_calibrate_t const *const p_expected,
sf_touch_panel_payload_t const *const p_actual, ULONG timeout)
```

Detailed description

Begin calibration routine based on provided expected coordinates. Returns SSP_SUCCESS only if the tolerance is longer than the distance from the expected touch point to the actual touch point (using the formula below): p_calibrate->tolerance_pixels ^ 2 > (p_calibrate->x - x_measured) ^ 2 + (p_calibrate->y - y_measured) ^ 2 Implemented as

- SF_TOUCH_PANEL_I2C_Calibrate

Table 284:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in touch_panel_api_t::open.
p_expected	in	Expected coordinates and tolerance for passing.
p_actual	in	Pointer to message payload received from SF_MESSAGE_EVENT_CLASS_TOUCH event class.
timeout	in	ThreadX timeout. Select TX_NO_WAIT, a value in system clock counts between 1 and 0xFFFFFFFF, or TX_WAIT_FOREVER.

Parameter p_ctrl

Definition: sf_touch_panel_ctrl_t*const p_ctrl

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

Parameter p_expected

Definition: sf_touch_panel_calibrate_tconst *const p_expected

Calibration data passed to SF_TOUCH_PANEL_Calibrate.

- sf_touch_panel_calibrate_t::x

  Expected x coordinate.
- sf_touch_panel_calibrate_t::y

  Expected y coordinate.
- sf_touch_panel_calibrate_t::tolerance_pixels

  Acceptable linear deviation from the expected coordinates in pixels.
- sf_touch_panel_calibrate_t::p_extend

  Pointer to hardware specific extension. See sf_touch_panel_<instance>_cfg_t in sf_touch_panel_<instance>.h.

Parameter p_actual

Definition: sf_touch_panel_payload_tconst *const p_actual

Touch data payload posted to the message queue.

- sf_touch_panel_payload_t::header

  Required header for messaging framework.
- sf_touch_panel_payload_t::x

  X coordinate.
- sf_touch_panel_payload_t::y

  Y coordinate.
- sf_touch_panel_payload_t::event_type

  Touch event type.

Parameter timeout

const

7.30.8.7 start

```
ssp_err_t(* sf_touch_panel_api_t::start) (sf_touch_panel_ctrl_t *const p_ctrl)
```

Brief description

Start scanning for touch events.

Detailed description

Implemented as

- SF_TOUCH_PANEL_I2C_Start

Table 285:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in touch_panel_api_t::open.

Parameter p_ctrl

Definition: `sf_touch_panel_ctrl_t*const p_ctrl`

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

7.30.8.8 stop

```
ssp_err_t(* sf_touch_panel_api_t::stop) (sf_touch_panel_ctrl_t *const p_ctrl)
```

Brief description

Stop scanning for touch events.

Detailed description

Implemented as

- SF_TOUCH_PANEL_I2C_Stop

Table 286:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in touch_panel_api_t::open.

Parameter p_ctrl

Definition: `sf_touch_panel_ctrl_t*const p_ctrl`

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

7.30.8.9 reset

```
ssp_err_t(* sf_touch_panel_api_t::reset) (sf_touch_panel_ctrl_t *const p_ctrl)
```

Brief description

Reset touch controller if reset pin is provided, and resets the I2C bus.

Detailed description

Implemented as

- SF_TOUCH_PANEL_I2C_Reset

NOTE: This does not include calibration. Use calibrate from the application after this function if calibration is required after reset.

Table 287:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in touch_panel_api_t::open.

Parameter p_ctrl

Definition: `sf_touch_panel_ctrl_t*const p_ctrl`

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

7.30.8.10 close

```
ssp_err_t(* sf_touch_panel_api_t::close) (sf_touch_panel_ctrl_t *const p_ctrl)
```

Brief description

Terminate touch thread and close channel at HAL layer.

Detailed description

Implemented as

- SF_TOUCH_PANEL_I2C_Close

Table 288:Parameters

Name	Direction	Description
p_ctrl	in	Handle set in touch_panel_api_t::open.

Parameter p_ctrl

Definition: `sf_touch_panel_ctrl_t*const p_ctrl`

Touch panel framework control block. Allocate an instance specific control block to pass into the touch panel framework API calls. Implemented assf_touch_panel_i2c_instance_ctrl_t

7.30.8.11 versionGet

```
ssp_err_t(* sf_touch_panel_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Implemented as

- SF_TOUCH_PANEL_I2C_VersionGet

Table 289:Parameters

Name	Direction	Description
p_version	out	Code and API version used stored here.

Parameter p_version

7.30.8.12 sf_touch_panel_instance_t

sf_touch_panel_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_touch_panel_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_touch_panel_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_touch_panel_api_t const * p_api

  Pointer to the API structure for this instance.

7.31 SF WIFI Framework Interface

RTOS-integrated SF WIFI Framework Interface.

7.31.1 Summary

This SSP Interface provides access to the ThreadX-aware SF WIFI Framework.

7.31.2 Interface API

sf_wifi_api_t

Function name	Description
.open	Initializes the network interface for data transfers. Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	De-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.
.multicastListAdd	Add the given MAC address to the multicast filter list.
.multicastListDelete	Delete the given MAC address from the multicast filter list.
.statisticsGet	Get the interface statistics.
.transmit	Transmit data packet.
.provisioningSet	Set WiFi module provisioning which will configure the module in AP or Client mode.
.provisioningGet	Get the provisioning information of WiFi module.
.infoGet	Get WiFi module information.
.scan	Scan for WiFi SSIDs.
.ACLAdd	Adds a MAC address to the Access Control List. Valid in AP mode only.
.ACLDelete	Deletes a MAC address from Access Control List. Valid in AP mode only.
.macAddressGet	Get WiFi module MAC address.
.macAddressSet	Set WiFi module MAC address.
.versionGet	Gets version and stores it in provided pointer p_version.

7.31.3 Data structures

- sf_wifi_ip_addr_t
- sf_wifi_info_t
- sf_wifi_callback_args_t

- sf_wifi_provisioning_t
- sf_wifi_cfg_t
- sf_wifi_stats_t
- sf_wifi_scan_t
- sf_wifi_ctrl_t
- sf_wifi_instance_t

7.31.4 Enumerations

- sf_wifi_ip_addr_version_t
- sf_wifi_interface_mode_t
- sf_wifi_wep_key_format_t
- sf_wifi_security_type_t
- sf_wifi_encryption_type_t
- sf_wifi_bss_type_t
- sf_wifi_interface_hw_mode_t
- sf_wifi_rts_t
- sf_wifi_preamble_t
- sf_wifi_wmm_t
- sf_wifi_high_throughput_t
- sf_wifi_ssid_broadcast_t
- sf_wifi_wds_t
- sf_wifi_mandatory_high_throughput_t
- sf_wifi_auto_negotiation_t
- sf_wifi_access_control_t
- sf_wifi_event_t

7.31.5 Defines

- #define SF_WIFI_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file

- #define SF_WIFI_API_VERSION_MINOR

  Initial value:`(0U)`

  Minor Version of the API defined in this file

- #define SF_WIFI_SSID_LENGTH

  Initial value:`(32U)`

  WiFi SSID length.
- #define SF_WIFI_SECURITY_KEY_LENGTH

  Initial value:`(128U)`

  WiFi Security Key length.
- #define SF_WIFI_MAC_ADDR_LENGTH

  Initial value:`(6U)`

  WiFi MAC address length.
- #define SF_WIFI_TRUE

  Initial value:`(1U)`

  Boolean True condition.
- #define SF_WIFI_FALSE

  Initial value:`(0U)`

  Boolean False condition.
- #define SF_WIFI_IPV4_ADDRESS

  Initial value:
```
((((uint32_t) a) << (24U)) | (((uint32_t) b) << (16U)) | \
(((uint32_t) c) << (8U)) | ((uint32_t) d))
```

  IP Address Generation Macro

7.31.6 API Data

7.31.6.1 sf_wifi_ip_addr_version_t

`sf_wifi_ip_addr_version_t`

Detailed description

IP address version

Enumerated values

Name	Description
SF_WIFI_IP_ADDR_VERSION_4	IPv4 address.
SF_WIFI_IP_ADDR_VERSION_6	IPv6 address.

7.31.6.2 sf_wifi_interface_mode_t

`sf_wifi_interface_mode_t`

Detailed description

WiFi Interface mode

Enumerated values

Name	Description
SF_WIFI_INTERFACE_MODE_AP	Access Point mode.
SF_WIFI_INTERFACE_MODE_CLIENT	Station Mode.

7.31.6.3 sf_wifi_wep_key_format_t

`sf_wifi_wep_key_format_t`

Detailed description

WiFi WEP Key Format

Enumerated values

Name	Description
SF_WIFI_WEP_KEY_FORMAT_ASCII	WEP Key in ASCII.
SF_WIFI_WEP_KEY_FORMAT_HEX	WEP Key in Hex.

7.31.6.4 sf_wifi_security_type_t

`sf_wifi_security_type_t`

Detailed description

WiFi Security type

Enumerated values

Name	Description
SF_WIFI_SECURITY_TYPE_OPEN	Open. No encryption used.
SF_WIFI_SECURITY_TYPE_WEP	128-bit WEP OPEN ASCII
SF_WIFI_SECURITY_TYPE_WPA	WiFi Protected Access.

Name	Description
SF_WIFI_SECURITY_TYPE_WPA2	WiFi Protected Access v2.

7.31.6.5 sf_wifi_encryption_type_t

`sf_wifi_encryption_type_t`

Detailed description

WiFi Encryption type

Enumerated values

Name	Description
SF_WIFI_ENCRYPTION_TYPE_AUTO	Automatic selection of encryption protocol.
SF_WIFI_ENCRYPTION_TYPE_TKIP	Temporal Key Integrity Protocol. Used by WPA.
SF_WIFI_ENCRYPTION_TYPE_CCMP	CTR mode with CBC-MAC Protocol. Used by WPA2.
SF_WIFI_ENCRYPTION_TYPE_WEP	WEP mode. Used by WEP.
SF_WIFI_ENCRYPTION_TYPE_NONE	No Encryption. Used by Open Security type.

7.31.6.6 sf_wifi_bss_type_t

`sf_wifi_bss_type_t`

Detailed description

WiFi BSS type

Enumerated values

Name	Description
SF_WIFI_BSS_TYPE_INFRASTRUCTURE	Infrastructure network.
SF_WIFI_BSS_TYPE_ADHOC	802.11 ad-hoc IBSS network
SF_WIFI_BSS_TYPE_ANY	Either infrastructure or ad-hoc network.
SF_WIFI_BSS_TYPE_UNKNOWN	BSS type is unknown.

7.31.6.7 sf_wifi_interface_hw_mode_t

`sf_wifi_interface_hw_mode_t`

Detailed description

WiFi Hardware mode

Enumerated values

Name	Description
SF_WIFI_INTERFACE_HW_MODE_11A	802.11a
SF_WIFI_INTERFACE_HW_MODE_11B	802.11b
SF_WIFI_INTERFACE_HW_MODE_11G	802.11g
SF_WIFI_INTERFACE_HW_MODE_11N	802.11n

7.31.6.8 sf_wifi_rts_t

`sf_wifi_rts_t`

Detailed description

WiFi RTS flag

Enumerated values

Name	Description
SF_WIFI_RTS_DISABLE	Disable RTS/CTS handshake.
SF_WIFI_RTS_ENABLE	Enable RTS/CTS handshake.

7.31.6.9 sf_wifi_preamble_t

`sf_wifi_preamble_t`

Detailed description

WiFi Preamble type

Enumerated values

Name	Description
SF_WIFI_PREAMBLE_SHORT	Use short preamble.

Name	Description
SF_WIFI_PREAMBLE_LONG	Use long preamble.

7.31.6.10 sf_wifi_wmm_t

`sf_wifi_wmm_t`

Detailed description

WiFi WMM flag

Enumerated values

Name	Description
SF_WIFI_WMM_DISABLE	Disable WMM.
SF_WIFI_WMM_ENABLE	Enable WMM.

7.31.6.11 sf_wifi_high_throughput_t

`sf_wifi_high_throughput_t`

Detailed description

WiFi High Throughput flag

Enumerated values

Name	Description
SF_WIFI_HIGH_THROUGHPUT_DISABLE	Disable high throughput mode.
SF_WIFI_HIGH_THROUGHPUT_ENABLE	Enable high throughput mode. Also requires WMM to be enabled.

7.31.6.12 sf_wifi_ssid_broadcast_t

`sf_wifi_ssid_broadcast_t`

Detailed description

WiFi SSID Broadcast flag

Enumerated values

Name	Description
SF_WIFI_SSID_BROADCAST_DISABLE	Disable SSID Broadcast.
SF_WIFI_SSID_BROADCAST_ENABLE	Enable SSID Broadcast.

7.31.6.13 sf_wifi_wds_t

`sf_wifi_wds_t`

Detailed description

WiFi WDS Flasg

Enumerated values

Name	Description
SF_WIFI_WDS_DISABLE	Disable WDS.
SF_WIFI_WDS_ENABLE	Enable WDS.

7.31.6.14 sf_wifi_mandatory_high_throughput_t

`sf_wifi_mandatory_high_throughput_t`

Detailed description

WiFi Mandatory High Throughput flag

Enumerated values

Name	Description
SF_WIFI_MANDATORY_HIGH_THROUGHPUT_DISABLE	Disable Mandatory HT requirement.
SF_WIFI_MANDATORY_HIGH_THROUGHPUT_ENABLE	Enable mandatory HT requirement.

7.31.6.15 sf_wifi_auto_negotiation_t

`sf_wifi_auto_negotiation_t`

Detailed description

WiFi Auto Negotiation flag

Enumerated values

Name	Description
SF_WIFI_AUTO_NEGOTIATION_DISABLE	Auto negotiation disable.
SF_WIFI_AUTO_NEGOTIATION_ENABLE	Auto negotiation enable.

7.31.6.16 sf_wifi_access_control_t

`sf_wifi_access_control_t`

Detailed description

WiFi Framework AccessContol mode

Enumerated values

Name	Description
SF_WIFI_ACCESS_CONTROL_DISABLE	Disable MAC address matching.
SF_WIFI_ACCESS_CONTROL_DENY	Deny association to stations on the MAC list.
SF_WIFI_ACCESS_CONTROL_ALLOW	Allow association to stations on the MAC list.

7.31.6.17 sf_wifi_event_t

`sf_wifi_event_t`

Detailed description

WiFi Framework event codes

Enumerated values

Name	Description
SF_WIFI_EVENT_RX	Packet received event.
SF_WIFI_EVENT_AP_CONNECT	Device Associated Successfully with AP.
SF_WIFI_EVENT_AP_DISCONNECT	Device Disconnected with AP.

7.31.7 API Structures

7.31.7.1 sf_wifi_ip_addr_t

sf_wifi_ip_addr_t

Detailed description

IP address information

Variables

- sf_wifi_ip_addr_version_t version

  IP Address Version : v4 or v6.

- uint32_t v4
- uint32_t v6[4]
- union{} addr

  IP address.

  See source code for definition of this member.

7.31.7.2 sf_wifi_info_t

sf_wifi_info_t

Detailed description

Configuration about underlying device driver.

Variables

- uint8_t * p_chipset

  Pointer to sting showing WiFi chipset/driver information.

- uint16_t rssi

  Received signal strength indication.

- uint16_t noise_level

  Signal to noise ratio.

- uint16_t link_quality

  Signal strength / quality.

7.31.7.3 sf_wifi_callback_args_t

sf_wifi_callback_args_t

Detailed description

WiFi framework callback parameter definition

Variables

- sf_wifi_event_t event

  Event code.
- uint8_t * p_data

  Packet data.
- uint32_t length

  Packet Data length.
- void const * p_context

  Context provided to user during callback.

7.31.7.4 sf_wifi_provisioning_t

sf_wifi_provisioning_t

Detailed description

WiFi Provisioning parameters

Variables

- sf_wifi_interface_mode_t mode

  Select AP or Client mode.
- uint8_t channel

  RF Channel number.
- uint8_t ssid[SF_WIFI_SSID_LENGTH]

  SSID.
- sf_wifi_security_type_t security

  Security type.
- sf_wifi_encryption_type_t encryption

  Encryption type.
- uint8_t key[SF_WIFI_SECURITY_KEY_LENGTH]

  Pre-shared key.
- void(* p_callback)(sf_wifi_callback_args_t *p_args)

  Pointer to AP Link UP/Down callback function.

7.31.7.5 sf_wifi_cfg_t

sf_wifi_cfg_t

Detailed description

Define the WiFi configuration parameters

Variables

- uint8_t mac_addr[6]

  MAC address of WiFi Device.
- sf_wifi_interface_hw_mode_t hw_mode

  Modulation type: 11a/b/g/n.
- uint8_t tx_power

  Sets transmit power in dBm.
- sf_wifi_rts_t rts

  RTS/CTS handshake flag.
- uint16_t fragmentation

  Fragmentation threshold.
- uint8_t dtim

  Delivery traffic indication message interval.
- sf_wifi_high_throughput_t high_throughput

  High-throughput mode. Only valid for 802.11n.
- sf_wifi_preamble_t preamble

  Preamble type.
- sf_wifi_wmm_t wmm

  WiFi Multimedia Mode flag. If enabled, also requires.
- uint8_t max_stations

  Maximum permitted stations. Valid in AP mode only.
- sf_wifi_ssid_broadcast_t ssid_broadcast

  SSID broadcast flag. Valid in AP mode only.
- sf_wifi_access_control_t access_control

  Mode of access control MAC list.
- uint32_t beacon

  Beacon interval. Valid in AP mode only.
- uint32_t station_inactivity_timeout

  Station inactivity timeout value. Valid in AP mode only.
- sf_wifi_wds_t wds

  WDS flag. Valid in AP mode only.
- void * p_buffer_pool_rx

  Pointer to Network stack Rx buffer pool.
- sf_wifi_mandatory_high_throughput_t req_high_throughput

  Only allow HT mode. Valid in AP mode only.

- void(* p_callback)(sf_wifi_callback_args_t *p_args)

  Pointer to callback function.
- void const * p_context

  User defined context passed into callback function.
- void const * p_extend

  Instance specific configuration.

7.31.7.6 sf_wifi_stats_t

sf_wifi_stats_t

Detailed description

Define the statistic and error counters for this IP instance.

Variables

- uint32_t rx_pkts

  Packets received successfully.
- uint32_t tx_pkts

  Packets transmitted successfully.
- uint32_t tx_err

  Transmit errors.

7.31.7.7 sf_wifi_scan_t

sf_wifi_scan_t

Detailed description

Define the structure to store the SSID scan information

Variables

- sf_wifi_interface_hw_mode_t hw_mode

  Hardware mode 802.11a/b/g/n.
- uint8_t rssi

  Signal Strength.
- uint8_t ssid[SF_WIFI_SSID_LENGTH]

  SSID name.
- uint8_t bssid[SF_WIFI_MAC_ADDR_LENGTH]

  Basic Service Set Identification (i.e. MAC address of Access Point)
- uint8_t channel

  Radio channel that the AP beacon was received on.

- sf_wifi_security_type_t security

  Security type.
- sf_wifi_encryption_type_t encryption

  Encryption type.
- sf_wifi_bss_type_t bss_type

  Network type.

7.31.7.8 sf_wifi_ctrl_t

sf_wifi_ctrl_t

Detailed description

WiFi Framework control structure

Variables

- void * p_driver_handle

  Storage for information needed for each WiFi device driver in the system.

7.31.7.9 sf_wifi_api_t

sf_wifi_api_t

Detailed description

Framework API structure. Implementations will use the following API.

7.31.7.10 open

```
ssp_err_t(* sf_wifi_api_t::open) (sf_wifi_ctrl_t *p_ctrl, sf_wifi_cfg_t const
*const p_cfg)
```

Brief description

Initializes the network interface for data transfers.

Detailed description

Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Table 290:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to WiFi configuration structure.

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_cfg

Definition: `sf_wifi_cfg_t const *const p_cfg`

Define the WiFi configuration parameters

- `sf_wifi_cfg_t`::`mac_addr`

  MAC address of WiFi Device.

- `sf_wifi_cfg_t`::`sf_wifi_interface_hw_mode_t`

  Modulation type: 11a/b/g/n.

  Enumerated as:

  – SF_WIFI_INTERFACE_HW_MODE_11A

  – SF_WIFI_INTERFACE_HW_MODE_11B

  – SF_WIFI_INTERFACE_HW_MODE_11G

  – SF_WIFI_INTERFACE_HW_MODE_11N

- `sf_wifi_cfg_t`::`tx_power`

  Sets transmit power in dBm.

- `sf_wifi_cfg_t`::`sf_wifi_rts_t`

  RTS/CTS handshake flag.

  Enumerated as:

  – SF_WIFI_RTS_DISABLE

  – SF_WIFI_RTS_ENABLE

- `sf_wifi_cfg_t`::`fragmentation`

  Fragmentation threshold.

- `sf_wifi_cfg_t`::`dtim`

  Delivery traffic indication message interval.

- `sf_wifi_cfg_t`::`sf_wifi_high_throughput_t`

  High-throughput mode. Only valid for 802.11n.

  Enumerated as:

  – SF_WIFI_HIGH_THROUGHPUT_DISABLE

  – SF_WIFI_HIGH_THROUGHPUT_ENABLE

- sf_wifi_cfg_t::sf_wifi_preamble_t

  Preamble type.

  Enumerated as:

  – SF_WIFI_PREAMBLE_SHORT

  – SF_WIFI_PREAMBLE_LONG
- sf_wifi_cfg_t::sf_wifi_wmm_t

  WiFi Multimedia Mode flag. If enabled, also requires.

  Enumerated as:

  – SF_WIFI_WMM_DISABLE

  – SF_WIFI_WMM_ENABLE
- sf_wifi_cfg_t::max_stations

  Maximum permitted stations. Valid in AP mode only.
- sf_wifi_cfg_t::sf_wifi_ssid_broadcast_t

  SSID broadcast flag. Valid in AP mode only.

  Enumerated as:

  – SF_WIFI_SSID_BROADCAST_DISABLE

  – SF_WIFI_SSID_BROADCAST_ENABLE
- sf_wifi_cfg_t::sf_wifi_access_control_t

  Mode of access control MAC list.

  Enumerated as:

  – SF_WIFI_ACCESS_CONTROL_DISABLE

  – SF_WIFI_ACCESS_CONTROL_DENY

  – SF_WIFI_ACCESS_CONTROL_ALLOW
- sf_wifi_cfg_t::beacon

  Beacon interval. Valid in AP mode only.
- sf_wifi_cfg_t::station_inactivity_timeout

  Station inactivity timeout value. Valid in AP mode only.
- sf_wifi_cfg_t::sf_wifi_wds_t

  WDS flag. Valid in AP mode only.

  Enumerated as:

  – SF_WIFI_WDS_DISABLE

  – SF_WIFI_WDS_ENABLE

- sf_wifi_cfg_t::p_buffer_pool_rx

  Pointer to Network stack Rx buffer pool.

- sf_wifi_cfg_t::sf_wifi_mandatory_high_throughput_t

  Only allow HT mode. Valid in AP mode only.

  Enumerated as:

  – SF_WIFI_MANDATORY_HIGH_THROUGHPUT_DISABLE

  – SF_WIFI_MANDATORY_HIGH_THROUGHPUT_ENABLE

- sf_wifi_cfg_t::p_callback

  Pointer to callback function.

- sf_wifi_cfg_t::p_context

  User defined context passed into callback function.

- sf_wifi_cfg_t::p_extend

  Instance specific configuration.

7.31.7.11 close

```
ssp_err_t(* sf_wifi_api_t::close) (sf_wifi_ctrl_t *const p_ctrl)
```

Brief description

De-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Detailed description

Table 291:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the control block for the WiFi module.

Parameter p_ctrl

Definition: sf_wifi_ctrl_t

WiFi Framework control structure

7.31.7.12 multicastListAdd

```
ssp_err_t(* sf_wifi_api_t::multicastListAdd) (sf_wifi_ctrl_t *const p_ctrl,
uint8_t const *const p_mac_addr)
```

Brief description

Add the given MAC address to the multicast filter list.

Detailed description

Table 292:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac_addr	in	Pointer to the Mac address.

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_mac_addr

`uint8_t`

7.31.7.13 multicastListDelete

```
ssp_err_t(* sf_wifi_api_t::multicastListDelete) (sf_wifi_ctrl_t *const p_ctrl,
uint8_t const *const p_mac_addr)
```

Detailed description

Delete the given MAC address from the multicast filter list.

Table 293:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac_addr	in	Pointer to the Mac address.

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_mac_addr

`uint8_t`

7.31.7.14 statisticsGet

```
ssp_err_t(* sf_wifi_api_t::statisticsGet) (sf_wifi_ctrl_t *const p_ctrl,
sf_wifi_stats_t *const p_wifi_device_stats)
```

Brief description

Get the interface statistics.

Detailed description

Table 294:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_wifi_device_stats	out	Pointer to the WiFi module data statistics.

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_wifi_device_stats

Definition: `sf_wifi_stats_t*const p_wifi_device_stats`

Define the statistic and error counters for this IP instance.

- `sf_wifi_stats_t::rx_pkts`

  Packets received successfully.

- `sf_wifi_stats_t::tx_pkts`

  Packets transmitted successfully.

- `sf_wifi_stats_t::tx_err`

  Transmit errors.

7.31.7.15 transmit

```
ssp_err_t(* sf_wifi_api_t::transmit) (sf_wifi_ctrl_t *const p_ctrl, uint8_t
*const p_buf, uint32_t length)
```

Brief description

Transmit data packet.

Detailed description

Table 295:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_buf	in	Pointer to transmit buffer
length	in	Transmit buffer length

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_buf

`uint8_t`

Parameter length

`uint32_t`

7.31.7.16 provisioningSet

```
ssp_err_t(* sf_wifi_api_t::provisioningSet) (sf_wifi_ctrl_t *const p_ctrl,
sf_wifi_provisioning_t const *const p_wifi_provisioning)
```

Brief description

Set WiFi module provisioning which will configure the module in AP or Client mode.

Detailed description

Table 296:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_wifi_provisioning	in	Pointer to WiFi provisioning structure

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_wifi_provisioning

Definition: `sf_wifi_provisioning_tconst *const p_wifi_provisioning`

WiFi Provisioning parameters

- `sf_wifi_provisioning_t::mode`

  Select AP or Client mode.
- `sf_wifi_provisioning_t::channel`

  RF Channel number.
- `sf_wifi_provisioning_t::ssid`

  SSID.
- `sf_wifi_provisioning_t::security`

  Security type.
- `sf_wifi_provisioning_t::encryption`

  Encryption type.
- `sf_wifi_provisioning_t::key`

  Pre-shared key.
- `sf_wifi_provisioning_t::p_callback`

  Pointer to AP Link UP/Down callback function.

7.31.7.17 provisioningGet

```
ssp_err_t(* sf_wifi_api_t::provisioningGet) (sf_wifi_ctrl_t *const p_ctrl,
sf_wifi_provisioning_t *const p_wifi_provisioning)
```

Brief description

Get the provisioning information of WiFi module.

Detailed description

Table 297:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_wifi_provisioning	out	Pointer to WiFi provisioning structure

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_wifi_provisioning

Definition: `sf_wifi_provisioning_t*const p_wifi_provisioning`

WiFi Provisioning parameters

- sf_wifi_provisioning_t::mode

  Select AP or Client mode.
- sf_wifi_provisioning_t::channel

  RF Channel number.
- sf_wifi_provisioning_t::ssid

  SSID.
- sf_wifi_provisioning_t::security

  Security type.
- sf_wifi_provisioning_t::encryption

  Encryption type.
- sf_wifi_provisioning_t::key

  Pre-shared key.
- sf_wifi_provisioning_t::p_callback

  Pointer to AP Link UP/Down callback function.

7.31.7.18 infoGet

```
ssp_err_t(* sf_wifi_api_t::infoGet) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_info_t
*const p_wifi_info)
```

Brief description

Get WiFi module information.

Detailed description

Table 298:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_wifi_info	out	Pointer to WiFi module information structure

Parameter p_ctrl

Definition: sf_wifi_ctrl_t

WiFi Framework control structure

Parameter p_wifi_info

Definition: sf_wifi_info_t*const p_wifi_info

Configuration about underlying device driver.

- `sf_wifi_info_t::p_chipset`

  Pointer to sting showing WiFi chipset/driver information.
- `sf_wifi_info_t::rssi`

  Received signal strength indication.
- `sf_wifi_info_t::noise_level`

  Signal to noise ratio.
- `sf_wifi_info_t::link_quality`

  Signal strength / quality.

7.31.7.19 scan

```
ssp_err_t(* sf_wifi_api_t::scan) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_scan_t
*const p_scan, uint8_t *const p_cnt)
```

Brief description

Scan for WiFi SSIDs.

Detailed description

Table 299:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_scan	out	Pointer to structure which will hold scan result. It is the responsibility of the caller to ensure that adequate space is available to hold scan results.
p_cnt	inout	Pointer to variable, specifying maximum number of SSID's to scan and will be updated to number of actual SSIDs scanned by device

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_scan

Definition: `sf_wifi_scan_t*const p_scan`

Define the structure to store the SSID scan information

- sf_wifi_scan_t::hw_mode

  Hardware mode 802.11a/b/g/n.
- sf_wifi_scan_t::rssi

  Signal Strength.
- sf_wifi_scan_t::ssid

  SSID name.
- sf_wifi_scan_t::bssid

  Basic Service Set Identification (i.e. MAC address of Access Point)
- sf_wifi_scan_t::channel

  Radio channel that the AP beacon was received on.
- sf_wifi_scan_t::security

  Security type.
- sf_wifi_scan_t::encryption

  Encryption type.
- sf_wifi_scan_t::bss_type

  Network type.

Parameter p_cnt

uint8_t

7.31.7.20 ACLAdd

```
ssp_err_t(* sf_wifi_api_t::ACLAdd) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const
*const p_mac)
```

Brief description

Adds a MAC address to the Access Control List. Valid in AP mode only.

Detailed description

Table 300:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac	in	Pointer to MAC address

Parameter p_ctrl

Definition: sf_wifi_ctrl_t

WiFi Framework control structure

Parameter p_mac

`uint8_t`

7.31.7.21 ACLDelete

```
ssp_err_t(* sf_wifi_api_t::ACLDelete) (sf_wifi_ctrl_t *const p_ctrl, uint8_t
const *const p_mac)
```

Brief description

Deletes a MAC address from Access Control List. Valid in AP mode only.

Detailed description

Table 301:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac	in	Pointer to MAC address

Parameter p_ctrl

Definition: `sf_wifi_ctrl_t`

WiFi Framework control structure

Parameter p_mac

`uint8_t`

7.31.7.22 macAddressGet

```
ssp_err_t(* sf_wifi_api_t::macAddressGet) (sf_wifi_ctrl_t *const p_ctrl, uint8_t
*const p_mac)
```

Brief description

Get WiFi module MAC address.

Detailed description

Table 302:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac	out	Pointer to MAC address

Parameter p_ctrl

Definition: sf_wifi_ctrl_t

WiFi Framework control structure

Parameter p_mac

uint8_t

7.31.7.23 macAddressSet

```
ssp_err_t(* sf_wifi_api_t::macAddressSet) (sf_wifi_ctrl_t *const p_ctrl, uint8_t
const *const p_mac)
```

Brief description

Set WiFi module MAC address.

Detailed description

Table 303:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the WiFi module.
p_mac	in	Pointer to MAC address

Parameter p_ctrl

Definition: sf_wifi_ctrl_t

WiFi Framework control structure

Parameter p_mac

uint8_t

7.31.7.24 versionGet

```
ssp_err_t(* sf_wifi_api_t::versionGet) (ssp_version_t *const p_version)
```

Brief description

Gets version and stores it in provided pointer p_version.

Detailed description

Table 304:Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

7.31.7.25 sf_wifi_instance_t

sf_wifi_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sf_wifi_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_wifi_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_wifi_api_t const * p_api

  Pointer to the API structure for this instance.

7.32 SF WIFI NSAL Interface

RTOS-integrated SF WIFI NSAL Framework Interface.

7.32.1 Summary

This SSP Interface provides access to the ThreadX-aware SF WIFI NSAL Framework.

7.32.2 Data structures

- sf_wifi_nsal_cfg_t
- sf_wifi_nsal_callback_args_t

7.32.3 Enumerations

- sf_wifi_nsal_zero_copy_t

7.32.4 Defines

- #define SF_WIFI_NSAL_MAC_CHANGED

  Initial value: `(NX_LINK_USER_COMMAND + 1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

7.32.5 API Data

7.32.5.1 sf_wifi_nsal_zero_copy_t

`sf_wifi_nsal_zero_copy_t`

Detailed description

Zero Copy Configuration Enumeration

Enumerated values

Name	Description
SF_WIFI_NSAL_ZERO_COPY_DISABLE	Zero copy is disabled.
SF_WIFI_NSAL_ZERO_COPY_ENABLE	Zero copy is enabled.

7.32.6 API Structures

7.32.6.1 sf_wifi_nsal_cfg_t

sf_wifi_nsal_cfg_t

Detailed description

Define the NSAL configuration parameters

Variables

- sf_wifi_nsal_zero_copy_t tx_zero_copy

  Transmit path zero copy support.

- sf_wifi_nsal_zero_copy_t rx_zero_copy

  Receive path zero copy support.

- uint8_t * p_tx_packet_buffer

  Pointer to Tx buffer used to consolidate data from chained NetX packets.

7.32.6.2 sf_wifi_nsal_callback_args_t

sf_wifi_nsal_callback_args_t

Detailed description

Define the NSAL callback arguments

Variables

- NX_IP * p_ip

  Pointer to NetX IP interface.

- sf_wifi_nsal_cfg_t * p_wifi_nsal_cfg
  pointer to NSAL configuration

7.33 SF WIFI On-Chip Stack Interface

RTOS-integrated SF WIFI On-Chip Stack Interface.

7.33.1 Summary

This SSP Interface provides access to the ThreadX-aware SF WIFI Framework.

7.33.2 Interface API

sf_wifi_onchip_stack_api_t

Function name	Description
.open	Pointer to function which initializes the network interface for data transfersInitial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.
.close	Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.
.ipAddressCfg	Configures IP address of the interface.
.dhcpServerStart	Starts DHCP server on the interface.
.dhcpServerStop	Stop DHCP server on the interface.
.versionGet	Gets version and stores it in provided pointer p_version.

7.33.3 Data structures

- sf_wifi_onchip_stack_ip_cfg_t
- sf_wifi_onchip_stack_cfg_t
- sf_wifi_onchip_stack_ctrl_t
- sf_wifi_onchip_stack_instance_t

7.33.4 Enumerations

- sf_wifi_onchip_stack_ip_addr_mode_t

7.33.5 Defines

- #define SF_WIFI_ONCHIP_STACK_API_VER_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Major Version of the API defined in this file
- #define SF_WIFI_ONCHIP_STACK_API_VER_MINOR

  Initial value:`(0U)`

  Minor Version of the API defined in this file

7.33.6 API Data

7.33.6.1 sf_wifi_onchip_stack_ip_addr_mode_t

`sf_wifi_onchip_stack_ip_addr_mode_t`

Detailed description

IP addressing modes

Enumerated values

Name	Description
SF_WIFI_IP_ADDR_GET	Read the IP address assigned to interface.
SF_WIFI_IP_ADDR_STATIC	Statically configure the IP address.
SF_WIFI_IP_ADDR_DHCP	Get the IP address from DHCP server, dynamic assignment.

7.33.7 API Structures

7.33.7.1 sf_wifi_onchip_stack_ip_cfg_t

sf_wifi_onchip_stack_ip_cfg_t

Detailed description

Define IP Interface configuration information

Variables

- sf_wifi_onchip_stack_ip_addr_mode_t ip_addr_mode

  Addressing mode.
- sf_wifi_ip_addr_t ip_addr

  Interface IP address.
- sf_wifi_ip_addr_t netmask

  Interface netmask.
- sf_wifi_ip_addr_t gateway

  Interface Gateway.

7.33.7.2 sf_wifi_onchip_stack_cfg_t

sf_wifi_onchip_stack_cfg_t

Detailed description

Define the WiFi configuration parameters

Variables

- sf_wifi_instance_t const * p_lower_lvl_wifi

  Pointer to SF WiFi instance.
- void * p_extend

  Extended configuration.

7.33.7.3 sf_wifi_onchip_stack_ctrl_t

sf_wifi_onchip_stack_ctrl_t

Detailed description

WiFi Framework control structure

Variables

- sf_wifi_instance_t * p_lower_lvl_wifi

  Pointer to SF WiFi instance.

  Storage for information needed for each WiFi device driver in the system.

7.33.7.4 sf_wifi_onchip_stack_api_t

sf_wifi_onchip_stack_api_t

Detailed description

Framework API structure. Implementations will use the following API.

7.33.7.5 open

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::open) (sf_wifi_onchip_stack_ctrl_t
*p_ctrl, sf_wifi_onchip_stack_cfg_t const *const p_cfg)
```

Detailed description

Pointer to function which initializes the network interface for data transfers

Initial driver configuration, enable the driver link, enable interrupts and make device ready for data transfer.

Table 305:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to configuration structure.

Parameter p_ctrl

Definition: `sf_wifi_onchip_stack_ctrl_t`

WiFi Framework control structure

Parameter p_cfg

Definition: `sf_wifi_onchip_stack_cfg_t const *const p_cfg`

Define the WiFi configuration parameters

- `sf_wifi_onchip_stack_cfg_t`::`sf_wifi_instance_t`

  Pointer to SF WiFi instance.

- `sf_wifi_onchip_stack_cfg_t`::`p_extend`

  Extended configuration.

7.33.7.6 close

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::close) (sf_wifi_onchip_stack_ctrl_t
*const p_ctrl)
```

Detailed description

Pointer to function which un-initialize the network interface and may put it in low power mode or power it off. Close the driver, disable the driver link, disable interrupt.

Table 306:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the control block

Parameter p_ctrl

Definition: sf_wifi_onchip_stack_ctrl_t

WiFi Framework control structure

7.33.7.7 ipAddressCfg

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::ipAddressCfg)
(sf_wifi_onchip_stack_ctrl_t *const p_ctrl, sf_wifi_onchip_stack_ip_cfg_t *const
p_cfg)
```

Detailed description

Configures IP address of the interface.

Table 307:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block
p_cfg	inout	Pointer to IP configuration structure.

Parameter p_ctrl

Definition: sf_wifi_onchip_stack_ctrl_t

WiFi Framework control structure

Parameter p_cfg

7.33.7.8 dhcpServerStart

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::dhcpServerStart)
(sf_wifi_onchip_stack_ctrl_t *const p_ctrl, sf_wifi_ip_addr_t const *const
p_start_ip, sf_wifi_ip_addr_t const *const p_end_ip)
```

Detailed description

Starts DHCP server on the interface.

Table 308:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block
p_start_ip	in	Pointer to start IP address
p_end_ip	in	Pointer to end IP address

Parameter p_ctrl

Definition: sf_wifi_onchip_stack_ctrl_t

WiFi Framework control structure

Parameter p_start_ip

Parameter p_end_ip

7.33.7.9 dhcpServerStop

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::dhcpServerStop)
(sf_wifi_onchip_stack_ctrl_t *const p_ctrl)
```

Detailed description

Stop DHCP server on the interface.

Table 309:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block

Parameter p_ctrl

Definition: sf_wifi_onchip_stack_ctrl_t

WiFi Framework control structure

7.33.7.10 versionGet

```
ssp_err_t(* sf_wifi_onchip_stack_api_t::versionGet) (ssp_version_t *const
p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 310:Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

7.33.7.11 sf_wifi_onchip_stack_instance_t

sf_wifi_onchip_stack_instance_t

Detailed description

SF WiFi On Chip Stack Instance structure

Variables

- sf_wifi_onchip_stack_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sf_wifi_onchip_stack_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sf_wifi_onchip_stack_api_t const * p_api

  Pointer to the API structure for this instance.

7.34 SF WIFI NSAL on NetX

NetX NSAL interface implementation header file.

7.34.1 Summary

This SSP Interface provides access to the ThreadX-aware SF WIFI NSAL NX Framework.

7.34.2 Functions

- nsal_netx_driver

7.34.3 nsal_netx_driver

```
 nsal_netx_driver ( NX_IP_DRIVER  * driver ,  sf_wifi_instance_t  const
* p_wifi_instance ,  sf_wifi_nsal_cfg_t  * p_wifi_nsal_cfg )
```

7.34.3.1 Detailed description

NSAL NetX Driver Entry function

7.34.3.2 Function steps

- Check if the link is up

Chapter 8 API Reference: Framework

8.1 API Reference: Framework Layer

The framework layer provides RTOS aware drivers for functional use cases.

- SF_EL_UX Framework
- SF_EL_UX API Framework
- ADC periodic Framework
- Audio Framework
- DAC Audio Playback Framework
- I2S Audio Playback Framework
- ADC Audio recording Framework
- BLOCK_MEDIA_SDMMC
- Cellular NSAL Implementation on NetX
- Console Framework
- FX_IO Framework
- GUIX Synergy Port
- Telnet Communication Framework
- USB Communication Framework
- USB Communication Framework V2
- External IRQ Framework
- I2C Framework
- JPEG Framework
- Messaging Framework
- Power Profiles Framework
- SPI Framework
- Thread Monitor Framework
- CTSU Framework
- CTSU Button Framework
- CTSU Slider Framework
- I2C Touch Panel Framework

- UART Framework Instance
- NetX Synergy Port
- NetX Synergy Port PHY Driver
- Power Profiles V2 Framework

8.2 ADC periodic Framework

RTOS-integrated ADC Framework.

8.2.1 Functions

- SF_ADC_PERIODIC_Open
- SF_ADC_PERIODIC_Start
- SF_ADC_PERIODIC_Stop
- SF_ADC_PERIODIC_Close
- SF_ADC_PERIODIC_VersionGet

8.2.2 Defines

- #define SF_ADC_PERIODIC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_ADC_PERIODIC_CODE_VERSION_MINOR

  Initial value:`(4U)`

8.2.3 SF_ADC_PERIODIC_Open

```
ssp_err_t SF_ADC_PERIODIC_Open ( sf_adc_periodic_ctrl_t  *const p_api_ctrl ,
 sf_adc_periodic_cfg_t  const *const p_cfg )
```

8.2.3.1 Brief description

Configures periodic ADC framework and optionally starts the timer.

8.2.3.2 Detailed description

The SF_ADC_PERIODIC_Open function acquires a mutex for the ADC Unit used, then calls the driver .open function in the p_api parameter. The mutex is released following the driver layer open function.

Table 311:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_api_ctrl or p_cfg. See HAL driver for other possible causes.
SSP_ERR_INVALID_ARGUMENT	An invalid argument is used
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.2.3.3 Function steps

- Save driver structure pointer for use in other framework layer functions
- Create a mutex
- Get mutex before calling lower layer, which will access HW registers.
- Initialize the HAL layer
- Post the Mutex
- If any of the HAL layer initializations failed, then delete the mutex and exit the function with the error code
- Delete the mutex
- If there was a mutex error, return it
- Mark control block open so other tasks know it is valid

8.2.4 SF_ADC_PERIODIC_Start

```
ssp_err_t SF_ADC_PERIODIC_Start ( sf_adc_periodic_ctrl_t  *const p_api_ctrl )
```

8.2.4.1 Brief description

Gets mutex, starts the periodic ADC scan, and releases mutex.

8.2.4.2 Detailed description

ATTENTION: The driver will enable the ADC to be triggered via timer event; there will be a time delay from the time this function is called to the time the hardware timer count expires and triggers the scan.

Table 312:Return values

Name	Description
SSP_SUCCESS	ADC Periodic Scan started successfully.
SSP_ERR_ASSERTION	One or more pointers point to NULL.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_ADC_PERIODIC_Open to configure.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.
SSP_ERR_IN_USE	The module is currently busy performing another operation

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls:

- scanStart
- start

8.2.4.3 Function steps

- Get mutex, start timer, then release mutex
- Enable the ADC to receive hardware triggers
- If the scan was successfully enabled in the ADC HAL,
- Start the timer to generate the ADC trigger events
- Return the mutex
- If there was a mutex error, return it

8.2.5 SF_ADC_PERIODIC_Stop

```
ssp_err_t SF_ADC_PERIODIC_Stop ( sf_adc_periodic_ctrl_t  *const p_api_ctrl )
```

8.2.5.1 Brief description

Gets mutex, stops the periodic ADC scan, and releases mutex.

8.2.5.2 Detailed description

Table 313:Return values

Name	Description
SSP_SUCCESS	Periodic ADC scan stopped successfully.
SSP_ERR_ASSERTION	One or more pointers point to NULL..
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_ADC_PERIODIC_Open to configure.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.
SSP_ERR_IN_USE	The module is currently busy performing another operation

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls:

- stop

8.2.5.3 Function steps

- Get mutex, stop timer, then release mutex
- If there was a mutex error, return it

8.2.6 SF_ADC_PERIODIC_Close

```
ssp_err_t SF_ADC_PERIODIC_Close ( sf_adc_periodic_ctrl_t  *const p_api_ctrl )
```

8.2.6.1 Brief description

The close function acquires the unit's mutex, closes all lower level drivers, releases and deletes the mutex.

8.2.6.2 Detailed description

Table 314:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	One or more pointers point to NULL.

Table 314:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_ADC_PERIODIC_Open to configure.
SSP_ERR_IN_USE	The module is currently busy performing another operation
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.2.6.3 Function steps

- Get mutex since this will access hardware registers
- Close the HAL layer modules
- Post the mutex
- If all the HAL layers closed successfully, then delete the mutex and mark module as un-initialized
- Delete RTOS services used
- Clear information from control block so other functions know this instance is closed
- If there was a mutex error, return it

8.2.7 SF_ADC_PERIODIC_VersionGet

```
ssp_err_t SF_ADC_PERIODIC_VersionGet ( ssp_version_t  *const p_version )
```

8.2.7.1 Brief description

Gets version and stores it in provided pointer p_version.

8.2.7.2 Detailed description

Table 315:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.2.8 Extensions

8.2.8.1 sf_adc_periodic_instance_ctrl_t

sf_adc_periodic_instance_ctrl_t

Detailed description

Channel instance control block. DO NOT INITIALIZE. Initialization occurs when SF_ADC_PERIODIC_Open is called

Variables

- uint32_t open

  Used by driver to check if pointer to control block is valid.
- TX_MUTEX mutex

  Mutex used to protect access to lower level driver hardware registers.
- adc_instance_t const * p_lower_lvl_adc

  Pointer to the ADC instance.
- timer_instance_t const * p_lower_lvl_timer

  Pointer to the Timer instance.
- transfer_instance_t const * p_lower_lvl_transfer

  Pointer to the Transfer instance.
- void const *volatile p_src_transfer

  Source pointer for the low level transfer method.
- adc_data_size_t * p_data_buffer

  Pointer to the buffer that will store the samples.
- uint32_t data_buffer_length

  Length of the data buffer that will store the samples.
- uint32_t data_buffer_index

  Index of the data buffer where data is to be written to next.
- uint32_t sample_count

  Samples per channel to be buffered before notifying the app.
- uint32_t dtc_transfer_length

  Total Length of DTC transfer for requested number of samples.
- void(* p_callback)(*p_args)

  Callback function.
- void const * p_context

  Placeholder for user data.

8.3 Audio Framework

RTOS-integrated Audio Framework.

8.3.1 Summary

This module is a ThreadX-aware Audio Framework. The module implements Audio Framework Interface.

Name of module used by error logger macro

8.3.2 Functions

- SF_AUDIO_PLAYBACK_Open
- SF_AUDIO_PLAYBACK_Close
- SF_AUDIO_PLAYBACK_Start
- SF_AUDIO_PLAYBACK_Pause
- SF_AUDIO_PLAYBACK_Stop
- SF_AUDIO_PLAYBACK_Resume
- SF_AUDIO_PLAYBACK_VolumeSet
- SF_AUDIO_PLAYBACK_VersionGet

8.3.3 Defines

- #define SF_AUDIO_PLAYBACK_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  Version of code that implements the API defined in this file
- #define SF_AUDIO_PLAYBACK_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_AUDIO_PLAYBACK_STACK_SIZE

  Initial value:`(SF_AUDIO_PLAYBACK_CFG_THREAD_STACK_SIZE)`

  Audio playback internal thread stack size. Varies by application, but rarely requires more than 256 bytes.

8.3.4 SF_AUDIO_PLAYBACK_Open

```
ssp_err_t SF_AUDIO_PLAYBACK_Open ( sf_audio_playback_ctrl_t  *const p_api_ctrl ,
 sf_audio_playback_cfg_t  const *const p_cfg )
```

8.3.4.1 Detailed description

Implements open.

Table 316:Return values

Name	Description
SSP_SUCCESS	Audio hardware successfully configured.
SSP_ERR_ASSERTION	A pointer is NULL or a parameter is invalid.
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is not reentrant.

8.3.4.2 Function steps

- Open hardware if it is not already open.
- Create event flags to notify playback thread when playback of a buffer is complete.
- Store stream pointer in common control block.
- Initialize stream variables.
- Store queue pointers for audio data and clear stream owner pointers.
- Mark stream opened so it can be used by other API's.

8.3.5 SF_AUDIO_PLAYBACK_Close

```
ssp_err_t SF_AUDIO_PLAYBACK_Close ( sf_audio_playback_ctrl_t
*const p_api_ctrl )
```

8.3.5.1 Detailed description

Implements close.

Table 317:Return values

Name	Description
SSP_SUCCESS	Audio instance successfully closed

Table 317:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	p_ctrl is NULL
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is not reentrant.

8.3.5.2 Function steps

- Mark stream as unused in hardware control block and determine if all streams are closed.
- Mark hardware control block as unused so it can be reconfigured.
- Close lower level drivers
- Delete RTOS services used
- Mark control block as unused so it can be reconfigured.

8.3.6 SF_AUDIO_PLAYBACK_Start

```
ssp_err_t SF_AUDIO_PLAYBACK_Start ( sf_audio_playback_ctrl_t
*const p_api_ctrl ,  sf_audio_playback_data_t  *const p_data ,   UINT
const timeout )
```

8.3.6.1 Detailed description

Implements start.

Table 318:Return values

Name	Description
SSP_SUCCESS	Buffer successfully sent to audio playback thread
SSP_ERR_ASSERTION	p_ctrl is NULL
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

NOTE: This function is reentrant.

8.3.6.2 Function steps

- Ensure that audio data is only posted from the current stream owner.

- Store new stream owner if stream is unowned.
- Set message header to audio start event. Set instance to stream instance.
- Send message with audio data to audio thread.

8.3.7 SF_AUDIO_PLAYBACK_Pause

```
ssp_err_t SF_AUDIO_PLAYBACK_Pause ( sf_audio_playback_ctrl_t
*const p_api_ctrl )
```

8.3.7.1 Detailed description

Implements pause.

Table 319:Return values

Name	Description
SSP_SUCCESS	Audio playback pause message sent to audio playback thread for this stream.
SSP_ERR_ASSERTION	p_ctrl is NULL
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant.

8.3.8 SF_AUDIO_PLAYBACK_Stop

```
ssp_err_t SF_AUDIO_PLAYBACK_Stop ( sf_audio_playback_ctrl_t  *const p_api_ctrl )
```

8.3.8.1 Detailed description

Implements stop.

Table 320:Return values

Name	Description
SSP_SUCCESS	Audio playback stop message sent to audio playback thread for this stream.
SSP_ERR_ASSERTION	p_ctrl is NULL

Table 320:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant.

8.3.9 SF_AUDIO_PLAYBACK_Resume

```
ssp_err_t SF_AUDIO_PLAYBACK_Resume ( sf_audio_playback_ctrl_t
*const p_api_ctrl )
```

8.3.9.1 Detailed description

Implements resume.

Table 321:Return values

Name	Description
SSP_SUCCESS	Audio playback resume message sent to audio playback thread for this stream.
SSP_ERR_ASSERTION	p_ctrl is NULL
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

NOTE: This function is reentrant.

8.3.10 SF_AUDIO_PLAYBACK_VolumeSet

```
ssp_err_t SF_AUDIO_PLAYBACK_VolumeSet ( sf_audio_playback_ctrl_t
*const p_api_ctrl ,   uint8_t const volume )
```

8.3.10.1 Detailed description

Implements volumeSet.

Table 322:Return values

Name	Description
SSP_SUCCESS	Audio playback software volume level updated (applies to all streams).
SSP_ERR_ASSERTION	p_ctrl is NULL
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

8.3.10.2 Function steps

- Update volume in control block.

8.3.11 SF_AUDIO_PLAYBACK_VersionGet

```
ssp_err_t SF_AUDIO_PLAYBACK_VersionGet ( ssp_version_t  *const p_version )
```

8.3.11.1 Detailed description

Implements versionGet.

Table 323:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.3.12 Extensions

8.3.12.1 sf_audio_playback_common_instance_ctrl_t

sf_audio_playback_common_instance_ctrl_t

Detailed description

Audio common instance control block. DO NOT INITIALIZE. Initialization the first time open is called. Shared by all streams.

Variables

- uint32_t open

  Used to determine if driver is initialized.

- void const * p_next_buffer

  Pointer to next buffer (to be played when the current buffer completes).
- uint32_t next_length

  Length of next buffer (to be played when the current buffer completes).
- sf_message_instance_t const * p_message

  Pointer to message control block.
- TX_QUEUE * p_queue

  Queue subscribed to SF_MESSAGE_EVENT_CLASS_AUDIO events.
- sf_audio_playback_hw_instance_t const * p_lower_lvl_hw

  Hardware API's used.
- sf_audio_playback_instance_ctrl_t * p_stream[SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS]

  Stream specific data.
- TX_THREAD thread

  Main audio thread.
- TX_EVENT_FLAGS_GROUP flags

  Event flags used to end wait in audio thread.
- sf_audio_playback_data_type_t data_type

  Sample format required by the hardware.
- uint8_t volume

  Volume from 0 (muted) to 255 (maximum, default on open).
- uint8_t buffer_index

  Which ping pong buffer to use.
- uint8_t stack[SF_AUDIO_PLAYBACK_STACK_SIZE]

  Stack for audio thread.
- int16_t samples[2][SF_AUDIO_PLAYBACK_CFG_BUFFER_SIZE_BYTES/sizeof(int16_t)]

  Ping pong buffers, used to store converted data during transfer.
- bool playing

  State of audio instance (currently playing if true)

8.3.12.2 st_sf_audio_playback_instance_ctrl

st_sf_audio_playback_instance_ctrl

Detailed description

Audio stream instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- uint32_t open

  Used to determine if driver is initialized.

- TX_THREAD * p_owner

  Pointer to thread that began the stream at this index. Used to ensure multiple threads don't interleave data on the same stream.

- void(* p_callback)(*p_args)

  Callback called when playback of a buffer passed to sf_audio_playback_api_t::start is complete.

- uint8_t class_instance

  Class instance used to identify the stream to the messaging framework.

- uint32_t samples_remaining

  Internal state of data samples remaining for this stream.

- uint32_t index

  Internal state of current data index for this stream.

- uint32_t end

  Used to track completion of looped playback.

- sf_audio_playback_data_t * p_data[2]

  Audio data read from queue.

- sf_audio_playback_status_t status

  Status of current stream.

- sf_audio_playback_common_instance_ctrl_t * p_common_ctrl

  Pointer to the hardware control block used by this stream.

8.4 DAC Audio Playback Framework

RTOS-integrated DAC implementation of Audio Playback Interface.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

The Audio Playback Framework DAC implementation uses a timer to generate events at the sampling frequency, and uses these events to transfer PCM samples to the DAC.

8.4.1 Functions

- SF_AUDIO_PLAYBACK_HW_DAC_Open
- SF_AUDIO_PLAYBACK_HW_DAC_Start
- SF_AUDIO_PLAYBACK_HW_DAC_Stop
- SF_AUDIO_PLAYBACK_HW_DAC_Play

- SF_AUDIO_PLAYBACK_HW_DAC_DataTypeGet
- SF_AUDIO_PLAYBACK_HW_DAC_Close
- SF_AUDIO_PLAYBACK_HW_DAC_VersionGet
- sf_audio_playback_hw_callback_timer
- sf_audio_playback_hw_callback_common

8.4.2 Variables

- g_sf_audio_playback_hw_on_sf_audio_playback_hw_dac

8.4.3 Defines

- #define SF_AUDIO_PLAYBACK_HW_DAC_CODE_VERSION_MAJOR
  Initial value:`(1U)`
- #define SF_AUDIO_PLAYBACK_HW_DAC_CODE_VERSION_MINOR
  Initial value:`(5U)`

8.4.4 SF_AUDIO_PLAYBACK_HW_DAC_Open

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_Open ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl ,  sf_audio_playback_hw_cfg_t  const *const p_cfg )
```

8.4.4.1 Detailed description

Open the DAC audio driver, including the DAC HAL driver and helper timer and transfer HAL drivers.

Table 324:Return values

Name	Description
SSP_SUCCESS	Configuration of lower level drivers completed successfully.
SSP_ERR_ASSERTION	Null Pointer.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- open
- open
- open
- close
- close

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.4.4.2 Function steps

- Open Timer driver at selected frequency
- If DTC is selected, register the audio callback with the timer ISR. DTC calls activation source ISR when the transfer is complete.
- Open DAC module
- Open transfer module to transfer from buffer to DAC output register
- Open transfer driver
- Store driver data.
- Play linear ramp data to get DAC up to half the maximum output.

8.4.5 SF_AUDIO_PLAYBACK_HW_DAC_Start

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_Start ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl )
```

8.4.5.1 Detailed description

Start the DAC and timer HAL drivers.

Table 325:Return values

Name	Description
SSP_SUCCESS	Audio playback hardware started successfully.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- start
- start

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.4.5.2 Function steps

- Start DAC.
- Start timer.

8.4.6 SF_AUDIO_PLAYBACK_HW_DAC_Stop

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_Stop ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl )
```

8.4.6.1 Detailed description

Stop the DAC and timer HAL drivers.

Table 326:Return values

Name	Description
SSP_SUCCESS	Audio playback hardware stopped successfully.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- stop
- stop

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.4.6.2 Function steps

- Stop timer.
- Stop DAC.

8.4.7 SF_AUDIO_PLAYBACK_HW_DAC_Play

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_Play ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl ,   int16_t const *const p_buffer ,   uint32_t length )
```

8.4.7.1 Detailed description

Play a single audio buffer by input samples to the DAC at the sampling frequency configured by the timer.

Table 327:Return values

Name	Description
SSP_SUCCESS	Buffer playback began successfully.

Table 327:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	The parameter p_ctrl or p_buffer is NULL or length is less than 0x10000UL.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- reset

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.4.7.2 Function steps

- Reset transfer.

8.4.8 SF_AUDIO_PLAYBACK_HW_DAC_DataTypeGet

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_DataTypeGet ( sf_audio_playback_hw_ctrl_t
*const p_ctrl ,  sf_audio_playback_data_type_t  *const p_data_type )
```

8.4.8.1 Detailed description

Provides the expected data type in the pointer p_data_type.

Table 328:Return values

Name	Description
SSP_SUCCESS	Data type stored in p_data_type.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_data_type is NULL.

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.4.8.2 Function steps

- Store data type. The Synergy DAC supports only 12-bit unsigned data.

8.4.9 SF_AUDIO_PLAYBACK_HW_DAC_Close

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_Close ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl )
```

8.4.9.1 Detailed description

Close open audio driver.

Table 329:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- close
- close
- close

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.4.9.2 Function steps

- Close timer driver.
- Close DAC driver.
- Close transfer driver.

8.4.10 SF_AUDIO_PLAYBACK_HW_DAC_VersionGet

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_DAC_VersionGet ( ssp_version_t
*const p_version )
```

8.4.10.1 Detailed description

Stores the version of the firmware and API in provided pointer p_version.

Table 330:Return values

Name	Description
SSP_ERR_ASSERTION	The parameter p_version is NULL.
SSP_SUCCESS	Module version successfully stored in p_version.

NOTE: This function is reentrant.

8.4.11 sf_audio_playback_hw_callback_timer

```
sf_audio_playback_hw_callback_timer ( timer_callback_args_t  * p_args )
```

8.4.11.1 Detailed description

Callback function intercepted from timer driver.

Table 331:Parameters

Name	Direction	Description
p_args	in	Callback data can be used to identify what triggered the interrupt.

8.4.12 sf_audio_playback_hw_callback_common

```
sf_audio_playback_hw_callback_common (  void const * p_context )
```

8.4.12.1 Detailed description

Callback function intercepted from HAL layer.

Table 332:Parameters

Name	Direction	Description
p_context	in	Pointer to control block, used to identify what triggered the interrupt.

8.4.12.2 Function steps

- Recover context from ISR.
- Create callback arguments.
- Call user callback if not NULL.

8.4.13 g_sf_audio_playback_hw_on_sf_audio_playback_hw_dac

```
sf_audio_playback_hw_api_t::g_sf_audio_playback_hw_on_sf_audio_playback_hw_dac
```

8.4.13.1 Detailed description

Function pointers for DAC implementation of audio playback API.

8.4.14 Extensions

8.4.14.1 sf_audio_playback_hw_dac_instance_ctrl_t

sf_audio_playback_hw_dac_instance_ctrl_t

Detailed description

Hardware dependent control block for DAC audio driver.

Variables

- void(* p_callback)(*p_args)

  Callback called when play is complete.

- void * p_context

  Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

- dac_instance_t const * p_lower_lvl_dac

  DAC API used to access DAC hardware.

- timer_instance_t const * p_lower_lvl_timer

  Timer API used to generate sampling frequency.

- transfer_instance_t const * p_lower_lvl_transfer

  Transfer API used to transfer data each sampling frequency.

8.4.14.2 sf_audio_playback_hw_dac_cfg_t

sf_audio_playback_hw_dac_cfg_t

Detailed description

Hardware dependent configuration for DAC audio driver.

Variables

- dac_instance_t const * p_lower_lvl_dac

  DAC API used to access DAC hardware.

- timer_instance_t const * p_lower_lvl_timer

  Timer API used to generate sampling frequency.

- transfer_instance_t const * p_lower_lvl_transfer

  Transfer API used to transfer data each sampling frequency.

8.5 I2S Audio Playback Framework

RTOS-integrated I2S implementation of Audio Playback Interface.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

The Audio Playback Framework I2S implementation uses the I2S interface for audio playback.

Name of module used by error logger macro

8.5.1 Functions

- SF_AUDIO_PLAYBACK_HW_I2S_Open
- SF_AUDIO_PLAYBACK_HW_I2S_Start
- SF_AUDIO_PLAYBACK_HW_I2S_Stop
- SF_AUDIO_PLAYBACK_HW_I2S_Play
- SF_AUDIO_PLAYBACK_HW_I2S_DataTypeGet
- SF_AUDIO_PLAYBACK_HW_I2S_Close
- SF_AUDIO_PLAYBACK_HW_I2S_VersionGet

8.5.2 Defines

- #define SF_AUDIO_PLAYBACK_HW_I2S_CODE_VERSION_MAJOR
  Initial value:`(1U)`
- #define SF_AUDIO_PLAYBACK_HW_I2S_CODE_VERSION_MINOR
  Initial value:`(4U)`

8.5.3 SF_AUDIO_PLAYBACK_HW_I2S_Open

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_Open ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl ,  sf_audio_playback_hw_cfg_t  const *const p_cfg )
```

8.5.3.1 Brief description

Open the I2S audio driver, including the I2S HAL driver and helper timer and transfer HAL drivers.

8.5.3.2 Detailed description

Table 333:Return values

Name	Description
SSP_SUCCESS	Configuration of lower level drivers completed successfully.

Table 333:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the following parameter is null: p_ctrl or p_cfg or p_cfg_extend->p_lower_lvl_i2s or p_cfg_extend->p_lower_lvl_i2s->p_api.

See Common Error Codes and lower level driver function for other possible return codes. This function calls:

- open

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.5.3.3 Function steps

- Open I2S module
- Store driver data.

8.5.4 SF_AUDIO_PLAYBACK_HW_I2S_Start

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_Start ( sf_audio_playback_hw_ctrl_t
*const p_ctrl )
```

8.5.4.1 Brief description

Start the I2S and timer HAL drivers.

8.5.4.2 Detailed description

Table 334:Return values

Name	Description
SSP_SUCCESS	Audio playback hardware started successfully.

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.5.4.3 Function steps

- This API is not used - I2S is started when write is called from SF_AUDIO_PLAYBACK_HW_I2S_Play.

8.5.5 SF_AUDIO_PLAYBACK_HW_I2S_Stop

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_Stop ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl )
```

8.5.5.1 Brief description

Stop the I2S and timer HAL drivers.

8.5.5.2 Detailed description

Table 335:Return values

Name	Description
SSP_SUCCESS	Audio playback hardware stopped successfully.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

See Common Error Codes and lower level driver function for other possible return codes. This function calls:

- stop

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.5.5.3 Function steps

- Stop I2S.

8.5.6 SF_AUDIO_PLAYBACK_HW_I2S_Play

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_Play ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl ,   int16_t const *const p_buffer ,   uint32_t length )
```

8.5.6.1 Brief description

Play a single audio buffer by input samples to the I2S at the sampling frequency configured by the timer.

8.5.6.2 Detailed description

Table 336:Return values

Name	Description
SSP_SUCCESS	Buffer playback began successfully.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_buffer is NULL or length is less than 0x10000U.

See Common Error Codes and lower level driver function for other possible return codes. This function calls:

- write

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.5.6.3 Function steps

- Reset transfer.

8.5.7 SF_AUDIO_PLAYBACK_HW_I2S_DataTypeGet

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_DataTypeGet ( sf_audio_playback_hw_ctrl_t
*const p_ctrl ,  sf_audio_playback_data_type_t  *const p_data_type )
```

8.5.7.1 Brief description

Provides the expected data type in the pointer p_data_type.

8.5.7.2 Detailed description

Table 337:Return values

Name	Description
SSP_SUCCESS	Data type stored in p_data_type.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_data_type is NULL.

NOTE: This function is reentrant if the lower level driver functions are reentrant.

8.5.7.3 Function steps

- Store data type. The audio framework supports only 16-bit signed data.

8.5.8 SF_AUDIO_PLAYBACK_HW_I2S_Close

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_Close ( sf_audio_playback_hw_ctrl_t
*const p_api_ctrl )
```

8.5.8.1 Brief description

Close open audio driver.

8.5.8.2 Detailed description

Table 338:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

See Common Error Codes and lower level driver function for other possible return codes. This function calls:

- close

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.5.8.3 Function steps

- Close I2S driver.

8.5.9 SF_AUDIO_PLAYBACK_HW_I2S_VersionGet

```
ssp_err_t SF_AUDIO_PLAYBACK_HW_I2S_VersionGet ( ssp_version_t
*const p_version )
```

8.5.9.1 Brief description

Stores the version of the firmware and API in provided pointer p_version.

8.5.9.2 Detailed description

Table 339:Return values

Name	Description
SSP_ERR_ASSERTION	The parameter p_version is NULL.
SSP_SUCCESS	Module version successfully stored in p_version.

NOTE: This function is reentrant.

8.5.10 Extensions

8.5.10.1 sf_audio_playback_hw_i2s_instance_ctrl_t

sf_audio_playback_hw_i2s_instance_ctrl_t

Detailed description

Hardware dependent control block for I2S audio driver.

Variables

- void(* p_callback)(*p_args)

  Callback called when play is complete.

- void * p_context

  Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

- i2s_instance_t const * p_lower_lvl_i2s

  I2S API used to access I2S hardware.

8.5.10.2 sf_audio_playback_hw_i2s_cfg_t

sf_audio_playback_hw_i2s_cfg_t

Detailed description

Hardware dependent configuration for I2S audio driver.

Variables

- i2s_instance_t const * p_lower_lvl_i2s

  I2S API used to access I2S hardware.

8.6 ADC Audio recording Framework

RTOS-integrated ADC implementation of Audio Recording Interface.

The Audio Recording Framework implementation uses the ADC periodic interface for audio recording.

8.6.1 Functions

- SF_AUDIO_RECORD_ADC_Open
- SF_AUDIO_RECORD_ADC_Close
- SF_AUDIO_RECORD_ADC_Start
- SF_AUDIO_RECORD_ADC_Stop
- SF_AUDIO_RECORD_ADC_InfoGet
- SF_AUDIO_RECORD_ADC_VersionGet

8.6.2 Defines

- #define SF_AUDIO_RECORD_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_AUDIO_RECORD_CODE_VERSION_MINOR

  Initial value:`(2U)`

8.6.3 SF_AUDIO_RECORD_ADC_Open

```
ssp_err_t SF_AUDIO_RECORD_ADC_Open ( sf_audio_record_ctrl_t  *const p_api_ctrl ,
 sf_audio_record_cfg_t  const *const p_cfg )
```

8.6.3.1 Brief description

Configure the ADC with user configurations.

8.6.3.2 Detailed description

The SF_AUDIO_RECORD_ADC_Open will initialize the configurations for the underlying ADC periodic framework. Implements

- open.

Table 340:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL. Or any one of the following p_cfg parameter is NULL/zero. p_cfg->p_capture_data_buffer, p_cfg->sample_count, p_cfg->capture_data_buffer_size, p_cfg->p_callback, p_cfg->sampling_rate_hz. Or resolution or time period is not matching. for other possible causes.
SSP_ERR_IN_USE	The channel specified has already been opened.

See Common Error Codes or sf_adc_periodic for other possible return codes or causes. This function calls

- open

NOTE: This function is reentrant for any unit.

8.6.3.3 Function steps

- Initialize the ADC periodic framework
- Initialize the configuration parameters for ADC periodic configuration structure
- Configure the ADC resolution depending on the data width in cfg
- Call the underlying ADC periodic open

8.6.4 SF_AUDIO_RECORD_ADC_Close

```
ssp_err_t SF_AUDIO_RECORD_ADC_Close ( sf_audio_record_ctrl_t
*const p_api_ctrl )
```

8.6.4.1 Brief description

Call the ADC periodic framework Close.

8.6.4.2 Detailed description

Implements

- close.

Table 341:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Control block p_ctrl is not initialized. Call SF_AUDIO_RECORD_ADC_Open to configure.

See Common Error Codes or sf_adc_periodic for other possible return codes or causes. This function calls

- close

8.6.4.3 Function steps

- Call the underlying ADC periodic close

8.6.5 SF_AUDIO_RECORD_ADC_Start

```
ssp_err_t SF_AUDIO_RECORD_ADC_Start ( sf_audio_record_ctrl_t
*const p_api_ctrl )
```

8.6.5.1 Brief description

Call the ADC periodic framework Start.

8.6.5.2 Detailed description

Implements

- start.

Table 342:Return values

Name	Description
SSP_SUCCESS	ADC Periodic Scan started successfully.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Control block p_ctrl is not initialized. Call SF_AUDIO_RECORD_ADC_Open to configure.

See Common Error Codes or sf_adc_periodic for other possible return codes or causes. This function calls

- start

8.6.5.3 Function steps

- Call the underlying ADC periodic start

8.6.6 SF_AUDIO_RECORD_ADC_Stop

```
ssp_err_t SF_AUDIO_RECORD_ADC_Stop ( sf_audio_record_ctrl_t  *const p_api_ctrl )
```

8.6.6.1 Brief description

Call the ADC periodic framework Stop.

8.6.6.2 Detailed description

Implements

- stop.

Table 343:Return values

Name	Description
SSP_SUCCESS	Periodic ADC scan stopped successfully.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Control block p_ctrl is not initialized. Call SF_AUDIO_RECORD_ADC_Open to configure.

See Common Error Codes or sf_adc_periodic for other possible return codes or causes. This function calls

- stop

8.6.6.3 Function steps

- Call the underlying ADC periodic start

8.6.7 SF_AUDIO_RECORD_ADC_InfoGet

```
ssp_err_t SF_AUDIO_RECORD_ADC_InfoGet ( sf_audio_record_ctrl_t
*const p_api_ctrl ,  sf_audio_record_info_t  * p_info )
```

8.6.7.1 Brief description

Provide information about the channel supported by audio recording framework(MONO/STERIO)

8.6.7.2 Detailed description

Implements

- infoGet.

Table 344:Return values

Name	Description
SSP_SUCCESS	InfoGet returns successfully.
SSP_ERR_ASSERTION	p_ctrl or p_info is null.
SSP_ERR_NOT_OPEN	Control block p_ctrl is not initialized. Call SF_AUDIO_RECORD_ADC_Open to configure.

See Common Error Codes or sf_adc_periodic for other possible return codes or causes.

8.6.8 SF_AUDIO_RECORD_ADC_VersionGet

```
ssp_err_t SF_AUDIO_RECORD_ADC_VersionGet ( ssp_version_t  *const p_version )
```

8.6.8.1 Brief description

Gets version and stores it in provided pointer p_version.

8.6.8.2 Detailed description

Implements

- versionGet.

Table 345:Return values

Name	Description
SSP_SUCCESS	VersionGet returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.6.9 Extensions

8.6.9.1 sf_audio_record_adc_instance_ctrl_t

sf_audio_record_adc_instance_ctrl_t

Detailed description

Control block for audio recording Initialization occurs when open is called

Variables

- uint32_t open

  Used by driver to check if pointer to control block is valid
- TX_MUTEX mutex

  Mutex used to protect access to lower level driver hardware registers
- void * p_capture_data_buffer

  Pointer to the buffer that will store the samples
- uint32_t sample_count

  Samples per channel to be buffered before notifying the app
- void(* p_callback)(*p_args)

  Callback function.
- void const * p_context

  Placeholder for user data.
- sf_adc_periodic_instance_t const * p_lower_lvl_adc_periodic

  Lower level ADC periodic instance.

8.6.9.2 sf_audio_record_adc_hw_cfg_t

sf_audio_record_adc_hw_cfg_t

Detailed description

Variables

- sf_adc_periodic_instance_t const * p_lower_lvl_adc_periodic

8.7 BLOCK_MEDIA_SDMMC

RTOS-integrated Block Media framework for SDMMC driver.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

8.7.1 Functions

- SF_Block_Media_SDMMC_Open
- SF_Block_Media_SDMMC_Read
- SF_Block_Media_SDMMC_Write
- SF_Block_Media_SDMMC_Control
- SF_Block_Media_SDMMC_Close
- SF_Block_Media_SDMMC_VersionGet
- sf_block_media_sdmmc_callback

8.7.2 Defines

- #define BLOCK_MEDIA_SDMMC_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define BLOCK_MEDIA_SDMMC_CODE_VERSION_MINOR

  Initial value:`(5U)`

8.7.3 SF_Block_Media_SDMMC_Open

```
ssp_err_t SF_Block_Media_SDMMC_Open ( sf_block_media_ctrl_t  *const p_api_ctrl ,
 sf_block_media_cfg_t  const *const p_cfg )
```

8.7.3.1 Brief description

Open device for read/write and control.

8.7.3.2 Detailed description

Open an SD or MMC device port for read/write and control. This function initializes the SDMMC driver and hardware the first time it is called out of reset.

Table 346:Return values

Name	Description
SSP_SUCCESS	Port is available and is now open for read, write, and control access.
SSP_ERR_ASSERTION	p_ctrl, p_cfg, p_block_media_cfg or p_sdmmc is NULL.
SSP_ERR_INTERNAL	OS service call fails.

Table 346:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- open

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.7.3.3 Function steps

- Create SDMMC event flag and put it into context
- Mark the stream as open by initializing "BMSO" in its ASCII equivalent.
- Cleanup before logging the error

8.7.4 SF_Block_Media_SDMMC_Read

```
ssp_err_t SF_Block_Media_SDMMC_Read ( sf_block_media_ctrl_t  *const p_api_ctrl ,
  uint8_t *const p_dest ,   uint32_t const start_block ,   uint32_t
const block_count )
```

8.7.4.1 Brief description

Read data from SD/MMC.

8.7.4.2 Detailed description

Read data from an SD or MMC device port.

Table 347:Return values

Name	Description
SSP_SUCCESS	Data read successfully.
SSP_ERR_ASSERTION	p_ctrl, p_sdmmc or p_dest is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INTERNAL	OS service call fails.

Table 347:Return values (Continued)

Name	Description
SSP_ERR_READ_FAILED	Data read failed.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- read

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.7.4.3 Function steps

- Wait until read operation is completed. Event is signaled in event flag object.

8.7.5 SF_Block_Media_SDMMC_Write

```
ssp_err_t SF_Block_Media_SDMMC_Write ( sf_block_media_ctrl_t
*const p_api_ctrl ,   uint8_t const *const p_src ,   uint32_t const start_block ,
  uint32_t const block_count )
```

8.7.5.1 Brief description

Write data to SDMMC channel.

8.7.5.2 Detailed description

Table 348:Return values

Name	Description
SSP_SUCCESS	Card write finished successfully.
SSP_ERR_ASSERTION	p_ctrl, p_sdmmc or p_src is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INTERNAL	OS service call fails.
SSP_ERR_WRITE_FAILED	Data write failed.
SSP_ERR_WRITE_PROTECTED	SD or MMC card is Write Protected.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- write

NOTE: This function is reentrant for different channels.

8.7.5.3 Function steps

- Wait until write operation is completed. Event is signaled in event flag object.

8.7.6 SF_Block_Media_SDMMC_Control

```
ssp_err_t SF_Block_Media_SDMMC_Control ( sf_block_media_ctrl_t
*const p_api_ctrl ,  ssp_command_t  const command ,   void * p_data )
```

8.7.6.1 Brief description

Send control commands to and receive status of SD/MMC port.

8.7.6.2 Detailed description

Send control commands to the SD/MMC port and receive the status of the SD/MMC port.

Table 349:Return values

Name	Description
SSP_SUCCESS	Command executed successfully.
SSP_ERR_ASSERTION	p_ctrl or p_sdmmc or p_data is Null.
SSP_ERR_NOT_OPEN	The channel is not opened.
SF_INFO_NOT_AVAILABLE	Information not available possibly because card has been removed or is defective.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- control

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.7.7 SF_Block_Media_SDMMC_Close

```
ssp_err_t SF_Block_Media_SDMMC_Close ( sf_block_media_ctrl_t
*const p_api_ctrl )
```

8.7.7.1 Brief description

Close open device port.

8.7.7.2 Detailed description

Close an open SD/MMC device port.

Table 350:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl or p_sdmmc is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INTERNAL	OS service call fails.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- close

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

8.7.7.3 Function steps

- Mark control block as unused so it can be reconfigured.

8.7.8 SF_Block_Media_SDMMC_VersionGet

```
ssp_err_t SF_Block_Media_SDMMC_VersionGet ( ssp_version_t  *const p_version )
```

8.7.8.1 Brief description

Get version of Block Media SD/MMC driver.

8.7.8.2 Detailed description

Return the version of the firmware and API.

Table 351:Return values

Name	Description
SSP_ERR_ASSERTION	p_version is Pointer.
SSP_SUCCESS	version read successfully.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant.

8.7.9 sf_block_media_sdmmc_callback

```
sf_block_media_sdmmc_callback ( sdmmc_callback_args_t  * pcb_arg )
```

8.7.9.1 Detailed description

SDMMC SSP framework level callback

Table 352:Parameters

Name	Direction	Description
pcb_arg	in	Pointer to callback parameters

Table 353:Return values

Name	Description
void	

8.7.9.2 Function steps

- Transfer complete event occurs wake up the suspended thread.

8.7.10 Extensions

8.7.10.1 sf_block_media_on_sdmmc_cfg_t

sf_block_media_on_sdmmc_cfg_t

Detailed description

Variables

- sdmmc_instance_t const *const p_lower_lvl_sdmmc

  Pointer to SDMMC instance structure.

8.7.10.2 sf_block_media_sdmmc_instance_ctrl_t

sf_block_media_sdmmc_instance_ctrl_t

Detailed description

SDMMC block media instance control block.

Variables

- uint32_t block_size

  Block size in bytes.

- sdmmc_instance_t * p_lower_lvl_sdmmc

  Pointer to SDMMC instance structure.

- TX_EVENT_FLAGS_GROUP eventflag

  Pointer to the event flag object for SDMMC data transfer.

- uint32_t open

  Used to determine if framework is initialized.

8.8 Cellular NSAL Implementation on NetX

Cellular NetX NSAL interface implementation header file.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

8.8.1 Functions

- sf_cellular_nsal_netx_driver
- sf_cellular_nsal_ppp_send_byte
- sf_cellular_nsal_invalid_packet_handler

8.8.2 Defines

- #define SF_CELR_NSALNX_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Version of code that implements the API defined in this file Major Version of code that implements the API defined in this file

- #define SF_CELR_NSALNX_CODE_VERSION_MINOR

  Initial value:`(0U)`

  Minor Version of code that implements the API defined in this file

8.8.3 sf_cellular_nsal_netx_driver

```
 sf_cellular_nsal_netx_driver ( NX_IP_DRIVER  * driver_req_ptr ,
 sf_cellular_instance_t  const * p_cellular_instance ,  sf_cellular_nsal_cfg_t
* p_cellular_nsal_cfg )
```

8.8.3.1 Brief description

NetX IP Driver entry function.

8.8.3.2 Detailed description

Table 354:Parameters

Name	Direction	Description
driver_req_ptr	in	Pointer to NetX IP Driver
p_cellular_instance	in	Pointer to cellular framework instance
p_cellular_nsal_cfg	in	Pointer to cellular nsal configuration structure

8.8.4 sf_cellular_nsal_ppp_send_byte

```
 sf_cellular_nsal_ppp_send_byte (  UCHAR byte ,  sf_cellular_instance_t  const
* p_celr_instance )
```

8.8.4.1 Detailed description

NSAL PPP Send bytes callback API

Table 355:Parameters

Name	Direction	Description
byte	in	Byte to send
p_celr_instance	in	Pointer to cellular framework instance

8.8.4.2 Function steps

- Transmit byte

8.8.5 sf_cellular_nsal_invalid_packet_handler

```
sf_cellular_nsal_invalid_packet_handler ( NX_PACKET  * p_packet_ptr ,
sf_cellular_instance_t  const * p_celr_instance )
```

8.8.5.1 Brief description

Cellular framework nsal invalid callback handler.

8.8.5.2 Detailed description

Table 356:Parameters

Name	Direction	Description
p_packet_ptr	in	Pointer to NetX Packet
p_celr_instance	in	Pointer to cellular framework instance

8.8.5.3 Function steps

- variable to get Memory compare result
- Check whether packet contains "NO CARRIER" string, we are ignoring first and last 2 bytes of packet which are Carriage Return and Line Feed.
- check whether no carrier string present
- Disconnect Network
- Soft reset the module
- Provision the Module using callback
- Reestablish data connection
- Restart PPP
- Release the packet

8.9 Console Framework

RTOS-integrated Console Framework.

This is a ThreadX aware console framework implemented using the SSP communications framework.

8.9.1 Functions

- SF_CONSOLE_Open
- SF_CONSOLE_Close
- SF_CONSOLE_Parse
- SF_CONSOLE_Prompt
- SF_CONSOLE_Read
- SF_CONSOLE_Write
- SF_CONSOLE_ArgumentFind
- SF_CONSOLE_CallbackNextMenu
- SF_CONSOLE_VersionGet

8.9.2 Defines

- #define SF_CONSOLE_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file
- #define SF_CONSOLE_CODE_VERSION_MINOR

  Initial value:`(5U)`

8.9.3 SF_CONSOLE_Open

```
ssp_err_t SF_CONSOLE_Open ( sf_console_ctrl_t  *const p_api_ctrl ,
 sf_console_cfg_t  const *const p_cfg )
```

8.9.3.1 Detailed description

Table 357:Return values

Name	Description
SSP_SUCCESS	Console channel is successfully opened

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.3.2 Function steps

- Open UART driver
- Store echo configuration and initial menu in control block
- Prompt for input autostart is true

8.9.4 SF_CONSOLE_Close

```
ssp_err_t SF_CONSOLE_Close ( sf_console_ctrl_t  *const p_api_ctrl )
```

8.9.4.1 Detailed description

Table 358:Return values

Name	Description
SSP_SUCCESS	Console successfully closed

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.4.2 Function steps

- Close UART driver

8.9.5 SF_CONSOLE_Parse

```
ssp_err_t SF_CONSOLE_Parse ( sf_console_ctrl_t  *const p_api_ctrl ,
 sf_console_menu_t  const *const p_menu ,   uint8_t const *const p_input ,
  uint32_t const bytes )
```

8.9.5.1 Detailed description

Table 359:Return values

Name	Description
SSP_SUCCESS	Data parsed successfully, command found and callback called.
SSP_ERR_UNSUPPORTED	Command not found in the current menu.

Table 359:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One or more input parameter pointers are invalid.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.5.2 Function steps

- Print help menu if help command is entered
- Go back one menu if previous menu command is entered.
- Go back to root menu if root menu command is entered.
- Look for matching commands, call callback if command found.

8.9.6 SF_CONSOLE_Prompt

```
ssp_err_t SF_CONSOLE_Prompt ( sf_console_ctrl_t  *const p_api_ctrl ,
 sf_console_menu_t  const *const p_menu ,   UINT const timeout )
```

8.9.6.1 Detailed description

Table 360:Return values

Name	Description
SSP_SUCCESS	Received valid command and called callback
SSP_ERR_ASSERTION	p_ctrl is NULL

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.6.2 Function steps

- Update stored current menu pointer if a new pointer is specified.
- Print menu name followed by ">" to prompt for user input.
- Lock the console UART framework to reserve exclusive access until the command completes. ***NOTE:*** *Transmission is only locked while the menu name is printed and while the input command is non-zero in length. This allow debug messages to print from other threads while echo is off or no input command has been entered.*Wait for input
- Parse input and call associated user callback.

- Command is complete, so unlock comms reception.

8.9.7 SF_CONSOLE_Read

```
ssp_err_t SF_CONSOLE_Read ( sf_console_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   uint32_t const timeout )
```

8.9.7.1 Detailed description

Table 361:Return values

Name	Description
SSP_SUCCESS	Data read completed successfully

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.7.2 Function steps

- Lock the communications framework reception until carriage return is received.
- Read one byte at a time, checking for carriage returns, backspace, delete, and escape codes.
- Unlock the communications framework reception

8.9.8 SF_CONSOLE_Write

```
ssp_err_t SF_CONSOLE_Write ( sf_console_ctrl_t  *const p_api_ctrl ,   uint8_t
const *const p_src ,   uint32_t const timeout )
```

8.9.8.1 Detailed description

Table 362:Return values

Name	Description
SSP_SUCCESS	Data write completed successfully

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.8.2 Function steps

- Write null terminated string. Calculate the length. If it isn't longer than the maximum, write the entire string to the console.

8.9.9 SF_CONSOLE_ArgumentFind

```
ssp_err_t SF_CONSOLE_ArgumentFind (  uint8_t const *const p_arg ,   uint8_t const
*const p_str ,   int32_t *const p_index ,   int32_t *const p_data )
```

8.9.9.1 Detailed description

Table 363:Return values

Name	Description
SSP_SUCCESS	Argument found successfully
SSP_ERR_ASSERTION	p_arg or p_str is NULL

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any channel.

8.9.9.2 Function steps

- Search for first letter match at beginning of word.
- If the input string matches the input argument, store the index of the character following the argument in p_index and the data at that index in p_data. Then return.

8.9.10 SF_CONSOLE_CallbackNextMenu

```
SF_CONSOLE_CallbackNextMenu ( sf_console_callback_args_t  * p_args )
```

8.9.10.1 Detailed description

Callback provided to continue parsing the next menu down.

Table 364:Parameters

Name	Direction	Description
p_args	in	Pointer to callback arguments to use in the next menu.

8.9.10.2 Function steps

- Next level menu is passed in as the user context parameter
- Update current menu.
- Check to see if the next menu command was input in the remaining string.

8.9.11 SF_CONSOLE_VersionGet

```
ssp_err_t SF_CONSOLE_VersionGet ( ssp_version_t  *const p_version )
```

8.9.11.1 Detailed description

Console version get function.

Table 365:Parameters

Name	Direction	Description
p_version	in	Version information stored here.

Table 366:Return values

Name	Description
SSP_SUCCESS	Version stored in provided pointer.
SSP_ERR_ASSERTION	p_version was null.

8.9.11.2 Function steps

- Set version pointer

8.9.12 Extensions

8.9.12.1 sf_console_instance_ctrl_t

sf_console_instance_ctrl_t

Detailed description

Console instance control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- sf_console_menu_t const * p_current_menu

  Current menu is stored here.
- sf_comms_instance_t const * p_comms

  Pointer to communications driver instance.
- uint8_t new_line

  Whether to echo input commands to transmitter.
- bool echo

  Whether to echo input commands to transmitter.
- uint8_t input[SF_CONSOLE_MAX_INPUT_LENGTH]

  Input buffer used to store user input.

8.10 FX_IO Framework

FileX adaptation layer for block media device drivers.
SF_EL_FX FileX I/O is a single entry function which adapts FileX to Renesas Synergy block media device drivers.

8.10.1 Summary

SF_EL_FX Has no API file.

8.10.2 Functions

- SF_EL_FX_BlockDriver
- check_partition_offset

8.10.3 Defines

- #define SF_EL_FX_API_VERSION_MAJOR

  Initial value:`(1)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  Version of the API defined in this file
- #define SF_EL_FX_API_VERSION_MINOR

  Initial value:`(0)`
- #define SF_EL_FX_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Version of code that implements the API defined in this file

- #define SF_EL_FX_CODE_VERSION_MINOR

  Initial value:`(4U)`
- #define SF_EL_FX_55AA_SIGNATURE_OFFSET

  Initial value:`(0x1FEU)`

  SSP FileX Support.
- #define SF_EL_FX_JUMP_INST_OFFSET

  Initial value:`(0U)`
- #define SF_EL_FX_FAT16_SECTORS_PER_FAT

  Initial value:`(0x16U)`
- #define SF_EL_FX_FAT32_SECTORS_PER_FAT

  Initial value:`(0x24U)`
- #define SF_EL_FX_FAT16_SECTOR_COUNT

  Initial value:`(0x13U)`
- #define SF_EL_FX_FAT32_SECTOR_COUNT

  Initial value:`(0x20U)`
- #define SF_EL_FX_PARTITION0_OFFSET

  Initial value:`(0x1C6U)`
- #define SF_EL_FX_PARTITION0_TYPE_OFFSET

  Initial value:`(0x1C2U)`

8.10.4 SF_EL_FX_BlockDriver

```
SF_EL_FX_BlockDriver ( FX_MEDIA  * media_ptr )
```

8.10.4.1 Brief description

Access Block Media device functions open, close, read, write and control.

8.10.4.2 Detailed description

The file system relies on the media to be formatted prior to creating directories and files The sector size and sector count will change depending on the media type and size.

The File Allocation Table (FAT) starts after the reserved sectors in the media. The FAT area is basically an array of 12-bit, 16-bit, or 32-bit entries that determine if that cluster is allocated or part of a chain of clusters comprising a subdirectory or a file. The size of each FAT entry is determined by the number of clusters that need to be represented. If the number of clusters (derived from the total sectors divided by the sectors per cluster) is less than 4,086, 12-bit FAT entries are used. If the total number of clusters is greater than 4,086 and less than or equal to 65,525, 16-bit FAT entries are used. Otherwise, if the total number of clusters is greater than 65,525, 32-bit FAT entries are used. The SF_EL_FX_BlockDriver function is called from the FileX file system driver and issues requests to a Block Media device through the Synergy Block Media Interface.

Table 367:Parameters

Name	Direction	Description
p_fx_media	inout	FileX media control block. All information about each open media device are maintained in by the FX_MEDIA data type. The I/O driver communicates the success or failure of the request through the fx_media_driver_status member of FX_MEDIA (p_fx_media->fx_media_driver_status). Possible values are documented in the FileX User Guide.

Table 368:Return values

Name	Description
noneUses	block media driver for accesses.

8.10.4.3 Function steps

- Initialize FileX I/O status to error. It will change to FX_SUCCESS unless an operation fails.
- Process the driver request specified in the media control block.
- FileX reads one or more sectors into memory by issuing a read request to the I/O driver.
- FileX writes one or more sectors to the physical media by issuing a write request to the I/O driver.
- FileX flushes all sectors currently in the driver's sector cache to the physical media by issuing a flush request to the I/O driver. Synergy drivers do not currently cache sectors.
- Command not currently supported in available Synergy modules. Return driver success.
- FileX informs the driver to abort all further physical I/O activity with the physical media by issuing an abort request to the I/O driver. The driver should not perform any I/O again until it is re-initialized.
- Close/un-initialize device.
- Although the actual driver initialization processing is application specific, it usually consists of data structure initialization and possibly some preliminary hardware initialization. This request is the first made by FileX and is done from within the fx_media_open service. If media write protection is detected, the driver fx_media_driver_write_protect member of FX_MEDIA should be set to FX_TRUE.
- Open the block media.
- Get write protection status.
- Close the block media.

- FileX uses the uninit command to close the media.
- Close/un-initialize device.
- Instead of using a standard read request, FileX makes a specific request to read the media's boot sector.
- Read the boot record and return to the caller.
- Open the block media.
- Use already allocated buffer
- Read the first non hidden sector.
- Check the partition offset to determine if the current sector is the boot record or partition table.
- Read the sector at the offset indicated in the partition table.
- Check the partition offset to confirm it is the boot record.
- check_partition_offset failed or the current buffer is not the boot sector
- Instead of using a standard write request, FileX makes a specific request to write the media's boot sector.
- Open driver.
- Write the boot record and return to the caller.
- Invalid driver request.
- Successful driver request.

8.10.5 check_partition_offset

```
ssp_err_t check_partition_offset (  uint8_t * p_sector ,   uint32_t
* partition_offset )
```

8.10.5.1 Brief description

Checks if the sector passed is a valid boot record. If not returns the offset to the partitions boot record.

8.10.5.2 Detailed description

(end addtogroup SF_EL_FX)

Table 369:Parameters

Name	Direction	Description
p_sector	in	Pointer to sector
partition_offset	in	Sector number of found partition table. 0 if boot sector.

Table 370:Return values

Name	Description
SSP_SUCCESS	Found boot record or partition offset
SSP_ERR_MEDIA_OPEN_FAILED	Not a valid boot record or partition table

8.10.5.3 Function steps

- Check signature to make sure the buffer is valid.
- Invalid, return an error.
- Check for a valid jump instruction.
- Check for a non zero sectors per FAT.
- Check for a non zero sector count.
- If the boot sector is valid and sectors per fat is not 0 and sector count is not 0 return valid boot rectord.
- Sector is not a valid boot record. Read the offset to partition 0.
- If the partition type is set and the offset is not zero report the partition offset.

8.10.6 Extensions

8.10.6.1 sf_el_fx_t

sf_el_fx_t

Detailed description

Block Media Control Block Type

Variables

- sf_block_media_instance_t * p_lower_lvl_block_media

  Lower level block media pointer.
- void * p_extend

8.11 GUIX Synergy Port

GUIX adaptation layer.

8.11.1 Functions

- SF_EL_GX_Open
- SF_EL_GX_Close
- SF_EL_GX_VersionGet
- SF_EL_GX_Setup
- SF_EL_GX_CanvasInit

8.11.2 SF_EL_GX_Open

```
ssp_err_t SF_EL_GX_Open ( sf_el_gx_ctrl_t  *const p_api_ctrl ,  sf_el_gx_cfg_t
const *const p_cfg )
```

8.11.2.1 Brief description

GUIX adaptation framework driver for Synergy, open function to configure the framework module. The function initialize RTOS resources used by the module, initialize the control block based on user configuration, and transition the module state to SF_EL_GX_OPENED. This function calls following functions:

8.11.2.2 Detailed description

- sf_el_gx_open_param_check() Check configuration parameters if parameter check is enabled.
- tx_mutex_create() Creates the mutex for lock the driver during the context update.
- tx_mutex_delete() Deletes the mutex if kernel service calls failed in the process.
- tx_semaphore_create() Creates the semaphore for rendering and displaying synchronization.

Table 371:Return values

Name	Description
SSP_SUCCESS	Opened the module successfully.
SSP_ERR_ASSERTION	NULL pointer error happened.
SSP_ERR_IN_USE	SF_EL_GX is in-use.
SSP_ERR_INTERNAL	Error happened in kernel service calls.
SSP_ERR_INVALID_ARGUMENT	An invalid argument was passed to the driver.

NOTE: This function does not initialize the display or rendering hardware but setup function will do.

NOTE: This function registers a user callback function but it is optional. Set NULL to p_cfg::p_callback if user callback is not required.

NOTE: The configuration for the frame buffer B (p_cfg::p_framebuffer_b) is optional. Set NULL to p_framebuffer_b in case of a single-buffered system.

8.11.2.3 Function steps

- Creates global mutex for SF_EL_GX
- Locks the SF_EL_GX instance until driver setup is done by SF_EL_GX_Setup.
- Creates a semaphore for frame buffer flip
- Initializes the SF_EL_GX control block
- Saves the control block to the global pointer inside the module temporarily. Stored data will be used in sf_el_gx_driver_setup() which will be invoked by GUIX. This pointer will be valid at last but be protected until SF_EL_GX_Setup is done.
- Changes the driver state

8.11.3 SF_EL_GX_Close

```
ssp_err_t SF_EL_GX_Close ( sf_el_gx_ctrl_t  *const p_api_ctrl )
```

8.11.3.1 Brief description

GUIX adaptation framework for Synergy, Close function. This function calls following functions:

8.11.3.2 Detailed description

- tx_mutex_get() Gets the mutex to lock the driver while device access.
- tx_mutex_put() Puts the mutex to unlock the driver while device access.
- tx_mutex_delete() Deletes the mutex if kernel service calls failed in the process.
- tx_semaphore_delete() Deletes the semaphore for rendering and displaying synchronization.
- sf_el_gx_d2_close() Finalizes 2D Drawing Engine hardware.
- sf_el_gx_display_close() Finalizes display hardware.

Table 372:Return values

Name	Description
SSP_SUCCESS	Closed the module successfully.
SSP_ERR_ASSERTION	NULL pointer error happens.
SSP_ERR_NOT_OPEN	SF_EL_GX is not opened.
SSP_ERR_INTERNAL	Error happened in kernel service calls.

Table 372:Return values (Continued)

Name	Description
SSP_ERR_TIMEOUT	Error occurred in display driver.
SSP_ERR_D2D_ERROR_DEINIT	Error occurred in D/AVE 2D driver.

NOTE: This function is re-entrant.

8.11.3.3 Function steps

- Locks the driver to update the context.
- Finalizes display hardware
- Changes the driver state
- Deletes a semaphore for frame buffer flip
- Unlocks the SF_EL_GX instance
- Deletes driver global mutex
- Clears the temporary storage for the pointer to a control block. This procedure has done in SF_EL_GX_Setup in the expected function call sequence, but clear it here as well in case SF_EL_GX_Setup being not called.

8.11.4 SF_EL_GX_VersionGet

```
ssp_err_t SF_EL_GX_VersionGet ( ssp_version_t  * p_version )
```

8.11.4.1 Brief description

GUIX adaptation framework for Synergy, Version get function.

8.11.4.2 Detailed description

Table 373:Parameters

Name	Direction	Description
p_version	inout	The version number.

Table 374:Return values

Name	Description
SSP_SUCCESS	Successfully returned the module version.
SSP_ERR_ASSERTION	NULL pointer is passed to function.

NOTE: This function is re-entrant.

8.11.5 SF_EL_GX_Setup

```
SF_EL_GX_Setup ( GX_DISPLAY  * p_display )
```

8.11.5.1 Brief description

GUIX adaptation framework for Synergy, Setup GUIX low level device drivers for Display and D/AVE 2D interface. This function has to be passed to the GUIX Studio display driver setup function gx_studio_display_configure() to let GUIX call this function and configure the GUIX low level device driver(s). This function calls following functions:

8.11.5.2 Detailed description

- tx_mutex_put() Puts the driver global mutex when the low level driver setup is done
- tx_mutex_delete() Deletes the mutex if kernel service calls failed in the process
- _gx_synergy_display_driver_565rgb_setup() Setups default GUIX callback functions for RGB565 in case of display format format is RGB565 format
- _gx_synergy_display_driver_24xrgb_setup() Setups default GUIX callback functions for RGB565 in case of display format format is RGB888, unpacked format
- _gx_display_driver_32argb_setup() Setups default GUIX callback functions for RGB565 in case of display format format is ARGB8888, unpacked format
- sf_el_gx_driver_setup() Setups low level device drivers and overrides the GUIX default callback functions with hardware accelerated functions.

Table 375:Return values

Name	Description
GX_SUCCESS	Device driver setup is successfully done.
GX_FAILURE	Device driver setup failed.

NOTE: Make sure SF_EL_GX_Open has been called when this function is called back by GUIX. The behavior is not defined if this function were not invoked by GUIX.

8.11.5.3 Function steps

- Copies the GX_DISPLAY context for later use.
- Setups GUIX low level device drivers
- Changes the driver state
- Clears the temporary storage for the pointer to a control block.
- Unlocks the SF_EL_GX instance since driver setup is done

8.11.6 SF_EL_GX_CanvasInit

```
ssp_err_t SF_EL_GX_CanvasInit ( sf_el_gx_ctrl_t  *const p_api_ctrl ,
 GX_WINDOW_ROOT  * p_window_root )
```

8.11.6.1 Brief description

GUIX adaptation framework for Synergy, Canvas initialization, setup the memory address of first canvas to be rendered.

8.11.6.2 Detailed description

Table 376:Return values

Name	Description
SSP_SUCCESS	Memory address is successfully configured to a canvas.
SSP_ERR_ASSERTION	Invalid control block (NULL pointer) or window root (NULL pointer) passed to driver.
SSP_ERR_INVALID_CALL	Function call was made when the driver is not in SF_EL_GX_CONFIGURED state.
SSP_ERR_INTERNAL	Mutex operation had an error.

8.11.6.3 Function steps

- Locks the driver to update the context.
- Lets GUIX know the first canvas
- Unlocks the driver.

8.11.7 Extensions

8.11.7.1 sf_el_gx_instance_ctrl_t

sf_el_gx_instance_ctrl_t

Detailed description

GUIX adaptation layer for SSP. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Instance control block for the SSP GUIX adaptation framework

Variables

- GX_DISPLAY * p_display

  Pointer to the GUIX display context.

- display_instance_t * p_display_instance

  Pointer to a display instance.

- display_runtime_cfg_t * p_display_runtime_cfg

  Pointer to a runtime display configuration.

- void * p_canvas

  Pointer to a canvas(reserved)

- void * p_framebuffer_read

  Pointer to a frame buffer (for displaying)

- void * p_framebuffer_write

  Pointer to a frame buffer (for rendering)

- void(* p_callback)(*p_args)

  Pointer to callback function.

- void * p_context

  Pointer to a context.

- TX_SEMAPHORE semaphore

  Semaphore for the frame buffer flip sync.

- bool rendering_enable

  Sync flag between Rendering and displaying.

- bool display_list_flushed

  Flag to show the display list is flushed.

- sf_el_gx_state_t state

  State of this module.

- void * p_jpegbuffer

  Pointer to a JPEG work buffer.

- uint32_t jpegbuffer_size

  Size of a JPEG work buffer.
- uint16_t rotation_angle

  Screen rotation angle(0/90/270)
- void * p_sf_jpeg_decode_instance

  Pointer to a JPEG framework instance.

8.12 Telnet Communication Framework

RTOS-integrated Communications Framework NetX telnet server implementation.

8.12.1 Functions

- SF_EL_NX_COMMS_Open
- SF_EL_NX_COMMS_Close
- SF_EL_NX_COMMS_Read
- SF_EL_NX_COMMS_Write
- SF_EL_NX_COMMS_Lock
- SF_EL_NX_COMMS_Unlock
- SF_EL_NX_COMMS_VersionGet
- netx_resource_setup
- netx_resource_release
- tx_resource_setup
- tx_resource_release
- receive_data
- telnet_new_connection
- telnet_connection_end
- telnet_receive_data
- sf_el_nx_comms_error

8.12.2 Variables

- gp_ctrls
- gps_ctrls_logical

8.12.3 Defines

- #define SF_EL_NX_COMMS_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_EL_NX_COMMS_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_EL_NX_COMMS_PACKET_POOL_MEMORY_SIZE_BYTES

  Initial value:`((1536U + 32U + (uint32_t) sizeof(NX_PACKET)) * 50U)`
- #define SF_EL_NX_COMMS_IP_MEMORY_SIZE_BYTES

  Initial value:`(2048U)`
- #define SF_EL_NX_COMMS_ARP_MEMORY_SIZE_BYTES

  Initial value:`(1024U)`
- #define SF_EL_NX_COMMS_TELNET_SERVER_MEMORY_SIZE_BYTES

  Initial value:`(2048U)`
- #define SF_EL_NX_COMMS_QUEUE_MEMORY_SIZE_BYTES

  Initial value:`(20U)`
- #define OPEN

  Initial value:`(0x4E58434DU)`

  0x4E58434DU is "NXCM" in ASCII, used to identify NetX comms handle

8.12.4 SF_EL_NX_COMMS_Open

```
ssp_err_t SF_EL_NX_COMMS_Open ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_cfg_t  const *const p_cfg )
```

8.12.4.1 Detailed description

Initializes the Telnet server.

Table 377:Return values

Name	Description
SSP_SUCCESS	Channel opened successfully

Table 377:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the following invalid parameter passed. • Pointer p_api_ctrl is NULL. • Pointer p_cfg is NULL • Pointer p_cfg->p_extend is NULL
SSP_ERR_INTERNAL	An internal ThreadX Or NetX error has occurred. This is typically a failure to create/use thread mutex or failure create/enable an internal thread/feature for NetX service.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

NOTE: - This function is reentrant.

8.12.4.2 Function steps

- Initialize and start NetX and NetX application resources
- Initialize and start ThreadX resources
- Mark control block open.

8.12.5 SF_EL_NX_COMMS_Close

```
ssp_err_t SF_EL_NX_COMMS_Close ( sf_comms_ctrl_t  *const p_api_ctrl )
```

8.12.5.1 Detailed description

Disconnect Telnet server and clean up variables.

Table 378:Return values

Name	Description
SSP_SUCCESS	Channel successfully closed
SSP_ERR_ASSERTION	Pointer p_api_ctrl is NULL
SSP_ERR_INTERNAL	An internal ThreadX Or NetX error has occurred. This is typically a failure to create/use thread mutex or failure create/enable an internal thread/feature for NetX service.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

NOTE: - This function is reentrant.

8.12.5.2 Function steps

- Stop and release NetX resources
- Release ThreadX resources
- Mark control block closed.

8.12.6 SF_EL_NX_COMMS_Read

```
ssp_err_t SF_EL_NX_COMMS_Read ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   UINT const timeout )
```

8.12.6.1 Detailed description

Read data from the Telnet server.

Table 379:Return values

Name	Description
SSP_SUCCESS	Data reception ends successfully.
SSP_ERR_ASSERTION	One of the following invalid parameter passed. • Pointer p_api_ctrl is NULL • Pointer p_dest is NULL • Invalid read length i.e. bytes value is zero
SSP_ERR_TIMEOUT	One of the following operation timed out. • 'Event flags get' timed out • 'Receive mutex get' timed out • 'Queue receive' timed out
SSP_ERR_INTERNAL	An internal ThreadX Or NetX error has occurred. This is typically a failure to create/use thread mutex or failure create/enable an internal thread/feature for NetX service.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

NOTE: - This API is reentrant.

8.12.6.2 Function steps

- Wait for a client to be connected.
- Get mutex.
- Receive data.
- Release mutex.

8.12.7 SF_EL_NX_COMMS_Write

```
ssp_err_t SF_EL_NX_COMMS_Write ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
const *const p_src ,   uint32_t const bytes ,   UINT const timeout )
```

8.12.7.1 Detailed description

Write data to the Telnet server.

Table 380:Return values

Name	Description
SSP_SUCCESS	Data transmission finished successfully.
SSP_ERR_ASSERTION	One of the following invalid parameter passed. • Pointer p_api_ctrl is NULL. • Pointer p_src is NULL. • Invalid write length i.e. bytes value is zero.
SSP_ERR_TIMEOUT	One of the following operation timed out. • 'Event flags get' timed out • 'Transmit mutex get' timed out
SSP_ERR_OUT_OF_MEMORY	Couldn't allocate pool memory for Telnet server
SSP_ERR_INTERNAL	An internal ThreadX Or NetX error has occurred. This is typically a failure to create/use thread mutex or failure create/enable an internal thread/feature for NetX service.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

NOTE: - This function is reentrant.

8.12.7.2 Function steps

- Get Transmit mutex.
- Allocate a packet for data.
- Build the message.
- Send the packet to the client.
- Release Transmit mutex.

8.12.8 SF_EL_NX_COMMS_Lock

```
ssp_err_t SF_EL_NX_COMMS_Lock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type ,   UINT timeout )
```

8.12.8.1 Detailed description

Table 381:Return values

Name	Description
SSP_SUCCESS	Locking NX_COMMS resource successful.
SSP_ERR_ASSERTION	Pointer p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel is not open.
SSP_ERR_TIMEOUT	Mutex not available in timeout.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

8.12.8.2 Function steps

- Get transmit mutex if transmit or all mutexes are requested.
- Get receive mutex if receive or all mutexes are requested.

8.12.9 SF_EL_NX_COMMS_Unlock

```
ssp_err_t SF_EL_NX_COMMS_Unlock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type )
```

8.12.9.1 Detailed description

Table 382:Return values

Name	Description
SSP_SUCCESS	Unlocking NX_COMMS resource successful.
SSP_ERR_ASSERTION	Pointer p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel is not open.

NOTE: - For Telnet server specific API details, refer Telnet server user manual.

NOTE: - For NetX specific API details, refer NetX user manual.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

8.12.9.2 Function steps

- Release transmit mutex if transmit or all mutexes are released.
- Release receive mutex if receive or all mutexes are released.

8.12.10 SF_EL_NX_COMMS_VersionGet

```
ssp_err_t SF_EL_NX_COMMS_VersionGet ( ssp_version_t  *const p_version )
```

8.12.10.1 Detailed description

Get driver version

Table 383:Return values

Name	Description
SSP_SUCCESS	Operation successful
SSP_ERR_ASSERTION	p_version pointer is NULL

NOTE: - This function is reentrant.

8.12.11 netx_resource_setup

```
ssp_err_t netx_resource_setup ( sf_el_nx_comms_instance_ctrl_t  * p_ctrl ,
 sf_el_nx_comms_on_comms_cfg_t  * p_cfg_extend )
```

8.12.11.1 Brief description

This is a helper routine called by open API to initialize NetX and NetX application resources.

8.12.11.2 Detailed description

Table 384:Parameters

Name	Direction	Description
p_ctrl	in	- Pointer to a control structure allocated by user.
p_cfg_extend	in	- Pointer to lower level communications control structure

8.12.11.3 Function steps

- Initialize the NetX system.
- Create a packet pool.
- Create an IP instance.
- Enable ARP and supply ARP cache memory for the IP Instance.
- Enable TCP traffic.
- Enable ICMP.
- Create the NetX TELNET Server.
- Start the TELNET Server.

8.12.12 netx_resource_release

```
ssp_err_t netx_resource_release ( sf_el_nx_comms_instance_ctrl_t  * p_ctrl )
```

8.12.12.1 Brief description

This is a helper routine called by close API to release NetX and NetX application resources.

8.12.12.2 Detailed description

Table 385:Parameters

Name	Direction	Description
p_ctrl	in	- Pointer to a control structure allocated by user.

8.12.12.3 Function steps

- Initiate server disconnection.
- Delete Telnet server instance.
- Delete IP instance.

8.12.13 tx_resource_setup

```
ssp_err_t tx_resource_setup ( sf_el_nx_comms_instance_ctrl_t  * p_ctrl )
```

8.12.13.1 Brief description

This is a helper routine called by open API to initialize ThreadX resources.

8.12.13.2 Detailed description

Table 386:Parameters

Name	Direction	Description
p_ctrl	in	- Pointer to a control structure allocated by user.

8.12.13.3 Function steps

- Create the transmit mutex.
- Create the receive mutex.

8.12.14 tx_resource_release

```
ssp_err_t tx_resource_release ( sf_el_nx_comms_instance_ctrl_t  * p_ctrl )
```

8.12.14.1 Brief description

This is a helper routine called by Close API to release ThreadX resources.

8.12.14.2 Detailed description

Table 387:Parameters

Name	Direction	Description
p_ctrl	in	- Pointer to a control structure allocated by user.

8.12.14.3 Function steps

- Delete transmit mutex.
- Delete receive mutex.
- Delete event flags group.
- Delete queue instance.

8.12.15 receive_data

```
ssp_err_t receive_data ( sf_el_nx_comms_instance_ctrl_t  * p_ctrl ,   uint8_t
*const p_dest ,   uint32_t bytes ,   UINT const timeout )
```

8.12.15.1 Brief description

Called by the NetX Telnet Server on new Telnet client connection request.

8.12.15.2 Detailed description

Table 388:Parameters

Name	Direction	Description
p_ctrl	in	- Pointer to a control structure allocated by user.
p_dest	in	- Pointer to destination
bytes	in	- Bytes to transfer

Table 388:Parameters (Continued)

Name	Direction	Description
timeout	in	- Timeout period

8.12.16 telnet_new_connection

```
telnet_new_connection ( NX_TELNET_SERVER  * server_ptr ,
 UINT logical_connection )
```

8.12.16.1 Brief description

Called by the NetX Telnet Server on new Telnet client connection request.

8.12.16.2 Detailed description

Table 389:Parameters

Name	Direction	Description
server_ptr		- Telnet server instance pointer
logical_connection		- Logical connection for this request

8.12.16.3 Function steps

- Find which control block belongs to this server.
- Set event flag to show connection is complete.

8.12.17 telnet_connection_end

```
telnet_connection_end ( NX_TELNET_SERVER  * server_ptr ,
 UINT logical_connection )
```

8.12.17.1 Brief description

Called by the NetX Telnet Server on telnet client disconnect.

8.12.17.2 Detailed description

Table 390:Parameters

Name	Direction	Description
server_ptr		- Telnet server instance pointer
logical_connection		- Logical connection for this request

8.12.17.3 Function steps

- Clear event flag.

8.12.18 telnet_receive_data

```
telnet_receive_data ( NX_TELNET_SERVER  * server_ptr ,
 UINT logical_connection ,  NX_PACKET  * packet_ptr )
```

8.12.18.1 Brief description

Called by the NetX Telnet Server on data receive from Telnet client.

8.12.18.2 Detailed description

Table 391:Parameters

Name	Direction	Description
server_ptr		- Telnet server instance pointer
logical_connection		- Logical connection for this request
packet_ptr		- received packet pointer

8.12.18.3 Function steps

- Send packet to read function.

8.12.19 sf_el_nx_comms_error

```
ssp_err_t sf_el_nx_comms_error (  UINT status ,  ssp_err_t  err )
```

8.12.19.1 Brief description

Helper function log an error when callers return type is void to avoid compiler warnings.

8.12.19.2 Detailed description

Table 392:Parameters

Name	Direction	Description
status		- Actual returned status value
err		- Error type to log and return on error

8.12.20 Extensions

8.12.20.1 sf_el_nx_comms_instance_ctrl_t

sf_el_nx_comms_instance_ctrl_t

Detailed description

NetX Telnet server communications instance control structure. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open
- NX_PACKET * p_current_packet
- uint32_t packet_index
- TX_MUTEX mutex[2]
- NX_PACKET_POOL pool
- uint8_t pool_memory[SF_EL_NX_COMMS_PACKET_POOL_MEMORY_SIZE_BYTES]
- NX_IP ip
- uint8_t ip_memory[SF_EL_NX_COMMS_IP_MEMORY_SIZE_BYTES]
- uint8_t arp_memory[SF_EL_NX_COMMS_ARP_MEMORY_SIZE_BYTES]
- NX_TELNET_SERVER telnet_server
- uint8_t telnet_server_memory[SF_EL_NX_COMMS_TELNET_SERVER_MEMORY_SIZE_BYTES]
- UINT logical_connection
- TX_EVENT_FLAGS_GROUP available

  Flag to tell if this connection is available or not.

- TX_QUEUE queue

  Queue of received bytes.
- uint8_t queue_memory[SF_EL_NX_COMMS_QUEUE_MEMORY_SIZE_BYTES]

8.12.20.2 sf_el_nx_comms_on_comms_cfg_t

sf_el_nx_comms_on_comms_cfg_t

Detailed description

NetX Telnet server device communications driver configuration

Variables

- uint32_t ip_address
- uint32_t subnet_mask
- VOID(* driver)(NX_IP_DRIVER *driver_req_ptr)

8.13 SF_EL_UX Framework

RTOS-integrated SF_EL_UX Framework.

USBX Component Application Interface (API)

NOTE: This file exists to maintain backward compatibility with Renesas Synergy SSP projects created with versions prior to v1.1.0. DO NOT USE THIS FILE IN NEW PROJECTS. Use ux_dcd_synergy.h instead.

USBX Component SYNERGY Controller Driver

USBX Component CDC Class

USBX Component Device Data Pump Class

USBX Component HID Class

USBX Component Device Storage Class

NOTE: This file exists to maintain backward compatibility with Renesas Synergy SSP projects created with versions prior to SSP version 1.1.0. DO NOT USE THIS FILE IN NEW PROJECTS. Use ux_hcd_synergy.h instead.

USBX Component CDC ACM Class

USBX Component Host Data Pump Class

USBX Component HID Keyboard Client

USBX Component HID Mouse Client Class

USBX Component HID Remote Control Class

USBX Component HUB Class

USBX Component Storage Class

USBX Component Port Specific

8.13.1 Functions

- _ux_device_class_cdc_acm_activate
- _ux_device_class_cdc_acm_control_complete
- _ux_device_class_cdc_acm_control_request
- _ux_device_class_cdc_acm_deactivate
- _ux_device_class_cdc_acm_entry
- _ux_device_class_cdc_acm_initialize
- _ux_device_class_cdc_acm_uninitialize
- _ux_device_class_cdc_acm_write
- _ux_device_class_cdc_acm_read
- _ux_device_class_cdc_acm_thread
- _ux_device_class_cdc_acm_ioctl
- usbhs_usb_int_resume_isr
- usbfs_int_isr
- usbfs_resume_isr
- ux_dcd_synergy_buffer_empty_interrupt
- ux_dcd_synergy_buffer_notready_interrupt
- ux_dcd_synergy_buffer_read
- ux_dcd_synergy_buffer_ready_interrupt
- ux_dcd_synergy_buffer_write
- ux_dcd_synergy_current_endpoint_change
- ux_dcd_synergy_data_buffer_size
- ux_dcd_synergy_endpoint_create
- ux_dcd_synergy_endpoint_destroy
- ux_dcd_synergy_endpoint_nak_set
- ux_dcd_synergy_endpoint_reset
- ux_dcd_synergy_endpoint_stall
- ux_dcd_synergy_endpoint_status
- ux_dcd_synergy_fifo_port_change
- ux_dcd_synergy_fifo_read
- ux_dcd_synergy_read_dma_set
- ux_dcd_synergy_read_dma_set_16bit
- ux_dcd_synergy_fifo_read_dma_start
- ux_dcd_synergy_fifo_read_software_copy

- ux_dcd_synergy_fifo_read_software_copy_16bit
- ux_dcd_synergy_fifo_read_last_bytes
- ux_dcd_synergy_fifoc_write
- ux_dcd_synergy_fifo_write_software_copy
- ux_dcd_synergy_fifo_write_software_copy_16bit
- ux_dcd_synergy_fifo_write_last_bytes
- ux_dcd_synergy_fifod_write
- ux_dcd_synergy_write_dma_set
- ux_dcd_synergy_write_dma_set_16bit
- ux_dcd_synergy_fifo_write_dma_start
- ux_dcd_synergy_frame_number_get
- ux_dcd_synergy_function
- ux_dcd_synergy_initialize
- ux_dcd_synergy_initialize_transfer_support
- ux_dcd_synergy_initialize_common
- ux_dcd_synergy_initialize_common_complete
- ux_dcd_dma_complete_tx
- ux_dcd_dma_complete_rx
- ux_dcd_synergy_initialize_complete
- ux_dcd_synergy_interrupt_handler
- ux_dcd_synergy_register_clear
- ux_dcd_synergy_register_read
- ux_dcd_synergy_register_set
- ux_dcd_synergy_register_write
- ux_dcd_synergy_transfer_callback
- ux_dcd_synergy_transfer_request
- ux_hcd_synergy_asynch_queue_process
- ux_hcd_synergy_asynch_queue_process_bemp
- ux_hcd_synergy_asynch_queue_process_brdy
- ux_hcd_synergy_asynch_queue_process_nrdy
- ux_hcd_synergy_asynch_queue_process_sign
- ux_hcd_synergy_asynch_schedule
- ux_hcd_synergy_asynchronous_endpoint_create
- ux_hcd_synergy_asynchronous_endpoint_destroy

- ux_hcd_synergy_buffer_empty_interrupt
- ux_hcd_synergy_buffer_notready_interrupt
- ux_hcd_synergy_buffer_read
- ux_hcd_synergy_buffer_ready_interrupt
- ux_hcd_synergy_buffer_write
- ux_hcd_synergy_bulk_endpoint_create
- ux_hcd_synergy_bulk_int_td_add
- ux_hcd_synergy_control_endpoint_create
- ux_hcd_synergy_control_td_add
- ux_hcd_synergy_controller_disable
- ux_hcd_synergy_current_endpoint_change
- ux_hcd_synergy_data_buffer_size
- ux_hcd_synergy_ed_obtain
- ux_hcd_synergy_ed_td_clean
- ux_hcd_synergy_endpoint_nak_set
- ux_hcd_synergy_endpoint_reset
- ux_hcd_synergy_entry
- ux_hcd_synergy_fifo_port_change
- ux_hcd_synergy_fifo_read
- ux_hcd_synergy_fifo_read_software_copy
- ux_hcd_synergy_fifo_read_software_copy_16bit
- ux_hcd_synergy_fifo_read_software_copy_8bit
- ux_hcd_synergy_fifo_read_dma_start
- ux_hcd_synergy_fifo_read_dma_set_16bit_or_8bit
- ux_hcd_synergy_fifoc_write
- ux_hcd_synergy_fifo_write_software_copy
- ux_hcd_synergy_fifo_write_software_copy_16bit
- ux_hcd_synergy_fifo_write_software_copy_8bit
- ux_hcd_synergy_fifo_write_software_copy_remaining_bytes
- ux_hcd_synergy_fifod_write
- ux_hcd_synergy_fifod_write_dma_start
- ux_hcd_synergy_fifod_write_dma_set_16bit_or_8bit
- ux_hcd_synergy_frame_number_get
- ux_hcd_synergy_frame_number_set

- ux_hcd_synergy_initialize
- ux_hcd_synergy_initialize_common
- ux_hcd_synergy_initialize_transfer_support
- ux_hcd_synergy_initialize_common_complete
- ux_hcd_dma_complete_tx
- ux_hcd_dma_complete_rx
- ux_hcd_synergy_interrupt_endpoint_create
- ux_hcd_synergy_interrupt_handler
- ux_hcd_synergy_iso_queue_process
- ux_hcd_synergy_iso_schedule
- ux_hcd_synergy_isochronous_endpoint_create
- ux_hcd_synergy_isochronous_td_obtain
- ux_hcd_synergy_least_traffic_list_get
- ux_hcd_synergy_periodic_endpoint_destroy
- ux_hcd_synergy_periodic_schedule
- ux_hcd_synergy_periodic_tree_create
- ux_hcd_synergy_port_disable
- ux_hcd_synergy_port_enable
- ux_hcd_synergy_port_reset
- ux_hcd_synergy_port_resume
- ux_hcd_synergy_port_status_get
- ux_hcd_synergy_port_suspend
- ux_hcd_synergy_power_down_port
- ux_hcd_synergy_power_on_port
- ux_hcd_synergy_power_root_hubs
- ux_hcd_synergy_register_clear
- ux_hcd_synergy_register_read
- ux_hcd_synergy_register_set
- ux_hcd_synergy_register_write
- ux_hcd_synergy_regular_td_obtain
- ux_hcd_synergy_request_bulk_transfer
- ux_hcd_synergy_request_control_transfer
- ux_hcd_synergy_request_interrupt_transfer
- ux_hcd_synergy_request_isochronous_transfer

- ux_hcd_synergy_request_transfer
- ux_hcd_synergy_td_add
- ux_hcd_synergy_transfer_abort

8.13.2 Typedefs

- SCHAR
- SLONG

8.13.3 Defines

- #define UX_PARAMETER_NOT_USED
  Initial value:`((VOID)(p))`
- #define UX_MAX_SLAVE_INTERFACES
  Initial value:`16`
- #define UX_DEBUG_LOG
  Initial value:
- #define UX_TRACE_OBJECT_REGISTER
  Initial value:
- #define UX_TRACE_OBJECT_UNREGISTER
  Initial value:
- #define UX_TRACE_IN_LINE_INSERT
  Initial value:
- #define UX_TRACE_EVENT_UPDATE
  Initial value:
- #define UX_SYSTEM_LEVEL_INTERRUPT
  Initial value:`1`
- #define UX_SYSTEM_LEVEL_THREAD
  Initial value:`2`
- #define UX_SYSTEM_CONTEXT_HCD
  Initial value:`1`
- #define UX_SYSTEM_CONTEXT_DCD
  Initial value:`2`
- #define UX_SYSTEM_CONTEXT_INIT
  Initial value:`3`

- #define UX_SYSTEM_CONTEXT_ENUMERATOR
  Initial value:`4`
- #define UX_SYSTEM_CONTEXT_ROOT_HUB
  Initial value:`5`
- #define UX_SYSTEM_CONTEXT_HUB
  Initial value:`6`
- #define UX_SYSTEM_CONTEXT_CLASS
  Initial value:`7`
- #define UX_SYSTEM_CONTEXT_UTILITY
  Initial value:`8`
- #define UX_NULL
  Initial value:`((void*)0)`
- #define UX_TRUE
  Initial value:`1`
- #define UX_FALSE
  Initial value:`0`
- #define UX_MAX_TT
  Initial value:`8`
- #define UX_TT_MASK
  Initial value:`0x1f7`
- #define UX_TT_BANDWIDTH
  Initial value:`900`
- #define UX_SLAVE_ENDPOINT_DEFAULT_BUFFER_SIZE
  Initial value:`256`
- #define UX_REGULAR_MEMORY
  Initial value:`0`
- #define UX_CACHE_SAFE_MEMORY
  Initial value:`1`
- #define UX_SETUP_REQUEST_TYPE
  Initial value:`0`
- #define UX_SETUP_VALUE
  Initial value:`2`
- #define UX_SETUP_INDEX
  Initial value:`4`

- #define UX_SETUP_LENGTH
  Initial value:6
- #define UX_SETUP_SIZE
  Initial value:8
- #define UX_GET_STATUS
  Initial value:0
- #define UX_CLEAR_FEATURE
  Initial value:1
- #define UX_SET_FEATURE
  Initial value:3
- #define UX_SET_ADDRESS
  Initial value:5
- #define UX_GET_DESCRIPTOR
  Initial value:6
- #define UX_SET_DESCRIPTOR
  Initial value:7
- #define UX_GET_CONFIGURATION
  Initial value:8
- #define UX_SET_CONFIGURATION
  Initial value:9
- #define UX_GET_INTERFACE
  Initial value:10
- #define UX_SET_INTERFACE
  Initial value:11
- #define UX_SYNCH_FRAME
  Initial value:12
- #define UX_ENDPOINT_HALT
  Initial value:0
- #define UX_WAIT_FOREVER
  Initial value:0xffffffff
- #define UX_UNUSED
  Initial value:0
- #define UX_USED
  Initial value:1

- #define UX_MEMORY_UNUSED
  Initial value:`0x12345678`
- #define UX_MEMORY_USED
  Initial value:`0x87654321`
- #define UX_NO_ALIGN
  Initial value:`0`
- #define UX_ALIGN_16
  Initial value:`0x0f`
- #define UX_ALIGN_MIN
  Initial value:`0x0f`
- #define UX_ALIGN_32
  Initial value:`0x1f`
- #define UX_ALIGN_64
  Initial value:`0x3f`
- #define UX_ALIGN_128
  Initial value:`0x7f`
- #define UX_ALIGN_256
  Initial value:`0xff`
- #define UX_ALIGN_512
  Initial value:`0x1ff`
- #define UX_ALIGN_1024
  Initial value:`0x3ff`
- #define UX_ALIGN_2048
  Initial value:`0x7ff`
- #define UX_ALIGN_4096
  Initial value:`0xfff`
- #define UX_SAFE_ALIGN
  Initial value:`0xffffffff`
- #define UX_MAX_SCATTER_GATHER_ALIGNMENT
  Initial value:`4096`
- #define UX_MAX_USB_DEVICES
  Initial value:`127`
- #define UX_ENDPOINT_DIRECTION
  Initial value:`0x80`

- #define UX_ENDPOINT_IN
  Initial value:`0x80`
- #define UX_ENDPOINT_OUT
  Initial value:`0x00`
- #define UX_MASK_ENDPOINT_TYPE
  Initial value:`3`
- #define UX_CONTROL_ENDPOINT
  Initial value:`0`
- #define UX_ISOCHRONOUS_ENDPOINT
  Initial value:`1`
- #define UX_BULK_ENDPOINT
  Initial value:`2`
- #define UX_INTERRUPT_ENDPOINT
  Initial value:`3`
- #define UX_ISOCHRONOUS_ENDPOINT_IN
  Initial value:`0x81`
- #define UX_ISOCHRONOUS_ENDPOINT_OUT
  Initial value:`0x01`
- #define UX_BULK_ENDPOINT_IN
  Initial value:`0x82`
- #define UX_BULK_ENDPOINT_OUT
  Initial value:`0x02`
- #define UX_INTERRUPT_ENDPOINT_IN
  Initial value:`0x83`
- #define UX_INTERRUPT_ENDPOINT_OUT
  Initial value:`0x03`
- #define UX_REQUEST_DIRECTION
  Initial value:`0x80`
- #define UX_REQUEST_IN
  Initial value:`0x80`
- #define UX_REQUEST_OUT
  Initial value:`0x00`
- #define UX_REQUEST_TYPE
  Initial value:`0x60`

- #define UX_REQUEST_TYPE_STANDARD
  Initial value:0x00
- #define UX_REQUEST_TYPE_CLASS
  Initial value:0x20
- #define UX_REQUEST_TYPE_VENDOR
  Initial value:0x40
- #define UX_REQUEST_TARGET
  Initial value:0x03
- #define UX_REQUEST_TARGET_DEVICE
  Initial value:0x00
- #define UX_REQUEST_TARGET_INTERFACE
  Initial value:0x01
- #define UX_REQUEST_TARGET_ENDPOINT
  Initial value:0x02
- #define UX_REQUEST_TARGET_OTHER
  Initial value:0x03
- #define UX_DEVICE_RESET
  Initial value:0
- #define UX_DEVICE_ATTACHED
  Initial value:1
- #define UX_DEVICE_ADDRESSED
  Initial value:2
- #define UX_DEVICE_CONFIGURED
  Initial value:3
- #define UX_DEVICE_SUSPENDED
  Initial value:4
- #define UX_DEVICE_RESUMED
  Initial value:5
- #define UX_DEVICE_SELF_POWERED_STATE
  Initial value:6
- #define UX_DEVICE_BUS_POWERED_STATE
  Initial value:7
- #define UX_DEVICE_REMOTE_WAKEUP
  Initial value:8

- #define UX_DEVICE_BUS_RESET_COMPLETED
  Initial value:9
- #define UX_DEVICE_REMOVED
  Initial value:10
- #define UX_DEVICE_FORCE_DISCONNECT
  Initial value:11
- #define UX_ENDPOINT_RESET
  Initial value:0
- #define UX_ENDPOINT_RUNNING
  Initial value:1
- #define UX_ENDPOINT_HALTED
  Initial value:2
- #define UX_DEVICE_DESCRIPTOR_ITEM
  Initial value:1
- #define UX_CONFIGURATION_DESCRIPTOR_ITEM
  Initial value:2
- #define UX_STRING_DESCRIPTOR_ITEM
  Initial value:3
- #define UX_INTERFACE_DESCRIPTOR_ITEM
  Initial value:4
- #define UX_ENDPOINT_DESCRIPTOR_ITEM
  Initial value:5
- #define UX_DEVICE_QUALIFIER_DESCRIPTOR_ITEM
  Initial value:6
- #define UX_OTHER_SPEED_DESCRIPTOR_ITEM
  Initial value:7
- #define UX_OTG_DESCRIPTOR_ITEM
  Initial value:9
- #define UX_INTERFACE_ASSOCIATION_DESCRIPTOR_ITEM
  Initial value:11
- #define UX_DFU_FUNCTIONAL_DESCRIPTOR_ITEM
  Initial value:0x21
- #define UX_HUB_DESCRIPTOR_ITEM
  Initial value:0x29

- #define UX_CONTROL_TRANSFER_TIMEOUT_IN_MS
  Initial value:`10000`
- #define UX_NON_CONTROL_TRANSFER_TIMEOUT_IN_MS
  Initial value:`50000`
- #define UX_PORT_ENABLE_WAIT_IN_MS
  Initial value:`500`
- #define UX_DEVCIE_ADDRESS_SET_WAIT_IN_MS
  Initial value:`500`
- #define UX_DEVICE_ADDRESS_SET_WAIT
  Initial value:`((UX_DEVCIE_ADDRESS_SET_WAIT_IN_MS) * (UX_PERIODIC_RATE) / 1000)`
- #define UX_HIGH_SPEED_DETECTION_HANDSHAKE_SUSPEND_WAIT_IN_MS
  Initial value:`2000`
- #define UX_ENUMERATION_THREAD_WAIT_IN_MS
  Initial value:`2000`
- #define UX_TRANSFER_PHASE_SETUP
  Initial value:`1`
- #define UX_TRANSFER_PHASE_DATA_IN
  Initial value:`2`
- #define UX_TRANSFER_PHASE_DATA_OUT
  Initial value:`3`
- #define UX_TRANSFER_PHASE_STATUS_IN
  Initial value:`4`
- #define UX_TRANSFER_PHASE_STATUS_OUT
  Initial value:`5`
- #define UX_DEVICE_INSERTION
  Initial value:`1`
- #define UX_DEVICE_REMOVAL
  Initial value:`2`
- #define UX_TRANSFER_STATUS_NOT_PENDING
  Initial value:`0`
- #define UX_TRANSFER_STATUS_PENDING
  Initial value:`1`
- #define UX_TRANSFER_STATUS_COMPLETED
  Initial value:`2`

- #define UX_TRANSFER_STATUS_ABORT
  Initial value:4
- #define UX_MAX_SELF_POWER
  Initial value:(500/2)
- #define UX_MAX_BUS_POWER
  Initial value:(100/2)
- #define UX_CONFIGURATION_DEVICE_BUS_POWERED
  Initial value:0x80
- #define UX_CONFIGURATION_DEVICE_SELF_POWERED
  Initial value:0x40
- #define UX_STATUS_DEVICE_SELF_POWERED
  Initial value:1
- #define UX_OTG_BM_ATTRIBUTES
  Initial value:2
- #define UX_OTG_SRP_SUPPORT
  Initial value:1
- #define UX_OTG_HNP_SUPPORT
  Initial value:2
- #define UX_HCD_OTG_CAPABLE
  Initial value:1
- #define UX_DCD_OTG_CAPABLE
  Initial value:1
- #define UX_OTG_FEATURE_B_HNP_ENABLE
  Initial value:3
- #define UX_OTG_FEATURE_A_HNP_SUPPORT
  Initial value:4
- #define UX_OTG_FEATURE_A_ALT_HNP_SUPPORT
  Initial value:5
- #define UX_OTG_STATUS_SELECTOR
  Initial value:0xF000
- #define UX_OTG_HOST_REQUEST_FLAG
  Initial value:0x01
- #define UX_OTG_IDLE
  Initial value:0

- #define UX_OTG_IDLE_TO_HOST
  Initial value:1
- #define UX_OTG_IDLE_TO_SLAVE
  Initial value:2
- #define UX_OTG_HOST_TO_IDLE
  Initial value:3
- #define UX_OTG_HOST_TO_SLAVE
  Initial value:4
- #define UX_OTG_SLAVE_TO_IDLE
  Initial value:5
- #define UX_OTG_SLAVE_TO_HOST
  Initial value:6
- #define UX_OTG_SLAVE_SRP
  Initial value:7
- #define UX_OTG_MODE_IDLE
  Initial value:0
- #define UX_OTG_MODE_SLAVE
  Initial value:1
- #define UX_OTG_MODE_HOST
  Initial value:2
- #define UX_OTG_DEVICE_IDLE
  Initial value:0
- #define UX_OTG_DEVICE_A
  Initial value:1
- #define UX_OTG_DEVICE_B
  Initial value:2
- #define UX_OTG_VBUS_IDLE
  Initial value:0
- #define UX_OTG_VBUS_ON
  Initial value:1
- #define UX_OTG_VBUS_OFF
  Initial value:2
- #define UX_OTG_HNP_THREAD_SLEEP_TIME
  Initial value:(2 * UX_PERIODIC_RATE)

- #define UX_DEFAULT_HS_MPS
  Initial value:64
- #define UX_DEFAULT_MPS
  Initial value:8
- #define UX_LOW_SPEED_DEVICE
  Initial value:0
- #define UX_FULL_SPEED_DEVICE
  Initial value:1
- #define UX_HIGH_SPEED_DEVICE
  Initial value:2
- #define UX_PS_CCS
  Initial value:0x01
- #define UX_PS_CPE
  Initial value:0x01
- #define UX_PS_PES
  Initial value:0x02
- #define UX_PS_PSS
  Initial value:0x04
- #define UX_PS_POCI
  Initial value:0x08
- #define UX_PS_PRS
  Initial value:0x10
- #define UX_PS_PPS
  Initial value:0x20
- #define UX_PS_DS_LS
  Initial value:0x00
- #define UX_PS_DS_FS
  Initial value:0x40
- #define UX_PS_DS_HS
  Initial value:0x80
- #define UX_SUCCESS
  Initial value:0
- #define UX_ERROR
  Initial value:0xff

- #define UX_TOO_MANY_DEVICES
  Initial value:`0x11`
- #define UX_MEMORY_INSUFFICIENT
  Initial value:`0x12`
- #define UX_NO_TD_AVAILABLE
  Initial value:`0x13`
- #define UX_NO_ED_AVAILABLE
  Initial value:`0x14`
- #define UX_SEMAPHORE_ERROR
  Initial value:`0x15`
- #define UX_THREAD_ERROR
  Initial value:`0x16`
- #define UX_MUTEX_ERROR
  Initial value:`0x17`
- #define UX_EVENT_ERROR
  Initial value:`0x18`
- #define UX_MEMORY_CORRUPTED
  Initial value:`0x19`
- #define UX_MEMORY_ARRAY_FULL
  Initial value:`0x1a`
- #define UX_TRANSFER_STALLED
  Initial value:`0x21`
- #define UX_TRANSFER_NO_ANSWER
  Initial value:`0x22`
- #define UX_TRANSFER_ERROR
  Initial value:`0x23`
- #define UX_TRANSFER_MISSED_FRAME
  Initial value:`0x24`
- #define UX_TRANSFER_NOT_READY
  Initial value:`0x25`
- #define UX_TRANSFER_BUS_RESET
  Initial value:`0x26`
- #define UX_TRANSFER_BUFFER_OVERFLOW
  Initial value:`0x27`

- #define UX_TRANSFER_APPLICATION_RESET
  Initial value:`0x28`
- #define UX_PORT_RESET_FAILED
  Initial value:`0x31`
- #define UX_CONTROLLER_INIT_FAILED
  Initial value:`0x32`
- #define UX_CONTROLLER_DEAD
  Initial value:`0x33`
- #define UX_NO_BANDWIDTH_AVAILABLE
  Initial value:`0x41`
- #define UX_DESCRIPTOR_CORRUPTED
  Initial value:`0x42`
- #define UX_OVER_CURRENT_CONDITION
  Initial value:`0x43`
- #define UX_DEVICE_ENUMERATION_FAILURE
  Initial value:`0x44`
- #define UX_DEVICE_HANDLE_UNKNOWN
  Initial value:`0x50`
- #define UX_CONFIGURATION_HANDLE_UNKNOWN
  Initial value:`0x51`
- #define UX_INTERFACE_HANDLE_UNKNOWN
  Initial value:`0x52`
- #define UX_ENDPOINT_HANDLE_UNKNOWN
  Initial value:`0x53`
- #define UX_FUNCTION_NOT_SUPPORTED
  Initial value:`0x54`
- #define UX_CONTROLLER_UNKNOWN
  Initial value:`0x55`
- #define UX_PORT_INDEX_UNKNOWN
  Initial value:`0x56`
- #define UX_NO_CLASS_MATCH
  Initial value:`0x57`
- #define UX_HOST_CLASS_ALREADY_INSTALLED
  Initial value:`0x58`

- #define UX_HOST_CLASS_UNKNOWN
  Initial value:`0x59`
- #define UX_CONNECTION_INCOMPATIBLE
  Initial value:`0x5a`
- #define UX_HOST_CLASS_INSTANCE_UNKNOWN
  Initial value:`0x5b`
- #define UX_TRANSFER_TIMEOUT
  Initial value:`0x5c`
- #define UX_BUFFER_OVERFLOW
  Initial value:`0x5d`
- #define UX_NO_ALTERNATE_SETTING
  Initial value:`0x5e`
- #define UX_NO_DEVICE_CONNECTED
  Initial value:`0x5f`
- #define UX_HOST_CLASS_PROTOCOL_ERROR
  Initial value:`0x60`
- #define UX_HOST_CLASS_MEMORY_ERROR
  Initial value:`0x61`
- #define UX_HOST_CLASS_MEDIA_NOT_SUPPORTED
  Initial value:`0x62`
- #define UX_HOST_CLASS_HID_REPORT_OVERFLOW
  Initial value:`0x70`
- #define UX_HOST_CLASS_HID_USAGE_OVERFLOW
  Initial value:`0x71`
- #define UX_HOST_CLASS_HID_TAG_UNSUPPORTED
  Initial value:`0x72`
- #define UX_HOST_CLASS_HID_PUSH_OVERFLOW
  Initial value:`0x73`
- #define UX_HOST_CLASS_HID_POP_UNDERFLOW
  Initial value:`0x74`
- #define UX_HOST_CLASS_HID_COLLECTION_OVERFLOW
  Initial value:`0x75`
- #define UX_HOST_CLASS_HID_COLLECTION_UNDERFLOW
  Initial value:`0x76`

- #define UX_HOST_CLASS_HID_MIN_MAX_ERROR
  Initial value:0x77
- #define UX_HOST_CLASS_HID_DELIMITER_ERROR
  Initial value:0x78
- #define UX_HOST_CLASS_HID_REPORT_ERROR
  Initial value:0x79
- #define UX_HOST_CLASS_HID_PERIODIC_REPORT_ERROR
  Initial value:0x7A
- #define UX_HOST_CLASS_HID_UNKNOWN
  Initial value:0x7B
- #define UX_HOST_CLASS_AUDIO_WRONG_TYPE
  Initial value:0x80
- #define UX_HOST_CLASS_AUDIO_WRONG_INTERFACE
  Initial value:0x81
- #define UX_HCD_DISABLE_CONTROLLER
  Initial value:1
- #define UX_HCD_GET_PORT_STATUS
  Initial value:2
- #define UX_HCD_ENABLE_PORT
  Initial value:3
- #define UX_HCD_DISABLE_PORT
  Initial value:4
- #define UX_HCD_POWER_ON_PORT
  Initial value:5
- #define UX_HCD_POWER_DOWN_PORT
  Initial value:6
- #define UX_HCD_SUSPEND_PORT
  Initial value:7
- #define UX_HCD_RESUME_PORT
  Initial value:8
- #define UX_HCD_RESET_PORT
  Initial value:9
- #define UX_HCD_GET_FRAME_NUMBER
  Initial value:10

- #define UX_HCD_SET_FRAME_NUMBER
  Initial value:11
- #define UX_HCD_TRANSFER_REQUEST
  Initial value:12
- #define UX_HCD_TRANSFER_ABORT
  Initial value:13
- #define UX_HCD_CREATE_ENDPOINT
  Initial value:14
- #define UX_HCD_DESTROY_ENDPOINT
  Initial value:15
- #define UX_HCD_RESET_ENDPOINT
  Initial value:16
- #define UX_HCD_PROCESS_DONE_QUEUE
  Initial value:17
- #define UX_DCD_DISABLE_CONTROLLER
  Initial value:1
- #define UX_DCD_GET_PORT_STATUS
  Initial value:2
- #define UX_DCD_ENABLE_PORT
  Initial value:3
- #define UX_DCD_DISABLE_PORT
  Initial value:4
- #define UX_DCD_POWER_ON_PORT
  Initial value:5
- #define UX_DCD_POWER_DOWN_PORT
  Initial value:6
- #define UX_DCD_SUSPEND_PORT
  Initial value:7
- #define UX_DCD_RESUME_PORT
  Initial value:8
- #define UX_DCD_RESET_PORT
  Initial value:9
- #define UX_DCD_GET_FRAME_NUMBER
  Initial value:10

- #define UX_DCD_SET_FRAME_NUMBER
  Initial value:11
- #define UX_DCD_TRANSFER_REQUEST
  Initial value:12
- #define UX_DCD_TRANSFER_ABORT
  Initial value:13
- #define UX_DCD_CREATE_ENDPOINT
  Initial value:14
- #define UX_DCD_DESTROY_ENDPOINT
  Initial value:15
- #define UX_DCD_RESET_ENDPOINT
  Initial value:16
- #define UX_DCD_SET_DEVICE_ADDRESS
  Initial value:17
- #define UX_DCD_ISR_PENDING
  Initial value:18
- #define UX_DCD_CHANGE_STATE
  Initial value:19
- #define UX_DCD_STALL_ENDPOINT
  Initial value:20
- #define UX_DCD_ENDPOINT_STATUS
  Initial value:21
- #define UX_HCD_STATUS_HALTED
  Initial value:0
- #define UX_HCD_STATUS_OPERATIONAL
  Initial value:1
- #define UX_HCD_STATUS_DEAD
  Initial value:2
- #define UX_DCD_STATUS_HALTED
  Initial value:0
- #define UX_DCD_STATUS_OPERATIONAL
  Initial value:1
- #define UX_DCD_STATUS_DEAD
  Initial value:2

- #define UX_DCD_VBUS_RESET
  Initial value:0
- #define UX_DCD_VBUS_SET
  Initial value:1
- #define UX_HOST_CLASS_COMMAND_QUERY
  Initial value:1
- #define UX_HOST_CLASS_COMMAND_ACTIVATE
  Initial value:2
- #define UX_HOST_CLASS_COMMAND_DEACTIVATE
  Initial value:3
- #define UX_SLAVE_CLASS_COMMAND_QUERY
  Initial value:1
- #define UX_SLAVE_CLASS_COMMAND_ACTIVATE
  Initial value:2
- #define UX_SLAVE_CLASS_COMMAND_DEACTIVATE
  Initial value:3
- #define UX_SLAVE_CLASS_COMMAND_REQUEST
  Initial value:4
- #define UX_SLAVE_CLASS_COMMAND_INITIALIZE
  Initial value:5
- #define UX_SLAVE_CLASS_COMMAND_CHANGE
  Initial value:6
- #define UX_SLAVE_CLASS_COMMAND_UNINITIALIZE
  Initial value:7
- #define UX_HOST_CLASS_COMMAND_USAGE_PIDVID
  Initial value:1
- #define UX_HOST_CLASS_COMMAND_USAGE_CSP
  Initial value:2
- #define UX_HOST_CLASS_INSTANCE_FREE
  Initial value:0
- #define UX_HOST_CLASS_INSTANCE_LIVE
  Initial value:1
- #define UX_HOST_CLASS_INSTANCE_SHUTDOWN
  Initial value:2

- #define UX_HOST_CLASS_INSTANCE_MOUNTING
  Initial value:`3`
- #define UX_RH_ENUMERATION_RETRY
  Initial value:`3`
- #define UX_RH_ENUMERATION_RETRY_DELAY
  Initial value:`100`
- #define UX_PCI_NB_FUNCTIONS
  Initial value:`7`
- #define UX_PCI_NB_DEVICE
  Initial value:`32`
- #define UX_PCI_NB_BUS
  Initial value:`0xff`
- #define UX_PCI_CMD_IO_ENABLE
  Initial value:`0x0001`
- #define UX_PCI_CMD_MEM_ENABLE
  Initial value:`0x0002`
- #define UX_PCI_CMD_MASTER_ENABLE
  Initial value:`0x0004`
- #define UX_PCI_CMD_MONITOR_ENABLE
  Initial value:`0x0008`
- #define UX_PCI_CMD_MEM_WRITE_INV_ENABLE
  Initial value:`0x0010`
- #define UX_PCI_CMD_SNOOP_PALETTE_ENABLE
  Initial value:`0x0020`
- #define UX_PCI_CMD_PARITY_ERROR_ENABLE
  Initial value:`0x0040`
- #define UX_PCI_CMD_WAIT_CYCLE_CTRL_ENABLE
  Initial value:`0x0080`
- #define UX_PCI_CMD_SERR_ENABLE
  Initial value:`0x0100`
- #define UX_PCI_CMD_FBB_ENABLE
  Initial value:`0x0200`
- #define UX_PCI_CFG_CTRL_ADDRESS
  Initial value:`0x0cf8`

- #define UX_PCI_CFG_DATA_ADDRESS
  Initial value:`0x0cfc`
- #define UX_PCI_CFG_VENDOR_ID
  Initial value:`0x00`
- #define UX_PCI_CFG_DEVICE_ID
  Initial value:`0x02`
- #define UX_PCI_CFG_COMMAND
  Initial value:`0x04`
- #define UX_PCI_CFG_STATUS
  Initial value:`0x06`
- #define UX_PCI_CFG_REVISION
  Initial value:`0x08`
- #define UX_PCI_CFG_PROGRAMMING_IF
  Initial value:`0x09`
- #define UX_PCI_CFG_SUBCLASS
  Initial value:`0x0a`
- #define UX_PCI_CFG_CLASS
  Initial value:`0x0b`
- #define UX_PCI_CFG_CACHE_LINE_SIZE
  Initial value:`0x0c`
- #define UX_PCI_CFG_LATENCY_TIMER
  Initial value:`0x0d`
- #define UX_PCI_CFG_HEADER_TYPE
  Initial value:`0x0e`
- #define UX_PCI_CFG_BIST
  Initial value:`0x0f`
- #define UX_PCI_CFG_BASE_ADDRESS_0
  Initial value:`0x10`
- #define UX_PCI_CFG_BASE_ADDRESS_1
  Initial value:`0x14`
- #define UX_PCI_CFG_BASE_ADDRESS_2
  Initial value:`0x18`
- #define UX_PCI_CFG_BASE_ADDRESS_3
  Initial value:`0x1c`

- #define UX_PCI_CFG_BASE_ADDRESS_4
  Initial value:`0x20`
- #define UX_PCI_CFG_BASE_ADDRESS_5
  Initial value:`0x24`
- #define UX_PCI_CFG_CARDBUS_CIS
  Initial value:`0x28`
- #define UX_PCI_CFG_SUB_VENDOR_ID
  Initial value:`0x2c`
- #define UX_PCI_CFG_SUB_SYSTEM_ID
  Initial value:`0x2e`
- #define UX_PCI_CFG_EXPANSION_ROM_ADDRESS
  Initial value:`0x30`
- #define UX_PCI_CFG_RESERVED_0
  Initial value:`0x34`
- #define UX_PCI_CFG_RESERVED_1
  Initial value:`0x38`
- #define UX_PCI_CFG_INT_LINE
  Initial value:`0x3c`
- #define UX_PCI_CFG_INT_PIN
  Initial value:`0x3d`
- #define UX_PCI_CFG_MIN_GNT
  Initial value:`0x3e`
- #define UX_PCI_CFG_MAX_LATENCY
  Initial value:`0x3f`
- #define UX_PCI_CFG_SBRN
  Initial value:`0x60`
- #define UX_PCI_CFG_FLADJ
  Initial value:`0x61`
- #define UX_SYSTEM_DFU_STATE_APP_IDLE
  Initial value:`0`
- #define UX_SYSTEM_DFU_STATE_APP_DETACH
  Initial value:`1`
- #define UX_SYSTEM_DFU_STATE_DFU_IDLE
  Initial value:`2`

- #define UX_SYSTEM_DFU_STATE_DFU_DNLOAD_SYNC
  Initial value:3
- #define UX_SYSTEM_DFU_STATE_DFU_DNBUSY
  Initial value:4
- #define UX_SYSTEM_DFU_STATE_DFU_DNLOAD_IDLE
  Initial value:5
- #define UX_SYSTEM_DFU_STATE_DFU_MANIFEST_SYNC
  Initial value:6
- #define UX_SYSTEM_DFU_STATE_DFU_MANIFEST_WAIT_RESET
  Initial value:8
- #define UX_SYSTEM_DFU_STATE_DFU_UPLOAD_IDLE
  Initial value:9
- #define UX_SYSTEM_DFU_STATE_DFU_ERROR
  Initial value:10
- #define UX_HOST_CLASS_PRINTER_NAME_LENGTH
  Initial value:64
- #define UX_ENDPOINT_DESCRIPTOR_ENTRIES
  Initial value:6
- #define UX_ENDPOINT_DESCRIPTOR_LENGTH
  Initial value:7
- #define UX_DEVICE_DESCRIPTOR_ENTRIES
  Initial value:14
- #define UX_DEVICE_DESCRIPTOR_LENGTH
  Initial value:18
- #define UX_DEVICE_QUALIFIER_DESCRIPTOR_ENTRIES
  Initial value:9
- #define UX_DEVICE_QUALIFIER_DESCRIPTOR_LENGTH
  Initial value:10
- #define UX_OTHER_SPEED_DESCRIPTOR_ENTRIES
  Initial value:8
- #define UX_OTHER_SPEED_DESCRIPTOR_LENGTH
  Initial value:9
- #define UX_OTG_DESCRIPTOR_ENTRIES
  Initial value:4

- #define UX_OTG_DESCRIPTOR_LENGTH
  Initial value:5
- #define UX_INTERFACE_ASSOCIATION_DESCRIPTOR_ENTRIES
  Initial value:8
- #define UX_INTERFACE_ASSOCIATION_DESCRIPTOR_LENGTH
  Initial value:8
- #define UX_CONFIGURATION_DESCRIPTOR_ENTRIES
  Initial value:8
- #define UX_CONFIGURATION_DESCRIPTOR_LENGTH
  Initial value:9
- #define UX_INTERFACE_DESCRIPTOR_ENTRIES
  Initial value:9
- #define UX_INTERFACE_DESCRIPTOR_LENGTH
  Initial value:9
- #define UX_STRING_DESCRIPTOR_ENTRIES
  Initial value:3
- #define UX_STRING_DESCRIPTOR_LENGTH
  Initial value:4
- #define UX_DFU_FUNCTIONAL_DESCRIPTOR_ENTRIES
  Initial value:6
- #define UX_DFU_FUNCTIONAL_DESCRIPTOR_LENGTH
  Initial value:9
- #define ux_system_initialize
  Initial value:`_ux_system_initialize`
- #define ux_system_uninitialize
  Initial value:`_ux_system_uninitialize`
- #define ux_host_class_hub_entry
  Initial value:`_ux_host_class_hub_entry`
- #define ux_host_class_storage_entry
  Initial value:`_ux_host_class_storage_entry`
- #define ux_host_stack_class_get
  Initial value:`_ux_host_stack_class_get`
- #define ux_host_stack_class_instance_create
  Initial value:`_ux_host_stack_class_instance_create`

- #define ux_host_stack_class_instance_destroy
  Initial value:`_ux_host_stack_class_instance_destroy`
- #define ux_host_stack_class_instance_get
  Initial value:`_ux_host_stack_class_instance_get`
- #define ux_host_stack_class_register
  Initial value:`_ux_host_stack_class_register`
- #define ux_host_stack_configuration_interface_get
  Initial value:`_ux_host_stack_configuration_interface_get`
- #define ux_host_stack_device_configuration_get
  Initial value:`_ux_host_stack_device_configuration_get`
- #define ux_host_stack_device_configuration_select
  Initial value:`_ux_host_stack_device_configuration_select`
- #define ux_host_stack_device_get
  Initial value:`_ux_host_stack_device_get`
- #define ux_host_stack_endpoint_transfer_abort
  Initial value:`_ux_host_stack_endpoint_transfer_abort`
- #define ux_host_stack_hcd_register
  Initial value:`_ux_host_stack_hcd_register`
- #define ux_host_stack_initialize
  Initial value:`_ux_host_stack_initialize`
- #define ux_host_stack_interface_endpoint_get
  Initial value:`_ux_host_stack_interface_endpoint_get`
- #define ux_host_stack_interface_setting_select
  Initial value:`_ux_host_stack_interface_setting_select`
- #define ux_host_stack_transfer_request
  Initial value:`_ux_host_stack_transfer_request`
- #define ux_host_stack_transfer_request_abort
  Initial value:`_ux_host_stack_transfer_request_abort`
- #define ux_host_stack_hnp_polling_thread_entry
  Initial value:`_ux_host_stack_hnp_polling_thread_entry`
- #define ux_host_stack_role_swap
  Initial value:`_ux_host_stack_role_swap`
- #define ux_utility_pci_class_scan
  Initial value:`_ux_utility_pci_class_scan`

- #define ux_utility_pci_read
  Initial value:`_ux_utility_pci_read`
- #define ux_utility_pci_write
  Initial value:`_ux_utility_pci_write`
- #define ux_device_stack_alternate_setting_get
  Initial value:`_ux_device_stack_alternate_setting_get`
- #define ux_device_stack_alternate_setting_set
  Initial value:`_ux_device_stack_alternate_setting_set`
- #define ux_device_stack_class_register
  Initial value:`_ux_device_stack_class_register`
- #define ux_device_stack_class_unregister
  Initial value:`_ux_device_stack_class_unregister`
- #define ux_device_stack_configuration_get
  Initial value:`_ux_device_stack_configuration_get`
- #define ux_device_stack_configuration_set
  Initial value:`_ux_device_stack_configuration_set`
- #define ux_device_stack_descriptor_send
  Initial value:`_ux_device_stack_descriptor_send`
- #define ux_device_stack_connect
  Initial value:`_ux_device_stack_connect`
- #define ux_device_stack_disconnect
  Initial value:`_ux_device_stack_disconnect`
- #define ux_device_stack_endpoint_stall
  Initial value:`_ux_device_stack_endpoint_stall`
- #define ux_device_stack_host_wakeup
  Initial value:`_ux_device_stack_host_wakeup`
- #define ux_device_stack_initialize
  Initial value:`_ux_device_stack_initialize`
- #define ux_device_stack_uninitialize
  Initial value:`_ux_device_stack_uninitialize`
- #define ux_device_stack_interface_delete
  Initial value:`_ux_device_stack_interface_delete`
- #define ux_device_stack_interface_get
  Initial value:`_ux_device_stack_interface_get`

- #define ux_device_stack_interface_set
  Initial value:`_ux_device_stack_interface_set`
- #define ux_device_stack_interface_start
  Initial value:`_ux_device_stack_interface_start`
- #define ux_device_stack_transfer_request
  Initial value:`_ux_device_stack_transfer_request`
- #define ux_device_stack_transfer_request_abort
  Initial value:`_ux_device_stack_transfer_request_abort`
- #define ux_hcd_ehci_initialize
  Initial value:`_ux_hcd_ehci_initialize`
- #define ux_hcd_isp1161_initialize
  Initial value:`_ux_hcd_isp1161_initialize`
- #define ux_hcd_ohci_initialize
  Initial value:`_ux_hcd_ohci_initialize`
- #define ux_hcd_sim_host_initialize
  Initial value:`_ux_hcd_sim_host_initialize`
- #define ux_dcd_sim_slave_initialize
  Initial value:`_ux_dcd_sim_slave_initialize`
- #define UX_DCD_RX_SLAVE_CONTROLLER
  Initial value:`(UX_DCD_SYNERGY_SLAVE_CONTROLLER)`
- #define UX_DCD_RX_MAX_ED
  Initial value:`(UX_DCD_SYNERGY_MAX_ED)`
- #define UX_DCD_RX_ENABLE
  Initial value:`(UX_DCD_SYNERGY_ENABLE)`
- #define UX_DCD_RX_DISABLE
  Initial value:`(UX_DCD_SYNERGY_DISABLE)`
- #define UX_DCD_RX_MAX_BULK_PAYLOAD
  Initial value:`(UX_DCD_SYNERGY_MAX_BULK_PAYLOAD)`
- #define UX_DCD_RX_MAX_CONTROL_PAYLOAD
  Initial value:`(UX_DCD_SYNERGY_MAX_CONTROL_PAYLOAD)`
- #define UX_DCD_RX_MAX_BUF_SIZE
  Initial value:`(UX_DCD_SYNERGY_MAX_BUF_SIZE)`
- #define UX_DCD_RX_MAX_BUF_NUM
  Initial value:`(UX_DCD_SYNERGY_MAX_BUF_NUM)`

- #define UX_RX_USB_BASE
  Initial value:(UX_SYNERGY_USB_BASE)
- #define UX_RX_DCD_SYSCFG
  Initial value:(UX_SYNERGY_DCD_SYSCFG)
- #define UX_RX_DCD_SYSSTS
  Initial value:(UX_SYNERGY_DCD_SYSSTS)
- #define UX_RX_DCD_DVSTCTR
  Initial value:(UX_SYNERGY_DCD_DVSTCTR)
- #define UX_RX_DCD_CFIFO
  Initial value:(UX_SYNERGY_DCD_CFIFO)
- #define UX_RX_DCD_D0FIFO
  Initial value:(UX_SYNERGY_DCD_D0FIFO)
- #define UX_RX_DCD_D1FIFO
  Initial value:(UX_SYNERGY_DCD_D1FIFO)
- #define UX_RX_DCD_CFIFOSEL
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL)
- #define UX_RX_DCD_CFIFOCTR
  Initial value:(UX_SYNERGY_DCD_CFIFOCTR)
- #define UX_RX_DCD_D0FIFOSEL
  Initial value:(UX_SYNERGY_DCD_D0FIFOSEL)
- #define UX_RX_DCD_D0FIFOCTR
  Initial value:(UX_SYNERGY_DCD_D0FIFOCTR)
- #define UX_RX_DCD_D1FIFOSEL
  Initial value:(UX_SYNERGY_DCD_D1FIFOSEL)
- #define UX_RX_DCD_D1FIFOCTR
  Initial value:(UX_SYNERGY_DCD_D1FIFOCTR)
- #define UX_RX_DCD_INTENB0
  Initial value:(UX_SYNERGY_DCD_INTENB0)
- #define UX_RX_DCD_INTENB1
  Initial value:(UX_SYNERGY_DCD_INTENB1)
- #define UX_RX_DCD_BRDYENB
  Initial value:(UX_SYNERGY_DCD_BRDYENB)
- #define UX_RX_DCD_NRDYENB
  Initial value:(UX_SYNERGY_DCD_NRDYENB)

- #define UX_RX_DCD_BEMPENB
  Initial value:(UX_SYNERGY_DCD_BEMPENB)
- #define UX_RX_DCD_SOFCFG
  Initial value:(UX_SYNERGY_DCD_SOFCFG)
- #define UX_RX_DCD_INTSTS0
  Initial value:(UX_SYNERGY_DCD_INTSTS0)
- #define UX_RX_DCD_INTSTS1
  Initial value:(UX_SYNERGY_DCD_INTSTS1)
- #define UX_RX_DCD_BRDYSTS
  Initial value:(UX_SYNERGY_DCD_BRDYSTS)
- #define UX_RX_DCD_NRDYSTS
  Initial value:(UX_SYNERGY_DCD_NRDYSTS)
- #define UX_RX_DCD_BEMPSTS
  Initial value:(UX_SYNERGY_DCD_BEMPSTS)
- #define UX_RX_DCD_FRMNUM
  Initial value:(UX_SYNERGY_DCD_FRMNUM)
- #define UX_RX_DCD_DVCHGR
  Initial value:(UX_SYNERGY_DCD_DVCHGR)
- #define UX_RX_DCD_USBADDR
  Initial value:(UX_SYNERGY_DCD_USBADDR)
- #define UX_RX_DCD_USBREQ
  Initial value:(UX_SYNERGY_DCD_USBREQ)
- #define UX_RX_DCD_USBVAL
  Initial value:(UX_SYNERGY_DCD_USBVAL)
- #define UX_RX_DCD_USBINDX
  Initial value:(UX_SYNERGY_DCD_USBINDX)
- #define UX_RX_DCD_USBLENG
  Initial value:(UX_SYNERGY_DCD_USBLENG)
- #define UX_RX_DCD_DCPCFG
  Initial value:(UX_SYNERGY_DCD_DCPCFG)
- #define UX_RX_DCD_DCPMAXP
  Initial value:(UX_SYNERGY_DCD_DCPMAXP)
- #define UX_RX_DCD_DCPCTR
  Initial value:(UX_SYNERGY_DCD_DCPCTR)

- #define UX_RX_DCD_PIPESEL
  Initial value:(UX_SYNERGY_DCD_PIPESEL)
- #define UX_RX_DCD_PIPECFG
  Initial value:(UX_SYNERGY_DCD_PIPECFG)
- #define UX_RX_DCD_PIPEMAXP
  Initial value:(UX_SYNERGY_DCD_PIPEMAXP)
- #define UX_RX_DCD_PIPEPERI
  Initial value:(UX_SYNERGY_DCD_PIPEPERI)
- #define UX_RX_DCD_PIPE1CTR
  Initial value:(UX_SYNERGY_DCD_PIPE1CTR)
- #define UX_RX_DCD_PIPE2CTR
  Initial value:(UX_SYNERGY_DCD_PIPE2CTR)
- #define UX_RX_DCD_PIPE3CTR
  Initial value:(UX_SYNERGY_DCD_PIPE3CTR)
- #define UX_RX_DCD_PIPE4CTR
  Initial value:(UX_SYNERGY_DCD_PIPE4CTR)
- #define UX_RX_DCD_PIPE5CTR
  Initial value:(UX_SYNERGY_DCD_PIPE5CTR)
- #define UX_RX_DCD_PIPE6CTR
  Initial value:(UX_SYNERGY_DCD_PIPE6CTR)
- #define UX_RX_DCD_PIPE7CTR
  Initial value:(UX_SYNERGY_DCD_PIPE7CTR)
- #define UX_RX_DCD_PIPE8CTR
  Initial value:(UX_SYNERGY_DCD_PIPE8CTR)
- #define UX_RX_DCD_PIPE9CTR
  Initial value:(UX_SYNERGY_DCD_PIPE9CTR)
- #define UX_RX_DCD_PIPE1TRE
  Initial value:(UX_SYNERGY_DCD_PIPE1TRE)
- #define UX_RX_DCD_PIPE1TRN
  Initial value:(UX_SYNERGY_DCD_PIPE1TRN)
- #define UX_RX_DCD_PIPE2TRE
  Initial value:(UX_SYNERGY_DCD_PIPE2TRE)
- #define UX_RX_DCD_PIPE2TRN
  Initial value:(UX_SYNERGY_DCD_PIPE2TRN)

- #define UX_RX_DCD_PIPE3TRE
  Initial value:(UX_SYNERGY_DCD_PIPE3TRE)
- #define UX_RX_DCD_PIPE3TRN
  Initial value:(UX_SYNERGY_DCD_PIPE3TRN)
- #define UX_RX_DCD_PIPE4TRE
  Initial value:(UX_SYNERGY_DCD_PIPE4TRE)
- #define UX_RX_DCD_PIPE4TRN
  Initial value:(UX_SYNERGY_DCD_PIPE4TRN)
- #define UX_RX_DCD_PIPE5TRE
  Initial value:(UX_SYNERGY_DCD_PIPE5TRE)
- #define UX_RX_DCD_PIPE5TRN
  Initial value:(UX_SYNERGY_DCD_PIPE5TRN)
- #define UX_RX_DCD_DEVADD0
  Initial value:(UX_SYNERGY_DCD_DEVADD0)
- #define UX_RX_DCD_DEVADD1
  Initial value:(UX_SYNERGY_DCD_DEVADD1)
- #define UX_RX_DCD_DEVADD2
  Initial value:(UX_SYNERGY_DCD_DEVADD2)
- #define UX_RX_DCD_DEVADD3
  Initial value:(UX_SYNERGY_DCD_DEVADD3)
- #define UX_RX_DCD_DEVADD4
  Initial value:(UX_SYNERGY_DCD_DEVADD4)
- #define UX_RX_DCD_DEVADD5
  Initial value:(UX_SYNERGY_DCD_DEVADD5)
- #define UX_RX_DCD_SYSCFG_SCKE
  Initial value:(UX_SYNERGY_DCD_SYSCFG_SCKE)
- #define UX_RX_DCD_SYSCFG_DCFM
  Initial value:(UX_SYNERGY_DCD_SYSCFG_DCFM)
- #define UX_RX_DCD_SYSCFG_DRPD
  Initial value:(UX_SYNERGY_DCD_SYSCFG_DRPD)
- #define UX_RX_DCD_SYSCFG_DPRPU
  Initial value:(UX_SYNERGY_DCD_SYSCFG_DPRPU)
- #define UX_RX_DCD_SYSCFG_USBE
  Initial value:(UX_SYNERGY_DCD_SYSCFG_USBE)

- #define UX_RX_DCD_SYSSTS_LNST
  Initial value:(UX_SYNERGY_DCD_SYSSTS_LNST)
- #define UX_RX_DCD_SYSSTS_SOFEN
  Initial value:(UX_SYNERGY_DCD_SYSSTS_SOFEN)
- #define UX_RX_DCD_DVSTCTR_UACKEY0
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_UACKEY0)
- #define UX_RX_DCD_DVSTCTR_UACKEY1
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_UACKEY1)
- #define UX_RX_DCD_DVSTCTR_WKUP
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_WKUP)
- #define UX_RX_DCD_DVSTCTR_RWUPE
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_RWUPE)
- #define UX_RX_DCD_DVSTCTR_USBRST
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_USBRST)
- #define UX_RX_DCD_DVSTCTR_RESUME
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_RESUME)
- #define UX_RX_DCD_DVSTCTR_UACT
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_UACT)
- #define UX_RX_DCD_DVSTCTR_RHST
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_RHST)
- #define UX_RX_DCD_DVSTCTR_SPEED_LOW
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_SPEED_LOW)
- #define UX_RX_DCD_DVSTCTR_SPEED_FULL
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_SPEED_FULL)
- #define UX_RX_DCD_DVSTCTR_SPEED_HIGH
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_SPEED_HIGH)
- #define UX_RX_DCD_DVSTCTR_RESET_IN_PROGRESS
  Initial value:(UX_SYNERGY_DCD_DVSTCTR_RESET_IN_PROGRESS)
- #define UX_RX_DCD_TESTMODE_HOSTPCC
  Initial value:(UX_SYNERGY_DCD_TESTMODE_HOSTPCC)
- #define UX_RX_DCD_DXFBCFG_DFACC
  Initial value:(UX_SYNERGY_DCD_DXFBCFG_DFACC)
- #define UX_RX_DCD_CFIFOSEL_RCNT
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_RCNT)

- #define UX_RX_DCD_CFIFOSEL_REW
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_REW)
- #define UX_RX_DCD_CFIFOSEL_MBW_8
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_MBW_8)
- #define UX_RX_DCD_CFIFOSEL_MBW_MASK
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_MBW_MASK)
- #define UX_RX_DCD_CFIFOSEL_BIGEND
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_BIGEND)
- #define UX_RX_DCD_CFIFOSEL_ISEL
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_ISEL)
- #define UX_RX_DCD_CFIFOSEL_CURPIPE_MASK
  Initial value:(UX_SYNERGY_DCD_CFIFOSEL_CURPIPE_MASK)
- #define UX_RX_DCD_DFIFOSEL_RCNT
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_RCNT)
- #define UX_RX_DCD_DFIFOSEL_REW
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_REW)
- #define UX_RX_DCD_DFIFOSEL_DCLRM
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_DCLRM)
- #define UX_RX_DCD_DFIFOSEL_DREQE
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_DREQE)
- #define UX_RX_DCD_DFIFOSEL_MBW_8
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_MBW_8)
- #define UX_RX_DCD_DFIFOSEL_BIGEND
  Initial value:(UX_SYNERGY_DCD_DFIFOSEL_BIGEND)
- #define UX_RX_DCD_FIFOCTR_BVAL
  Initial value:(UX_SYNERGY_DCD_FIFOCTR_BVAL)
- #define UX_RX_DCD_FIFOCTR_BCLR
  Initial value:(UX_SYNERGY_DCD_FIFOCTR_BCLR)
- #define UX_RX_DCD_FIFOCTR_FRDY
  Initial value:(UX_SYNERGY_DCD_FIFOCTR_FRDY)
- #define UX_RX_DCD_FIFOCTR_DTLN
  Initial value:(UX_SYNERGY_DCD_FIFOCTR_DTLN)
- #define UX_RX_DCD_INTENB0_VBSE
  Initial value:(UX_SYNERGY_DCD_INTENB0_VBSE)

- #define UX_RX_DCD_INTENB0_RSME
  Initial value:(UX_SYNERGY_DCD_INTENB0_RSME)
- #define UX_RX_DCD_INTENB0_SOFE
  Initial value:(UX_SYNERGY_DCD_INTENB0_SOFE)
- #define UX_RX_DCD_INTENB0_DVSE
  Initial value:(UX_SYNERGY_DCD_INTENB0_DVSE)
- #define UX_RX_DCD_INTENB0_CTRE
  Initial value:(UX_SYNERGY_DCD_INTENB0_CTRE)
- #define UX_RX_DCD_INTENB0_BEMPE
  Initial value:(UX_SYNERGY_DCD_INTENB0_BEMPE)
- #define UX_RX_DCD_INTENB0_NRDYE
  Initial value:(UX_SYNERGY_DCD_INTENB0_NRDYE)
- #define UX_RX_DCD_INTENB0_BRDYE
  Initial value:(UX_SYNERGY_DCD_INTENB0_BRDYE)
- #define UX_RX_DCD_INTENB1_BCHGE
  Initial value:(UX_SYNERGY_DCD_INTENB1_BCHGE)
- #define UX_RX_DCD_INTENB1_DTCHE
  Initial value:(UX_SYNERGY_DCD_INTENB1_DTCHE)
- #define UX_RX_DCD_INTENB1_ATTCHE
  Initial value:(UX_SYNERGY_DCD_INTENB1_ATTCHE)
- #define UX_RX_DCD_INTENB1_EOFERRE
  Initial value:(UX_SYNERGY_DCD_INTENB1_EOFERRE)
- #define UX_RX_DCD_INTENB1_SIGNE
  Initial value:(UX_SYNERGY_DCD_INTENB1_SIGNE)
- #define UX_RX_DCD_INTENB1_SACKE
  Initial value:(UX_SYNERGY_DCD_INTENB1_SACKE)
- #define UX_RX_DCD_PIPE0
  Initial value:(UX_SYNERGY_DCD_PIPE0)
- #define UX_RX_DCD_PIPE_ALL
  Initial value:(UX_SYNERGY_DCD_PIPE_ALL)
- #define UX_RX_DCD_SOFCFG_TRNENSEL
  Initial value:(UX_SYNERGY_DCD_SOFCFG_TRNENSEL)
- #define UX_RX_DCD_SOFCFG_BRDYM
  Initial value:(UX_SYNERGY_DCD_SOFCFG_BRDYM)

- #define UX_RX_DCD_SOFCFG_INIT
  Initial value:(UX_SYNERGY_DCD_SOFCFG_INIT)
- #define UX_RX_DCD_INTSTS0_VBINT
  Initial value:(UX_SYNERGY_DCD_INTSTS0_VBINT)
- #define UX_RX_DCD_INTSTS0_RESM
  Initial value:(UX_SYNERGY_DCD_INTSTS0_RESM)
- #define UX_RX_DCD_INTSTS0_SOFR
  Initial value:(UX_SYNERGY_DCD_INTSTS0_SOFR)
- #define UX_RX_DCD_INTSTS0_DVST
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVST)
- #define UX_RX_DCD_INTSTS0_CTRT
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTRT)
- #define UX_RX_DCD_INTSTS0_BEMP
  Initial value:(UX_SYNERGY_DCD_INTSTS0_BEMP)
- #define UX_RX_DCD_INTSTS0_NRDY
  Initial value:(UX_SYNERGY_DCD_INTSTS0_NRDY)
- #define UX_RX_DCD_INTSTS0_BRDY
  Initial value:(UX_SYNERGY_DCD_INTSTS0_BRDY)
- #define UX_RX_DCD_INTSTS0_VBSTS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_VBSTS)
- #define UX_RX_DCD_INTSTS0_DVSQ_MASK
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_MASK)
- #define UX_RX_DCD_INTSTS0_VALID
  Initial value:(UX_SYNERGY_DCD_INTSTS0_VALID)
- #define UX_RX_DCD_INTSTS0_CTSQ_MASK
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_MASK)
- #define UX_RX_DCD_INTSTS0_DVSQ_POWERED
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_POWERED)
- #define UX_RX_DCD_INTSTS0_DVSQ_DEFAULT
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_DEFAULT)
- #define UX_RX_DCD_INTSTS0_DVSQ_ADDRESS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_ADDRESS)
- #define UX_RX_DCD_INTSTS0_DVSQ_CONFIGURED
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_CONFIGURED)

- #define UX_RX_DCD_INTSTS0_DVSQ_SUSPENDED_POWERED
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_POWERED)
- #define UX_RX_DCD_INTSTS0_DVSQ_SUSPENDED_DEFAULT
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_DEFAULT)
- #define UX_RX_DCD_INTSTS0_DVSQ_SUSPENDED_ADDRESS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_ADDRESS)
- #define UX_RX_DCD_INTSTS0_DVSQ_SUSPENDED_CONFIGURED
  Initial value:(UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_CONFIGURED)
- #define UX_RX_DCD_INTSTS0_CTSQ_SETUP
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_SETUP)
- #define UX_RX_DCD_INTSTS0_CTSQ_CRDS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CRDS)
- #define UX_RX_DCD_INTSTS0_CTSQ_CRSS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CRSS)
- #define UX_RX_DCD_INTSTS0_CTSQ_CWDS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CWDS)
- #define UX_RX_DCD_INTSTS0_CTSQ_CWSS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CWSS)
- #define UX_RX_DCD_INTSTS0_CTSQ_CWNDSS
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CWNDSS)
- #define UX_RX_DCD_INTSTS0_CTSQ_CTSE
  Initial value:(UX_SYNERGY_DCD_INTSTS0_CTSQ_CTSE)
- #define UX_RX_DCD_INTSTS1_BCHG
  Initial value:(UX_SYNERGY_DCD_INTSTS1_BCHG)
- #define UX_RX_DCD_INTSTS1_DTCH
  Initial value:(UX_SYNERGY_DCD_INTSTS1_DTCH)
- #define UX_RX_DCD_INTSTS1_ATTCH
  Initial value:(UX_SYNERGY_DCD_INTSTS1_ATTCH)
- #define UX_RX_DCD_INTSTS1_EOFERR
  Initial value:(UX_SYNERGY_DCD_INTSTS1_EOFERR)
- #define UX_RX_DCD_INTSTS1_SIGN
  Initial value:(UX_SYNERGY_DCD_INTSTS1_SIGN)
- #define UX_RX_DCD_INTSTS1_SACK
  Initial value:(UX_SYNERGY_DCD_INTSTS1_SACK)

- #define UX_RX_DCD_FRMNUM_OVRN
  Initial value:(UX_SYNERGY_DCD_FRMNUM_OVRN)
- #define UX_RX_DCD_FRMNUM_CRCE
  Initial value:(UX_SYNERGY_DCD_FRMNUM_CRCE)
- #define UX_RX_DCD_FRMNUM_FRNM_MASK
  Initial value:(UX_SYNERGY_DCD_FRMNUM_FRNM_MASK)
- #define UX_RX_DCD_DCPCFG_DIR
  Initial value:(UX_SYNERGY_DCD_DCPCFG_DIR)
- #define UX_RX_DCD_DCPCFG_SHTNAK
  Initial value:(UX_SYNERGY_DCD_DCPCFG_SHTNAK)
- #define UX_RX_DCD_DCPCFG_CNTMD
  Initial value:(UX_SYNERGY_DCD_DCPCFG_CNTMD)
- #define UX_RX_DCD_DCPMAXP_DEVADDR_SHIFT
  Initial value:(UX_SYNERGY_DCD_DCPMAXP_DEVADDR_SHIFT)
- #define UX_RX_DCD_DCPMAXP_DEVADDR_MASK
  Initial value:(UX_SYNERGY_DCD_DCPMAXP_DEVADDR_MASK)
- #define UX_RX_DCD_DCPCTR_BSTS
  Initial value:(UX_SYNERGY_DCD_DCPCTR_BSTS)
- #define UX_RX_DCD_DCPCTR_INBUFM
  Initial value:(UX_SYNERGY_DCD_DCPCTR_INBUFM)
- #define UX_RX_DCD_DCPCTR_CSCLR
  Initial value:(UX_SYNERGY_DCD_DCPCTR_CSCLR)
- #define UX_RX_DCD_DCPCTR_CSSTS
  Initial value:(UX_SYNERGY_DCD_DCPCTR_CSSTS)
- #define UX_RX_DCD_DCPCTR_SUREQCLR
  Initial value:(UX_SYNERGY_DCD_DCPCTR_SUREQCLR)
- #define UX_RX_DCD_DCPCTR_SQCLR
  Initial value:(UX_SYNERGY_DCD_DCPCTR_SQCLR)
- #define UX_RX_DCD_DCPCTR_SQSET
  Initial value:(UX_SYNERGY_DCD_DCPCTR_SQSET)
- #define UX_RX_DCD_DCPCTR_SQMON
  Initial value:(UX_SYNERGY_DCD_DCPCTR_SQMON)
- #define UX_RX_DCD_DCPCTR_PBUSY
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PBUSY)

- #define UX_RX_DCD_DCPCTR_PINGE
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PINGE)
- #define UX_RX_DCD_DCPCTR_CCPL
  Initial value:(UX_SYNERGY_DCD_DCPCTR_CCPL)
- #define UX_RX_DCD_DCPCTR_PID_MASK
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PID_MASK)
- #define UX_RX_DCD_DCPCTR_PIDNAK
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PIDNAK)
- #define UX_RX_DCD_DCPCTR_PIDBUF
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PIDBUF)
- #define UX_RX_DCD_DCPCTR_PIDSTALL
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PIDSTALL)
- #define UX_RX_DCD_DCPCTR_PIDSTALL2
  Initial value:(UX_SYNERGY_DCD_DCPCTR_PIDSTALL2)
- #define UX_RX_DCD_PIPECFG_TYPE_MASK
  Initial value:(UX_SYNERGY_DCD_PIPECFG_TYPE_MASK)
- #define UX_RX_DCD_PIPECFG_TYPE_BIT_USED
  Initial value:(UX_SYNERGY_DCD_PIPECFG_TYPE_BIT_USED)
- #define UX_RX_DCD_PIPECFG_TYPE_BULK
  Initial value:(UX_SYNERGY_DCD_PIPECFG_TYPE_BULK)
- #define UX_RX_DCD_PIPECFG_TYPE_INTERRUPT
  Initial value:(UX_SYNERGY_DCD_PIPECFG_TYPE_INTERRUPT)
- #define UX_RX_DCD_PIPECFG_TYPE_ISOCHRONOUS
  Initial value:(UX_SYNERGY_DCD_PIPECFG_TYPE_ISOCHRONOUS)
- #define UX_RX_DCD_PIPECFG_BFRE
  Initial value:(UX_SYNERGY_DCD_PIPECFG_BFRE)
- #define UX_RX_DCD_PIPECFG_DBLB
  Initial value:(UX_SYNERGY_DCD_PIPECFG_DBLB)
- #define UX_RX_DCD_PIPECFG_CNTMD
  Initial value:(UX_SYNERGY_DCD_PIPECFG_CNTMD)
- #define UX_RX_DCD_PIPECFG_SHTNAK
  Initial value:(UX_SYNERGY_DCD_PIPECFG_SHTNAK)
- #define UX_RX_DCD_PIPECFG_DIR
  Initial value:(UX_SYNERGY_DCD_PIPECFG_DIR)

- #define UX_RX_DCD_PIPECFG_EPNUM_MASK
  Initial value:(UX_SYNERGY_DCD_PIPECFG_EPNUM_MASK)
- #define UX_RX_DCD_PIPEBUF_SIZEMASK
  Initial value:(UX_SYNERGY_DCD_PIPEBUF_SIZEMASK)
- #define UX_RX_DCD_PIPEBUF_BUFNMBMASK
  Initial value:(UX_SYNERGY_DCD_PIPEBUF_BUFNMBMASK)
- #define UX_RX_DCD_PIPEBUF_SHIFT
  Initial value:(UX_SYNERGY_DCD_PIPEBUF_SHIFT)
- #define UX_RX_DCD_PIPEMAXP_DEVSELMASK
  Initial value:(UX_SYNERGY_DCD_PIPEMAXP_DEVSELMASK)
- #define UX_RX_DCD_PIPEMAXP_DEVSEL_SHIFT
  Initial value:(UX_SYNERGY_DCD_PIPEMAXP_DEVSEL_SHIFT)
- #define UX_RX_DCD_PIPEMAXP_MXPSMASK
  Initial value:(UX_SYNERGY_DCD_PIPEMAXP_MXPSMASK)
- #define UX_RX_DCD_PIPE1TRE_TRCLR
  Initial value:(UX_SYNERGY_DCD_PIPE1TRE_TRCLR)
- #define UX_RX_DCD_PIPE1TRE_TRENB
  Initial value:(UX_SYNERGY_DCD_PIPE1TRE_TRENB)
- #define UX_RX_DCD_FIFO_D0
  Initial value:(UX_SYNERGY_DCD_FIFO_D0)
- #define UX_RX_DCD_FIFO_D1
  Initial value:(UX_SYNERGY_DCD_FIFO_D1)
- #define UX_RX_DCD_FIFO_C
  Initial value:(UX_SYNERGY_DCD_FIFO_C)
- #define UX_RX_DCD_DEVADDX_SPEED_LOW
  Initial value:(UX_SYNERGY_DCD_DEVADDX_SPEED_LOW)
- #define UX_RX_DCD_DEVADDX_SPEED_FULL
  Initial value:(UX_SYNERGY_DCD_DEVADDX_SPEED_FULL)
- #define UX_RX_DCD_DEVADDX_SPEED_HIGH
  Initial value:(UX_SYNERGY_DCD_DEVADDX_SPEED_HIGH)
- #define UX_RX_DCD_DEVADDX_UPPHUB_SHIFT
  Initial value:(UX_SYNERGY_DCD_DEVADDX_UPPHUB_SHIFT)
- #define UX_RX_DCD_DEVADDX_HUBPORT_SHIFT
  Initial value:(UX_SYNERGY_DCD_DEVADDX_HUBPORT_SHIFT)

- #define UX_RX_DCD_DCPCTR_DATA1
  Initial value:(UX_SYNERGY_DCD_DCPCTR_DATA1)
- #define UX_RX_DCD_DCPCTR_DATA0
  Initial value:(UX_SYNERGY_DCD_DCPCTR_DATA0)
- #define UX_RX_DCD_PIPESEL_NO_PIPE
  Initial value:(UX_SYNERGY_DCD_PIPESEL_NO_PIPE)
- #define UX_RX_DCD_PIPE0_SIZE
  Initial value:(UX_SYNERGY_DCD_PIPE0_SIZE)
- #define UX_RX_DCD_PIPE_NB_BUFFERS
  Initial value:(UX_SYNERGY_DCD_PIPE_NB_BUFFERS)
- #define UX_RX_DCD_PIPECTR_INBUFM
  Initial value:(UX_SYNERGY_DCD_PIPECTR_INBUFM)
- #define UX_RX_DCD_PIPECTR_BSTS
  Initial value:(UX_SYNERGY_DCD_PIPECTR_BSTS)
- #define UX_RX_DCD_PIPECTR_CSCLR
  Initial value:(UX_SYNERGY_DCD_PIPECTR_CSCLR)
- #define UX_RX_DCD_PIPECTR_CSSTS
  Initial value:(UX_SYNERGY_DCD_PIPECTR_CSSTS)
- #define UX_RX_DCD_PIPECTR_ATREPM
  Initial value:(UX_SYNERGY_DCD_PIPECTR_ATREPM)
- #define UX_RX_DCD_PIPECTR_SQCLR
  Initial value:(UX_SYNERGY_DCD_PIPECTR_SQCLR)
- #define UX_RX_DCD_PIPECTR_SQSET
  Initial value:(UX_SYNERGY_DCD_PIPECTR_SQSET)
- #define UX_RX_DCD_PIPECTR_SQMON
  Initial value:(UX_SYNERGY_DCD_PIPECTR_SQMON)
- #define UX_RX_DCD_PIPECTR_PBUSY
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PBUSY)
- #define UX_RX_DCD_PIPECTR_PID_MASK
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PID_MASK)
- #define UX_RX_DCD_PIPECTR_PIDNAK
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PIDNAK)
- #define UX_RX_DCD_PIPECTR_PIDBUF
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PIDBUF)

- #define UX_RX_DCD_PIPECTR_PIDSTALL
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PIDSTALL)
- #define UX_RX_DCD_PIPECTR_PIDSTALL2
  Initial value:(UX_SYNERGY_DCD_PIPECTR_PIDSTALL2)
- #define UX_RX_DCD_PIPECTR_DATA1
  Initial value:(UX_SYNERGY_DCD_PIPECTR_DATA1)
- #define UX_RX_DCD_PIPECTR_DATA0
  Initial value:(UX_SYNERGY_DCD_PIPECTR_DATA0)
- #define UX_RX_DCD_COMMAND_STATUS_RESET
  Initial value:(UX_SYNERGY_DCD_COMMAND_STATUS_RESET)
- #define UX_RX_DCD_INIT_RESET_DELAY
  Initial value:(UX_SYNERGY_DCD_INIT_RESET_DELAY)
- #define UX_RX_DCD_MAX_BUF_SIZE
  Initial value:(UX_SYNERGY_DCD_MAX_BUF_SIZE)
- #define UX_RX_DCD_MAX_BUF_NUM
  Initial value:(UX_SYNERGY_DCD_MAX_BUF_NUM)
- #define UX_RX_DCD_FIFO_WRITING
  Initial value:(UX_SYNERGY_DCD_FIFO_WRITING)
- #define UX_RX_DCD_FIFO_WRITE_END
  Initial value:(UX_SYNERGY_DCD_FIFO_WRITE_END)
- #define UX_RX_DCD_FIFO_WRITE_SHORT
  Initial value:(UX_SYNERGY_DCD_FIFO_WRITE_SHORT)
- #define UX_RX_DCD_FIFO_WRITE_DMA
  Initial value:(UX_SYNERGY_DCD_FIFO_WRITE_DMA)
- #define UX_RX_DCD_FIFO_WRITE_ERROR
  Initial value:(UX_SYNERGY_DCD_FIFO_WRITE_ERROR)
- #define UX_RX_DCD_FIFO_READING
  Initial value:(UX_SYNERGY_DCD_FIFO_READING)
- #define UX_RX_DCD_FIFO_READ_END
  Initial value:(UX_SYNERGY_DCD_FIFO_READ_END)
- #define UX_RX_DCD_FIFO_READ_SHORT
  Initial value:(UX_SYNERGY_DCD_FIFO_READ_SHORT)
- #define UX_RX_DCD_FIFO_READ_DMA
  Initial value:(UX_SYNERGY_DCD_FIFO_READ_DMA)

- #define UX_RX_DCD_FIFO_READ_ERROR
  Initial value:(UX_SYNERGY_DCD_FIFO_READ_ERROR)
- #define UX_RX_DCD_FIFO_READ_OVER
  Initial value:(UX_SYNERGY_DCD_FIFO_READ_OVER)
- #define UX_RX_DCD_ED_BRDY
  Initial value:(UX_SYNERGY_DCD_ED_BRDY)
- #define UX_RX_DCD_ED_NRDY
  Initial value:(UX_SYNERGY_DCD_ED_NRDY)
- #define UX_RX_DCD_ED_BEMP
  Initial value:(UX_SYNERGY_DCD_ED_BEMP)
- #define UX_RX_DCD_ED_EOFERR
  Initial value:(UX_SYNERGY_DCD_ED_EOFERR)
- #define UX_RX_DCD_ED_SIGN
  Initial value:(UX_SYNERGY_DCD_ED_SIGN)
- #define UX_RX_DCD_ED_SACK
  Initial value:(UX_SYNERGY_DCD_ED_SACK)
- #define UX_DCD_RX_ED_STATUS_UNUSED
  Initial value:(UX_DCD_SYNERGY_ED_STATUS_UNUSED)
- #define UX_DCD_RX_ED_STATUS_USED
  Initial value:(UX_DCD_SYNERGY_ED_STATUS_USED)
- #define UX_DCD_RX_ED_STATUS_TRANSFER
  Initial value:(UX_DCD_SYNERGY_ED_STATUS_TRANSFER)
- #define UX_DCD_RX_ED_STATUS_STALLED
  Initial value:(UX_DCD_SYNERGY_ED_STATUS_STALLED)
- #define UX_DCD_RX_ED_STATE_IDLE
  Initial value:(UX_DCD_SYNERGY_ED_STATE_IDLE)
- #define UX_DCD_RX_ED_STATE_DATA_TX
  Initial value:(UX_DCD_SYNERGY_ED_STATE_DATA_TX)
- #define UX_DCD_RX_ED_STATE_DATA_RX
  Initial value:(UX_DCD_SYNERGY_ED_STATE_DATA_RX)
- #define UX_DCD_RX_ED_STATE_STATUS_TX
  Initial value:(UX_DCD_SYNERGY_ED_STATE_STATUS_TX)
- #define UX_DCD_RX_ED_STATE_STATUS_RX
  Initial value:(UX_DCD_SYNERGY_ED_STATE_STATUS_RX)

- #define _ux_dcd_rx_address_set
  Initial value:`_ux_dcd_synergy_address_set`
- #define _ux_dcd_rx_endpoint_create
  Initial value:`_ux_dcd_synergy_endpoint_create`
- #define _ux_dcd_rx_endpoint_destroy
  Initial value:`_ux_dcd_synergy_endpoint_destroy`
- #define _ux_dcd_rx_endpoint_reset
  Initial value:`_ux_dcd_synergy_endpoint_reset`
- #define _ux_dcd_rx_endpoint_stall
  Initial value:`_ux_dcd_synergy_endpoint_stall`
- #define _ux_dcd_rx_endpoint_status
  Initial value:`_ux_dcd_synergy_endpoint_status`
- #define _ux_dcd_rx_frame_number_get
  Initial value:`_ux_dcd_synergy_frame_number_get`
- #define _ux_dcd_rx_function
  Initial value:`_ux_dcd_synergy_function`
- #define _ux_dcd_rx_initialize
  Initial value:`_ux_dcd_synergy_initialize((ULONG) R_USBFS)`
- #define _ux_dcd_rx_initialize_complete
  Initial value:`_ux_dcd_synergy_initialize_complete`
- #define _ux_dcd_rx_interrupt_handler
  Initial value:`_ux_dcd_synergy_interrupt_handler`
- #define _ux_dcd_rx_register_clear
  Initial value:`_ux_dcd_synergy_register_clear`
- #define _ux_dcd_rx_register_read
  Initial value:`_ux_dcd_synergy_register_read`
- #define _ux_dcd_rx_register_set
  Initial value:`_ux_dcd_synergy_register_set`
- #define _ux_dcd_rx_register_write
  Initial value:`_ux_dcd_synergy_register_write`
- #define _ux_dcd_rx_state_change
  Initial value:`_ux_dcd_synergy_state_change`
- #define _ux_dcd_rx_transfer_callback
  Initial value:`_ux_dcd_synergy_transfer_callback`

- #define _ux_dcd_rx_transfer_request
  Initial value:`_ux_dcd_synergy_transfer_request`
- #define _ux_dcd_rx_buffer_read
  Initial value:`_ux_dcd_synergy_buffer_read`
- #define _ux_dcd_rx_fifo_read
  Initial value:`_ux_dcd_synergy_fifo_read`
- #define _ux_dcd_rx_buffer_notready_interrupt
  Initial value:`_ux_dcd_synergy_buffer_notready_interrupt`
- #define _ux_dcd_rx_buffer_empty_interrupt
  Initial value:`_ux_dcd_synergy_buffer_empty_interrupt`
- #define _ux_dcd_rx_buffer_ready_interrupt
  Initial value:`_ux_dcd_synergy_buffer_ready_interrupt`
- #define _ux_dcd_rx_fifo_port_change
  Initial value:`_ux_dcd_synergy_fifo_port_change`
- #define _ux_dcd_rx_fifod_write
  Initial value:`_ux_dcd_synergy_fifod_write`
- #define _ux_dcd_rx_fifoc_write
  Initial value:`_ux_dcd_synergy_fifoc_write`
- #define _ux_dcd_rx_buffer_write
  Initial value:`_ux_dcd_synergy_buffer_write`
- #define _ux_dcd_rx_data_buffer_size
  Initial value:`_ux_dcd_synergy_data_buffer_size`
- #define _ux_dcd_rx_endpoint_nak_set
  Initial value:`_ux_dcd_synergy_endpoint_nak_set`
- #define _ux_dcd_rx_current_endpoint_change
  Initial value:`_ux_dcd_synergy_current_endpoint_change`
- #define UX_DCD_SYNERGY_SLAVE_CONTROLLER
  Initial value:`(0x80U)`
- #define UX_DCD_SYNERGY_MAX_ED
  Initial value:`(7U)`
- #define UX_DCD_SYNERGY_ENABLE
  Initial value:`(1U)`
- #define UX_DCD_SYNERGY_DISABLE
  Initial value:`(0U)`

- #define UX_DCD_SYNERGY_MAX_BULK_PAYLOAD
  Initial value:`(512U)`
- #define UX_DCD_SYNERGY_MAX_CONTROL_PAYLOAD
  Initial value:`(512U)`
- #define UX_DCD_SYNERGY_MAX_BUF_SIZE
  Initial value:`(64U)`
- #define UX_DCD_SYNERGY_MAX_BUF_NUM
  Initial value:`(127U)`
- #define UX_DCD_SYNERGY_MIN_PIPE
  Initial value:`(4UL)`
- #define UX_DCD_SYNERGY_MAX_PIPE
  Initial value:`(7UL)`
- #define UX_SYNERGY_DCD_SYSCFG
  Initial value:`(0x00UL)`
- #define UX_SYNERGY_DCD_BUSWAIT
  Initial value:`(0x02UL)`
- #define UX_SYNERGY_DCD_SYSSTS
  Initial value:`(0x04UL)`
- #define UX_SYNERGY_DCD_PLLSTA
  Initial value:`(0x06UL)`
- #define UX_SYNERGY_DCD_DVSTCTR
  Initial value:`(0x08UL)`
- #define UX_SYNERGY_DCD_CFIFO
  Initial value:`(0x14UL)`
- #define UX_SYNERGY_DCD_CFIFOH
  Initial value:`(0x16UL)`
- #define UX_SYNERGY_DCD_CFIFOHH
  Initial value:`(0x17UL)`
- #define UX_SYNERGY_DCD_D0FIFO
  Initial value:`(0x18UL)`
- #define UX_SYNERGY_DCD_D0FIFOH
  Initial value:`(0x1AUL)`
- #define UX_SYNERGY_DCD_D0FIFOHH
  Initial value:`(0x1BUL)`

- #define UX_SYNERGY_DCD_D1FIFO
  Initial value:(0x1CUL)
- #define UX_SYNERGY_DCD_D1FIFOH
  Initial value:(0x1EUL)
- #define UX_SYNERGY_DCD_D1FIFOHH
  Initial value:(0x1FUL)
- #define UX_SYNERGY_DCD_CFIFOSEL
  Initial value:(0x20UL)
- #define UX_SYNERGY_DCD_CFIFOCTR
  Initial value:(0x22UL)
- #define UX_SYNERGY_DCD_D0FIFOSEL
  Initial value:(0x28UL)
- #define UX_SYNERGY_DCD_D0FIFOCTR
  Initial value:(0x2AUL)
- #define UX_SYNERGY_DCD_D1FIFOSEL
  Initial value:(0x2CUL)
- #define UX_SYNERGY_DCD_D1FIFOCTR
  Initial value:(0x2EUL)
- #define UX_SYNERGY_DCD_INTENB0
  Initial value:(0x30UL)
- #define UX_SYNERGY_DCD_INTENB1
  Initial value:(0x32UL)
- #define UX_SYNERGY_DCD_BRDYENB
  Initial value:(0x36UL)
- #define UX_SYNERGY_DCD_NRDYENB
  Initial value:(0x38UL)
- #define UX_SYNERGY_DCD_BEMPENB
  Initial value:(0x3AUL)
- #define UX_SYNERGY_DCD_SOFCFG
  Initial value:(0x3CUL)
- #define UX_SYNERGY_DCD_PHYSET
  Initial value:(0x3EUL)
- #define UX_SYNERGY_DCD_INTSTS0
  Initial value:(0x40UL)

- #define UX_SYNERGY_DCD_INTSTS1
  Initial value:(0x42UL)
- #define UX_SYNERGY_DCD_BRDYSTS
  Initial value:(0x46UL)
- #define UX_SYNERGY_DCD_NRDYSTS
  Initial value:(0x48UL)
- #define UX_SYNERGY_DCD_BEMPSTS
  Initial value:(0x4AUL)
- #define UX_SYNERGY_DCD_FRMNUM
  Initial value:(0x4CUL)
- #define UX_SYNERGY_DCD_DVCHGR
  Initial value:(0x4EUL)
- #define UX_SYNERGY_DCD_USBADDR
  Initial value:(0x50UL)
- #define UX_SYNERGY_DCD_USBREQ
  Initial value:(0x54UL)
- #define UX_SYNERGY_DCD_USBVAL
  Initial value:(0x56UL)
- #define UX_SYNERGY_DCD_USBINDX
  Initial value:(0x58UL)
- #define UX_SYNERGY_DCD_USBLENG
  Initial value:(0x5AUL)
- #define UX_SYNERGY_DCD_DCPCFG
  Initial value:(0x5CUL)
- #define UX_SYNERGY_DCD_DCPMAXP
  Initial value:(0x5EUL)
- #define UX_SYNERGY_DCD_DCPCTR
  Initial value:(0x60UL)
- #define UX_SYNERGY_DCD_PIPESEL
  Initial value:(0x64UL)
- #define UX_SYNERGY_DCD_PIPECFG
  Initial value:(0x68UL)
- #define UX_SYNERGY_DCD_PIPEMAXP
  Initial value:(0x6CUL)

- #define UX_SYNERGY_DCD_PIPEPERI
  Initial value:(0x6EUL)
- #define UX_SYNERGY_DCD_PIPE1CTR
  Initial value:(0x70UL)
- #define UX_SYNERGY_DCD_PIPE2CTR
  Initial value:(0x72UL)
- #define UX_SYNERGY_DCD_PIPE3CTR
  Initial value:(0x74UL)
- #define UX_SYNERGY_DCD_PIPE4CTR
  Initial value:(0x76UL)
- #define UX_SYNERGY_DCD_PIPE5CTR
  Initial value:(0x78UL)
- #define UX_SYNERGY_DCD_PIPE6CTR
  Initial value:(0x7AUL)
- #define UX_SYNERGY_DCD_PIPE7CTR
  Initial value:(0x7CUL)
- #define UX_SYNERGY_DCD_PIPE8CTR
  Initial value:(0x7EUL)
- #define UX_SYNERGY_DCD_PIPE9CTR
  Initial value:(0x80UL)
- #define UX_SYNERGY_DCD_PIPE1TRE
  Initial value:(0x90UL)
- #define UX_SYNERGY_DCD_PIPE1TRN
  Initial value:(0x92UL)
- #define UX_SYNERGY_DCD_PIPE2TRE
  Initial value:(0x94UL)
- #define UX_SYNERGY_DCD_PIPE2TRN
  Initial value:(0x96UL)
- #define UX_SYNERGY_DCD_PIPE3TRE
  Initial value:(0x98UL)
- #define UX_SYNERGY_DCD_PIPE3TRN
  Initial value:(0x9AUL)
- #define UX_SYNERGY_DCD_PIPE4TRE
  Initial value:(0x9CUL)

- #define UX_SYNERGY_DCD_PIPE4TRN
  Initial value:(0x9EUL)
- #define UX_SYNERGY_DCD_PIPE5TRE
  Initial value:(0xA0UL)
- #define UX_SYNERGY_DCD_PIPE5TRN
  Initial value:(0xA2UL)
- #define UX_SYNERGY_DCD_USBMC
  Initial value:(0xCCUL)
- #define UX_SYNERGY_DCD_DEVADD0
  Initial value:(0xD0UL)
- #define UX_SYNERGY_DCD_DEVADD1
  Initial value:(0xD2UL)
- #define UX_SYNERGY_DCD_DEVADD2
  Initial value:(0xD4UL)
- #define UX_SYNERGY_DCD_DEVADD3
  Initial value:(0xD6UL)
- #define UX_SYNERGY_DCD_DEVADD4
  Initial value:(0xD8UL)
- #define UX_SYNERGY_DCD_DEVADD5
  Initial value:(0xDAUL)
- #define UX_SYNERGY_DCD_PHYSLEW
  Initial value:(0xF0UL)
- #define UX_SYNERGY_DCD_LPSTS
  Initial value:(0x102UL)
- #define UX_SYNERGY_DCD_UCKSEL
  Initial value:(0xC4U)
- #define UX_SYNERGY_DCD_BUSWAIT_DEFAULT_VAL
  Initial value:(0xFU)
- #define UX_SYNERGY_DCD_BUSWAIT_MASK
  Initial value:(0xFU)
- #define UX_SYNERGY_DCD_BUSWAIT_CALC_FREQ_PCLK_CYC
  Initial value:(17142857U)
- #define UX_SYNERGY_DCD_BUSWAIT_CALC_FREQ_PCLK_CYCx10
  Initial value:(1714288U)

- #define UX_SYNERGY_DCD_SYSCFG_SCKE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_SYSCFG_CNEN
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_SYSCFG_HSE
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_SYSCFG_DCFM
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_SYSCFG_DRPD
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_SYSCFG_DPRPU
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_SYSCFG_DMRPU
  Initial value:`(1U<<3)`
- #define UX_SYNERGY_DCD_SYSCFG_USBE
  Initial value:`(1)`
- #define UX_SYNERGY_DCD_SYSSTS_LNST
  Initial value:`(3)`
- #define UX_SYNERGY_DCD_SYSSTS_SOFEN
  Initial value:`(0x20U)`
- #define UX_SYNERGY_DCD_DVSTCTR_UACKEY0
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_DVSTCTR_UACKEY1
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_DCD_DVSTCTR_WKUP
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_DVSTCTR_RWUPE
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_DVSTCTR_USBRST
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_DVSTCTR_RESUME
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_DVSTCTR_UACT
  Initial value:`(1U<<4)`

- #define UX_SYNERGY_DCD_DVSTCTR_RHST
  Initial value:`(0x7U)`
- #define UX_SYNERGY_DCD_DVSTCTR_SPEED_LOW
  Initial value:`(1)`
- #define UX_SYNERGY_DCD_DVSTCTR_SPEED_FULL
  Initial value:`(2)`
- #define UX_SYNERGY_DCD_DVSTCTR_SPEED_HIGH
  Initial value:`(3)`
- #define UX_SYNERGY_DCD_DVSTCTR_RESET_IN_PROGRESS
  Initial value:`(4)`
- #define UX_SYNERGY_DCD_TESTMODE_HOSTPCC
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_DXFBCFG_DFACC
  Initial value:`(0U<<12)`
- #define UX_SYNERGY_DCD_CFIFOSEL_RCNT
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_CFIFOSEL_REW
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_CFIFOSEL_MBW_8
  Initial value:`(0U<<10)`
- #define UX_SYNERGY_DCD_CFIFOSEL_MBW_16
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_CFIFOSEL_MBW_32
  Initial value:`(2U<<10)`
- #define UX_SYNERGY_DCD_CFIFOSEL_MBW_MASK
  Initial value:`(3U<<10)`
- #define UX_SYNERGY_DCD_CFIFOSEL_BIGEND
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_CFIFOSEL_ISEL
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_CFIFOSEL_CURPIPE_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_DCD_DFIFOSEL_RCNT
  Initial value:`(1U<<15)`

- #define UX_SYNERGY_DCD_DFIFOSEL_REW
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_DFIFOSEL_DCLRM
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_DFIFOSEL_DREQE
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_DCD_DFIFOSEL_MBW_8
  Initial value:`(0U<<10)`
- #define UX_SYNERGY_DCD_DFIFOSEL_MBW_16
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_DFIFOSEL_MBW_32
  Initial value:`(2U<<10)`
- #define UX_SYNERGY_DCD_DFIFOSEL_MBW_MASK
  Initial value:`(3U<<10)`
- #define UX_SYNERGY_DCD_DFIFOSEL_BIGEND
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_DFIFOSEL_CURPIPE_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_DCD_FIFOCTR_BVAL
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_FIFOCTR_BCLR
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_FIFOCTR_FRDY
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_FIFOCTR_DTLN
  Initial value:`(0xFFFU)`
- #define UX_SYNERGY_DCD_INTENB0_VBSE
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_INTENB0_RSME
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_INTENB0_SOFE
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_INTENB0_DVSE
  Initial value:`(1U<<12)`

- #define UX_SYNERGY_DCD_INTENB0_CTRE
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_DCD_INTENB0_BEMPE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_INTENB0_NRDYE
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_DCD_INTENB0_BRDYE
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_INTENB1_BCHGE
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_INTENB1_DTCHE
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_DCD_INTENB1_ATTCHE
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_DCD_INTENB1_EOFERRE
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_INTENB1_SIGNE
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_INTENB1_SACKE
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_PIPE0
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_DCD_PIPE_ALL
  Initial value:`(0x3FFU)`
- #define UX_SYNERGY_DCD_SOFCFG_TRNENSEL
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_SOFCFG_BRDYM
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_SOFCFG_INIT
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_PHYSET_HSEB
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_PHYSET_REPSTART
  Initial value:`(1U<<11)`

- #define UX_SYNERGY_DCD_PHYSET_REPSEL
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_PHYSET_CLKSEL_1
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_PHYSET_CLKSEL_0
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_PHYSET_CDPEN
  Initial value:`(1U<<3)`
- #define UX_SYNERGY_DCD_PHYSET_PLLRESET
  Initial value:`(1U<<1)`
- #define UX_SYNERGY_DCD_PHYSET_DIRPD
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_DCD_INTSTS0_VBINT
  Initial value:`(USHORT)(1U<<15)`
- #define UX_SYNERGY_DCD_INTSTS0_RESM
  Initial value:`(USHORT)(1U<<14)`
- #define UX_SYNERGY_DCD_INTSTS0_SOFR
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_INTSTS0_DVST
  Initial value:`(USHORT)(1U<<12)`
- #define UX_SYNERGY_DCD_INTSTS0_CTRT
  Initial value:`(USHORT)(1U<<11)`
- #define UX_SYNERGY_DCD_INTSTS0_BEMP
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_INTSTS0_NRDY
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_DCD_INTSTS0_BRDY
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_INTSTS0_VBSTS
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_MASK
  Initial value:`(7U<<4)`
- #define UX_SYNERGY_DCD_INTSTS0_VALID
  Initial value:`(USHORT)(1U<<3)`

- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_MASK
  Initial value:`(7U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_POWERED
  Initial value:`(0x0000U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_DEFAULT
  Initial value:`(0x0010U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_ADDRESS
  Initial value:`(0x0020U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_CONFIGURED
  Initial value:`(0x0030U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_POWERED
  Initial value:`(0x0040U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_DEFAULT
  Initial value:`(0x0050U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_ADDRESS
  Initial value:`(0x0060U)`
- #define UX_SYNERGY_DCD_INTSTS0_DVSQ_SUSPENDED_CONFIGURED
  Initial value:`(0x0070U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_SETUP
  Initial value:`(0x0000U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CRDS
  Initial value:`(0x0001U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CRSS
  Initial value:`(0x0002U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CWDS
  Initial value:`(0x0003U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CWSS
  Initial value:`(0x0004U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CWNDSS
  Initial value:`(0x0005U)`
- #define UX_SYNERGY_DCD_INTSTS0_CTSQ_CTSE
  Initial value:`(0x0006U)`
- #define UX_SYNERGY_DCD_INTSTS1_BCHG
  Initial value:`(1U<<14)`

- #define UX_SYNERGY_DCD_INTSTS1_DTCH
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_DCD_INTSTS1_ATTCH
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_DCD_INTSTS1_EOFERR
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_INTSTS1_SIGN
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_INTSTS1_SACK
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_FRMNUM_OVRN
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_FRMNUM_CRCE
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_FRMNUM_FRNM_MASK
  Initial value:`(0x7FF)`
- #define UX_SYNERGY_DCD_DCPCFG_DIR
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_DCPCFG_SHTNAK
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_DCPCFG_CNTMD
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_DCPMAXP_DEVADDR_SHIFT
  Initial value:`(12)`
- #define UX_SYNERGY_DCD_DCPMAXP_DEVADDR_MASK
  Initial value:`(0xF000U)`
- #define UX_SYNERGY_DCD_DCPCTR_BSTS
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_DCPCTR_INBUFM
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_DCPCTR_CSCLR
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_DCPCTR_CSSTS
  Initial value:`(1U<<12)`

- #define UX_SYNERGY_DCD_DCPCTR_SUREQCLR
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_DCD_DCPCTR_SQCLR
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_DCPCTR_SQSET
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_DCPCTR_SQMON
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_DCPCTR_PBUSY
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_DCPCTR_PINGE
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_DCPCTR_CCPL
  Initial value:`(1U<<2)`
- #define UX_SYNERGY_DCD_DCPCTR_PID_MASK
  Initial value:`(3U)`
- #define UX_SYNERGY_DCD_DCPCTR_PIDNAK
  Initial value:`(0)`
- #define UX_SYNERGY_DCD_DCPCTR_PIDBUF
  Initial value:`(1U)`
- #define UX_SYNERGY_DCD_DCPCTR_PIDSTALL
  Initial value:`(2U)`
- #define UX_SYNERGY_DCD_DCPCTR_PIDSTALL2
  Initial value:`(3U)`
- #define UX_SYNERGY_DCD_PIPECFG_TYPE_MASK
  Initial value:`(0xC000U)`
- #define UX_SYNERGY_DCD_PIPECFG_TYPE_BIT_USED
  Initial value:`(0)`
- #define UX_SYNERGY_DCD_PIPECFG_TYPE_BULK
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_PIPECFG_TYPE_INTERRUPT
  Initial value:`(2U<<14)`
- #define UX_SYNERGY_DCD_PIPECFG_TYPE_ISOCHRONOUS
  Initial value:`(3U<<14)`

- #define UX_SYNERGY_DCD_PIPECFG_BFRE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_DCD_PIPECFG_DBLB
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_DCD_PIPECFG_CNTMD
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_PIPECFG_SHTNAK
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_PIPECFG_DIR
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_DCD_PIPECFG_EPNUM_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_DCD_PIPEBUF_SIZEMASK
  Initial value:`(0x1FU<<10)`
- #define UX_SYNERGY_DCD_PIPEBUF_BUFNMBMASK
  Initial value:`(0xFFU<<10)`
- #define UX_SYNERGY_DCD_PIPEBUF_SHIFT
  Initial value:`(10)`
- #define UX_SYNERGY_DCD_PIPEMAXP_DEVSELMASK
  Initial value:`(0xFU<<12)`
- #define UX_SYNERGY_DCD_PIPEMAXP_DEVSEL_SHIFT
  Initial value:`(12)`
- #define UX_SYNERGY_DCD_PIPEMAXP_MXPSMASK
  Initial value:`(0x7FF)`
- #define UX_SYNERGY_DCD_PIPE1TRE_TRCLR
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_PIPE1TRE_TRENB
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_DCD_FIFO_D0
  Initial value:`(0UL)`
- #define UX_SYNERGY_DCD_FIFO_D1
  Initial value:`(1UL)`
- #define UX_SYNERGY_DCD_FIFO_C
  Initial value:`(2UL)`

- #define UX_SYNERGY_DCD_DEVADDX_SPEED_LOW
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_DEVADDX_SPEED_FULL
  Initial value:`(2U<<6)`
- #define UX_SYNERGY_DCD_DEVADDX_SPEED_HIGH
  Initial value:`(3U<<6)`
- #define UX_SYNERGY_DCD_DEVADDX_UPPHUB_SHIFT
  Initial value:`(11)`
- #define UX_SYNERGY_DCD_DEVADDX_HUBPORT_SHIFT
  Initial value:`(8)`
- #define UX_SYNERGY_DCD_USBMC_VDCEN
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_DCPCTR_DATA1
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_DCPCTR_DATA0
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_PIPESEL_NO_PIPE
  Initial value:`(0x000FU)`
- #define UX_SYNERGY_DCD_UCKSEL_UCKSELC
  Initial value:`(1)`
- #define UX_SYNERGY_DCD_PIPE0_SIZE
  Initial value:`(256U)`
- #define UX_SYNERGY_DCD_PIPE_NB_BUFFERS
  Initial value:`(64)`
- #define UX_SYNERGY_DCD_PIPECTR_INBUFM
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_DCD_PIPECTR_BSTS
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_DCD_PIPECTR_CSCLR
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_DCD_PIPECTR_CSSTS
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_DCD_PIPECTR_ATREPM
  Initial value:`(1U<<11)`

- #define UX_SYNERGY_DCD_PIPECTR_SQCLR
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_PIPECTR_SQSET
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_PIPECTR_SQMON
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_DCD_PIPECTR_PBUSY
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_DCD_PIPECTR_PID_MASK
  Initial value:`(3U)`
- #define UX_SYNERGY_DCD_PIPECTR_PIDNAK
  Initial value:`(0U)`
- #define UX_SYNERGY_DCD_PIPECTR_PIDBUF
  Initial value:`(1U)`
- #define UX_SYNERGY_DCD_PIPECTR_PIDSTALL
  Initial value:`(2U)`
- #define UX_SYNERGY_DCD_PIPECTR_PIDSTALL2
  Initial value:`(3U)`
- #define UX_SYNERGY_DCD_PIPECTR_DATA1
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_DCD_PIPECTR_DATA0
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_DCD_COMMAND_STATUS_RESET
  Initial value:`(0)`
- #define UX_SYNERGY_DCD_INIT_RESET_DELAY
  Initial value:`(10)`
- #define UX_SYNERGY_DCD_MAX_BUF_SIZE
  Initial value:`(64)`
- #define UX_SYNERGY_DCD_MAX_BUF_NUM
  Initial value:`(135UL)`
- #define UX_SYNERGY_DCD_PIPE1_BUF_START_NUM
  Initial value:`(8UL)`
- #define UX_SYNERGY_DCD_PIPE6_BUF_START_NUM
  Initial value:`(4UL)`

- #define UX_SYNERGY_DCD_FIFO_WRITING
  Initial value:(2U)
- #define UX_SYNERGY_DCD_FIFO_WRITE_END
  Initial value:(3U)
- #define UX_SYNERGY_DCD_FIFO_WRITE_SHORT
  Initial value:(4U)
- #define UX_SYNERGY_DCD_FIFO_WRITE_DMA
  Initial value:(5U)
- #define UX_SYNERGY_DCD_FIFO_WRITE_ERROR
  Initial value:(6U)
- #define UX_SYNERGY_DCD_FIFO_READING
  Initial value:(2U)
- #define UX_SYNERGY_DCD_FIFO_READ_END
  Initial value:(3U)
- #define UX_SYNERGY_DCD_FIFO_READ_SHORT
  Initial value:(4U)
- #define UX_SYNERGY_DCD_FIFO_READ_DMA
  Initial value:(5U)
- #define UX_SYNERGY_DCD_FIFO_READ_ERROR
  Initial value:(6U)
- #define UX_SYNERGY_DCD_FIFO_READ_OVER
  Initial value:(7U)
- #define UX_SYNERGY_DCD_ED_BRDY
  Initial value:(0x00000001U)
- #define UX_SYNERGY_DCD_ED_NRDY
  Initial value:(0x00000002U)
- #define UX_SYNERGY_DCD_ED_BEMP
  Initial value:(0x00000004U)
- #define UX_SYNERGY_DCD_ED_EOFERR
  Initial value:(0x00000010U)
- #define UX_SYNERGY_DCD_ED_SIGN
  Initial value:(0x00000020U)
- #define UX_SYNERGY_DCD_ED_SACK
  Initial value:(0x00000040U)

- #define UX_SYNERGY_DCD_PHYSLEW_SLEW_SLEWR00
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_DCD_PHYSLEW_SLEW_SLEWR01
  Initial value:`(1U<<1)`
- #define UX_SYNERGY_DCD_PHYSLEW_SLEW_SLEWF00
  Initial value:`(1U<<2)`
- #define UX_SYNERGY_DCD_PHYSLEW_SLEW_SLEWF01
  Initial value:`(1U<<3)`
- #define UX_SYNERGY_DCD_LPSTS_SUSPENDM
  Initial value:`(1U<<14)`
- #define UX_DCD_SYNERGY_ED_STATUS_UNUSED
  Initial value:`(0U)`
- #define UX_DCD_SYNERGY_ED_STATUS_USED
  Initial value:`(1U)`
- #define UX_DCD_SYNERGY_ED_STATUS_TRANSFER
  Initial value:`(2U)`
- #define UX_DCD_SYNERGY_ED_STATUS_STALLED
  Initial value:`(4U)`
- #define UX_DCD_SYNERGY_ED_STATE_IDLE
  Initial value:`(0U)`
- #define UX_DCD_SYNERGY_ED_STATE_DATA_TX
  Initial value:`(1U)`
- #define UX_DCD_SYNERGY_ED_STATE_DATA_RX
  Initial value:`(2U)`
- #define UX_DCD_SYNERGY_ED_STATE_STATUS_TX
  Initial value:`(3U)`
- #define UX_DCD_SYNERGY_ED_STATE_STATUS_RX
  Initial value:`(4U)`
- #define UX_SLAVE_CLASS_CDC_ACM_CLASS
  Initial value:`10`
- #define UX_SLAVE_CLASS_CDC_ACM_SEND_ENCAPSULATED_COMMAND
  Initial value:`0x00`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_ENCAPSULATED_RESPONSE
  Initial value:`0x01`

- #define UX_SLAVE_CLASS_CDC_ACM_SET_COMM_FEATURE
  Initial value:`0x02`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_COMM_FEATURE
  Initial value:`0x03`
- #define UX_SLAVE_CLASS_CDC_ACM_CLEAR_COMM_FEATURE
  Initial value:`0x04`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_AUX_LINE_STATE
  Initial value:`0x10`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_HOOK_STATE
  Initial value:`0x11`
- #define UX_SLAVE_CLASS_CDC_ACM_PULSE_SETUP
  Initial value:`0x12`
- #define UX_SLAVE_CLASS_CDC_ACM_SEND_PULSE
  Initial value:`0x13`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_PULSE_TIME
  Initial value:`0x14`
- #define UX_SLAVE_CLASS_CDC_ACM_RING_AUX_JACK
  Initial value:`0x15`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_LINE_CODING
  Initial value:`0x20`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_LINE_CODING
  Initial value:`0x21`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_CONTROL_LINE_STATE
  Initial value:`0x22`
- #define UX_SLAVE_CLASS_CDC_ACM_SEND_BREAK
  Initial value:`0x23`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_RINGER_PARMS
  Initial value:`0x30`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_RINGER_PARMS
  Initial value:`0x31`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_OPERATION_PARMS
  Initial value:`0x32`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_OPERATION_PARMS
  Initial value:`0x33`

- #define UX_SLAVE_CLASS_CDC_ACM_SET_LINE_PARMS
  Initial value:`0x34`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_LINE_PARMS
  Initial value:`0x35`
- #define UX_SLAVE_CLASS_CDC_ACM_DIAL_DIGITS
  Initial value:`0x36`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_UNIT_PARAMETER
  Initial value:`0x37`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_UNIT_PARAMETER
  Initial value:`0x38`
- #define UX_SLAVE_CLASS_CDC_ACM_CLEAR_UNIT_PARAMETER
  Initial value:`0x39`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_PROFILE
  Initial value:`0x3A`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_ETHERNET_MULTICAST_FILTERS
  Initial value:`0x40`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_ETHERNET_POWER_MANAGEMENT_PATTERN
  Initial value:`0x41`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_ETHERNET_POWER_MANAGEMENT_PATTERN
  Initial value:`0x42`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_ETHERNET_PACKET_FILTER
  Initial value:`0x43`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_ETHERNET_STATISTIC
  Initial value:`0x44`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_ATM_DATA_FORMAT
  Initial value:`0x50`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_ATM_DEVICE_STATISTICS
  Initial value:`0x51`
- #define UX_SLAVE_CLASS_CDC_ACM_SET_ATM_DEFAULT_VC
  Initial value:`0x52`
- #define UX_SLAVE_CLASS_CDC_ACM_GET_ATM_VC_STATISTICS
  Initial value:`0x53`
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_BAUDRATE
  Initial value:`115200`

- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_STOP_BIT
  Initial value:1
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_PARITY
  Initial value:0
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_DATA_BIT
  Initial value:8
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_BAUDRATE_STRUCT
  Initial value:0
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_STOP_BIT_STRUCT
  Initial value:4
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_PARITY_STRUCT
  Initial value:5
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_DATA_BIT_STRUCT
  Initial value:6
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_CODING_RESPONSE_SIZE
  Initial value:7
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_STATE_DTR
  Initial value:1
- #define UX_SLAVE_CLASS_CDC_ACM_LINE_STATE_RTS
  Initial value:2
- #define UX_SLAVE_CLASS_CDC_ACM_IOCTL_SET_LINE_CODING
  Initial value:1
- #define UX_SLAVE_CLASS_CDC_ACM_IOCTL_GET_LINE_CODING
  Initial value:2
- #define UX_SLAVE_CLASS_CDC_ACM_IOCTL_GET_LINE_STATE
  Initial value:3
- #define
  UX_SLAVE_CLASS_CDC_ACM_SET_ETHERNET_POWER_MANAGEMENT_PATTERN_FILTER
  Initial value:0x41
- #define
  UX_SLAVE_CLASS_CDC_ACM_GET_ETHERNET_POWER_MANAGEMENT_PATTERN_FILTER
  Initial value:0x42
- #define UX_SLAVE_CLASS_CDC_ACM_BUFFER_SIZE
  Initial value:4096

- #define ux_device_class_cdc_acm_entry
  Initial value:_ux_device_class_cdc_acm_entry
- #define ux_device_class_cdc_acm_read
  Initial value:_ux_device_class_cdc_acm_read
- #define ux_device_class_cdc_acm_write
  Initial value:_ux_device_class_cdc_acm_write
- #define UX_SLAVE_CLASS_DPUMP_CLASS
  Initial value:0x99
- #define UX_SLAVE_CLASS_DPUMP_SUBCLASS
  Initial value:0x99
- #define UX_SLAVE_CLASS_DPUMP_PROTOCOL
  Initial value:0x99
- #define UX_DEVICE_CLASS_DPUMP_PACKET_SIZE
  Initial value:128
- #define UX_DEVICE_CLASS_HID_CLASS
  Initial value:0x03
- #define UX_DEVICE_CLASS_HID_SUBCLASS
  Initial value:0X00
- #define UX_DEVICE_CLASS_HID_PROTOCOL
  Initial value:0X00
- #define UX_DEVICE_CLASS_HID_COMMAND_GET_REPORT
  Initial value:0x01
- #define UX_DEVICE_CLASS_HID_COMMAND_GET_IDLE
  Initial value:0x02
- #define UX_DEVICE_CLASS_HID_COMMAND_GET_PROTOCOL
  Initial value:0x03
- #define UX_DEVICE_CLASS_HID_COMMAND_SET_REPORT
  Initial value:0x09
- #define UX_DEVICE_CLASS_HID_COMMAND_SET_IDLE
  Initial value:0x0A
- #define UX_DEVICE_CLASS_HID_COMMAND_SET_PROTOCOL
  Initial value:0x0B
- #define UX_DEVICE_CLASS_HID_DESCRIPTOR_HID
  Initial value:0x21

- #define UX_DEVICE_CLASS_HID_DESCRIPTOR_REPORT
  Initial value:0x22
- #define UX_DEVICE_CLASS_HID_DESCRIPTOR_PHYSICAL
  Initial value:0x23
- #define UX_DEVICE_CLASS_HID_REPORT_TYPE_INPUT
  Initial value:0x1
- #define UX_DEVICE_CLASS_HID_REPORT_TYPE_OUTPUT
  Initial value:0x2
- #define UX_DEVICE_CLASS_HID_REPORT_TYPE_FEATURE
  Initial value:0x3
- #define UX_DEVICE_CLASS_HID_EVENT_BUFFER_LENGTH
  Initial value:32
- #define UX_DEVICE_CLASS_HID_NEW_EVENT
  Initial value:1
- #define UX_DEVICE_CLASS_HID_MAX_EVENTS_QUEUE
  Initial value:16
- #define UX_MAX_SLAVE_LUN
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_CLASS
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_SUBCLASS_RBC
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_SUBCLASS_SFF8020
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_SUBCLASS_UFI
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_SUBCLASS_SFF8070
  Initial value:5
- #define UX_SLAVE_CLASS_STORAGE_SUBCLASS_SCSI
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_PROTOCOL_CBI
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_PROTOCOL_BO
  Initial value:0x50

- #define UX_SLAVE_CLASS_STORAGE_MEDIA_FAT_DISK
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_MEDIA_CDROM
  Initial value:5
- #define UX_SLAVE_CLASS_STORAGE_MEDIA_OPTICAL_DISK
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_MEDIA_IOMEGA_CLICK
  Initial value:0x55
- #define UX_SLAVE_CLASS_STORAGE_SCSI_TEST_READY
  Initial value:0x00
- #define UX_SLAVE_CLASS_STORAGE_SCSI_REQUEST_SENSE
  Initial value:0x03
- #define UX_SLAVE_CLASS_STORAGE_SCSI_FORMAT
  Initial value:0x04
- #define UX_SLAVE_CLASS_STORAGE_SCSI_INQUIRY
  Initial value:0x12
- #define UX_SLAVE_CLASS_STORAGE_SCSI_MODE_SENSE_SHORT
  Initial value:0x1a
- #define UX_SLAVE_CLASS_STORAGE_SCSI_START_STOP
  Initial value:0x1b
- #define UX_SLAVE_CLASS_STORAGE_SCSI_PREVENT_ALLOW_MEDIA_REMOVAL
  Initial value:0x1e
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ_FORMAT_CAPACITY
  Initial value:0x23
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ_CAPACITY
  Initial value:0x25
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ16
  Initial value:0x28
- #define UX_SLAVE_CLASS_STORAGE_SCSI_WRITE16
  Initial value:0x2a
- #define UX_SLAVE_CLASS_STORAGE_SCSI_VERIFY
  Initial value:0x2f
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ_TOC
  Initial value:0x43

- #define UX_SLAVE_CLASS_STORAGE_SCSI_GET_CONFIGURATION
  Initial value:`0x46`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_GET_STATUS_NOTIFICATION
  Initial value:`0x4A`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ_DISK_INFORMATION
  Initial value:`0x51`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_MODE_SELECT
  Initial value:`0x55`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ32
  Initial value:`0xa8`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_REPORT_KEY
  Initial value:`0xa4`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_WRITE32
  Initial value:`0xaa`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_GET_PERFORMANCE
  Initial value:`0xac`
- #define UX_SLAVE_CLASS_STORAGE_SCSI_READ_DVD_STRUCTURE
  Initial value:`0xad`
- #define UX_SLAVE_CLASS_STORAGE_CBW_SIGNATURE_MASK
  Initial value:`0x43425355`
- #define UX_SLAVE_CLASS_STORAGE_CBW_TAG
  Initial value:`4`
- #define UX_SLAVE_CLASS_STORAGE_CBW_DATA_LENGTH
  Initial value:`8`
- #define UX_SLAVE_CLASS_STORAGE_CBW_FLAGS
  Initial value:`12`
- #define UX_SLAVE_CLASS_STORAGE_CBW_LUN
  Initial value:`13`
- #define UX_SLAVE_CLASS_STORAGE_CBW_CB_LENGTH
  Initial value:`14`
- #define UX_SLAVE_CLASS_STORAGE_CBW_LENGTH
  Initial value:`31`
- #define UX_SLAVE_CLASS_STORAGE_CSW_SIGNATURE_MASK
  Initial value:`0x53425355`

- #define UX_SLAVE_CLASS_STORAGE_CSW_TAG
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_CSW_DATA_RESIDUE
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_CSW_STATUS
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_CSW_LENGTH
  Initial value:13
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_PAGE_CODE
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_ALLOCATION_LENGTH
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_COMMAND_LENGTH_SBC
  Initial value:06
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_PERIPHERAL_TYPE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_REMOVABLE_MEDIA
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_DATA_FORMAT
  Initial value:3
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_ADDITIONAL_LENGTH
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_VENDOR_INFORMATION
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_PRODUCT_ID
  Initial value:16
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_PRODUCT_REVISION
  Initial value:32

- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_LENGTH
  Initial value:36
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_RESPONSE_LENGTH_CD_ROM
  Initial value:0x5b
- #define UX_SLAVE_CLASS_STORAGE_START_STOP_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_START_STOP_LBUFLAGS
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_START_STOP_START_BIT
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_START_STOP_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_START_STOP_COMMAND_LENGTH_SBC
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_PC_PAGE_CODE
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_PARAMETER_LIST_LENGTH_6
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_PARAMETER_LIST_LENGTH_10
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_MODE_SENSE_COMMAND_LENGTH_SBC
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_ALLOCATION_LENGTH
  Initial value:4

- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_COMMAND_LENGTH_SBC
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_ERROR_CODE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_SENSE_KEY
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_INFORMATION
  Initial value:3
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_ADD_LENGTH
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_CODE
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_CODE_QUALIFIER
  Initial value:13
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_LENGTH
  Initial value:18
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_HEADER_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_LBA
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_COMMAND_LENGTH_SBC
  Initial value:10
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_RESPONSE_LAST_LBA
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_RESPONSE_BLOCK_SIZE
  Initial value:4

- #define UX_SLAVE_CLASS_STORAGE_READ_CAPACITY_RESPONSE_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_READ_FORMAT_CAPACITY_RESPONSE_SIZE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_FORMAT_CAPACITY_RESPONSE_LAST_LBA
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_READ_FORMAT_CAPACITY_RESPONSE_DESC_CODE
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_READ_FORMAT_CAPACITY_RESPONSE_BLOCK_SIZE
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_READ_FORMAT_CAPACITY_RESPONSE_LENGTH
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_TEST_READY_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_TEST_READY_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_TEST_READY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_TEST_READY_COMMAND_LENGTH_SBC
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_READ_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_READ_LBA
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_READ_TRANSFER_LENGTH_32
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_READ_TRANSFER_LENGTH_16
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_READ_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_READ_COMMAND_LENGTH_SBC
  Initial value:10

- #define UX_SLAVE_CLASS_STORAGE_WRITE_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_WRITE_LUN
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_WRITE_LBA
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_WRITE_TRANSFER_LENGTH_32
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_WRITE_TRANSFER_LENGTH_16
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_WRITE_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_SLAVE_CLASS_STORAGE_WRITE_COMMAND_LENGTH_SBC
  Initial value:10
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_NO_SENSE
  Initial value:0x0
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_RECOVERED_ERROR
  Initial value:0x1
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_NOT_READY
  Initial value:0x2
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_MEDIUM_ERROR
  Initial value:0x3
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_HARDWARE_ERROR
  Initial value:0x4
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_ILLEGAL_REQUEST
  Initial value:0x5
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_UNIT_ATTENTION
  Initial value:0x6
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_DATA_PROTECT
  Initial value:0x7
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_BLANK_CHECK
  Initial value:0x8
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_ABORTED_COMMAND
  Initial value:0x0b

- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_VOLUME_OVERFLOW
  Initial value:`0x0d`
- #define UX_SLAVE_CLASS_STORAGE_SENSE_KEY_MISCOMPARE
  Initial value:`0x0e`
- #define UX_SLAVE_CLASS_STORAGE_GET_CONFIGURATION_OPERATION
  Initial value:`0`
- #define UX_SLAVE_CLASS_STORAGE_GET_CONFIGURATION_RT
  Initial value:`1`
- #define UX_SLAVE_CLASS_STORAGE_GET_CONFIGURATION_STARTING_FEATURE
  Initial value:`2`
- #define UX_SLAVE_CLASS_STORAGE_GET_CONFIGURATION_ALLOCATION_LENGTH
  Initial value:`7`
- #define UX_SLAVE_CLASS_STORAGE_GET_CONFIGURATION_COMMAND_LENGTH_SBC
  Initial value:`9`
- #define UX_SLAVE_CLASS_STORAGE_ASC_KEY_INVALID_COMMAND
  Initial value:`0x20`
- #define UX_SLAVE_CLASS_STORAGE_CSW_PASSED
  Initial value:`0`
- #define UX_SLAVE_CLASS_STORAGE_CSW_FAILED
  Initial value:`1`
- #define UX_SLAVE_CLASS_STORAGE_CSW_PHASE_ERROR
  Initial value:`2`
- #define UX_SLAVE_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_ERROR_CODE_VALUE
  Initial value:`0x70`
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_PAGE_CODE_STANDARD
  Initial value:`0x00`
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_PAGE_CODE_SERIAL
  Initial value:`0x80`
- #define UX_SLAVE_CLASS_STORAGE_INQUIRY_PERIPHERAL_TYPE
  Initial value:`0x00`
- #define UX_SLAVE_CLASS_STORAGE_RESET
  Initial value:`0xff`
- #define UX_SLAVE_CLASS_STORAGE_GET_MAX_LUN
  Initial value:`0xfe`

- #define UX_SLAVE_CLASS_STORAGE_MAX_LUN
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_RESPONSE_LENGTH
  Initial value:64
- #define UX_SLAVE_CLASS_STORAGE_READ_DISK_INFORMATION_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_DISK_INFORMATION_STATUS
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_READ_DISK_INFORMATION_ALLOCATION_LENGTH
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_READ_DISK_INFORMATION_LENGTH
  Initial value:10
- #define USBX_DEVICE_CLASS_STORAGE_FEATURE_DESCRIPTOR_FEATURE_CODE
  Initial value:0
- #define USBX_DEVICE_CLASS_STORAGE_FEATURE_DESCRIPTOR_FEATURE_PARAMETER
  Initial value:2
- #define USBX_DEVICE_CLASS_STORAGE_FEATURE_DESCRIPTOR_FEATURE_ADD_LENGTH
  Initial value:3
- #define USBX_DEVICE_CLASS_STORAGE_FEATURE_DESCRIPTOR_FEATURE_LENGTH
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_OPERATION
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ALLOCATION_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT
  Initial value:10
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ANSWER_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ANSWER_PAYLOAD
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ANSWER_RESET_FIELD
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ANSWER_REGION_MASK
  Initial value:5

- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_ANSWER_RPC_SCHEME
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT_AGID
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT_CHALLENGE
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT_KEY2
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT_TITLE
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_REPORT_KEY_FORMAT_RPC
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_EVENT_NOTIFICATION_OPCODE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_EVENT_NOTIFICATION_IMMEDIATE
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_EVENT_NOTIFICATION_CLASS_REQUEST
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_EVENT_NOTIFICATION_ALLOCATION_LENGTH
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_EVENT_NOTIFICATION_LENGTH
  Initial value:10
- #define UX_SLAVE_CLASS_STORAGE_READ_TOC_OPCODE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_TOC_FORMAT
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_READ_TOC_TRACK_NUMBER
  Initial value:6
- #define UX_SLAVE_CLASS_STORAGE_READ_TOC_ALLOCATION_LENGTH
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_OPCODE
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_ADDRESS
  Initial value:2

- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_FORMAT
  Initial value:7
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_ALLOCATION_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_RESP_LENGTH
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_BOOK_TYPE
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_READ_DVD_STRUCTURE_DS_MR
  Initial value:5
- #define UX_SLAVE_CLASS_STORAGE_GET_PERFORMANCE_PAGE
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_GET_PERFORMANCE_PAGE_14
  Initial value:0x14
- #define UX_SLAVE_CLASS_STORAGE_GET_PERFORMANCE_PAGE_0
  Initial value:0x00
- #define UX_SLAVE_CLASS_STORAGE_GET_PERFORMANCE_PAYLOAD_LENGTH
  Initial value:4
- #define UX_SLAVE_CLASS_STORAGE_GET_PERFORMANCE_RESPONSE_LENGTH
  Initial value:8
- #define UX_SLAVE_CLASS_STORAGE_BUFFER_SIZE
  Initial value:UX_SLAVE_REQUEST_DATA_MAX_LENGTH
- #define UX_SLAVE_CLASS_STORAGE_MMC2_DISK_STATUS_EMPTY
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_MMC2_DISK_STATUS_INCOMPLETE
  Initial value:1
- #define UX_SLAVE_CLASS_STORAGE_MMC2_DISK_STATUS_COMPLETE
  Initial value:2
- #define UX_SLAVE_CLASS_STORAGE_MMC2_DISK_STATUS_OTHERS
  Initial value:3
- #define UX_SLAVE_CLASS_STORAGE_MMC2_LAT_SESSION_EMPTY
  Initial value:0
- #define UX_SLAVE_CLASS_STORAGE_MMC2_LAT_SESSION_INCOMPLETE
  Initial value:1

- #define UX_SLAVE_CLASS_STORAGE_MMC2_LAT_SESSION_COMPLETE
  Initial value:3
- #define UX_SLAVE_CLASS_STORAGE_MMC2_PAGE_CODE_CDROM
  Initial value:0x2a
- #define UX_RX_CONTROLLER
  Initial value:(UX_SYNERGY_CONTROLLER)
- #define UX_RX_CONTROLLER_RX62N
  Initial value:(UX_SYNERGY_CONTROLLER_S7G2)
- #define UX_RX_MAX_BULK_PAYLOAD
  Initial value:(UX_SYNERGY_MAX_BULK_PAYLOAD)
- #define UX_RX_MAX_CONTROL_PAYLOAD
  Initial value:(UX_SYNERGY_MAX_CONTROL_PAYLOAD)
- #define UX_RX_FRAME_DELAY
  Initial value:(UX_SYNERGY_FRAME_DELAY)
- #define UX_RX_PERIODIC_ENTRY_NB
  Initial value:(UX_SYNERGY_PERIODIC_ENTRY_NB)
- #define UX_RX_PERIODIC_ENTRY_MASK
  Initial value:(UX_SYNERGY_PERIODIC_ENTRY_MASK)
- #define UX_RX_ENABLE
  Initial value:(UX_SYNERGY_ENABLE)
- #define UX_RX_DISABLE
  Initial value:(UX_SYNERGY_DISABLE)
- #define UX_RX_HC_SYSCFG
  Initial value:(UX_SYNERGY_HC_SYSCFG)
- #define UX_RX_HC_SYSSTS0
  Initial value:(UX_SYNERGY_HC_SYSSTS0)
- #define UX_RX_HC_PLLSTA
  Initial value:(UX_SYNERGY_HC_PLLSTA)
- #define UX_RX_HC_DVSTCTR0
  Initial value:(UX_SYNERGY_HC_DVSTCTR0)
- #define UX_RX_HC_CFIFO
  Initial value:(UX_SYNERGY_HC_CFIFO)
- #define UX_RX_HC_D0FIFO
  Initial value:(UX_SYNERGY_HC_D0FIFO)

- #define UX_RX_HC_D1FIFO
  Initial value:(UX_SYNERGY_HC_D1FIFO)
- #define UX_RX_HC_CFIFOSEL
  Initial value:(UX_SYNERGY_HC_CFIFOSEL)
- #define UX_RX_HC_CFIFOCTR
  Initial value:(UX_SYNERGY_HC_CFIFOCTR)
- #define UX_RX_HC_D0FIFOSEL
  Initial value:(UX_SYNERGY_HC_D0FIFOSEL)
- #define UX_RX_HC_D0FIFOCTR
  Initial value:(UX_SYNERGY_HC_D0FIFOCTR)
- #define UX_RX_HC_D1FIFOSEL
  Initial value:(UX_SYNERGY_HC_D1FIFOSEL)
- #define UX_RX_HC_D1FIFOCTR
  Initial value:(UX_SYNERGY_HC_D1FIFOCTR)
- #define UX_RX_HC_INTENB0
  Initial value:(UX_SYNERGY_HC_INTENB0)
- #define UX_RX_HC_INTENB1
  Initial value:(UX_SYNERGY_HC_INTENB1)
- #define UX_RX_HC_BRDYENB
  Initial value:(UX_SYNERGY_HC_BRDYENB)
- #define UX_RX_HC_NRDYENB
  Initial value:(UX_SYNERGY_HC_NRDYENB)
- #define UX_RX_HC_BEMPENB
  Initial value:(UX_SYNERGY_HC_BEMPENB)
- #define UX_RX_HC_SOFCFG
  Initial value:(UX_SYNERGY_HC_SOFCFG)
- #define UX_RX_HC_PHYSET
  Initial value:(UX_SYNERGY_HC_PHYSET)
- #define UX_RX_HC_INTSTS0
  Initial value:(UX_SYNERGY_HC_INTSTS0)
- #define UX_RX_HC_INTSTS1
  Initial value:(UX_SYNERGY_HC_INTSTS1)
- #define UX_RX_HC_BRDYSTS
  Initial value:(UX_SYNERGY_HC_BRDYSTS)

- #define UX_RX_HC_NRDYSTS
  Initial value:(UX_SYNERGY_HC_NRDYSTS)
- #define UX_RX_HC_BEMPSTS
  Initial value:(UX_SYNERGY_HC_BEMPSTS)
- #define UX_RX_HC_FRMNUM
  Initial value:(UX_SYNERGY_HC_FRMNUM)
- #define UX_RX_HC_DVCHGR
  Initial value:(UX_SYNERGY_HC_DVCHGR)
- #define UX_RX_HC_USBADDR
  Initial value:(UX_SYNERGY_HC_USBADDR)
- #define UX_RX_HC_USBREQ
  Initial value:(UX_SYNERGY_HC_USBREQ)
- #define UX_RX_HC_USBVAL
  Initial value:(UX_SYNERGY_HC_USBVAL)
- #define UX_RX_HC_USBINDX
  Initial value:(UX_SYNERGY_HC_USBINDX)
- #define UX_RX_HC_USBLENG
  Initial value:(UX_SYNERGY_HC_USBLENG)
- #define UX_RX_HC_DCPCFG
  Initial value:(UX_SYNERGY_HC_DCPCFG)
- #define UX_RX_HC_DCPMAXP
  Initial value:(UX_SYNERGY_HC_DCPMAXP)
- #define UX_RX_HC_DCPCTR
  Initial value:(UX_SYNERGY_HC_DCPCTR)
- #define UX_RX_HC_PIPESEL
  Initial value:(UX_SYNERGY_HC_PIPESEL)
- #define UX_RX_HC_PIPECFG
  Initial value:(UX_SYNERGY_HC_PIPECFG)
- #define UX_RX_HC_PIPEMAXP
  Initial value:(UX_SYNERGY_HC_PIPEMAXP)
- #define UX_RX_HC_PIPEPERI
  Initial value:(UX_SYNERGY_HC_PIPEPERI)
- #define UX_RX_HC_PIPE1CTR
  Initial value:(UX_SYNERGY_HC_PIPE1CTR)

- #define UX_RX_HC_PIPE2CTR
  Initial value:(UX_SYNERGY_HC_PIPE2CTR)
- #define UX_RX_HC_PIPE3CTR
  Initial value:(UX_SYNERGY_HC_PIPE3CTR)
- #define UX_RX_HC_PIPE4CTR
  Initial value:(UX_SYNERGY_HC_PIPE4CTR)
- #define UX_RX_HC_PIPE5CTR
  Initial value:(UX_SYNERGY_HC_PIPE5CTR)
- #define UX_RX_HC_PIPE6CTR
  Initial value:(UX_SYNERGY_HC_PIPE6CTR)
- #define UX_RX_HC_PIPE7CTR
  Initial value:(UX_SYNERGY_HC_PIPE7CTR)
- #define UX_RX_HC_PIPE8CTR
  Initial value:(UX_SYNERGY_HC_PIPE8CTR)
- #define UX_RX_HC_PIPE9CTR
  Initial value:(UX_SYNERGY_HC_PIPE9CTR)
- #define UX_RX_HC_PIPE1TRE
  Initial value:(UX_SYNERGY_HC_PIPE1TRE)
- #define UX_RX_HC_PIPE1TRN
  Initial value:(UX_SYNERGY_HC_PIPE1TRN)
- #define UX_RX_HC_PIPE2TRE
  Initial value:(UX_SYNERGY_HC_PIPE2TRE)
- #define UX_RX_HC_PIPE2TRN
  Initial value:(UX_SYNERGY_HC_PIPE2TRN)
- #define UX_RX_HC_PIPE3TRE
  Initial value:(UX_SYNERGY_HC_PIPE3TRE)
- #define UX_RX_HC_PIPE3TRN
  Initial value:(UX_SYNERGY_HC_PIPE3TRN)
- #define UX_RX_HC_PIPE4TRE
  Initial value:(UX_SYNERGY_HC_PIPE4TRE)
- #define UX_RX_HC_PIPE4TRN
  Initial value:(UX_SYNERGY_HC_PIPE4TRN)
- #define UX_RX_HC_PIPE5TRE
  Initial value:(UX_SYNERGY_HC_PIPE5TRE)

- #define UX_RX_HC_PIPE5TRN
  Initial value:(UX_SYNERGY_HC_PIPE5TRN)
- #define UX_RX_HC_DEVADD0
  Initial value:(UX_SYNERGY_HC_DEVADD0)
- #define UX_RX_HC_DEVADD1
  Initial value:(UX_SYNERGY_HC_DEVADD1)
- #define UX_RX_HC_DEVADD2
  Initial value:(UX_SYNERGY_HC_DEVADD2)
- #define UX_RX_HC_DEVADD3
  Initial value:(UX_SYNERGY_HC_DEVADD3)
- #define UX_RX_HC_DEVADD4
  Initial value:(UX_SYNERGY_HC_DEVADD4)
- #define UX_RX_HC_DEVADD5
  Initial value:(UX_SYNERGY_HC_DEVADD5)
- #define UX_RX_HC_LPSTS
  Initial value:(UX_SYNERGY_HC_LPSTS)
- #define UX_RX_HC_PFKUSB_ENABLED
  Initial value:(UX_SYNERGY_HC_PFKUSB_ENABLED)
- #define UX_RX_HC_PFKUSB_MODE_HOST
  Initial value:(UX_SYNERGY_HC_PFKUSB_MODE_HOST)
- #define UX_RX_HC_SYSCFG_CNEN
  Initial value:(UX_SYNERGY_HC_SYSCFG_CNEN)
- #define UX_RX_HC_SYSCFG_HSE
  Initial value:(UX_SYNERGY_HC_SYSCFG_HSE)
- #define UX_RX_HC_SYSCFG_DCFM
  Initial value:(UX_SYNERGY_HC_SYSCFG_DCFM)
- #define UX_RX_HC_SYSCFG_DRPD
  Initial value:(UX_SYNERGY_HC_SYSCFG_DRPD)
- #define UX_RX_HC_SYSCFG_DPRPU
  Initial value:(UX_SYNERGY_HC_SYSCFG_DPRPU)
- #define UX_RX_HC_SYSCFG_USBE
  Initial value:(UX_SYNERGY_HC_SYSCFG_USBE)
- #define UX_RX_HC_SYSSTS0_LNST
  Initial value:(UX_SYNERGY_HC_SYSSTS0_LNST)

- #define UX_RX_HC_SYSSTS0_IDMON
  Initial value:(UX_SYNERGY_HC_SYSSTS0_IDMON)
- #define UX_RX_HC_SYSSTS0_SOFEA
  Initial value:(UX_SYNERGY_HC_SYSSTS0_SOFEA)
- #define UX_RX_HC_SYSSTS0_HTACT
  Initial value:(UX_SYNERGY_HC_SYSSTS0_HTACT)
- #define UX_RX_HC_SYSSTS0_OVCMON
  Initial value:(UX_SYNERGY_HC_SYSSTS0_OVCMON)
- #define UX_RX_HC_PLLSTA_PLLLOCK
  Initial value:(UX_SYNERGY_HC_PLLSTA_PLLLOCK)
- #define UX_RX_HC_DVSTCTR0_HNPBTOA
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_HNPBTOA)
- #define UX_RX_HC_DVSTCTR0_EXICEN
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_EXICEN)
- #define UX_RX_HC_DVSTCTR0_VBUSEN
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_VBUSEN)
- #define UX_RX_HC_DVSTCTR0_WKUP
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_WKUP)
- #define UX_RX_HC_DVSTCTR0_RWUPE
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_RWUPE)
- #define UX_RX_HC_DVSTCTR0_USBRST
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_USBRST)
- #define UX_RX_HC_DVSTCTR0_RESUME
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_RESUME)
- #define UX_RX_HC_DVSTCTR0_UACT
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_UACT)
- #define UX_RX_HC_DVSTCTR0_RHST
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_RHST)
- #define UX_RX_HC_DVSTCTR0_SPEED_LOW
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_SPEED_LOW)
- #define UX_RX_HC_DVSTCTR0_SPEED_FULL
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_SPEED_FULL)
- #define UX_RX_HC_DVSTCTR0_SPEED_HIGH
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_SPEED_HIGH)

- #define UX_RX_HC_DVSTCTR0_RESET_IN_PROGRESS
  Initial value:(UX_SYNERGY_HC_DVSTCTR0_RESET_IN_PROGRESS)
- #define UX_RX_HC_CFIFOSEL_RCNT
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_RCNT)
- #define UX_RX_HC_CFIFOSEL_REW
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_REW)
- #define UX_RX_HC_CFIFOSEL_MBW_8
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_MBW_8)
- #define UX_RX_HC_CFIFOSEL_MBW_16
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_MBW_16)
- #define UX_RX_HC_CFIFOSEL_MBW_32
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_MBW_32)
- #define UX_RX_HC_CFIFOSEL_MBW_MASK
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_MBW_MASK)
- #define UX_RX_HC_CFIFOSEL_BIGEND
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_BIGEND)
- #define UX_RX_HC_CFIFOSEL_ISEL
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_ISEL)
- #define UX_RX_HC_CFIFOSEL_CURPIPE_MASK
  Initial value:(UX_SYNERGY_HC_CFIFOSEL_CURPIPE_MASK)
- #define UX_RX_HC_DFIFOSEL_RCNT
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_RCNT)
- #define UX_RX_HC_DFIFOSEL_REW
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_REW)
- #define UX_RX_HC_DFIFOSEL_DCLRM
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_DCLRM)
- #define UX_RX_HC_DFIFOSEL_DREQE
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_DREQE)
- #define UX_RX_HC_DFIFOSEL_MBW_8
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_MBW_8)
- #define UX_RX_HC_DFIFOSEL_MBW_16
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_MBW_16)
- #define UX_RX_HC_DFIFOSEL_MBW_32
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_MBW_32)

- #define UX_RX_HC_DFIFOSEL_MBW_MASK
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_MBW_MASK)
- #define UX_RX_HC_DFIFOSEL_BIGEND
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_BIGEND)
- #define UX_RX_HC_DFIFOSEL_CURPIPE_MASK
  Initial value:(UX_SYNERGY_HC_DFIFOSEL_CURPIPE_MASK)
- #define UX_RX_HC_FIFOCTR_BVAL
  Initial value:(UX_SYNERGY_HC_FIFOCTR_BVAL)
- #define UX_RX_HC_FIFOCTR_BCLR
  Initial value:(UX_SYNERGY_HC_FIFOCTR_BCLR)
- #define UX_RX_HC_FIFOCTR_FRDY
  Initial value:(UX_SYNERGY_HC_FIFOCTR_FRDY)
- #define UX_RX_HC_FIFOCTR_DTLN
  Initial value:(UX_SYNERGY_HC_FIFOCTR_DTLN)
- #define UX_RX_HC_INTENB0_VBSE
  Initial value:(UX_SYNERGY_HC_INTENB0_VBSE)
- #define UX_RX_HC_INTENB0_RSME
  Initial value:(UX_SYNERGY_HC_INTENB0_RSME)
- #define UX_RX_HC_INTENB0_SOFE
  Initial value:(UX_SYNERGY_HC_INTENB0_SOFE)
- #define UX_RX_HC_INTENB0_DVSE
  Initial value:(UX_SYNERGY_HC_INTENB0_DVSE)
- #define UX_RX_HC_INTENB0_CTRE
  Initial value:(UX_SYNERGY_HC_INTENB0_CTRE)
- #define UX_RX_HC_INTENB0_BEMPE
  Initial value:(UX_SYNERGY_HC_INTENB0_BEMPE)
- #define UX_RX_HC_INTENB0_NRDYE
  Initial value:(UX_SYNERGY_HC_INTENB0_NRDYE)
- #define UX_RX_HC_INTENB0_BRDYE
  Initial value:(UX_SYNERGY_HC_INTENB0_BRDYE)
- #define UX_RX_HC_INTENB1_OVRCRE
  Initial value:(UX_SYNERGY_HC_INTENB1_OVRCRE)
- #define UX_RX_HC_INTENB1_BCHGE
  Initial value:(UX_SYNERGY_HC_INTENB1_BCHGE)

- #define UX_RX_HC_INTENB1_DTCHE
  Initial value:(UX_SYNERGY_HC_INTENB1_DTCHE)
- #define UX_RX_HC_INTENB1_ATTCHE
  Initial value:(UX_SYNERGY_HC_INTENB1_ATTCHE)
- #define UX_RX_HC_INTENB1_L1RSMENDE
  Initial value:(UX_SYNERGY_HC_INTENB1_L1RSMENDE)
- #define UX_RX_HC_INTENB1_LPSMENDE
  Initial value:(UX_SYNERGY_HC_INTENB1_LPSMENDE)
- #define UX_RX_HC_INTENB1_EOFERRE
  Initial value:(UX_SYNERGY_HC_INTENB1_EOFERRE)
- #define UX_RX_HC_INTENB1_SIGNE
  Initial value:(UX_SYNERGY_HC_INTENB1_SIGNE)
- #define UX_RX_HC_INTENB1_SACKE
  Initial value:(UX_SYNERGY_HC_INTENB1_SACKE)
- #define UX_RX_HC_INTENB1_PDDETINTE
  Initial value:(UX_SYNERGY_HC_INTENB1_PDDETINTE)
- #define UX_RX_HC_PIPE0
  Initial value:(UX_SYNERGY_HC_PIPE0)
- #define UX_RX_HC_PIPE_ALL
  Initial value:(UX_SYNERGY_HC_PIPE_ALL)
- #define UX_RX_HC_SOFCFG_TRNENSEL
  Initial value:(UX_SYNERGY_HC_SOFCFG_TRNENSEL)
- #define UX_RX_HC_SOFCFG_BRDYM
  Initial value:(UX_SYNERGY_HC_SOFCFG_BRDYM)
- #define UX_RX_HC_SOFCFG_INTL
  Initial value:(UX_SYNERGY_HC_SOFCFG_INTL)
- #define UX_RX_HC_SOFCFG_EDGESTS
  Initial value:(UX_SYNERGY_HC_SOFCFG_EDGESTS)
- #define UX_RX_HC_PHYSET_HSEB
  Initial value:(UX_SYNERGY_HC_PHYSET_HSEB)
- #define UX_RX_HC_PHYSET_REPSTART
  Initial value:(UX_SYNERGY_HC_PHYSET_REPSTART)
- #define UX_RX_HC_PHYSET_REPSEL
  Initial value:(UX_SYNERGY_HC_PHYSET_REPSEL)

- #define UX_RX_HC_PHYSET_CLKSEL_1
  Initial value:(UX_SYNERGY_HC_PHYSET_CLKSEL_1)
- #define UX_RX_HC_PHYSET_CLKSEL_0
  Initial value:(UX_SYNERGY_HC_PHYSET_CLKSEL_0)
- #define UX_RX_HC_PHYSET_CDPEN
  Initial value:(UX_SYNERGY_HC_PHYSET_CDPEN)
- #define UX_RX_HC_PHYSET_PLLRESET
  Initial value:(UX_SYNERGY_HC_PHYSET_PLLRESET)
- #define UX_RX_HC_PHYSET_DIRPD
  Initial value:(UX_SYNERGY_HC_PHYSET_DIRPD)
- #define UX_RX_HC_INTSTS0_VBINT
  Initial value:(UX_SYNERGY_HC_INTSTS0_VBINT)
- #define UX_RX_HC_INTSTS0_RESM
  Initial value:(UX_SYNERGY_HC_INTSTS0_RESM)
- #define UX_RX_HC_INTSTS0_SOFR
  Initial value:(UX_SYNERGY_HC_INTSTS0_SOFR)
- #define UX_RX_HC_INTSTS0_DVST
  Initial value:(UX_SYNERGY_HC_INTSTS0_DVST)
- #define UX_RX_HC_INTSTS0_CTRT
  Initial value:(UX_SYNERGY_HC_INTSTS0_CTRT)
- #define UX_RX_HC_INTSTS0_BEMP
  Initial value:(UX_SYNERGY_HC_INTSTS0_BEMP)
- #define UX_RX_HC_INTSTS0_NRDY
  Initial value:(UX_SYNERGY_HC_INTSTS0_NRDY)
- #define UX_RX_HC_INTSTS0_BRDY
  Initial value:(UX_SYNERGY_HC_INTSTS0_BRDY)
- #define UX_RX_HC_INTSTS0_VBSTS
  Initial value:(UX_SYNERGY_HC_INTSTS0_VBSTS)
- #define UX_RX_HC_INTSTS0_VALID
  Initial value:(UX_SYNERGY_HC_INTSTS0_VALID)
- #define UX_RX_HC_INTSTS0_CTSQ_MASK
  Initial value:(UX_SYNERGY_HC_INTSTS0_CTSQ_MASK)
- #define UX_RX_HC_INTSTS1_OVRCRE
  Initial value:(UX_SYNERGY_HC_INTSTS1_OVRCRE)

- #define UX_RX_HC_INTSTS1_BCHG
  Initial value:(UX_SYNERGY_HC_INTSTS1_BCHG)
- #define UX_RX_HC_INTSTS1_DTCH
  Initial value:(UX_SYNERGY_HC_INTSTS1_DTCH)
- #define UX_RX_HC_INTSTS1_ATTCH
  Initial value:(UX_SYNERGY_HC_INTSTS1_ATTCH)
- #define UX_RX_HC_INTSTS1_EOFERR
  Initial value:(UX_SYNERGY_HC_INTSTS1_EOFERR)
- #define UX_RX_HC_INTSTS1_SIGN
  Initial value:(UX_SYNERGY_HC_INTSTS1_SIGN)
- #define UX_RX_HC_INTSTS1_SACK
  Initial value:(UX_SYNERGY_HC_INTSTS1_SACK)
- #define UX_RX_HC_FRMNUM_OVRN
  Initial value:(UX_SYNERGY_HC_FRMNUM_OVRN)
- #define UX_RX_HC_FRMNUM_CRCE
  Initial value:(UX_SYNERGY_HC_FRMNUM_CRCE)
- #define UX_RX_HC_FRMNUM_FRNM_MASK
  Initial value:(UX_SYNERGY_HC_FRMNUM_FRNM_MASK)
- #define UX_RX_HC_DCPCFG_DIR
  Initial value:(UX_SYNERGY_HC_DCPCFG_DIR)
- #define UX_RX_HC_DCPMAXP_DEVADDR_SHIFT
  Initial value:(UX_SYNERGY_HC_DCPMAXP_DEVADDR_SHIFT)
- #define UX_RX_HC_DCPMAXP_DEVADDR_MASK
  Initial value:(UX_SYNERGY_HC_DCPMAXP_DEVADDR_MASK)
- #define UX_RX_HC_DCPCTR_BSTS
  Initial value:(UX_SYNERGY_HC_DCPCTR_BSTS)
- #define UX_RX_HC_DCPCTR_SUREQ
  Initial value:(UX_SYNERGY_HC_DCPCTR_SUREQ)
- #define UX_RX_HC_DCPCTR_SUREQCLR
  Initial value:(UX_SYNERGY_HC_DCPCTR_SUREQCLR)
- #define UX_RX_HC_DCPCTR_SQCLR
  Initial value:(UX_SYNERGY_HC_DCPCTR_SQCLR)
- #define UX_RX_HC_DCPCTR_SQSET
  Initial value:(UX_SYNERGY_HC_DCPCTR_SQSET)

- #define UX_RX_HC_DCPCTR_SQMON
  Initial value:(UX_SYNERGY_HC_DCPCTR_SQMON)
- #define UX_RX_HC_DCPCTR_PBUSY
  Initial value:(UX_SYNERGY_HC_DCPCTR_PBUSY)
- #define UX_RX_HC_DCPCTR_CCPL
  Initial value:(UX_SYNERGY_HC_DCPCTR_CCPL)
- #define UX_RX_HC_DCPCTR_PID_MASK
  Initial value:(UX_SYNERGY_HC_DCPCTR_PID_MASK)
- #define UX_RX_HC_DCPCTR_PIDNAK
  Initial value:(UX_SYNERGY_HC_DCPCTR_PIDNAK)
- #define UX_RX_HC_DCPCTR_PIDBUF
  Initial value:(UX_SYNERGY_HC_DCPCTR_PIDBUF)
- #define UX_RX_HC_DCPCTR_PIDSTALL
  Initial value:(UX_SYNERGY_HC_DCPCTR_PIDSTALL)
- #define UX_RX_HC_DCPCTR_PIDSTALL2
  Initial value:(UX_SYNERGY_HC_DCPCTR_PIDSTALL2)
- #define UX_RX_HC_PIPECFG_TYPE_MASK
  Initial value:(UX_SYNERGY_HC_PIPECFG_TYPE_MASK)
- #define UX_RX_HC_PIPECFG_TYPE_BIT_USED
  Initial value:(UX_SYNERGY_HC_PIPECFG_TYPE_BIT_USED)
- #define UX_RX_HC_PIPECFG_TYPE_BULK
  Initial value:(UX_SYNERGY_HC_PIPECFG_TYPE_BULK)
- #define UX_RX_HC_PIPECFG_TYPE_INTERRUPT
  Initial value:(UX_SYNERGY_HC_PIPECFG_TYPE_INTERRUPT)
- #define UX_RX_HC_PIPECFG_TYPE_ISOCHRONOUS
  Initial value:(UX_SYNERGY_HC_PIPECFG_TYPE_ISOCHRONOUS)
- #define UX_RX_HC_PIPECFG_BFRE
  Initial value:(UX_SYNERGY_HC_PIPECFG_BFRE)
- #define UX_RX_HC_PIPECFG_DBLB
  Initial value:(UX_SYNERGY_HC_PIPECFG_DBLB)
- #define UX_RX_HC_PIPECFG_SHTNAK
  Initial value:(UX_SYNERGY_HC_PIPECFG_SHTNAK)
- #define UX_RX_HC_PIPECFG_DIR
  Initial value:(UX_SYNERGY_HC_PIPECFG_DIR)

- #define UX_RX_HC_PIPECFG_EPNUM_MASK
  Initial value:(UX_SYNERGY_HC_PIPECFG_EPNUM_MASK)
- #define UX_RX_HC_PIPEBUF_SIZEMASK
  Initial value:(UX_SYNERGY_HC_PIPEBUF_SIZEMASK)
- #define UX_RX_HC_PIPEBUF_BUFNMBMASK
  Initial value:(UX_SYNERGY_HC_PIPEBUF_BUFNMBMASK)
- #define UX_RX_HC_PIPEBUF_SHIFT
  Initial value:(UX_SYNERGY_HC_PIPEBUF_SHIFT)
- #define UX_RX_HC_PIPEMAXP_DEVSELMASK
  Initial value:(UX_SYNERGY_HC_PIPEMAXP_DEVSELMASK)
- #define UX_RX_HC_PIPEMAXP_DEVSEL_SHIFT
  Initial value:(UX_SYNERGY_HC_PIPEMAXP_DEVSEL_SHIFT)
- #define UX_RX_HC_PIPEMAXP_MXPSMASK
  Initial value:(UX_SYNERGY_HC_PIPEMAXP_MXPSMASK)
- #define UX_RX_HC_PIPE1TRE_TRCLR
  Initial value:(UX_SYNERGY_HC_PIPE1TRE_TRCLR)
- #define UX_RX_HC_PIPE1TRE_TRENB
  Initial value:(UX_SYNERGY_HC_PIPE1TRE_TRENB)
- #define UX_RX_HC_FIFO_D0
  Initial value:(UX_SYNERGY_HC_FIFO_D0)
- #define UX_RX_HC_FIFO_D1
  Initial value:(UX_SYNERGY_HC_FIFO_D1)
- #define UX_RX_HC_FIFO_C
  Initial value:(UX_SYNERGY_HC_FIFO_C)
- #define UX_RX_HC_DEVADD_SPEED_LOW
  Initial value:(UX_SYNERGY_HC_DEVADD_SPEED_LOW)
- #define UX_RX_HC_DEVADD_SPEED_FULL
  Initial value:(UX_SYNERGY_HC_DEVADD_SPEED_FULL)
- #define UX_RX_HC_DEVADD_SPEED_HIGH
  Initial value:(UX_SYNERGY_HC_DEVADD_SPEED_HIGH)
- #define UX_RX_HC_DEVADD_UPPHUB_SHIFT
  Initial value:(UX_SYNERGY_HC_DEVADD_UPPHUB_SHIFT)
- #define UX_RX_HC_DEVADD_HUBPORT_SHIFT
  Initial value:(UX_SYNERGY_HC_DEVADD_HUBPORT_SHIFT)

- #define UX_RX_HC_DCPCTR_DATA1
  Initial value:(UX_SYNERGY_HC_DCPCTR_DATA1)
- #define UX_RX_HC_DCPCTR_DATA0
  Initial value:(UX_SYNERGY_HC_DCPCTR_DATA0)
- #define UX_RX_HC_PIPESEL_NO_PIPE
  Initial value:(UX_SYNERGY_HC_PIPESEL_NO_PIPE)
- #define UX_RX_HC_PIPE0_SIZE
  Initial value:(UX_SYNERGY_HC_PIPE0_SIZE)
- #define UX_RX_HC_PIPE_NB_BUFFERS
  Initial value:(UX_SYNERGY_HC_PIPE_NB_BUFFERS)
- #define UX_RX_HC_PIPECTR_BSTS
  Initial value:(UX_SYNERGY_HC_PIPECTR_BSTS)
- #define UX_RX_HC_PIPECTR_INBUFM
  Initial value:(UX_SYNERGY_HC_PIPECTR_INBUFM)
- #define UX_RX_HC_PIPECTR_ATREPM
  Initial value:(UX_SYNERGY_HC_PIPECTR_ATREPM)
- #define UX_RX_HC_PIPECTR_ACLRM
  Initial value:(UX_SYNERGY_HC_PIPECTR_ACLRM)
- #define UX_RX_HC_PIPECTR_SQCLR
  Initial value:(UX_SYNERGY_HC_PIPECTR_SQCLR)
- #define UX_RX_HC_PIPECTR_SQSET
  Initial value:(UX_SYNERGY_HC_PIPECTR_SQSET)
- #define UX_RX_HC_PIPECTR_SQMON
  Initial value:(UX_SYNERGY_HC_PIPECTR_SQMON)
- #define UX_RX_HC_PIPECTR_PBUSY
  Initial value:(UX_SYNERGY_HC_PIPECTR_PBUSY)
- #define UX_RX_HC_PIPECTR_PID_MASK
  Initial value:(UX_SYNERGY_HC_PIPECTR_PID_MASK)
- #define UX_RX_HC_PIPECTR_PIDNAK
  Initial value:(UX_SYNERGY_HC_PIPECTR_PIDNAK)
- #define UX_RX_HC_PIPECTR_PIDBUF
  Initial value:(UX_SYNERGY_HC_PIPECTR_PIDBUF)
- #define UX_RX_HC_PIPECTR_PIDSTALL
  Initial value:(UX_SYNERGY_HC_PIPECTR_PIDSTALL)

- #define UX_RX_HC_PIPECTR_PIDSTALL2
  Initial value:(UX_SYNERGY_HC_PIPECTR_PIDSTALL2)
- #define UX_RX_HC_PIPECTR_DATA1
  Initial value:(UX_SYNERGY_HC_PIPECTR_DATA1)
- #define UX_RX_HC_PIPECTR_DATA0
  Initial value:(UX_SYNERGY_HC_PIPECTR_DATA0)
- #define UX_RX_HC_AVAILABLE_BANDWIDTH
  Initial value:(UX_SYNERGY_HC_AVAILABLE_BANDWIDTH)
- #define UX_RX_HC_INIT_DELAY
  Initial value:(UX_SYNERGY_HC_INIT_DELAY)
- #define UX_RX_HC_RESET_RETRY
  Initial value:(UX_SYNERGY_HC_RESET_RETRY)
- #define UX_RX_HC_RESET_DELAY
  Initial value:(UX_SYNERGY_HC_RESET_DELAY)
- #define UX_RX_HC_PORT_RESET_RETRY
  Initial value:(UX_SYNERGY_HC_PORT_RESET_RETRY)
- #define UX_RX_HC_FORCE_PORT_RESET_RETRY
  Initial value:(UX_SYNERGY_HC_FORCE_PORT_RESET_RETRY)
- #define UX_RX_HC_FORCE_PORT_RESET_DELAY
  Initial value:(UX_SYNERGY_HC_FORCE_PORT_RESET_DELAY)
- #define UX_RX_HC_CHECK_PORT_RESET_RETRY
  Initial value:(UX_SYNERGY_HC_CHECK_PORT_RESET_RETRY)
- #define UX_RX_HC_PORT_RESET_DELAY
  Initial value:(UX_SYNERGY_HC_PORT_RESET_DELAY)
- #define UX_RX_HC_PORT_RESET_RECOVERY_DELAY
  Initial value:(UX_SYNERGY_HC_PORT_RESET_RECOVERY_DELAY)
- #define UX_RX_HC_COMMAND_STATUS_RESET
  Initial value:(UX_SYNERGY_HC_COMMAND_STATUS_RESET)
- #define UX_RX_HC_INIT_RESET_DELAY
  Initial value:(UX_SYNERGY_HC_INIT_RESET_DELAY)
- #define UX_RX_HC_MAX_BUF_SIZE
  Initial value:(UX_SYNERGY_HC_MAX_BUF_SIZE)
- #define UX_RX_HC_MAX_BUF_NUM
  Initial value:(UX_SYNERGY_HC_MAX_BUF_NUM)

- #define UX_RX_HC_FIFO_WRITING
  Initial value:(UX_SYNERGY_HC_FIFO_WRITING)
- #define UX_RX_HC_FIFO_WRITE_END
  Initial value:(UX_SYNERGY_HC_FIFO_WRITE_END)
- #define UX_RX_HC_FIFO_WRITE_SHORT
  Initial value:(UX_SYNERGY_HC_FIFO_WRITE_SHORT)
- #define UX_RX_HC_FIFO_WRITE_DMA
  Initial value:(UX_SYNERGY_HC_FIFO_WRITE_DMA)
- #define UX_RX_HC_FIFO_WRITE_ERROR
  Initial value:(UX_SYNERGY_HC_FIFO_WRITE_ERROR)
- #define UX_RX_HC_FIFO_READING
  Initial value:(UX_SYNERGY_HC_FIFO_READING)
- #define UX_RX_HC_FIFO_READ_END
  Initial value:(UX_SYNERGY_HC_FIFO_READ_END)
- #define UX_RX_HC_FIFO_READ_SHORT
  Initial value:(UX_SYNERGY_HC_FIFO_READ_SHORT)
- #define UX_RX_HC_FIFO_READ_DMA
  Initial value:(UX_SYNERGY_HC_FIFO_READ_DMA)
- #define UX_RX_HC_FIFO_READ_ERROR
  Initial value:(UX_SYNERGY_HC_FIFO_READ_ERROR)
- #define UX_RX_HC_FIFO_READ_OVER
  Initial value:(UX_SYNERGY_HC_FIFO_READ_OVER)
- #define UX_RX_HC_ED_BRDY
  Initial value:(UX_SYNERGY_HC_ED_BRDY)
- #define UX_RX_HC_ED_NRDY
  Initial value:(UX_SYNERGY_HC_ED_NRDY)
- #define UX_RX_HC_ED_BEMP
  Initial value:(UX_SYNERGY_HC_ED_BEMP)
- #define UX_RX_HC_ED_EOFERR
  Initial value:(UX_SYNERGY_HC_ED_EOFERR)
- #define UX_RX_HC_ED_SIGN
  Initial value:(UX_SYNERGY_HC_ED_SIGN)
- #define UX_RX_HC_ED_SACK
  Initial value:(UX_SYNERGY_HC_ED_SACK)

- #define UX_RX_HC_LPSTS_SUSPENDM
  Initial value:(UX_SYNERGY_HC_LPSTS_SUSPENDM)
- #define UX_RX_HC_PORT_ENABLED
  Initial value:(UX_SYNERGY_HC_PORT_ENABLED)
- #define UX_RX_HC_PORT_DISABLED
  Initial value:(UX_SYNERGY_HC_PORT_DISABLED)
- #define UX_RX_ED_STATIC
  Initial value:(UX_SYNERGY_ED_STATIC)
- #define UX_RX_ED_SKIP
  Initial value:(UX_SYNERGY_ED_SKIP)
- #define UX_RX_TD_SETUP_PHASE
  Initial value:(UX_SYNERGY_TD_SETUP_PHASE)
- #define UX_RX_TD_DATA_PHASE
  Initial value:(UX_SYNERGY_TD_DATA_PHASE)
- #define UX_RX_TD_STATUS_PHASE
  Initial value:(UX_SYNERGY_TD_STATUS_PHASE)
- #define UX_RX_TD_OUT
  Initial value:(UX_SYNERGY_TD_OUT)
- #define UX_RX_TD_IN
  Initial value:(UX_SYNERGY_TD_IN)
- #define UX_RX_TD_TOGGLE_FROM_ED
  Initial value:(UX_SYNERGY_TD_TOGGLE_FROM_ED)
- #define UX_RX_TD_ACK_PENDING
  Initial value:(UX_SYNERGY_TD_ACK_PENDING)
- #define UX_RX_TD_SETUP_TYPE
  Initial value:(UX_SYNERGY_TD_SETUP_TYPE)
- #define UX_RX_TD_DATA_TYPE
  Initial value:(UX_SYNERGY_TD_DATA_TYPE)
- #define UX_RX_TD_STATUS_TYPE
  Initial value:(UX_SYNERGY_TD_STATUS_TYPE)
- #define UX_RX_TD_MAX_ERROR
  Initial value:(UX_SYNERGY_TD_MAX_ERROR)
- #define _ux_hcd_rx_asynch_queue_process
  Initial value:_ux_hcd_synergy_asynch_queue_process

- #define _ux_hcd_rx_asynch_schedule
  Initial value:`_ux_hcd_synergy_asynch_schedule`
- #define _ux_hcd_rx_asynchronous_endpoint_create
  Initial value:`_ux_hcd_synergy_asynchronous_endpoint_create`
- #define _ux_hcd_rx_asynchronous_endpoint_destroy
  Initial value:`_ux_hcd_synergy_asynchronous_endpoint_destroy`
- #define _ux_hcd_rx_control_endpoint_create
  Initial value:`_ux_hcd_synergy_control_endpoint_create`
- #define _ux_hcd_rx_bulk_endpoint_create
  Initial value:`_ux_hcd_synergy_bulk_endpoint_create`
- #define _ux_hcd_rx_control_td_add
  Initial value:`_ux_hcd_synergy_control_td_add`
- #define _ux_hcd_rx_bulk_int_td_add
  Initial value:`_ux_hcd_synergy_bulk_int_td_add`
- #define _ux_hcd_rx_controller_disable
  Initial value:`_ux_hcd_synergy_controller_disable`
- #define _ux_hcd_rx_ed_obtain
  Initial value:`_ux_hcd_synergy_ed_obtain`
- #define _ux_hcd_rx_ed_td_clean
  Initial value:`_ux_hcd_synergy_ed_td_clean`
- #define _ux_hcd_rx_endpoint_reset
  Initial value:`_ux_hcd_synergy_endpoint_reset`
- #define _ux_hcd_rx_entry
  Initial value:`_ux_hcd_synergy_entry`
- #define _ux_hcd_rx_frame_number_get
  Initial value:`_ux_hcd_synergy_frame_number_get`
- #define _ux_hcd_rx_frame_number_set
  Initial value:`_ux_hcd_synergy_frame_number_set`
- #define _ux_hcd_rx_register_read
  Initial value:`_ux_hcd_synergy_register_read`
- #define _ux_hcd_rx_register_write
  Initial value:`_ux_hcd_synergy_register_write`
- #define _ux_hcd_rx_register_clear
  Initial value:`_ux_hcd_synergy_register_clear`

- #define _ux_hcd_rx_register_set
  Initial value:_ux_hcd_synergy_register_set
- #define _ux_hcd_rx_initialize
  Initial value:_ux_hcd_synergy_initialize
- #define _ux_hcd_rx_interrupt_endpoint_create
  Initial value:_ux_hcd_synergy_interrupt_endpoint_create
- #define _ux_hcd_rx_interrupt_handler
  Initial value:_ux_hcd_synergy_interrupt_handler
- #define _ux_hcd_rx_iso_queue_process
  Initial value:_ux_hcd_synergy_iso_queue_process
- #define _ux_hcd_rx_iso_schedule
  Initial value:_ux_hcd_synergy_iso_schedule
- #define _ux_hcd_rx_isochronous_endpoint_create
  Initial value:_ux_hcd_synergy_isochronous_endpoint_create
- #define _ux_hcd_rx_isochronous_td_obtain
  Initial value:_ux_hcd_synergy_isochronous_td_obtain
- #define _ux_hcd_rx_least_traffic_list_get
  Initial value:_ux_hcd_synergy_least_traffic_list_get
- #define _ux_hcd_rx_periodic_endpoint_destroy
  Initial value:_ux_hcd_synergy_periodic_endpoint_destroy
- #define _ux_hcd_rx_periodic_schedule
  Initial value:_ux_hcd_synergy_periodic_schedule
- #define _ux_hcd_rx_periodic_tree_create
  Initial value:_ux_hcd_synergy_periodic_tree_create
- #define _ux_hcd_rx_port_disable
  Initial value:_ux_hcd_synergy_port_disable
- #define _ux_hcd_rx_port_enable
  Initial value:_ux_hcd_synergy_port_enable
- #define _ux_hcd_rx_port_reset
  Initial value:_ux_hcd_synergy_port_reset
- #define _ux_hcd_rx_port_resume
  Initial value:_ux_hcd_synergy_port_resume
- #define _ux_hcd_rx_port_status_get
  Initial value:_ux_hcd_synergy_port_status_get

- #define _ux_hcd_rx_port_suspend
  Initial value:_ux_hcd_synergy_port_suspend
- #define _ux_hcd_rx_power_down_port
  Initial value:_ux_hcd_synergy_power_down_port
- #define _ux_hcd_rx_power_on_port
  Initial value:_ux_hcd_synergy_power_on_port
- #define _ux_hcd_rx_power_root_hubs
  Initial value:_ux_hcd_synergy_power_root_hubs
- #define _ux_hcd_rx_td_add
  Initial value:_ux_hcd_synergy_td_add
- #define _ux_hcd_rx_regular_td_obtain
  Initial value:_ux_hcd_synergy_regular_td_obtain
- #define _ux_hcd_rx_request_bulk_transfer
  Initial value:_ux_hcd_synergy_request_bulk_transfer
- #define _ux_hcd_rx_request_control_transfer
  Initial value:_ux_hcd_synergy_request_control_transfer
- #define _ux_hcd_rx_request_interrupt_transfer
  Initial value:_ux_hcd_synergy_request_interrupt_transfer
- #define _ux_hcd_rx_request_isochronous_transfer
  Initial value:_ux_hcd_synergy_request_isochronous_transfer
- #define _ux_hcd_rx_request_transfer
  Initial value:_ux_hcd_synergy_request_transfer
- #define _ux_hcd_rx_transfer_abort
  Initial value:_ux_hcd_synergy_transfer_abort
- #define _ux_hcd_rx_fifo_port_change
  Initial value:_ux_hcd_synergy_fifo_port_change
- #define _ux_hcd_rx_current_endpoint_change
  Initial value:_ux_hcd_synergy_current_endpoint_change
- #define _ux_hcd_rx_data_buffer_size
  Initial value:_ux_hcd_synergy_data_buffer_size
- #define _ux_hcd_rx_buffer_write
  Initial value:_ux_hcd_synergy_buffer_write
- #define _ux_hcd_rx_fifod_write
  Initial value:_ux_hcd_synergy_fifod_write

- #define _ux_hcd_rx_fifoc_write
  Initial value:`_ux_hcd_synergy_fifoc_write`
- #define _ux_hcd_rx_fifo_read
  Initial value:`_ux_hcd_synergy_fifo_read`
- #define _ux_hcd_rx_buffer_read
  Initial value:`_ux_hcd_synergy_buffer_read`
- #define _ux_hcd_rx_buffer_ready_interrupt
  Initial value:`_ux_hcd_synergy_buffer_ready_interrupt`
- #define _ux_hcd_rx_buffer_empty_interrupt
  Initial value:`_ux_hcd_synergy_buffer_empty_interrupt`
- #define _ux_hcd_rx_endpoint_nak_set
  Initial value:`_ux_hcd_synergy_endpoint_nak_set`
- #define _ux_hcd_rx_asynch_queue_process_bemp
  Initial value:`_ux_hcd_synergy_asynch_queue_process_bemp`
- #define _ux_hcd_rx_asynch_queue_process_nrdy
  Initial value:`_ux_hcd_synergy_asynch_queue_process_nrdy`
- #define _ux_hcd_rx_asynch_queue_process_brdy
  Initial value:`_ux_hcd_synergy_asynch_queue_process_brdy`
- #define _ux_hcd_rx_asynch_queue_process_sign
  Initial value:`_ux_hcd_synergy_asynch_queue_process_sign`
- #define _ux_hcd_rx_buffer_notready_interrupt
  Initial value:`_ux_hcd_synergy_buffer_notready_interrupt`
- #define UX_SYNERGY_CONTROLLER
  Initial value:`(0)`
- #define UX_SYNERGY_CONTROLLER_S7G2
  Initial value:`(1)`
- #define UX_SYNERGY_CONTROLLER_S3A7
  Initial value:`(2)`
- #define UX_SYNERGY_MAX_BULK_PAYLOAD
  Initial value:`(512)`
- #define UX_SYNERGY_MAX_CONTROL_PAYLOAD
  Initial value:`(512)`
- #define UX_SYNERGY_FRAME_DELAY
  Initial value:`(4UL)`

- #define UX_SYNERGY_PERIODIC_ENTRY_NB
  Initial value:(32)
- #define UX_SYNERGY_PERIODIC_ENTRY_MASK
  Initial value:(0x1FU)
- #define UX_SYNERGY_ENABLE
  Initial value:(1UL)
- #define UX_SYNERGY_DISABLE
  Initial value:(0UL)
- #define UX_SYNERGY_HC_SYSCFG
  Initial value:(0x00UL)
- #define UX_SYNERGY_HC_BUSWAIT
  Initial value:(0x02UL)
- #define UX_SYNERGY_HC_SYSSTS0
  Initial value:(0x04UL)
- #define UX_SYNERGY_HC_PLLSTA
  Initial value:(0x06UL)
- #define UX_SYNERGY_HC_DVSTCTR0
  Initial value:(0x08UL)
- #define UX_SYNERGY_HC_CFIFO
  Initial value:(0x14UL)
- #define UX_SYNERGY_HC_CFIFOH
  Initial value:(0x16UL)
- #define UX_SYNERGY_HC_CFIFOHH
  Initial value:(0x17UL)
- #define UX_SYNERGY_HC_D0FIFO
  Initial value:(0x18UL)
- #define UX_SYNERGY_HC_D1FIFO
  Initial value:(0x1CUL)
- #define UX_SYNERGY_HC_CFIFOSEL
  Initial value:(0x20UL)
- #define UX_SYNERGY_HC_CFIFOCTR
  Initial value:(0x22UL)
- #define UX_SYNERGY_HC_D0FIFOSEL
  Initial value:(0x28UL)

- #define UX_SYNERGY_HC_D0FIFOCTR
  Initial value:(0x2AUL)
- #define UX_SYNERGY_HC_D1FIFOSEL
  Initial value:(0x2CUL)
- #define UX_SYNERGY_HC_D1FIFOCTR
  Initial value:(0x2EUL)
- #define UX_SYNERGY_HC_INTENB0
  Initial value:(0x30UL)
- #define UX_SYNERGY_HC_INTENB1
  Initial value:(0x32UL)
- #define UX_SYNERGY_HC_BRDYENB
  Initial value:(0x36UL)
- #define UX_SYNERGY_HC_NRDYENB
  Initial value:(0x38UL)
- #define UX_SYNERGY_HC_BEMPENB
  Initial value:(0x3AUL)
- #define UX_SYNERGY_HC_SOFCFG
  Initial value:(0x3CUL)
- #define UX_SYNERGY_HC_PHYSET
  Initial value:(0x3EUL)
- #define UX_SYNERGY_HC_INTSTS0
  Initial value:(0x40UL)
- #define UX_SYNERGY_HC_INTSTS1
  Initial value:(0x42UL)
- #define UX_SYNERGY_HC_BRDYSTS
  Initial value:(0x46UL)
- #define UX_SYNERGY_HC_NRDYSTS
  Initial value:(0x48UL)
- #define UX_SYNERGY_HC_BEMPSTS
  Initial value:(0x4AUL)
- #define UX_SYNERGY_HC_FRMNUM
  Initial value:(0x4CUL)
- #define UX_SYNERGY_HC_DVCHGR
  Initial value:(0x4EUL)

- #define UX_SYNERGY_HC_USBADDR
  Initial value:(0x50UL)
- #define UX_SYNERGY_HC_USBREQ
  Initial value:(0x54UL)
- #define UX_SYNERGY_HC_USBVAL
  Initial value:(0x56UL)
- #define UX_SYNERGY_HC_USBINDX
  Initial value:(0x58UL)
- #define UX_SYNERGY_HC_USBLENG
  Initial value:(0x5AUL)
- #define UX_SYNERGY_HC_DCPCFG
  Initial value:(0x5CUL)
- #define UX_SYNERGY_HC_DCPMAXP
  Initial value:(0x5EUL)
- #define UX_SYNERGY_HC_DCPCTR
  Initial value:(0x60UL)
- #define UX_SYNERGY_HC_PIPESEL
  Initial value:(0x64UL)
- #define UX_SYNERGY_HC_PIPECFG
  Initial value:(0x68UL)
- #define UX_SYNERGY_HC_PIPEBUF
  Initial value:(0x6AUL)
- #define UX_SYNERGY_HC_PIPEMAXP
  Initial value:(0x6CUL)
- #define UX_SYNERGY_HC_PIPEPERI
  Initial value:(0x6EUL)
- #define UX_SYNERGY_HC_PIPE1CTR
  Initial value:(0x70UL)
- #define UX_SYNERGY_HC_PIPE2CTR
  Initial value:(0x72UL)
- #define UX_SYNERGY_HC_PIPE3CTR
  Initial value:(0x74UL)
- #define UX_SYNERGY_HC_PIPE4CTR
  Initial value:(0x76UL)

- #define UX_SYNERGY_HC_PIPE5CTR
  Initial value:(0x78UL)
- #define UX_SYNERGY_HC_PIPE6CTR
  Initial value:(0x7AUL)
- #define UX_SYNERGY_HC_PIPE7CTR
  Initial value:(0x7CUL)
- #define UX_SYNERGY_HC_PIPE8CTR
  Initial value:(0x7EUL)
- #define UX_SYNERGY_HC_PIPE9CTR
  Initial value:(0x80UL)
- #define UX_SYNERGY_HC_PIPE1TRE
  Initial value:(0x90UL)
- #define UX_SYNERGY_HC_PIPE1TRN
  Initial value:(0x92UL)
- #define UX_SYNERGY_HC_PIPE2TRE
  Initial value:(0x94UL)
- #define UX_SYNERGY_HC_PIPE2TRN
  Initial value:(0x96UL)
- #define UX_SYNERGY_HC_PIPE3TRE
  Initial value:(0x98UL)
- #define UX_SYNERGY_HC_PIPE3TRN
  Initial value:(0x9AUL)
- #define UX_SYNERGY_HC_PIPE4TRE
  Initial value:(0x9CUL)
- #define UX_SYNERGY_HC_PIPE4TRN
  Initial value:(0x9EUL)
- #define UX_SYNERGY_HC_PIPE5TRE
  Initial value:(0xA0UL)
- #define UX_SYNERGY_HC_PIPE5TRN
  Initial value:(0xA2UL)
- #define UX_SYNERGY_HC_USBMC
  Initial value:(0xCCUL)
- #define UX_SYNERGY_HC_DEVADD0
  Initial value:(0xD0UL)

- #define UX_SYNERGY_HC_DEVADD1
  Initial value:`(0xD2UL)`
- #define UX_SYNERGY_HC_DEVADD2
  Initial value:`(0xD4UL)`
- #define UX_SYNERGY_HC_DEVADD3
  Initial value:`(0xD6UL)`
- #define UX_SYNERGY_HC_DEVADD4
  Initial value:`(0xD8UL)`
- #define UX_SYNERGY_HC_DEVADD5
  Initial value:`(0xDAUL)`
- #define UX_SYNERGY_HC_PHYSLEW
  Initial value:`(0xF0UL)`
- #define UX_SYNERGY_HC_LPSTS
  Initial value:`(0x102UL)`
- #define UX_SYNERGY_HC_PFKUSB_ENABLED
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_PFKUSB_MODE_HOST
  Initial value:`(1)`
- #define UX_SYNERGY_HC_BUSWAIT_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_HC_BUSWAIT_CALC_FREQ_PCLK_CYC
  Initial value:`(17142857U)`
- #define UX_SYNERGY_HC_BUSWAIT_CALC_FREQ_PCLK_CYCX10
  Initial value:`(1714288U)`
- #define UX_SYNERGY_HC_SYSCFG_SCKE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_SYSCFG_CNEN
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_SYSCFG_HSE
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_SYSCFG_DCFM
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_SYSCFG_DRPD
  Initial value:`(1U<<5)`

- #define UX_SYNERGY_HC_SYSCFG_DPRPU
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_SYSCFG_USBE
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_HC_SYSSTS0_LNST
  Initial value:`(3)`
- #define UX_SYNERGY_HC_SYSSTS0_IDMON
  Initial value:`(1U<<2)`
- #define UX_SYNERGY_HC_SYSSTS0_SOFEA
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_SYSSTS0_HTACT
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_SYSSTS0_OVCMON
  Initial value:`(0xC0000U)`
- #define UX_SYNERGY_HC_SYSSTS0_LNST_J_STATE_FS
  Initial value:`(1U)`
- #define UX_SYNERGY_HC_SYSSTS0_LNST_J_STATE_LS
  Initial value:`(2U)`
- #define UX_SYNERGY_HC_PLLSTA_PLLLOCK
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_HC_DVSTCTR0_HNPBTOA
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_DVSTCTR0_EXICEN
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_DVSTCTR0_VBUSEN
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_HC_DVSTCTR0_WKUP
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_DVSTCTR0_RWUPE
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_DVSTCTR0_USBRST
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_DVSTCTR0_RESUME
  Initial value:`(1U<<5)`

- #define UX_SYNERGY_HC_DVSTCTR0_UACT
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_DVSTCTR0_RHST
  Initial value:`(0x7U)`
- #define UX_SYNERGY_HC_DVSTCTR0_SPEED_LOW
  Initial value:`(1)`
- #define UX_SYNERGY_HC_DVSTCTR0_SPEED_FULL
  Initial value:`(2)`
- #define UX_SYNERGY_HC_DVSTCTR0_SPEED_HIGH
  Initial value:`(3)`
- #define UX_SYNERGY_HC_DVSTCTR0_RESET_IN_PROGRESS
  Initial value:`(4)`
- #define UX_SYNERGY_HC_CFIFOSEL_RCNT
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_CFIFOSEL_REW
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_CFIFOSEL_MBW_8
  Initial value:`(0U<<10)`
- #define UX_SYNERGY_HC_CFIFOSEL_MBW_16
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_CFIFOSEL_MBW_32
  Initial value:`(2U<<10)`
- #define UX_SYNERGY_HC_CFIFOSEL_MBW_MASK
  Initial value:`(3U<<10)`
- #define UX_SYNERGY_HC_CFIFOSEL_BIGEND
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_CFIFOSEL_ISEL
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_CFIFOSEL_CURPIPE_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_HC_DFIFOSEL_RCNT
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_DFIFOSEL_REW
  Initial value:`(1U<<14)`

- #define UX_SYNERGY_HC_DFIFOSEL_DCLRM
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_HC_DFIFOSEL_DREQE
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_HC_DFIFOSEL_MBW_8
  Initial value:`(0U<<10)`
- #define UX_SYNERGY_HC_DFIFOSEL_MBW_16
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_DFIFOSEL_MBW_32
  Initial value:`(2U<<10)`
- #define UX_SYNERGY_HC_DFIFOSEL_MBW_MASK
  Initial value:`(3U<<10)`
- #define UX_SYNERGY_HC_DFIFOSEL_BIGEND
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_DFIFOSEL_CURPIPE_MASK
  Initial value:`(0xF)`
- #define UX_SYNERGY_HC_FIFOCTR_BVAL
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_FIFOCTR_BCLR
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_FIFOCTR_FRDY
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_HC_FIFOCTR_DTLN
  Initial value:`(0xFFF)`
- #define UX_SYNERGY_HC_INTENB0_VBSE
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_INTENB0_RSME
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_INTENB0_SOFE
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_HC_INTENB0_DVSE
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_HC_INTENB0_CTRE
  Initial value:`(1U<<11)`

- #define UX_SYNERGY_HC_INTENB0_BEMPE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_INTENB0_NRDYE
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_HC_INTENB0_BRDYE
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_INTENB1_OVRCRE
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_INTENB1_BCHGE
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_INTENB1_DTCHE
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_HC_INTENB1_ATTCHE
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_INTENB1_L1RSMENDE
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_HC_INTENB1_LPSMENDE
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_INTENB1_EOFERRE
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_INTENB1_SIGNE
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_INTENB1_SACKE
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_INTENB1_PDDETINTE
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_HC_PIPE0
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_HC_PIPE_ALL
  Initial value:`(0x3FF)`
- #define UX_SYNERGY_HC_SOFCFG_TRNENSEL
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_SOFCFG_BRDYM
  Initial value:`(1U<<6)`

- #define UX_SYNERGY_HC_SOFCFG_INTL
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_SOFCFG_EDGESTS
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_PHYSET_HSEB
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_PHYSET_REPSTART
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_PHYSET_REPSEL
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_PHYSET_CLKSEL_1
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_PHYSET_CLKSEL_0
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_PHYSET_CDPEN
  Initial value:`(1U<<3)`
- #define UX_SYNERGY_HC_PHYSET_PLLRESET
  Initial value:`(1U<<1)`
- #define UX_SYNERGY_HC_PHYSET_DIRPD
  Initial value:`(1U<<0)`
- #define UX_SYNERGY_HC_INTSTS0_VBINT
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_INTSTS0_RESM
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_INTSTS0_SOFR
  Initial value:`(1U<<13)`
- #define UX_SYNERGY_HC_INTSTS0_DVST
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_HC_INTSTS0_CTRT
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_INTSTS0_BEMP
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_INTSTS0_NRDY
  Initial value:`(1U<<9)`

- #define UX_SYNERGY_HC_INTSTS0_BRDY
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_INTSTS0_VBSTS
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_INTSTS0_VALID
  Initial value:`(1U<<3)`
- #define UX_SYNERGY_HC_INTSTS0_CTSQ_MASK
  Initial value:`(7)`
- #define UX_SYNERGY_HC_INTSTS0_ALL_CLEAR
  Initial value:`(0U)`
- #define UX_SYNERGY_HC_INTSTS1_OVRCRE
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_INTSTS1_BCHG
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_INTSTS1_DTCH
  Initial value:`(1U<<12)`
- #define UX_SYNERGY_HC_INTSTS1_ATTCH
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_INTSTS1_EOFERR
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_INTSTS1_SIGN
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_INTSTS1_SACK
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_INTSTS1_ALL_CLEAR
  Initial value:`(0U)`
- #define UX_SYNERGY_HC_FRMNUM_OVRN
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_FRMNUM_CRCE
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_FRMNUM_FRNM_MASK
  Initial value:`(0x7FFU)`
- #define UX_SYNERGY_HC_DCPCFG_DIR
  Initial value:`(1U<<4)`

- #define UX_SYNERGY_HC_DCPMAXP_DEVADDR_SHIFT
  Initial value:`(12U)`
- #define UX_SYNERGY_HC_DCPMAXP_DEVADDR_MASK
  Initial value:`(0xf000U)`
- #define UX_SYNERGY_HC_DCPCTR_BSTS
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_DCPCTR_SUREQ
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_DCPCTR_SUREQCLR
  Initial value:`(1U<<11)`
- #define UX_SYNERGY_HC_DCPCTR_SQCLR
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_DCPCTR_SQSET
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_DCPCTR_SQMON
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_DCPCTR_PBUSY
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_DCPCTR_CCPL
  Initial value:`(1U<<2)`
- #define UX_SYNERGY_HC_DCPCTR_PID_MASK
  Initial value:`(3UL)`
- #define UX_SYNERGY_HC_DCPCTR_PIDNAK
  Initial value:`(0UL)`
- #define UX_SYNERGY_HC_DCPCTR_PIDBUF
  Initial value:`(1UL)`
- #define UX_SYNERGY_HC_DCPCTR_PIDSTALL
  Initial value:`(2UL)`
- #define UX_SYNERGY_HC_DCPCTR_PIDSTALL2
  Initial value:`(3UL)`
- #define UX_SYNERGY_HC_PIPECFG_TYPE_MASK
  Initial value:`(0xC000)`
- #define UX_SYNERGY_HC_PIPECFG_TYPE_BIT_USED
  Initial value:`(0)`

- #define UX_SYNERGY_HC_PIPECFG_TYPE_BULK
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_PIPECFG_TYPE_INTERRUPT
  Initial value:`(2U<<14)`
- #define UX_SYNERGY_HC_PIPECFG_TYPE_ISOCHRONOUS
  Initial value:`(3U<<14)`
- #define UX_SYNERGY_HC_PIPECFG_BFRE
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_PIPECFG_DBLB
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_HC_PIPECFG_CNTMD
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_PIPECFG_SHTNAK
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_PIPECFG_DIR
  Initial value:`(1U<<4)`
- #define UX_SYNERGY_HC_PIPECFG_EPNUM_MASK
  Initial value:`(0xFU)`
- #define UX_SYNERGY_HC_PIPEBUF_SIZEMASK
  Initial value:`(0x1FU<<10)`
- #define UX_SYNERGY_HC_PIPEBUF_BUFNMBMASK
  Initial value:`(0xFFU<<10)`
- #define UX_SYNERGY_HC_PIPEBUF_SHIFT
  Initial value:`(10U)`
- #define UX_SYNERGY_HC_PIPEMAXP_DEVSELMASK
  Initial value:`(0xFU<<12)`
- #define UX_SYNERGY_HC_PIPEMAXP_DEVSEL_SHIFT
  Initial value:`(12U)`
- #define UX_SYNERGY_HC_PIPEMAXP_MXPSMASK
  Initial value:`(0x7FF)`
- #define UX_SYNERGY_HC_PIPETRE_TRENB
  Initial value:`(1U<<9)`
- #define UX_SYNERGY_HC_FIFO_D0
  Initial value:`(0UL)`

- #define UX_SYNERGY_HC_FIFO_D1
  Initial value:`(1UL)`
- #define UX_SYNERGY_HC_FIFO_C
  Initial value:`(2UL)`
- #define UX_SYNERGY_HC_DEVADD_SPEED_LOW
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_DEVADD_SPEED_FULL
  Initial value:`(2U<<6)`
- #define UX_SYNERGY_HC_DEVADD_SPEED_HIGH
  Initial value:`(3U<<6)`
- #define UX_SYNERGY_HC_DEVADD_UPPHUB_SHIFT
  Initial value:`(11U)`
- #define UX_SYNERGY_HC_DEVADD_HUBPORT_SHIFT
  Initial value:`(8U)`
- #define UX_SYNERGY_HC_USBMC_VDCEN
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_DCPCTR_DATA1
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_DCPCTR_DATA0
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_PIPESEL_NO_PIPE
  Initial value:`0x000f`
- #define UX_SYNERGY_HC_PIPE0_SIZE
  Initial value:`(256)`
- #define UX_SYNERGY_HC_PIPE_NB_BUFFERS
  Initial value:`(64)`
- #define UX_SYNERGY_HC_PIPECTR_BSTS
  Initial value:`(1U<<15)`
- #define UX_SYNERGY_HC_PIPECTR_INBUFM
  Initial value:`(1U<<14)`
- #define UX_SYNERGY_HC_PIPECTR_ATREPM
  Initial value:`(1U<<10)`
- #define UX_SYNERGY_HC_PIPECTR_ACLRM
  Initial value:`(1U<<9)`

- #define UX_SYNERGY_HC_PIPECTR_SQCLR
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_PIPECTR_SQSET
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_PIPECTR_SQMON
  Initial value:`(1U<<6)`
- #define UX_SYNERGY_HC_PIPECTR_PBUSY
  Initial value:`(1U<<5)`
- #define UX_SYNERGY_HC_PIPECTR_PID_MASK
  Initial value:`(3)`
- #define UX_SYNERGY_HC_PIPECTR_PIDNAK
  Initial value:`(0)`
- #define UX_SYNERGY_HC_PIPECTR_PIDBUF
  Initial value:`(1)`
- #define UX_SYNERGY_HC_PIPECTR_PIDSTALL
  Initial value:`(2)`
- #define UX_SYNERGY_HC_PIPECTR_PIDSTALL2
  Initial value:`(3)`
- #define UX_SYNERGY_HC_PIPECTR_DATA1
  Initial value:`(1U<<7)`
- #define UX_SYNERGY_HC_PIPECTR_DATA0
  Initial value:`(1U<<8)`
- #define UX_SYNERGY_HC_AVAILABLE_BANDWIDTH
  Initial value:`(920UL)`
- #define UX_SYNERGY_HC_INIT_DELAY
  Initial value:`(1000)`
- #define UX_SYNERGY_HC_RESET_RETRY
  Initial value:`(1000)`
- #define UX_SYNERGY_HC_RESET_DELAY
  Initial value:`(10)`
- #define UX_SYNERGY_HC_PORT_RESET_RETRY
  Initial value:`(50)`
- #define UX_SYNERGY_HC_FORCE_PORT_RESET_RETRY
  Initial value:`(50)`

- #define UX_SYNERGY_HC_FORCE_PORT_RESET_DELAY
  Initial value:`(1)`
- #define UX_SYNERGY_HC_CHECK_PORT_RESET_RETRY
  Initial value:`(500)`
- #define UX_SYNERGY_HC_PORT_RESET_DELAY
  Initial value:`(300)`
- #define UX_SYNERGY_HC_PORT_RESET_RECOVERY_DELAY
  Initial value:`(100)`
- #define UX_SYNERGY_HC_COMMAND_STATUS_RESET
  Initial value:`(0)`
- #define UX_SYNERGY_HC_INIT_RESET_DELAY
  Initial value:`(10)`
- #define UX_SYNERGY_HC_MAX_BUF_SIZE
  Initial value:`(64)`
- #define UX_SYNERGY_HC_MAX_BUF_NUM
  Initial value:`(135)`
- #define UX_SYNERGY_HC_PIPE1_BUF_START_NUM
  Initial value:`(8)`
- #define UX_SYNERGY_HC_FIFO_WRITING
  Initial value:`(2)`
- #define UX_SYNERGY_HC_FIFO_WRITE_END
  Initial value:`(3)`
- #define UX_SYNERGY_HC_FIFO_WRITE_SHORT
  Initial value:`(4)`
- #define UX_SYNERGY_HC_FIFO_WRITE_DMA
  Initial value:`(5)`
- #define UX_SYNERGY_HC_FIFO_WRITE_ERROR
  Initial value:`(6)`
- #define UX_SYNERGY_HC_FIFO_READING
  Initial value:`(2)`
- #define UX_SYNERGY_HC_FIFO_READ_END
  Initial value:`(3)`
- #define UX_SYNERGY_HC_FIFO_READ_SHORT
  Initial value:`(4)`

- #define UX_SYNERGY_HC_FIFO_READ_DMA
  Initial value:(5)
- #define UX_SYNERGY_HC_FIFO_READ_ERROR
  Initial value:(6)
- #define UX_SYNERGY_HC_FIFO_READ_OVER
  Initial value:(7)
- #define UX_SYNERGY_HC_ED_BRDY
  Initial value:(0x00000001U)
- #define UX_SYNERGY_HC_ED_NRDY
  Initial value:(0x00000002U)
- #define UX_SYNERGY_HC_ED_BEMP
  Initial value:(0x00000004U)
- #define UX_SYNERGY_HC_ED_EOFERR
  Initial value:(0x00000010U)
- #define UX_SYNERGY_HC_ED_SIGN
  Initial value:(0x00000020U)
- #define UX_SYNERGY_HC_ED_SACK
  Initial value:(0x00000040U)
- #define UX_SYNERGY_HC_ED_TIMEOUT
  Initial value:(0x00000080U)
- #define UX_SYNERGY_HC_PHYSLEW_SLEW_SLEWR00
  Initial value:(1U<<0)
- #define UX_SYNERGY_HC_PHYSLEW_SLEW_SLEWR01
  Initial value:(1U<<1)
- #define UX_SYNERGY_HC_PHYSLEW_SLEW_SLEWF00
  Initial value:(1U<<2)
- #define UX_SYNERGY_HC_PHYSLEW_SLEW_SLEWF01
  Initial value:(1U<<3)
- #define UX_SYNERGY_HC_LPSTS_SUSPENDM
  Initial value:(1U<<14)
- #define UX_SYNERGY_HC_PORT_ENABLED
  Initial value:(1)
- #define UX_SYNERGY_HC_PORT_DISABLED
  Initial value:(0)

- #define UX_SYNERGY_ED_STATIC
  Initial value:`(0x80000000)`
- #define UX_SYNERGY_ED_SKIP
  Initial value:`(0x40000000UL)`
- #define UX_SYNERGY_TD_SETUP_PHASE
  Initial value:`(0x00010000)`
- #define UX_SYNERGY_TD_DATA_PHASE
  Initial value:`(0x00020000)`
- #define UX_SYNERGY_TD_STATUS_PHASE
  Initial value:`(0x00040000)`
- #define UX_SYNERGY_TD_OUT
  Initial value:`(0x00000800)`
- #define UX_SYNERGY_TD_IN
  Initial value:`(0x00001000)`
- #define UX_SYNERGY_TD_TOGGLE_FROM_ED
  Initial value:`(0x80000000)`
- #define UX_SYNERGY_TD_ACK_PENDING
  Initial value:`(0x00080000)`
- #define UX_SYNERGY_TD_SETUP_TYPE
  Initial value:`(1)`
- #define UX_SYNERGY_TD_DATA_TYPE
  Initial value:`(2)`
- #define UX_SYNERGY_TD_STATUS_TYPE
  Initial value:`(3)`
- #define UX_SYNERGY_TD_MAX_ERROR
  Initial value:`(3)`
- #define UX_HOST_CLASS_CDC_ACM_DEVICE_INIT_DELAY
  Initial value:`(1 * UX_PERIODIC_RATE)`
- #define UX_HOST_CLASS_CDC_ACM_CLASS_TRANSFER_TIMEOUT
  Initial value:`300000`
- #define UX_HOST_CLASS_CDC_DATA_CLASS
  Initial value:`0x0A`
- #define UX_HOST_CLASS_CDC_CONTROL_CLASS
  Initial value:`0x02`

- #define UX_HOST_CLASS_CDC_ACM_SUBCLASS
  Initial value:0X02
- #define UX_HOST_CLASS_CDC_DLC_SUBCLASS
  Initial value:0X01
- #define UX_HOST_CLASS_CDC_ACM_CS_INTERFACE
  Initial value:0x24
- #define UX_HOST_CLASS_CDC_ACM_HEADER_DESCRIPTOR
  Initial value:0X00
- #define UX_HOST_CLASS_CDC_ACM_CALL_MANAGEMENT_DESCRIPTOR
  Initial value:0X01
- #define UX_HOST_CLASS_CDC_ACM_ABSTRACT_CONTROL_MGT_DESCRIPTOR
  Initial value:0X02
- #define UX_HOST_CLASS_CDC_ACM_DIRECT_LINE_MGT_DESCRIPTOR
  Initial value:0X03
- #define UX_HOST_CLASS_CDC_ACM_TELEPHONE_RINGER_DESCRIPTOR
  Initial value:0X04
- #define UX_HOST_CLASS_CDC_ACM_REPORT_CAPABILITY_DESCRIPTOR
  Initial value:0X05
- #define UX_HOST_CLASS_CDC_ACM_UNION_DESCRIPTOR
  Initial value:0X06
- #define UX_HOST_CLASS_CDC_ACM_COUNTRY_SELECTION_DESCRIPTOR
  Initial value:0X07
- #define UX_HOST_CLASS_CDC_ACM_TELEPHONE_OPERATIONAL_DESCRIPTOR
  Initial value:0X08
- #define UX_HOST_CLASS_CDC_ACM_USB_TERMINAL_DESCRIPTOR
  Initial value:0X09
- #define UX_HOST_CLASS_CDC_ACM_CALL_MANAGEMENT_CAPABILITIES
  Initial value:0x03
- #define UX_HOST_CLASS_CDC_ACM_CALL_MANAGEMENT_DCM
  Initial value:0x01
- #define UX_HOST_CLASS_CDC_ACM_CALL_MANAGEMENT_DCI
  Initial value:0x02
- #define UX_HOST_CLASS_CDC_ACM_REQ_SEND_ENCAPSULATED_COMMAND
  Initial value:0x00

- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_ENCAPSULATED_COMMAND
  Initial value:`0x01`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_COMM_FEATURE
  Initial value:`0x02`
- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_COMM_FEATURE
  Initial value:`0x03`
- #define UX_HOST_CLASS_CDC_ACM_REQ_CLEAR_COMM_FEATURE
  Initial value:`0x04`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_AUX_LINE_STATE
  Initial value:`0x10`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_HOOK_STATE
  Initial value:`0x11`
- #define UX_HOST_CLASS_CDC_ACM_REQ_PULSE_SETUP
  Initial value:`0x12`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SEND_PULSE
  Initial value:`0x13`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_PUSLE_TIME
  Initial value:`0x14`
- #define UX_HOST_CLASS_CDC_ACM_REQ_RING_AUX_JACK
  Initial value:`0x15`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_LINE_CODING
  Initial value:`0x20`
- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_LINE_CODING
  Initial value:`0x21`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_LINE_STATE
  Initial value:`0x22`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SEND_BREAK
  Initial value:`0x23`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_RINGER_PARMS
  Initial value:`0x30`
- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_RINGER_PARMS
  Initial value:`0x31`
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_OPERATION_PARMS
  Initial value:`0x32`

- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_OPERATION_PARMS
  Initial value:0x33
- #define UX_HOST_CLASS_CDC_ACM_REQ_SET_LINE_PARMS
  Initial value:0x34
- #define UX_HOST_CLASS_CDC_ACM_REQ_GET_LINE_PARMS
  Initial value:0x35
- #define UX_HOST_CLASS_CDC_ACM_CTRL_DTR
  Initial value:0x01
- #define UX_HOST_CLASS_CDC_ACM_CTRL_RTS
  Initial value:0x02
- #define UX_HOST_CLASS_CDC_ACM_CTRL_DCD
  Initial value:0x01
- #define UX_HOST_CLASS_CDC_ACM_CTRL_DSR
  Initial value:0x02
- #define UX_HOST_CLASS_CDC_ACM_CTRL_BRK
  Initial value:0x04
- #define UX_HOST_CLASS_CDC_ACM_CTRL_RI
  Initial value:0x08
- #define UX_HOST_CLASS_CDC_ACM_CTRL_FRAMING
  Initial value:0x10
- #define UX_HOST_CLASS_CDC_ACM_CTRL_PARITY
  Initial value:0x20
- #define UX_HOST_CLASS_CDC_ACM_CTRL_OVERRUN
  Initial value:0x40
- #define UX_HOST_CLASS_CDC_ACM_PACKET_SIZE
  Initial value:128
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_DEFAULT_RATE
  Initial value:9600
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_DEFAULT_DATA_BIT
  Initial value:8
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_STOP_BIT_0
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_STOP_BIT_15
  Initial value:1

- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_STOP_BIT_2
  Initial value:2
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_PARITY_NONE
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_PARITY_ODD
  Initial value:1
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_PARITY_EVEN
  Initial value:2
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_PARITY_MARK
  Initial value:3
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_PARITY_SPACE
  Initial value:4
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_RATE
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_LINE_CODING_DATA_BIT
  Initial value:6
- #define UX_HOST_CLASS_CDC_ACM_LINE_STATE_STOP_BIT_0
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_SET_LINE_CODING
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_GET_LINE_CODING
  Initial value:1
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_SET_LINE_STATE
  Initial value:2
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_SEND_BREAK
  Initial value:3
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_ABORT_IN_PIPE
  Initial value:5
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_ABORT_OUT_PIPE
  Initial value:6
- #define UX_HOST_CLASS_CDC_ACM_IOCTL_NOTIFICATION_CALLBACK
  Initial value:7

- #define UX_HOST_CLASS_CDC_ACM_IOCTL_GET_DEVICE_STATUS
  Initial value:8
- #define UX_HOST_CLASS_CDC_ACM_RECEPTION_STATE_STOPPED
  Initial value:0
- #define UX_HOST_CLASS_CDC_ACM_RECEPTION_STATE_STARTED
  Initial value:1
- #define UX_HOST_CLASS_CDC_ACM_RECEPTION_STATE_IN_TRANSFER
  Initial value:2
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_NETWORK_CONNECTION
  Initial value:0x00
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_RESPONSE_AVAILABLE
  Initial value:0x01
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_SERIAL_STATE
  Initial value:0x20
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_CALL_STATE_CHANGE
  Initial value:0x28
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_LINE_STATE_CHANGE
  Initial value:0x29
- #define UX_HOST_CLASS_CDC_ACM_NOTIFICATION_SPEED_CHANGE
  Initial value:0x2A
- #define UX_HOST_CLASS_CDC_ACM_NPF_REQUEST_TYPE
  Initial value:0x00
- #define UX_HOST_CLASS_CDC_ACM_NPF_NOTIFICATION_TYPE
  Initial value:0x01
- #define UX_HOST_CLASS_CDC_ACM_NPF_VALUE
  Initial value:0x02
- #define UX_HOST_CLASS_CDC_ACM_NPF_INDEX
  Initial value:0x04
- #define UX_HOST_CLASS_CDC_ACM_NPF_LENGTH
  Initial value:0x06
- #define UX_HOST_CLASS_DPUMP_CLASS_TRANSFER_TIMEOUT
  Initial value:300000
- #define UX_HOST_CLASS_DPUMP_SUBCLASS
  Initial value:0x99

- #define UX_HOST_CLASS_DPUMP_PROTOCOL
  Initial value:`0x99`
- #define UX_HOST_CLASS_DPUMP_PACKET_SIZE
  Initial value:`128`
- #define UX_HOST_CLASS_DPUMP_SELECT_ALTERNATE_SETTING
  Initial value:`1`
- #define UX_HOST_CLASS_DPUMP_GENERIC_NAME
  Initial value:`"USB DPUMP"`
- #define UX_HOST_CLASS_HID_CLASS
  Initial value:`3`
- #define UX_HOST_CLASS_HID_FIELDS
  Initial value:`16`
- #define UX_HOST_CLASS_HID_MAX_COLLECTION
  Initial value:`4`
- #define UX_HOST_CLASS_HID_MAX_REPORT
  Initial value:`8`
- #define UX_HOST_CLASS_HID_REPORT_SIZE
  Initial value:`32`
- #define UX_HOST_CLASS_HID_DESCRIPTOR
  Initial value:`0x21`
- #define UX_HOST_CLASS_HID_ITEM_LENGTH_MASK
  Initial value:`3`
- #define UX_HOST_CLASS_HID_ITEM_TAG_MASK
  Initial value:`0xf0`
- #define UX_HOST_CLASS_HID_ITEM_TAG_SHORT
  Initial value:`1`
- #define UX_HOST_CLASS_HID_ITEM_TAG_LONG
  Initial value:`0xf0`
- #define UX_HOST_CLASS_HID_MAX_CLIENTS
  Initial value:`8`
- #define UX_HOST_CLASS_HID_DECOMPRESSION_BUFFER
  Initial value:`4096`
- #define UX_HOST_CLASS_HID_REPORT_DECOMPRESSED
  Initial value:`0`

- #define UX_HOST_CLASS_HID_REPORT_RAW
  Initial value:`1`
- #define UX_HOST_CLASS_HID_REPORT_INDIVIDUAL_USAGE
  Initial value:`2`
- #define UX_HOST_CLASS_HID_INTERRUPT_ENDPOINT_READY
  Initial value:`1`
- #define UX_HOST_CLASS_HID_INTERRUPT_ENDPOINT_ACTIVE
  Initial value:`2`
- #define UX_HOST_CLASS_HID_TYPE_MAIN
  Initial value:`0x0`
- #define UX_HOST_CLASS_HID_TYPE_GLOBAL
  Initial value:`0x1`
- #define UX_HOST_CLASS_HID_TYPE_LOCAL
  Initial value:`0x2`
- #define UX_HOST_CLASS_HID_TYPE_RESERVED
  Initial value:`0x3`
- #define UX_HOST_CLASS_HID_MAIN_TAG_INPUT
  Initial value:`0x8`
- #define UX_HOST_CLASS_HID_MAIN_TAG_OUTPUT
  Initial value:`0x9`
- #define UX_HOST_CLASS_HID_MAIN_TAG_FEATURE
  Initial value:`0xb`
- #define UX_HOST_CLASS_HID_MAIN_TAG_COLLECTION
  Initial value:`0xa`
- #define UX_HOST_CLASS_HID_MAIN_TAG_END_COLLECTION
  Initial value:`0xc`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_USAGE_PAGE
  Initial value:`0x0`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_LOGICAL_MINIMUM
  Initial value:`0x1`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_LOGICAL_MAXIMUM
  Initial value:`0x2`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_PHYSICAL_MINIMUM
  Initial value:`0x3`

- #define UX_HOST_CLASS_HID_GLOBAL_TAG_PHYSICAL_MAXIMUM
  Initial value:`0x4`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_UNIT_EXPONENT
  Initial value:`0x5`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_REPORT_SIZE
  Initial value:`0x7`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_REPORT_ID
  Initial value:`0x8`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_REPORT_COUNT
  Initial value:`0x9`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_PUSH
  Initial value:`0xa`
- #define UX_HOST_CLASS_HID_GLOBAL_TAG_POP
  Initial value:`0xb`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_USAGE
  Initial value:`0x0`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_USAGE_MINIMUM
  Initial value:`0x1`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_USAGE_MAXIMUM
  Initial value:`0x2`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_DESIGNATOR_INDEX
  Initial value:`0x3`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_DESIGNATOR_MINIMUM
  Initial value:`0x4`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_DESIGNATOR_MAXIMUM
  Initial value:`0x5`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_STRING_INDEX
  Initial value:`0x7`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_STRING_MINIMUM
  Initial value:`0x8`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_STRING_MAXIMUM
  Initial value:`0x9`
- #define UX_HOST_CLASS_HID_LOCAL_TAG_DELIMITER
  Initial value:`0xa`

- #define UX_HOST_CLASS_HID_COLLECTION_PHYSICAL
  Initial value:`0`
- #define UX_HOST_CLASS_HID_COLLECTION_APPLICATION
  Initial value:`1`
- #define UX_HOST_CLASS_HID_COLLECTION_LOGICAL
  Initial value:`2`
- #define UX_HOST_CLASS_HID_DELIMITER_OPEN
  Initial value:`1`
- #define UX_HOST_CLASS_HID_DELIMITER_CLOSE
  Initial value:`0`
- #define UX_HOST_CLASS_HID_ITEM_CONSTANT
  Initial value:`0x0001`
- #define UX_HOST_CLASS_HID_ITEM_VARIABLE
  Initial value:`0x0002`
- #define UX_HOST_CLASS_HID_ITEM_RELATIVE
  Initial value:`0x0004`
- #define UX_HOST_CLASS_HID_ITEM_WRAP
  Initial value:`0x0008`
- #define UX_HOST_CLASS_HID_ITEM_NON_LINEAR
  Initial value:`0x0010`
- #define UX_HOST_CLASS_HID_ITEM_NO_PREFERRED_STATE
  Initial value:`0x0020`
- #define UX_HOST_CLASS_HID_ITEM_NULL_STATE
  Initial value:`0x0040`
- #define UX_HOST_CLASS_HID_ITEM_VOLATILE
  Initial value:`0x0080`
- #define UX_HOST_CLASS_HID_ITEM_BUFFERED_BYTES
  Initial value:`0x0100`
- #define UX_HOST_CLASS_HID_GET_REPORT
  Initial value:`0x01`
- #define UX_HOST_CLASS_HID_GET_IDLE
  Initial value:`0x02`
- #define UX_HOST_CLASS_HID_GET_PROTOCOL
  Initial value:`0x03`

- #define UX_HOST_CLASS_HID_SET_REPORT
  Initial value:0x09
- #define UX_HOST_CLASS_HID_SET_IDLE
  Initial value:0x0A
- #define UX_HOST_CLASS_HID_SET_PROTOCOL
  Initial value:0x0B
- #define UX_HOST_CLASS_HID_REPORT_DESCRIPTOR
  Initial value:0x22
- #define UX_HOST_CLASS_HID_PHYSICAL_DESCRIPTOR
  Initial value:0x23
- #define UX_HID_DESCRIPTOR_ENTRIES
  Initial value:7
- #define UX_HID_DESCRIPTOR_LENGTH
  Initial value:9
- #define UX_HOST_CLASS_HID_PAGE_GENERIC_DESKTOP_CONTROLS
  Initial value:0x01
- #define UX_HOST_CLASS_HID_PAGE_SIMULATION_CONTROLS
  Initial value:0x02
- #define UX_HOST_CLASS_HID_PAGE_VR_CONTROLS
  Initial value:0x03
- #define UX_HOST_CLASS_HID_PAGE_SPORT_CONTROLS
  Initial value:0x04
- #define UX_HOST_CLASS_HID_PAGE_GAME_CONTROLS
  Initial value:0x05
- #define UX_HOST_CLASS_HID_PAGE_GENERIC_DEVICE_CONTROLS
  Initial value:0x06
- #define UX_HOST_CLASS_HID_PAGE_KEYBOARD_KEYPAD
  Initial value:0x07
- #define UX_HOST_CLASS_HID_PAGE_LEDS
  Initial value:0x08
- #define UX_HOST_CLASS_HID_PAGE_BUTTON
  Initial value:0x09
- #define UX_HOST_CLASS_HID_PAGE_ORDINAL
  Initial value:0x0A

- #define UX_HOST_CLASS_HID_PAGE_TELEPHONY
  Initial value:`0x0B`
- #define UX_HOST_CLASS_HID_PAGE_CONSUMER
  Initial value:`0x0C`
- #define UX_HOST_CLASS_HID_PAGE_DIGITIZER
  Initial value:`0x0D`
- #define UX_HOST_CLASS_HID_PAGE_PHYSICAL_INTERFACE_DEVICE
  Initial value:`0x0F`
- #define UX_HOST_CLASS_HID_PAGE_UNICODE
  Initial value:`0x10`
- #define UX_HOST_CLASS_HID_PAGE_ALPHANUMERIC_DISPLAY
  Initial value:`0x14`
- #define UX_HOST_CLASS_HID_PAGE_MEDICAL_INSTRUMENTS
  Initial value:`0x40`
- #define UX_HOST_CLASS_HID_PAGE_BAR_CODE_SCANNER
  Initial value:`0x8C`
- #define UX_HOST_CLASS_HID_PAGE_SCALE_PAGE
  Initial value:`0x8D`
- #define UX_HOST_CLASS_HID_PAGE_MAGNETIC_STRIPE_READING
  Initial value:`0x8E`
- #define UX_HOST_CLASS_HID_PAGE_CAMERA_CONTROL_PAGE
  Initial value:`0x90`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_UNDEFINED
  Initial value:`0x00`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_POINTER
  Initial value:`0x01`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_MOUSE
  Initial value:`0x02`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_RESERVED
  Initial value:`0x03`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_JOYSTICK
  Initial value:`0x04`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_GAME
  Initial value:`PAD 0x05`

- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_KEYBOARD
  Initial value:`0x06`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_KEYPAD
  Initial value:`0x07`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_MULTI_AXIS_CONTROLLER
  Initial value:`0x08`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_X
  Initial value:`0x30`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_Y
  Initial value:`0x31`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_Z
  Initial value:`0x32`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_RX
  Initial value:`0x33`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_RY
  Initial value:`0x34`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_RZ
  Initial value:`0x35`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SLIDER
  Initial value:`0x36`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_DIAL
  Initial value:`0x37`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_WHEEL
  Initial value:`0x38`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_HAT_SWITCH
  Initial value:`0x39`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_COUNTED_BUFFER
  Initial value:`0x3A`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_BYTE_COUNT
  Initial value:`0x3B`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_MOTION_WAKEUP
  Initial value:`0x3C`
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_START
  Initial value:`0x3D`

- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SELECT
  Initial value:0x3E
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VX
  Initial value:0x40
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VY
  Initial value:0x41
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VZ
  Initial value:0x42
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VBRX
  Initial value:0x43
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VBRY
  Initial value:0x44
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VBRZ
  Initial value:0x45
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_VNO
  Initial value:0x46
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_FEATURE_NOTIFICATION
  Initial value:0x47
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_CONTROL
  Initial value:0x80
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_POWER_DOWN
  Initial value:0x81
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_SLEEP
  Initial value:0x82
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_WAKE_UP
  Initial value:0x83
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_CONTEXT_MENU
  Initial value:0x84
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MAIN_MENU
  Initial value:0x85
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_APP_MENU
  Initial value:0x86
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_HELP
  Initial value:0x87

- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_EXIT
  Initial value:0x88
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_SELECT
  Initial value:0x89
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_RIGHT
  Initial value:0x8A
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_LEFT
  Initial value:0x8B
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_UP
  Initial value:0x8C
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_MENU_DOWN
  Initial value:0x8D
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_COLD_RESTART
  Initial value:0x8E
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_WARM_RESTART
  Initial value:0x8F
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_D_PAD_UP
  Initial value:0x90
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_D_PAD_DOWN
  Initial value:0x91
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_D_PAD_RIGHT
  Initial value:0x92
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_D_PAD_LEFT
  Initial value:0x93
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DOCK
  Initial value:0xA0
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_UNDOCK
  Initial value:0xA1
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_SETUP
  Initial value:0xA2
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_BREAK
  Initial value:0xA3
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DEBUGGER_BREAK
  Initial value:0xA4

- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_APPLICAION_BREAK
  Initial value:0xA5
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_APPLICATION_DEBUGGER_BREAK
  Initial value:0xA6
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_SPEAKER_MUTE
  Initial value:0xA7
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_HIBERNATE
  Initial value:0xA8
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_INVERT
  Initial value:0xB0
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_INTERNAL
  Initial value:0xB1
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_EXTERNAL
  Initial value:0xB2
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_BOTH
  Initial value:0xB3
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_DUAL
  Initial value:0xB4
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_TOGGLE
  Initial value:0xB5
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_SWAP
  Initial value:0xB6
- #define UX_HOST_CLASS_HID_GENERIC_DESKTOP_SYSTEM_DISPLAY_LCD_AUTOSCALE
  Initial value:0xB7
- #define UX_HOST_CLASS_HID_GAME_CONTROL_UNDEFINED
  Initial value:0x00
- #define UX_HOST_CLASS_HID_GAME_CONTROL_3D_GAME_CONTROLLER
  Initial value:0x01
- #define UX_HOST_CLASS_HID_GAME_CONTROL_PINBALL_DEVICE
  Initial value:0x02
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_DEVICE
  Initial value:0x03
- #define UX_HOST_CLASS_HID_GAME_CONTROL_POINT_OF_VIEW
  Initial value:0x20

- #define UX_HOST_CLASS_HID_GAME_CONTROL_TURN_RIGHT_LEFT
  Initial value:`0x21`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_PITCH_FORWARD_BACKWARD
  Initial value:`0x22`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_ROLL_RIGHT_LEFT
  Initial value:`0x23`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_MOVE_RIGHT_LEFT
  Initial value:`0x24`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_MOVE_FORWARD_BACKWARD
  Initial value:`0x25`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_MOVE_UP_DOWN
  Initial value:`0x26`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_LEAN_RIGHT_LEFT
  Initial value:`0x27`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_LEAN_FORWARD_BACKWARD
  Initial value:`0x28`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_HEIGHT_OF_POV
  Initial value:`0x29`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_FLIPPER
  Initial value:`0x2A`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_SECONDARY_FLIPPER
  Initial value:`0x2B`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_BUMP
  Initial value:`0x2C`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_NEW_GAME
  Initial value:`0x2D`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_SHOOT_BALL
  Initial value:`0x2E`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_PLAYER
  Initial value:`0x2F`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_BOLT
  Initial value:`0x30`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_CLIP
  Initial value:`0x31`

- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_SELECTOR
  Initial value:`0x32`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_SINGLE_SHOT
  Initial value:`0x33`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_BURST
  Initial value:`0x34`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_AUTOMATIC
  Initial value:`0x35`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GUN_SAFETY
  Initial value:`0x36`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GAMEAD_FIRE_JUMP
  Initial value:`0x37`
- #define UX_HOST_CLASS_HID_GAME_CONTROL_GAMEPAD_TRIGGER
  Initial value:`0x39`
- #define UX_HOST_CLASS_HID_LED_UNDEFINED
  Initial value:`0x00`
- #define UX_HOST_CLASS_HID_LED_NUM_LOCK
  Initial value:`0x01`
- #define UX_HOST_CLASS_HID_LED_CAPS_LOCK
  Initial value:`0x02`
- #define UX_HOST_CLASS_HID_LED_SCROLL_LOCK
  Initial value:`0x03`
- #define UX_HOST_CLASS_HID_LED_COMPOSE
  Initial value:`0x04`
- #define UX_HOST_CLASS_HID_LED_KANA
  Initial value:`0x05`
- #define UX_HOST_CLASS_HID_LED_POWER
  Initial value:`0x06`
- #define UX_HOST_CLASS_HID_LED_SHIFT
  Initial value:`0x07`
- #define UX_HOST_CLASS_HID_LED_DO_NOT_DISTURB
  Initial value:`0x08`
- #define UX_HOST_CLASS_HID_LED_MUTE
  Initial value:`0x09`

- #define UX_HOST_CLASS_HID_LED_TONE_ENABLE
  Initial value:0x0A
- #define UX_HOST_CLASS_HID_LED_HIGH_CUT_FILTER
  Initial value:0x0B
- #define UX_HOST_CLASS_HID_LED_LOW_CUT_FILTER
  Initial value:0x0C
- #define UX_HOST_CLASS_HID_LED_EQUALIZER_ENABLE
  Initial value:0x0D
- #define UX_HOST_CLASS_HID_LED_SOUND_FIELD_ON
  Initial value:0x0E
- #define UX_HOST_CLASS_HID_LED_SURROUND_ON
  Initial value:0x0F
- #define UX_HOST_CLASS_HID_LED_REPEAT
  Initial value:0x10
- #define UX_HOST_CLASS_HID_LED_STEREO
  Initial value:0x11
- #define UX_HOST_CLASS_HID_LED_SAMPLING_RATE_DETECT
  Initial value:0x12
- #define UX_HOST_CLASS_HID_LED_SPINNING
  Initial value:0x13
- #define UX_HOST_CLASS_HID_LED_CAV
  Initial value:0x14
- #define UX_HOST_CLASS_HID_LED_CLV
  Initial value:0x15
- #define UX_HOST_CLASS_HID_LED_RECORDING_FORMAT_DETECT
  Initial value:0x16
- #define UX_HOST_CLASS_HID_LED_OFF_HOOK
  Initial value:0x17
- #define UX_HOST_CLASS_HID_LED_RING
  Initial value:0x18
- #define UX_HOST_CLASS_HID_LED_MESSAGE_WAITING
  Initial value:0x19
- #define UX_HOST_CLASS_HID_LED_DATA_MODE
  Initial value:0x1A

- #define UX_HOST_CLASS_HID_LED_BATTERY_OPERATION
  Initial value:`0x1B`
- #define UX_HOST_CLASS_HID_LED_BATTERY_OK
  Initial value:`0x1C`
- #define UX_HOST_CLASS_HID_LED_BATTERY_LOW
  Initial value:`0x1D`
- #define UX_HOST_CLASS_HID_LED_SPEAKER
  Initial value:`0x1E`
- #define UX_HOST_CLASS_HID_LED_HEAD_SET
  Initial value:`0x1F`
- #define UX_HOST_CLASS_HID_LED_HOLD
  Initial value:`0x20`
- #define UX_HOST_CLASS_HID_LED_MICROPHONE
  Initial value:`0x21`
- #define UX_HOST_CLASS_HID_LED_COVERAGE
  Initial value:`0x22`
- #define UX_HOST_CLASS_HID_LED_NIGHT_MODE
  Initial value:`0x23`
- #define UX_HOST_CLASS_HID_LED_SEND_CALLS
  Initial value:`0x24`
- #define UX_HOST_CLASS_HID_LED_CALL_PICKUP
  Initial value:`0x25`
- #define UX_HOST_CLASS_HID_LED_CONFERENCE
  Initial value:`0x26`
- #define UX_HOST_CLASS_HID_LED_STAND_BY
  Initial value:`0x27`
- #define UX_HOST_CLASS_HID_LED_CAMERA_ON
  Initial value:`0x28`
- #define UX_HOST_CLASS_HID_LED_CAMERA_OFF
  Initial value:`0x29`
- #define UX_HOST_CLASS_HID_LED_ON_LINE
  Initial value:`0x2A`
- #define UX_HOST_CLASS_HID_LED_OFF_LINE
  Initial value:`0x2B`

- #define UX_HOST_CLASS_HID_LED_BUSY
  Initial value:0x2C
- #define UX_HOST_CLASS_HID_LED_READY
  Initial value:0x2D
- #define UX_HOST_CLASS_HID_LED_PAPER_OUT
  Initial value:0x2E
- #define UX_HOST_CLASS_HID_LED_PAPER_JAM
  Initial value:0x2F
- #define UX_HOST_CLASS_HID_LED_REMOTE
  Initial value:0x30
- #define UX_HOST_CLASS_HID_LED_FORWARD
  Initial value:0x31
- #define UX_HOST_CLASS_HID_LED_REVERSE
  Initial value:0x32
- #define UX_HOST_CLASS_HID_LED_STOP
  Initial value:0x33
- #define UX_HOST_CLASS_HID_LED_REWIND
  Initial value:0x34
- #define UX_HOST_CLASS_HID_LED_FAST_FORWARD
  Initial value:0x35
- #define UX_HOST_CLASS_HID_LED_PLAY
  Initial value:0x36
- #define UX_HOST_CLASS_HID_LED_PAUSE
  Initial value:0x37
- #define UX_HOST_CLASS_HID_LED_ERROR
  Initial value:0x39
- #define UX_HOST_CLASS_HID_LED_USAGE_SELECTED_INDICATOR
  Initial value:0x3A
- #define UX_HOST_CLASS_HID_LED_USAGE_IN_USE_INDICATOR
  Initial value:0x3B
- #define UX_HOST_CLASS_HID_LED_INDICATOR_ON
  Initial value:0x3D
- #define UX_HOST_CLASS_HID_LED_INDICATOR_FLASH
  Initial value:0x3E

- #define UX_HOST_CLASS_HID_LED_INDICATOR_SLOW_BLINK
  Initial value:`0x3F`
- #define UX_HOST_CLASS_HID_LED_INDICATOR_FAST_BLINK
  Initial value:`0x40`
- #define UX_HOST_CLASS_HID_LED_INDICATOR_OFF
  Initial value:`0x41`
- #define UX_HOST_CLASS_HID_LED_FLASH_ON_TIME
  Initial value:`0x42`
- #define UX_HOST_CLASS_HID_LED_SLOW_BLINK_ON_TIME
  Initial value:`0x43`
- #define UX_HOST_CLASS_HID_LED_SLOW_BLINK_OFF_TIME
  Initial value:`0x44`
- #define UX_HOST_CLASS_HID_LED_FAST_BLINK_ON_TIME
  Initial value:`0x45`
- #define UX_HOST_CLASS_HID_LED_FAST_BLINK_OFF_TIME
  Initial value:`0x46`
- #define UX_HOST_CLASS_HID_LED_USAGE_INDICATOR_COLOR
  Initial value:`0x47`
- #define UX_HOST_CLASS_HID_LED_INDICATOR_RED
  Initial value:`0x48`
- #define UX_HOST_CLASS_HID_LED_INDICATOR_GREEN
  Initial value:`0x49`
- #define UX_HOST_CLASS_HID_LED_INDICATOR_AMBER
  Initial value:`0x4A`
- #define UX_HOST_CLASS_HID_LED_GENERIC_INDICATOR
  Initial value:`0x4B`
- #define UX_HOST_CLASS_HID_LED_SYSTEM_SUSPEND
  Initial value:`0x4C`
- #define UX_HOST_CLASS_HID_LED_EXTERNAL_POWER_CONNECTED
  Initial value:`0x4D`
- #define UX_HOST_CLASS_HID_CONSUMER_UNASSIGNED
  Initial value:`0x00`
- #define UX_HOST_CLASS_HID_CONSUMER_REMOTE_CONTROL
  Initial value:`0x01`

- #define UX_HOST_CLASS_HID_CONSUMER_NUMERIC_KEY_PAD
  Initial value:`0x02`
- #define UX_HOST_CLASS_HID_CONSUMER_PROGRAMMABLE_BUTTONS
  Initial value:`0x03`
- #define UX_HOST_CLASS_HID_CONSUMER_MICROPHONE
  Initial value:`0x04`
- #define UX_HOST_CLASS_HID_CONSUMER_HEADPHONE
  Initial value:`0x05`
- #define UX_HOST_CLASS_HID_CONSUMER_GRAPHIC_EQUALIZER
  Initial value:`0x06`
- #define UX_HOST_CLASS_HID_CONSUMER_PLUS_10
  Initial value:`0x20`
- #define UX_HOST_CLASS_HID_CONSUMER_PLUS_100
  Initial value:`0x21`
- #define UX_HOST_CLASS_HID_CONSUMER_AM_PM
  Initial value:`0x22`
- #define UX_HOST_CLASS_HID_CONSUMER_POWER
  Initial value:`0x30`
- #define UX_HOST_CLASS_HID_CONSUMER_RESET
  Initial value:`0x31`
- #define UX_HOST_CLASS_HID_CONSUMER_SLEEP
  Initial value:`0x32`
- #define UX_HOST_CLASS_HID_CONSUMER_SLEEP_AFTER
  Initial value:`0x33`
- #define UX_HOST_CLASS_HID_CONSUMER_SLEEP_MODE_RTC
  Initial value:`0x34`
- #define UX_HOST_CLASS_HID_CONSUMER_ILLUMINATION
  Initial value:`0x35`
- #define UX_HOST_CLASS_HID_CONSUMER_FUNCTION_BUTTONS
  Initial value:`0x36`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU
  Initial value:`0x40`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_PICK
  Initial value:`0x41`

- #define UX_HOST_CLASS_HID_CONSUMER_MENU_UP
  Initial value:`0x42`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_DOWN
  Initial value:`0x43`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_LEFT
  Initial value:`0x44`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_RIGHT
  Initial value:`0x45`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_ESCAPE
  Initial value:`0x46`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_VALUE_INCREASE
  Initial value:`0x47`
- #define UX_HOST_CLASS_HID_CONSUMER_MENU_VALUE_DECREASE
  Initial value:`0x48`
- #define UX_HOST_CLASS_HID_CONSUMER_DATA_ON_SCREEN
  Initial value:`0x60`
- #define UX_HOST_CLASS_HID_CONSUMER_CLOSED_CAPTION
  Initial value:`0x61`
- #define UX_HOST_CLASS_HID_CONSUMER_CLOSED_CAPTION_SELECT
  Initial value:`0x62`
- #define UX_HOST_CLASS_HID_CONSUMER_VCR_TV
  Initial value:`0x63`
- #define UX_HOST_CLASS_HID_CONSUMER_BROADCAST_MODE
  Initial value:`0x64`
- #define UX_HOST_CLASS_HID_CONSUMER_SNAPSHOT
  Initial value:`0x65`
- #define UX_HOST_CLASS_HID_CONSUMER_STILL
  Initial value:`0x66`
- #define UX_HOST_CLASS_HID_CONSUMER_SELECTION
  Initial value:`0x80`
- #define UX_HOST_CLASS_HID_CONSUMER_ASSIGN_SELECTION
  Initial value:`0x81`
- #define UX_HOST_CLASS_HID_CONSUMER_MODE_STEP
  Initial value:`0x82`

- #define UX_HOST_CLASS_HID_CONSUMER_RECALL_LAST
  Initial value:`0x83`
- #define UX_HOST_CLASS_HID_CONSUMER_ENTER_CHANNEL
  Initial value:`0x84`
- #define UX_HOST_CLASS_HID_CONSUMER_ORDER_MOVIE
  Initial value:`0x85`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_L
  Initial value:`0x86`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECTION
  Initial value:`0x87`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_COMPUTER
  Initial value:`0x88`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_TV
  Initial value:`0x89`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_WWW
  Initial value:`0x8A`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_DVD
  Initial value:`0x8B`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_TELEPHONE
  Initial value:`0x8C`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_PROGRAM_GUIDE
  Initial value:`0x8D`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_VIDEO_PHONE
  Initial value:`0x8E`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_GAMES
  Initial value:`0x8F`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_MESSAGES
  Initial value:`0x90`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_CD
  Initial value:`0x91`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_VCR
  Initial value:`0x92`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_TUNER
  Initial value:`0x93`

- #define UX_HOST_CLASS_HID_CONSUMER_QUIT
  Initial value:`0x94`
- #define UX_HOST_CLASS_HID_CONSUMER_HELP
  Initial value:`0x95`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_TAPE
  Initial value:`0x96`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_CABLE
  Initial value:`0x97`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_SATELLITE
  Initial value:`0x98`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_SECURITY
  Initial value:`0x99`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_HOME
  Initial value:`0x9A`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDIA_SELECT_CALL
  Initial value:`0x9B`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_INCREMENT
  Initial value:`0x9C`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_DECREMENT
  Initial value:`0x9D`
- #define UX_HOST_CLASS_HID_CONSUMER_VCR_PLUS
  Initial value:`0xA0`
- #define UX_HOST_CLASS_HID_CONSUMER_ONCE
  Initial value:`0xA1`
- #define UX_HOST_CLASS_HID_CONSUMER_DAILY
  Initial value:`0xA2`
- #define UX_HOST_CLASS_HID_CONSUMER_WEEKLY
  Initial value:`0xA3`
- #define UX_HOST_CLASS_HID_CONSUMER_MONTHLY
  Initial value:`0xA4`
- #define UX_HOST_CLASS_HID_CONSUMER_PLAY
  Initial value:`0xB0`
- #define UX_HOST_CLASS_HID_CONSUMER_PAUSE
  Initial value:`0xB1`

- #define UX_HOST_CLASS_HID_CONSUMER_RECORD
  Initial value:`0xB2`
- #define UX_HOST_CLASS_HID_CONSUMER_FAST_FORWARD
  Initial value:`0xB3`
- #define UX_HOST_CLASS_HID_CONSUMER_REWIND
  Initial value:`0xB4`
- #define UX_HOST_CLASS_HID_CONSUMER_SCAN_NEXT_TRACK
  Initial value:`0xB5`
- #define UX_HOST_CLASS_HID_CONSUMER_SCAN_PREVIOUS_TRACK
  Initial value:`0xB6`
- #define UX_HOST_CLASS_HID_CONSUMER_STOP
  Initial value:`0xB7`
- #define UX_HOST_CLASS_HID_CONSUMER_EJECT
  Initial value:`0xB8`
- #define UX_HOST_CLASS_HID_CONSUMER_RANDOM_PLAY
  Initial value:`0xB9`
- #define UX_HOST_CLASS_HID_CONSUMER_SELECT_DISC
  Initial value:`0xBA`
- #define UX_HOST_CLASS_HID_CONSUMER_ENTER_DISC
  Initial value:`0xBB`
- #define UX_HOST_CLASS_HID_CONSUMER_REPEAT
  Initial value:`0xBC`
- #define UX_HOST_CLASS_HID_CONSUMER_TRACKING
  Initial value:`0xBD`
- #define UX_HOST_CLASS_HID_CONSUMER_TRACK_NORMAL
  Initial value:`0xBE`
- #define UX_HOST_CLASS_HID_CONSUMER_SLOW_TRACKING
  Initial value:`0xBF`
- #define UX_HOST_CLASS_HID_CONSUMER_FRAME_FORWARD
  Initial value:`0xC0`
- #define UX_HOST_CLASS_HID_CONSUMER_FRAME_BACK
  Initial value:`0xC1`
- #define UX_HOST_CLASS_HID_CONSUMER_MARK
  Initial value:`0xC2`

- #define UX_HOST_CLASS_HID_CONSUMER_CLEAR_MARK
  Initial value:`0xC3`
- #define UX_HOST_CLASS_HID_CONSUMER_REPEAT_FROM_MARK
  Initial value:`0xC4`
- #define UX_HOST_CLASS_HID_CONSUMER_RETURN_TO_MARK
  Initial value:`0xC5`
- #define UX_HOST_CLASS_HID_CONSUMER_SEARCH_MARK_FORWARD
  Initial value:`0xC6`
- #define UX_HOST_CLASS_HID_CONSUMER_SEARCH_MARK_BACKWARDS
  Initial value:`0xC7`
- #define UX_HOST_CLASS_HID_CONSUMER_COUNTER_RESET
  Initial value:`0xC8`
- #define UX_HOST_CLASS_HID_CONSUMER_SHOW_COUNTER
  Initial value:`0xC9`
- #define UX_HOST_CLASS_HID_CONSUMER_TRACKING_INCREMENT
  Initial value:`0xCA`
- #define UX_HOST_CLASS_HID_CONSUMER_TRACKING_DECREMENT
  Initial value:`0xCB`
- #define UX_HOST_CLASS_HID_CONSUMER_STOP_EJECT
  Initial value:`0xCC`
- #define UX_HOST_CLASS_HID_CONSUMER_PLAY_PAUSE
  Initial value:`0xCD`
- #define UX_HOST_CLASS_HID_CONSUMER_PLAY_SKIP
  Initial value:`0xCE`
- #define UX_HOST_CLASS_HID_CONSUMER_VOLUME
  Initial value:`0xE0`
- #define UX_HOST_CLASS_HID_CONSUMER_BALANCE
  Initial value:`0xE1`
- #define UX_HOST_CLASS_HID_CONSUMER_MUTE
  Initial value:`0xE2`
- #define UX_HOST_CLASS_HID_CONSUMER_BASS
  Initial value:`0xE3`
- #define UX_HOST_CLASS_HID_CONSUMER_TREBLE
  Initial value:`0xE4`

- #define UX_HOST_CLASS_HID_CONSUMER_BASS_BOOST
  Initial value:`0xE5`
- #define UX_HOST_CLASS_HID_CONSUMER_SURROUND_MODE
  Initial value:`0xE6`
- #define UX_HOST_CLASS_HID_CONSUMER_LOUDNESS
  Initial value:`0xE7`
- #define UX_HOST_CLASS_HID_CONSUMER_MPX
  Initial value:`0xE8`
- #define UX_HOST_CLASS_HID_CONSUMER_VOLUME_INCREMENT
  Initial value:`0xE9`
- #define UX_HOST_CLASS_HID_CONSUMER_VOLUME_DECREMENT
  Initial value:`0xEA`
- #define UX_HOST_CLASS_HID_CONSUMER_SPEED_SELECT
  Initial value:`0xF0`
- #define UX_HOST_CLASS_HID_CONSUMER_PLAYBACK_SPEED
  Initial value:`0xF1`
- #define UX_HOST_CLASS_HID_CONSUMER_STANDARD_PLAY
  Initial value:`0xF2`
- #define UX_HOST_CLASS_HID_CONSUMER_LONG_PLAY
  Initial value:`0xF3`
- #define UX_HOST_CLASS_HID_CONSUMER_EXTENDED_PLAY
  Initial value:`0xF4`
- #define UX_HOST_CLASS_HID_CONSUMER_FAN_ENABLE
  Initial value:`0xF6`
- #define UX_HOST_CLASS_HID_CONSUMER_FAN_SPEED
  Initial value:`0x100`
- #define UX_HOST_CLASS_HID_CONSUMER_LIGHT_ENABLE
  Initial value:`0x101`
- #define UX_HOST_CLASS_HID_CONSUMER_LIGHT_ILLUMINATION_LEVEL
  Initial value:`0x102`
- #define UX_HOST_CLASS_HID_CONSUMER_CLIMATE_CONTROL_ENABLE
  Initial value:`0x103`
- #define UX_HOST_CLASS_HID_CONSUMER_ROOM_TEMPERATURE
  Initial value:`0x104`

- #define UX_HOST_CLASS_HID_CONSUMER_SECURITY_ENABLE
  Initial value:`0x105`
- #define UX_HOST_CLASS_HID_CONSUMER_FIRE_ALARM
  Initial value:`0x106`
- #define UX_HOST_CLASS_HID_CONSUMER_POLICE_ALARM
  Initial value:`0x107`
- #define UX_HOST_CLASS_HID_CONSUMER_PROXIMITY
  Initial value:`0x108`
- #define UX_HOST_CLASS_HID_CONSUMER_MOTION
  Initial value:`0x109`
- #define UX_HOST_CLASS_HID_CONSUMER_DURESS_ALARM
  Initial value:`0x10A`
- #define UX_HOST_CLASS_HID_CONSUMER_HOLDUP_ALARM
  Initial value:`0x10B`
- #define UX_HOST_CLASS_HID_CONSUMER_MEDICAL_ALARM
  Initial value:`0x10C`
- #define UX_HOST_CLASS_HID_CONSUMER_BALANCE_RIGHT
  Initial value:`0x10D`
- #define UX_HOST_CLASS_HID_CONSUMER_BALANCE_LEFT
  Initial value:`0x150`
- #define UX_HOST_CLASS_HID_CONSUMER_BASS_INCREMENT
  Initial value:`0x151`
- #define UX_HOST_CLASS_HID_CONSUMER_BASS_DECREMENT
  Initial value:`0x152`
- #define UX_HOST_CLASS_HID_CONSUMER_TREBLE_INCREMENT
  Initial value:`0x153`
- #define UX_HOST_CLASS_HID_CONSUMER_TREBLE_DECREMENT
  Initial value:`0x154`
- #define UX_HOST_CLASS_HID_CONSUMER_SPEAKER_SYSTEM
  Initial value:`0x155`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_LEFT
  Initial value:`0x160`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_RIGHT
  Initial value:`0x161`

- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_CENTER
  Initial value:`0x162`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_FRONT
  Initial value:`0x163`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_CENTER_FRONT
  Initial value:`0x164`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_SIDE
  Initial value:`0x165`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_SURROUND
  Initial value:`0x166`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_LOW_FREQUENCY
  Initial value:`0x167`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_TOP
  Initial value:`0x168`
- #define UX_HOST_CLASS_HID_CONSUMER_CHANNEL_UNKNOWN
  Initial value:`0x169`
- #define UX_HOST_CLASS_HID_CONSUMER_SUB_CHANNEL
  Initial value:`0x16A`
- #define UX_HOST_CLASS_HID_CONSUMER_SUB_CHANNEL_INCREMENT
  Initial value:`0x170`
- #define UX_HOST_CLASS_HID_CONSUMER_SUB_CHANNEL_DECREMENT
  Initial value:`0x171`
- #define UX_HOST_CLASS_HID_CONSUMER_ALTERNATE_AUDIO_INCREMENT
  Initial value:`0x172`
- #define UX_HOST_CLASS_HID_CONSUMER_ALTERNATE_AUDIO_DECREMENT
  Initial value:`0x173`
- #define UX_HOST_CLASS_HID_CONSUMER_APPLICATION_LAUNCH_BUTTONS
  Initial value:`0x174`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LAUNCH_BUTTON_CONFIGURATION
  Initial value:`0x180`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_PROGRAMMABLE_BUTTON
  Initial value:`0x181`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CONSUMER_CONTROL_CONFIGURATION
  Initial value:`0x182`

- #define UX_HOST_CLASS_HID_CONSUMER_AL_WORD_PROCESSOR
  Initial value:`0x183`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_TEXT_EDITOR
  Initial value:`0x184`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_SPREADSHEET
  Initial value:`0x185`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_GRAPHICS_EDITOR
  Initial value:`0x186`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_PRESENTATION_APP
  Initial value:`0x187`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_DATABASE_APP
  Initial value:`0x188`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_EMAIL_READER
  Initial value:`0x189`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_NEWSREADER
  Initial value:`0x18A`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_VOICEMAIL
  Initial value:`0x18B`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CONTACTS_ADDRESS_BOOK
  Initial value:`0x18C`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CALENDAR_SCHEDULE
  Initial value:`0x18D`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_TASK_PROJECT_MANAGER
  Initial value:`0x18E`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LOG_JOURNAL_TIMECARD
  Initial value:`0x18F`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CHECKBOOK_FINANCE
  Initial value:`0x190`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CALCULATOR
  Initial value:`0x191`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_A_V_CAPTURE_PLAYBACK
  Initial value:`0x192`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LOCAL_MACHINE_BROWSER
  Initial value:`0x193`

- #define UX_HOST_CLASS_HID_CONSUMER_AL_LAN_WAN_BROWSER
  Initial value:`0x194`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_INTERNET_BROWSER
  Initial value:`0x195`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_REMOTE_NETWORKING
  Initial value:`0x196`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_NETWORK_CONFERENCE
  Initial value:`0x197`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_NETWORK_CHAT
  Initial value:`0x198`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_TELEPHONY_DIALER
  Initial value:`0x199`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LOGON
  Initial value:`0x19A`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LOGOFF
  Initial value:`0x19B`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_LOGON_LOGOFF
  Initial value:`0x19C`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_SCREENSAVER
  Initial value:`0x19D`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CONTROL_PANEL
  Initial value:`0x19E`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_COMMAND_LINE_PROCESSOR
  Initial value:`0x19F`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_PROCESS_MANAGER
  Initial value:`0x1A0`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_SELECT_APPLICATION
  Initial value:`0x1A1`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_NEXT_APPLICATION
  Initial value:`0x1A2`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_PREVIOUS_APPLICATION
  Initial value:`0x1A3`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_PREEMPTIVE_HALT_APPLICATION
  Initial value:`0x1A4`

- #define UX_HOST_CLASS_HID_CONSUMER_AL_INTEGRATED_HELP_CENTER
  Initial value:`0x1A5`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_DOCUMENTS
  Initial value:`0x1A6`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_THESAURUS
  Initial value:`0x1A7`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_DICTIONARY
  Initial value:`0x1A8`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_DESKTOP
  Initial value:`0x1A9`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_SPELL_CHECK
  Initial value:`0x1AA`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_GRAMMAR_CHECK
  Initial value:`0x1AB`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_WIRELESS_STATUS
  Initial value:`0x1AC`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_KEYBOARD_LAYOUT
  Initial value:`0x1AD`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_VIRUS_PROTECTION
  Initial value:`0x1AE`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_ENCRYPTION
  Initial value:`0x1AF`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_SCREEN_SAVER
  Initial value:`0x1B0`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_ALARMS
  Initial value:`0x1B1`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_CLOCK
  Initial value:`0x1B2`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_FILE_BROWSER
  Initial value:`0x1B3`
- #define UX_HOST_CLASS_HID_CONSUMER_AL_POWER_STATUS
  Initial value:`0x1B4`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_NEW
  Initial value:`0x1B5`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_OPEN
  Initial value:`0x201`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_CLOSE
  Initial value:`0x202`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_EXIT
  Initial value:`0x203`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_MAXIMIZE
  Initial value:`0x204`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_MINIMIZE
  Initial value:`0x205`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SAVE
  Initial value:`0x206`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PRINT
  Initial value:`0x207`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PROPERTIES
  Initial value:`0x208`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_UNDO
  Initial value:`0x209`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_COPY
  Initial value:`0x21A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_CUT
  Initial value:`0x21B`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PASTE
  Initial value:`0x21C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_ALL
  Initial value:`0x21D`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FIND
  Initial value:`0x21E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FIND_AND_REPLACE
  Initial value:`0x21F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SEARCH
  Initial value:`0x220`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_GO_TO
  Initial value:`0x221`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_HOME
  Initial value:0x222
- #define UX_HOST_CLASS_HID_CONSUMER_AC_BACK
  Initial value:0x223
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FORWARD
  Initial value:0x224
- #define UX_HOST_CLASS_HID_CONSUMER_AC_STOP
  Initial value:0x225
- #define UX_HOST_CLASS_HID_CONSUMER_AC_REFRESH
  Initial value:0x226
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PREVIOUS_LINK
  Initial value:0x227
- #define UX_HOST_CLASS_HID_CONSUMER_AC_NEXT_LINK
  Initial value:0x228
- #define UX_HOST_CLASS_HID_CONSUMER_AC_BOOKMARKS
  Initial value:0x229
- #define UX_HOST_CLASS_HID_CONSUMER_AC_HISTORY
  Initial value:0x22A
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SUBSCRIPTIONS
  Initial value:0x22B
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ZOOM_IN
  Initial value:0x22C
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ZOOM_OUT
  Initial value:0x22D
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FULL_SCREEN_VIEW
  Initial value:0x22F
- #define UX_HOST_CLASS_HID_CONSUMER_AC_NORMAL_VIEW
  Initial value:0x230
- #define UX_HOST_CLASS_HID_CONSUMER_AC_VIEW_TOGGLE
  Initial value:0x231
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SCROLL_UP
  Initial value:0x232
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SCROLL_DOWN
  Initial value:0x233

- #define UX_HOST_CLASS_HID_CONSUMER_AC_PAN_LEFT
  Initial value:`0x235`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PAN_RIGHT
  Initial value:`0x236`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_NEW_WINDOW
  Initial value:`0x238`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_TILE_HORIZONTALLY
  Initial value:`0x239`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_TILE_VERTICALLY
  Initial value:`0x23A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FORMAT
  Initial value:`0x23B`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_EDIT
  Initial value:`0x23C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_BOLD
  Initial value:`0x23D`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ITALICS
  Initial value:`0x23E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_UNDERLINE
  Initial value:`0x23F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_STRIKETHROUGH
  Initial value:`0x240`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SUPERSCRIPT
  Initial value:`0x242`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ALL_CAPS
  Initial value:`0x243`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ROTATE
  Initial value:`0x244`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_RESIZE
  Initial value:`0x245`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FLIP_HORIZONTAL
  Initial value:`0x246`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FLIP_VERTICAL
  Initial value:`0x247`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_MIRROR_HORIZONTAL
  Initial value:`0x248`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_MIRROR_VERTICAL
  Initial value:`0x249`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FONT_SELECT
  Initial value:`0x24A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FONT_COLOR
  Initial value:`0x24B`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FONT_SIZE
  Initial value:`0x24C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_LEFT
  Initial value:`0x24D`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_CENTER_H
  Initial value:`0x24E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_RIGHT
  Initial value:`0x24F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_BLOCK_H
  Initial value:`0x250`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_TOP
  Initial value:`0x251`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_CENTER_V
  Initial value:`0x252`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_BOTTOM
  Initial value:`0x253`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_JUSTIFY_BLOCK_V
  Initial value:`0x254`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INDENT_DECREASE
  Initial value:`0x255`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INDENT_INCREASE
  Initial value:`0x256`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_NUMBERED_LIST
  Initial value:`0x257`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_RESTART_NUMBERING
  Initial value:`0x258`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_BULLETED_LIST
  Initial value:`0x259`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PROMOTE
  Initial value:`0x25A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_DEMOTE
  Initial value:`0x25B`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_YES
  Initial value:`0x25C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_CANCEL
  Initial value:`0x25E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_CATALOG
  Initial value:`0x25F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_BUY_CHECKOUT
  Initial value:`0x260`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ADD_TO_CART
  Initial value:`0x261`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_EXPAND
  Initial value:`0x262`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_EXPAND_ALL
  Initial value:`0x263`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_COLLAPSE
  Initial value:`0x264`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_COLLAPSE_ALL
  Initial value:`0x265`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PRINT_PREVIEW
  Initial value:`0x266`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PASTE_SPECIAL
  Initial value:`0x267`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_MODE
  Initial value:`0x268`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_DELETE
  Initial value:`0x269`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_LOCK
  Initial value:`0x26A`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_UNLOCK
  Initial value:`0x26B`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_PROTECT
  Initial value:`0x26C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_UNPROTECT
  Initial value:`0x26D`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ATTACH_COMMENT
  Initial value:`0x26E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_DELETE_COMMENT
  Initial value:`0x26F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_VIEW_COMMENT
  Initial value:`0x270`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_WORD
  Initial value:`0x271`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_SENTENCE
  Initial value:`0x272`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_PARAGRAPH
  Initial value:`0x273`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_COLUMN
  Initial value:`0x274`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_ROW
  Initial value:`0x275`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_TABLE
  Initial value:`0x276`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_OBJECT
  Initial value:`0x277`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_REDO_REPEAT
  Initial value:`0x278`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SORT
  Initial value:`0x279`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SORT_ASCENDING
  Initial value:`0x27A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SORT_DESCENDING
  Initial value:`0x27B`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_FILTER
  Initial value:`0x27C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SET_CLOCK
  Initial value:`0x27D`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_VIEW_CLOCK
  Initial value:`0x27E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SELECT_TIME_ZONE
  Initial value:`0x27F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_EDIT_TIME_ZONES
  Initial value:`0x280`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SET_ALARM
  Initial value:`0x281`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_CLEAR_ALARM
  Initial value:`0x282`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SNOOZE_ALARM
  Initial value:`0x283`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_RESET_ALARM
  Initial value:`0x284`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SYNCHRONIZE
  Initial value:`0x285`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SEND_RECEIVE
  Initial value:`0x286`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SEND_TO
  Initial value:`0x287`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_REPLY
  Initial value:`0x288`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_REPLY_ALL
  Initial value:`0x289`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_FORWARD_MSG
  Initial value:`0x28A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_ATTACH_FILE
  Initial value:`0x28C`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_UPLOAD
  Initial value:`0x28D`

- #define UX_HOST_CLASS_HID_CONSUMER_AC_DOWNLOAD
  Initial value:`0x28E`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SET_BORDERS
  Initial value:`0x28F`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_ROW
  Initial value:`0x290`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_COLUMN
  Initial value:`0x291`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_FILE
  Initial value:`0x292`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_PICTURE
  Initial value:`0x293`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_OBJECT
  Initial value:`0x294`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_INSERT_SYMBOL
  Initial value:`0x295`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SAVE_CLOSE
  Initial value:`0x296`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_RENAME
  Initial value:`0x297`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_MERGE
  Initial value:`0x298`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_SPLIT
  Initial value:`0x299`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_DISRIBUTE_HORIZONTALLY
  Initial value:`0x29A`
- #define UX_HOST_CLASS_HID_CONSUMER_AC_DISTRIBUTE_VERTICALLY
  Initial value:`0x29B`
- #define UX_HOST_CLASS_HID_REPORT_TYPE_INPUT
  Initial value:`0x1`
- #define UX_HOST_CLASS_HID_REPORT_TYPE_OUTPUT
  Initial value:`0x2`
- #define UX_HOST_CLASS_HID_REPORT_TYPE_FEATURE
  Initial value:`0x3`

- #define UX_HOST_CLASS_HID_USAGES
  Initial value:`1024`
- #define UX_HOST_CLASS_HID_MAX_GLOBAL
  Initial value:`4`
- #define UX_HOST_CLASS_HID_KEYBOARD_BUFFER_LENGTH
  Initial value:`128`
- #define UX_HOST_CLASS_HID_KEYBOARD_USAGE_ARRAY_LENGTH
  Initial value:`64`
- #define UX_HID_LED_KEY_CAPS_LOCK
  Initial value:`0x39`
- #define UX_HID_LED_KEY_NUM_LOCK
  Initial value:`0x53`
- #define UX_HID_LED_KEY_SCROLL_LOCK
  Initial value:`0x47`
- #define UX_HID_MODIFIER_KEY_LEFT_CONTROL
  Initial value:`0xe0`
- #define UX_HID_MODIFIER_KEY_LEFT_SHIFT
  Initial value:`0xe1`
- #define UX_HID_MODIFIER_KEY_LEFT_ALT
  Initial value:`0xe2`
- #define UX_HID_MODIFIER_KEY_LEFT_GUI
  Initial value:`0xe3`
- #define UX_HID_MODIFIER_KEY_RIGHT_CONTROL
  Initial value:`0xe4`
- #define UX_HID_MODIFIER_KEY_RIGHT_SHIFT
  Initial value:`0xe5`
- #define UX_HID_MODIFIER_KEY_RIGHT_ALT
  Initial value:`0xe6`
- #define UX_HID_MODIFIER_KEY_RIGHT_GUI
  Initial value:`0xe7`
- #define UX_HID_KEYBOARD_STATE_NUM_LOCK
  Initial value:`0x0001`
- #define UX_HID_KEYBOARD_STATE_CAPS_LOCK
  Initial value:`0x0002`

- #define UX_HID_KEYBOARD_STATE_SCROLL_LOCK
  Initial value:`0x0004`
- #define UX_HID_KEYBOARD_STATE_MASK_LOCK
  Initial value:`0x0007`
- #define UX_HID_KEYBOARD_STATE_LEFT_SHIFT
  Initial value:`0x0100`
- #define UX_HID_KEYBOARD_STATE_RIGHT_SHIFT
  Initial value:`0x0200`
- #define UX_HID_KEYBOARD_STATE_SHIFT
  Initial value:`0x0300`
- #define UX_HID_KEYBOARD_STATE_LEFT_ALT
  Initial value:`0x0400`
- #define UX_HID_KEYBOARD_STATE_RIGHT_ALT
  Initial value:`0x0800`
- #define UX_HID_KEYBOARD_STATE_ALT
  Initial value:`0x0a00`
- #define UX_HID_KEYBOARD_STATE_LEFT_CTRL
  Initial value:`0x1000`
- #define UX_HID_KEYBOARD_STATE_RIGHT_CTRL
  Initial value:`0x2000`
- #define UX_HID_KEYBOARD_STATE_CTRL
  Initial value:`0x3000`
- #define UX_HID_KEYBOARD_STATE_LEFT_GUI
  Initial value:`0x4000`
- #define UX_HID_KEYBOARD_STATE_RIGHT_GUI
  Initial value:`0x8000`
- #define UX_HID_KEYBOARD_STATE_GUI
  Initial value:`0xa000`
- #define UX_HID_KEYBOARD_NO_KEY
  Initial value:`0`
- #define UX_HID_KEYBOARD_KEYS_KEYPAD_LOWER_RANGE
  Initial value:`0x54`
- #define UX_HID_KEYBOARD_KEYS_KEYPAD_UPPER_RANGE
  Initial value:`0x67`

- #define UX_HID_KEYBOARD_KEYS_UPPER_RANGE
  Initial value:116
- #define UX_HOST_CLASS_HID_MOUSE_BUFFER_LENGTH
  Initial value:128
- #define UX_HOST_CLASS_HID_MOUSE_USAGE_ARRAY_LENGTH
  Initial value:64
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_1
  Initial value:0x00090001
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_2
  Initial value:0x00090002
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_3
  Initial value:0x00090003
- #define UX_HOST_CLASS_HID_MOUSE_AXIS_X
  Initial value:0x00010030
- #define UX_HOST_CLASS_HID_MOUSE_AXIS_Y
  Initial value:0x00010031
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_1_PRESSED
  Initial value:0x01
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_2_PRESSED
  Initial value:0x02
- #define UX_HOST_CLASS_HID_MOUSE_BUTTON_3_PRESSED
  Initial value:0x04
- #define UX_HOST_CLASS_HID_REMOTE_CONTROL_BUFFER_LENGTH
  Initial value:128
- #define UX_HOST_CLASS_HID_REMOTE_CONTROL_USAGE_ARRAY_LENGTH
  Initial value:64
- #define UX_HOST_CLASS_HUB_CLASS
  Initial value:9
- #define UX_HOST_CLASS_HUB_PROTOCOL_FS
  Initial value:0
- #define UX_HOST_CLASS_HUB_PROTOCOL_SINGLE_TT
  Initial value:1
- #define UX_HOST_CLASS_HUB_PROTOCOL_MULTIPLE_TT
  Initial value:2

- #define UX_HOST_CLASS_HUB_GANG_POWER_SWITCHING
  Initial value:`0x00`
- #define UX_HOST_CLASS_HUB_INDIVIDUAL_POWER_SWITCHING
  Initial value:`0x01`
- #define UX_HOST_CLASS_HUB_NO_POWER_SWITCHING
  Initial value:`0x02`
- #define UX_HOST_CLASS_HUB_COMPOUND_DEVICE
  Initial value:`0x04`
- #define UX_HOST_CLASS_HUB_GLOBAL_OVERCURRENT
  Initial value:`0x00`
- #define UX_HOST_CLASS_HUB_INDIVIDUAL_OVERCURRENT
  Initial value:`0x08`
- #define UX_HOST_CLASS_HUB_NO_OVERCURRENT
  Initial value:`0x10`
- #define UX_HOST_CLASS_HUB_GET_STATUS
  Initial value:`0x00`
- #define UX_HOST_CLASS_HUB_CLEAR_FEATURE
  Initial value:`0x01`
- #define UX_HOST_CLASS_HUB_GET_STATE
  Initial value:`0x02`
- #define UX_HOST_CLASS_HUB_SET_FEATURE
  Initial value:`0x03`
- #define UX_HOST_CLASS_HUB_GET_DESCRIPTOR
  Initial value:`0x06`
- #define UX_HOST_CLASS_HUB_SET_DESCRIPTOR
  Initial value:`0x07`
- #define UX_HOST_CLASS_HUB_PORT_CONNECTION
  Initial value:`0x00`
- #define UX_HOST_CLASS_HUB_PORT_ENABLE
  Initial value:`0x01`
- #define UX_HOST_CLASS_HUB_PORT_SUSPEND
  Initial value:`0x02`
- #define UX_HOST_CLASS_HUB_PORT_OVER_CURRENT
  Initial value:`0x03`

- #define UX_HOST_CLASS_HUB_PORT_RESET
  Initial value:`0x04`
- #define UX_HOST_CLASS_HUB_PORT_POWER
  Initial value:`0x08`
- #define UX_HOST_CLASS_HUB_PORT_LOW_SPEED
  Initial value:`0x09`
- #define UX_HOST_CLASS_HUB_C_PORT_CONNECTION
  Initial value:`0x10`
- #define UX_HOST_CLASS_HUB_C_PORT_ENABLE
  Initial value:`0x11`
- #define UX_HOST_CLASS_HUB_C_PORT_SUSPEND
  Initial value:`0x12`
- #define UX_HOST_CLASS_HUB_C_PORT_OVER_CURRENT
  Initial value:`0x13`
- #define UX_HOST_CLASS_HUB_C_PORT_RESET
  Initial value:`0x14`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_CONNECTION
  Initial value:`0x0001`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_ENABLE
  Initial value:`0x0002`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_SUSPEND
  Initial value:`0x0004`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_OVER_CURRENT
  Initial value:`0x0008`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_RESET
  Initial value:`0x0010`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_POWER
  Initial value:`0x0100`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_LOW_SPEED
  Initial value:`0x0200`
- #define UX_HOST_CLASS_HUB_PORT_STATUS_HIGH_SPEED
  Initial value:`0x0400`
- #define UX_HOST_CLASS_HUB_PORT_CHANGE_CONNECTION
  Initial value:`0x00001`

- #define UX_HOST_CLASS_HUB_PORT_CHANGE_ENABLE
  Initial value:0x00002
- #define UX_HOST_CLASS_HUB_PORT_CHANGE_SUSPEND
  Initial value:0x00004
- #define UX_HOST_CLASS_HUB_PORT_CHANGE_OVER_CURRENT
  Initial value:0x00008
- #define UX_HOST_CLASS_HUB_PORT_CHANGE_RESET
  Initial value:0x00010
- #define UX_HOST_CLASS_HUB_ENABLE_RETRY_COUNT
  Initial value:3
- #define UX_HOST_CLASS_HUB_ENABLE_RETRY_DELAY
  Initial value:100
- #define UX_HOST_CLASS_HUB_ENUMERATION_RETRY
  Initial value:3
- #define UX_HOST_CLASS_HUB_ENUMERATION_RETRY_DELAY
  Initial value:300
- #define UX_HOST_CLASS_HUB_NOT_POWERED
  Initial value:8
- #define UX_HUB_DESCRIPTOR_ENTRIES
  Initial value:8
- #define UX_HUB_DESCRIPTOR_LENGTH
  Initial value:9
- #define UX_MAX_HUB_PORTS
  Initial value:15
- #define UX_MAX_HOST_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_MAX_MEDIA
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_MEMORY_BUFFER_SIZE
  Initial value:(1024)
- #define UX_HOST_CLASS_STORAGE_MAX_TRANSFER_SIZE
  Initial value:(1024)
- #define UX_HOST_CLASS_STORAGE_THREAD_STACK_SIZE
  Initial value:UX_THREAD_STACK_SIZE

- #define UX_HOST_CLASS_STORAGE_DEVICE_INIT_DELAY
  Initial value:`(2 * UX_PERIODIC_RATE)`
- #define UX_HOST_CLASS_STORAGE_THREAD_SLEEP_TIME
  Initial value:`(2 * UX_PERIODIC_RATE)`
- #define UX_HOST_CLASS_STORAGE_INSTANCE_SHUTDOWN_TIMER
  Initial value:`(10)`
- #define UX_HOST_CLASS_STORAGE_THREAD_PRIORITY_CLASS
  Initial value:`20`
- #define UX_HOST_CLASS_STORAGE_TRANSFER_TIMEOUT
  Initial value:`10000`
- #define UX_HOST_CLASS_STORAGE_CBI_STATUS_TIMEOUT
  Initial value:`3000`
- #define UX_HOST_CLASS_STORAGE_CLASS
  Initial value:`8`
- #define UX_HOST_CLASS_STORAGE_SUBCLASS_RBC
  Initial value:`1`
- #define UX_HOST_CLASS_STORAGE_SUBCLASS_SFF8020
  Initial value:`2`
- #define UX_HOST_CLASS_STORAGE_SUBCLASS_UFI
  Initial value:`4`
- #define UX_HOST_CLASS_STORAGE_SUBCLASS_SFF8070
  Initial value:`5`
- #define UX_HOST_CLASS_STORAGE_SUBCLASS_SCSI
  Initial value:`6`
- #define UX_HOST_CLASS_STORAGE_CBW_SIZE
  Initial value:`64`
- #define UX_HOST_CLASS_STORAGE_PROTOCOL_CBI
  Initial value:`0`
- #define UX_HOST_CLASS_STORAGE_PROTOCOL_BO
  Initial value:`0x50`
- #define UX_HOST_CLASS_STORAGE_DATA_OUT
  Initial value:`0`
- #define UX_HOST_CLASS_STORAGE_DATA_IN
  Initial value:`0x80`

- #define UX_HOST_CLASS_STORAGE_CSW_PASSED
  Initial value:`0`
- #define UX_HOST_CLASS_STORAGE_CSW_FAILED
  Initial value:`1`
- #define UX_HOST_CLASS_STORAGE_CSW_PHASE_ERROR
  Initial value:`2`
- #define UX_HOST_CLASS_STORAGE_CBW_SIGNATURE_MASK
  Initial value:`0x43425355`
- #define UX_HOST_CLASS_STORAGE_CBW_TAG_MASK
  Initial value:`0x55534243`
- #define UX_HOST_CLASS_STORAGE_MEDIA_NAME
  Initial value:`"usb disk"`
- #define UX_HOST_CLASS_STORAGE_MEDIA_REMOVABLE
  Initial value:`0x80`
- #define UX_HOST_CLASS_STORAGE_MEDIA_UNKNOWN
  Initial value:`0`
- #define UX_HOST_CLASS_STORAGE_MEDIA_KNOWN
  Initial value:`1`
- #define UX_HOST_CLASS_STORAGE_MEDIA_FAT_DISK
  Initial value:`0`
- #define UX_HOST_CLASS_STORAGE_MEDIA_CDROM
  Initial value:`5`
- #define UX_HOST_CLASS_STORAGE_MEDIA_OPTICAL_DISK
  Initial value:`7`
- #define UX_HOST_CLASS_STORAGE_MEDIA_IOMEGA_CLICK
  Initial value:`0x55`
- #define UX_HOST_CLASS_STORAGE_RESET
  Initial value:`0xff`
- #define UX_HOST_CLASS_STORAGE_GET_MAX_LUN
  Initial value:`0xfe`
- #define UX_HOST_CLASS_STORAGE_TRANSPORT_ERROR
  Initial value:`1`
- #define UX_HOST_CLASS_STORAGE_COMMAND_ERROR
  Initial value:`2`

- #define UX_HOST_CLASS_STORAGE_SENSE_ERROR
  Initial value:3
- #define UX_HOST_CLASS_STORAGE_SECTOR_SIZE_FAT
  Initial value:512
- #define UX_HOST_CLASS_STORAGE_SECTOR_SIZE_OTHER
  Initial value:2048
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RETRY
  Initial value:10
- #define UX_HOST_CLASS_STORAGE_START_MEDIA
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_STOP_MEDIA
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_MEDIA_UNMOUNTED
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_MEDIA_MOUNTED
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_SCSI_TEST_READY
  Initial value:0x00
- #define UX_HOST_CLASS_STORAGE_SCSI_REQUEST_SENSE
  Initial value:0x03
- #define UX_HOST_CLASS_STORAGE_SCSI_FORMAT
  Initial value:0x04
- #define UX_HOST_CLASS_STORAGE_SCSI_INQUIRY
  Initial value:0x12
- #define UX_HOST_CLASS_STORAGE_SCSI_MODE_SENSE_SHORT
  Initial value:0x1a
- #define UX_HOST_CLASS_STORAGE_SCSI_START_STOP
  Initial value:0x1b
- #define UX_HOST_CLASS_STORAGE_SCSI_READ_FORMAT_CAPACITY
  Initial value:0x23
- #define UX_HOST_CLASS_STORAGE_SCSI_READ_CAPACITY
  Initial value:0x25
- #define UX_HOST_CLASS_STORAGE_SCSI_READ16
  Initial value:0x28

- #define UX_HOST_CLASS_STORAGE_SCSI_WRITE16
  Initial value:0x2a
- #define UX_HOST_CLASS_STORAGE_SCSI_VERIFY
  Initial value:0x2f
- #define UX_HOST_CLASS_STORAGE_SCSI_MODE_SELECT
  Initial value:0x55
- #define UX_HOST_CLASS_STORAGE_SCSI_READ32
  Initial value:0xa8
- #define UX_HOST_CLASS_STORAGE_SCSI_WRITE32
  Initial value:0xaa
- #define UX_HOST_CLASS_STORAGE_CBW_DATA_LENGTH
  Initial value:8
- #define UX_HOST_CLASS_STORAGE_CBW_FLAGS
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_CBW_LUN
  Initial value:13
- #define UX_HOST_CLASS_STORAGE_CBW_CB_LENGTH
  Initial value:14
- #define UX_HOST_CLASS_STORAGE_CSW_SIGNATURE
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_CSW_TAG
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_CSW_DATA_RESIDUE
  Initial value:8
- #define UX_HOST_CLASS_STORAGE_CSW_STATUS
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_CSW_LENGTH
  Initial value:13
- #define UX_HOST_CLASS_STORAGE_INQUIRY_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_INQUIRY_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_INQUIRY_PAGE_CODE
  Initial value:2

- #define UX_HOST_CLASS_STORAGE_INQUIRY_ALLOCATION_LENGTH
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_INQUIRY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_INQUIRY_COMMAND_LENGTH_SBC
  Initial value:06
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_PERIPHERAL_TYPE
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_REMOVABLE_MEDIA
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_DATA_FORMAT
  Initial value:3
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_ADDITIONAL_LENGTH
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_VENDOR_INFORMATION
  Initial value:8
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_PRODUCT_ID
  Initial value:16
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_PRODUCT_REVISION
  Initial value:32
- #define UX_HOST_CLASS_STORAGE_INQUIRY_RESPONSE_LENGTH
  Initial value:36
- #define UX_HOST_CLASS_STORAGE_START_STOP_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_START_STOP_LBUFLAGS
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_START_STOP_START_BIT
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_START_STOP_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_START_STOP_COMMAND_LENGTH_SBC
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_OPERATION
  Initial value:0

- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_PC_PAGE_CODE
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_PARAMETER_LIST_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_COMMAND_LENGTH_SBC
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RESPONSE_MODE_DATA_LENGTH
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RESPONSE_MEDIUM_TYPE_CODE
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RESPONSE_ATTRIBUTES_SHORT
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RESPONSE_ATTRIBUTES_WP
  Initial value:0x80
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_ALLOCATION_LENGTH
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_COMMAND_LENGTH_SBC
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_ERROR_CODE
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_SENSE_KEY
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_INFORMATION
  Initial value:3

- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_ADD_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_CODE
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_CODE_QUALIFIER
  Initial value:13
- #define UX_HOST_CLASS_STORAGE_REQUEST_SENSE_RESPONSE_LENGTH
  Initial value:18
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_LBA
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_PARAMETER_LIST_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_COMMAND_LENGTH_SBC
  Initial value:10
- #define UX_HOST_CLASS_STORAGE_READ_FORMAT_RESPONSE_LENGTH
  Initial value:0xFC
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_LBA
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_COMMAND_LENGTH_SBC
  Initial value:10
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_RESPONSE_LENGTH
  Initial value:8

- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_DATA_LBA
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_READ_CAPACITY_DATA_SECTOR_SIZE
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_TEST_READY_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_TEST_READY_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_TEST_READY_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_TEST_READY_COMMAND_LENGTH_SBC
  Initial value:6
- #define UX_HOST_CLASS_STORAGE_READ_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_READ_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_READ_LBA
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_READ_TRANSFER_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_READ_COMMAND_LENGTH_UFI
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_READ_COMMAND_LENGTH_SBC
  Initial value:10
- #define UX_HOST_CLASS_STORAGE_WRITE_OPERATION
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_WRITE_LUN
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_WRITE_LBA
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_WRITE_TRANSFER_LENGTH
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_WRITE_COMMAND_LENGTH_UFI
  Initial value:12

- #define UX_HOST_CLASS_STORAGE_WRITE_COMMAND_LENGTH_SBC
  Initial value:`10`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_NO_SENSE
  Initial value:`0x0`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_RECOVERED_ERROR
  Initial value:`0x1`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_NOT_READY
  Initial value:`0x2`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_MEDIUM_ERROR
  Initial value:`0x3`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_HARDWARE_ERROR
  Initial value:`0x4`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_ILLEGAL_REQUEST
  Initial value:`0x5`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_UNIT_ATTENTION
  Initial value:`0x6`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_DATA_PROTECT
  Initial value:`0x7`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_BLANK_CHECK
  Initial value:`0x8`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_ABORTED_COMMAND
  Initial value:`0x0b`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_VOLUME_OVERFLOW
  Initial value:`0x0d`
- #define UX_HOST_CLASS_STORAGE_SENSE_KEY_MISCOMPARE
  Initial value:`0x0e`
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RWER_PAGE
  Initial value:`0x01`
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_FD_PAGE
  Initial value:`0x05`
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RBAC_PAGE
  Initial value:`0x1B`
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_TP_PAGE
  Initial value:`0x1C`

- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_ALL_PAGE
  Initial value:0x3F
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_HEADER_PAGE_LENGTH
  Initial value:0x08
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RWER_PAGE_LENGTH
  Initial value:0x0c
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_FD_PAGE_LENGTH
  Initial value:0x20
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_RBAC_PAGE_LENGTH
  Initial value:0x0c
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_TP_PAGE_LENGTH
  Initial value:0x08
- #define UX_HOST_CLASS_STORAGE_MODE_SENSE_ALL_PAGE_LENGTH
  Initial value:0xC0
- #define UX_HOST_CLASS_STORAGE_ERROR_MEDIA_NOT_READ
  Initial value:0x023A00
- #define UX_HOST_CLASS_STORAGE_PARTITION_SIGNATURE
  Initial value:0xaa55
- #define UX_HOST_CLASS_STORAGE_PARTITION_TABLE_START
  Initial value:446
- #define UX_HOST_CLASS_STORAGE_PARTITION_BOOT_FLAG
  Initial value:0
- #define UX_HOST_CLASS_STORAGE_PARTITION_START_HEAD
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_PARTITION_START_SECTOR
  Initial value:2
- #define UX_HOST_CLASS_STORAGE_PARTITION_START_TRACK
  Initial value:3
- #define UX_HOST_CLASS_STORAGE_PARTITION_TYPE
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_PARTITION_END_HEAD
  Initial value:5
- #define UX_HOST_CLASS_STORAGE_PARTITION_END_SECTOR
  Initial value:6

- #define UX_HOST_CLASS_STORAGE_PARTITION_END_TRACK
  Initial value:7
- #define UX_HOST_CLASS_STORAGE_PARTITION_SECTORS_BEFORE
  Initial value:8
- #define UX_HOST_CLASS_STORAGE_PARTITION_NUMBER_SECTORS
  Initial value:12
- #define UX_HOST_CLASS_STORAGE_PARTITION_TABLE_SIZE
  Initial value:16
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_12
  Initial value:1
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_16
  Initial value:4
- #define UX_HOST_CLASS_STORAGE_PARTITION_EXTENDED
  Initial value:5
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_16L
  Initial value:6
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_32_1
  Initial value:0x0b
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_32_2
  Initial value:0x0c
- #define UX_HOST_CLASS_STORAGE_PARTITION_FAT_16_LBA_MAPPED
  Initial value:0x0e
- #define UX_HOST_CLASS_STORAGE_PARTITION_EXTENDED_LBA_MAPPED
  Initial value:0x0f
- #define UX_HOST_CLASS_STORAGE_CBW_LENGTH
  Initial value:31
- #define UX_HOST_CLASS_STORAGE_CBW_LENGTH_ALIGNED
  Initial value:64
- #define UX_HOST_CLASS_STORAGE_CSW_LENGTH_ALIGNED
  Initial value:64
- #define UX_INCLUDE_USER_DEFINE_FILE
  Initial value:
- #define UX_DISABLE_INCLUDE_SOURCE_CODE
  Initial value:

- #define UX_SYSTEM_DEVICE_ONLY

  Initial value:
- #define UX_DEVICE_SIDE_ONLY

  Initial value:
- #define UX_PERIODIC_RATE

  Initial value:`100`
- #define UX_MAX_CLASS_DRIVER

  Initial value:`8`
- #define UX_MAX_SLAVE_CLASS_DRIVER

  Initial value:`3`
- #define UX_MAX_HCD

  Initial value:`2`
- #define UX_MAX_DEVICES

  Initial value:`8`
- #define UX_MAX_ED

  Initial value:`80`
- #define UX_MAX_TD

  Initial value:`128`
- #define UX_MAX_ISO_TD

  Initial value:`128`
- #define UX_THREAD_STACK_SIZE

  Initial value:`512`
- #define UX_THREAD_PRIORITY_ENUM

  Initial value:`20`
- #define UX_THREAD_PRIORITY_CLASS

  Initial value:`20`
- #define UX_THREAD_PRIORITY_KEYBOARD

  Initial value:`20`
- #define UX_THREAD_PRIORITY_HCD

  Initial value:`2`
- #define UX_THREAD_PRIORITY_DCD

  Initial value:`2`
- #define UX_NO_TIME_SLICE

  Initial value:`0`

- #define UX_SLAVE_REQUEST_CONTROL_MAX_LENGTH
  Initial value:256
- #define UX_SLAVE_REQUEST_DATA_MAX_LENGTH
  Initial value:512

8.13.4 _ux_device_class_cdc_acm_activate

```
_ux_device_class_cdc_acm_activate ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.5 _ux_device_class_cdc_acm_control_complete

```
_ux_device_class_cdc_acm_control_complete ( UX_SLAVE_TRANSFER
* transfer_request )
```

8.13.6 _ux_device_class_cdc_acm_control_request

```
_ux_device_class_cdc_acm_control_request ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.7 _ux_device_class_cdc_acm_deactivate

```
_ux_device_class_cdc_acm_deactivate ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.8 _ux_device_class_cdc_acm_entry

```
_ux_device_class_cdc_acm_entry ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.9 _ux_device_class_cdc_acm_initialize

```
_ux_device_class_cdc_acm_initialize ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.10 _ux_device_class_cdc_acm_uninitialize

```
_ux_device_class_cdc_acm_uninitialize ( UX_SLAVE_CLASS_COMMAND  * command )
```

8.13.11 _ux_device_class_cdc_acm_write

```
_ux_device_class_cdc_acm_write ( UX_SLAVE_CLASS_CDC_ACM  * cdc_acm ,   UCHAR
* buffer ,   ULONG requested_length ,   ULONG * actual_length )
```

8.13.12 _ux_device_class_cdc_acm_read

```
_ux_device_class_cdc_acm_read ( UX_SLAVE_CLASS_CDC_ACM  * cdc_acm ,   UCHAR
* buffer ,   ULONG requested_length ,   ULONG * actual_length )
```

8.13.13 _ux_device_class_cdc_acm_thread

```
_ux_device_class_cdc_acm_thread (  ULONG cdc_acm_class )
```

8.13.14 _ux_device_class_cdc_acm_ioctl

```
_ux_device_class_cdc_acm_ioctl ( UX_SLAVE_CLASS_CDC_ACM  * cdc_acm ,
 ULONG ioctl_function ,   VOID * parameter )
```

8.13.15 usbhs_usb_int_resume_isr

```
usbhs_usb_int_resume_isr (  void  )
```

8.13.15.1 Brief description

This function calls the host interrupt handler or the device interrupt handler.

8.13.16 usbfs_int_isr

```
usbfs_int_isr (  void  )
```

8.13.16.1 Brief description

All the USBFS interrupts are handled by this ISR.

8.13.17 usbfs_resume_isr

```
usbfs_resume_isr (  void  )
```

8.13.17.1 Brief description

VBINT(VBUS interrupt), RESM(Resume interrupt) OVRCR(Over current input), BCHG(Change interrupt), and PDDETINT0(Bus change interrupt) fire this ISR. This is only used for canceling following standby modes.

8.13.17.2 Detailed description

- Canceling the software standby mode
- Canceling deep standby mode

8.13.18 ux_dcd_synergy_buffer_empty_interrupt

```
ux_dcd_synergy_buffer_empty_interrupt ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.18.1 Brief description

This function processes the BEMP interrupt for the specific endpoint.

8.13.18.2 Detailed description

Table 393:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to physical Endpoint(ED) control block
flag	in	Flag to enable or disable the buffer empty interrupt.

8.13.19 ux_dcd_synergy_buffer_notready_interrupt

```
ux_dcd_synergy_buffer_notready_interrupt ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.19.1 Brief description

This function processes the NRDY(Not Ready) interrupt for the specific endpoint.

8.13.19.2 Detailed description

Table 394:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to physical Endpoint(ED) control block

Table 394:Parameters (Continued)

Name	Direction	Description
flag	in	Flag for DCD synergy enable or disable.

8.13.20 ux_dcd_synergy_buffer_read

```
ux_dcd_synergy_buffer_read ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.20.1 Brief description

This function reads from a specified pipe into a buffer.

8.13.20.2 Detailed description

Table 395:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block

Table 396:Return values

Name	Description
UX_SUCCESS	Read a data from buffer successfully.
UX_ERROR	Unable to read a data from buffer.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_dcd_synergy_fifo_read

8.13.21 ux_dcd_synergy_buffer_ready_interrupt

```
ux_dcd_synergy_buffer_ready_interrupt ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.21.1 Brief description

This function enable or disable the BRDY(Ready) interrupt for the pipe.

8.13.21.2 Detailed description

Table 397:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block
flag	in	Check whether DCD synergy is enable or disable.

8.13.22 ux_dcd_synergy_buffer_write

```
ux_dcd_synergy_buffer_write ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.22.1 Brief description

This function writes a data from input buffer to the specified PIPE.

8.13.22.2 Detailed description

Table 398:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block

Table 399:Return values

Name	Description
UX_SUCCESS	Write a data to FIFO(D0, D1 and C) successfully.

Table 399:Return values (Continued)

Name	Description
UX_ERROR	Unable to write a data to FIFO(D0, D1 and C).

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_dcd_synergy_fifod_write
- ux_dcd_synergy_fifoc_write

8.13.23 ux_dcd_synergy_current_endpoint_change

```
ux_dcd_synergy_current_endpoint_change ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed ,   ULONG direction )
```

8.13.23.1 Brief description

This function configures the FIFO as per the request specified in endpoint descriptor.

8.13.23.2 Detailed description

Table 400:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block
direction	in	Endpoint direction

8.13.24 ux_dcd_synergy_data_buffer_size

```
ux_dcd_synergy_data_buffer_size ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.24.1 Brief description

This function returns the size of the data buffer and selects the specified pipe.

8.13.24.2 Detailed description

Table 401:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block

Table 402:Return values

Name	Description
buffer_size	Maximum packet size.

8.13.25 ux_dcd_synergy_endpoint_create

```
ux_dcd_synergy_endpoint_create ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_ENDPOINT  * endpoint )
```

8.13.25.1 Brief description

This function creates a physical endpoint.

8.13.25.2 Detailed description

Table 403:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
endpoint	in	Pointer to a Device Controller Endpoint structure.

Table 404:Return values

Name	Description
UX_SUCCESS	Endpoint is created successfully.
UX_ERROR	Buffer is not free or endpoint creation is unsuccessful.
UX_NO_ED_AVAILABLE	Endpoint is already in use.

8.13.26 ux_dcd_synergy_endpoint_destroy

```
ux_dcd_synergy_endpoint_destroy ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_ENDPOINT  * endpoint )
```

8.13.26.1 Brief description

This function will destroy a physical endpoint.

8.13.26.2 Detailed description

Table 405:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
endpoint	in	Pointer to a Device Controller Endpoint structure.

Table 406:Return values

Name	Description
UX_SUCCESS	Endpoint is destroyed successfully.

8.13.27 ux_dcd_synergy_endpoint_nak_set

```
ux_dcd_synergy_endpoint_nak_set ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.27.1 Brief description

This function sets a NAK(Not Acknowledged) to an endpoint.

8.13.27.2 Detailed description

Table 407:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block

8.13.28 ux_dcd_synergy_endpoint_reset

```
ux_dcd_synergy_endpoint_reset ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_ENDPOINT  * endpoint )
```

8.13.28.1 Brief description

This function will reset a physical endpoint.

8.13.28.2 Detailed description

Table 408:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
endpoint	in	Pointer to a Device Controller Endpoint structure.

Table 409:Return values

Name	Description
UX_SUCCESS	Endpoint is reset successfully.
UX_NO_ED_AVAILABLE	Device Controller Endpoint structure pointer is NULL

8.13.29 ux_dcd_synergy_endpoint_stall

```
ux_dcd_synergy_endpoint_stall ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_ENDPOINT  * endpoint )
```

8.13.29.1 Brief description

This function will stall a physical endpoint.

8.13.29.2 Detailed description

Table 410:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
endpoint	in	Pointer to a Device Controller Endpoint structure.

Table 411:Return values

Name	Description
UX_SUCCESS	Endpoint is stalled successfully.
UX_NO_ED_AVAILABLE	Device Controller Endpoint control pointer is Null.

8.13.30 ux_dcd_synergy_endpoint_status

```
ux_dcd_synergy_endpoint_status ( UX_DCD_SYNERGY  * dcd_synergy ,
 ULONG endpoint_index )
```

8.13.30.1 Brief description

This function will retrieve the status of the endpoint.

8.13.30.2 Detailed description

Table 412:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
endpoint_index	in	Endpoint number who's status is to be known.

Table 413:Return values

Name	Description
UX_ERROR	Endpoint already in use.
UX_FALSE	Endpoint is stalled.
UX_TRUE	Endpoint is not stalled.

8.13.31 ux_dcd_synergy_fifo_port_change

```
ux_dcd_synergy_fifo_port_change ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed ,   ULONG direction )
```

8.13.31.1 Brief description

This function configures the FIFO port.

8.13.31.2 Detailed description

Table 414:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to a DCD control block
ed	in	Pointer to a physical Endpoint(ED) control block
direction	in	Direction to switch

Table 415:Return values

Name	Description
UX_ERROR	Unable to change fifo port.

synergy_register Current endpoint index(pipe)

8.13.32 ux_dcd_synergy_fifo_read

```
ux_dcd_synergy_fifo_read ( UX_DCD_SYNERGY  * dcd_synergy ,  UX_DCD_SYNERGY_ED
* ed )
```

8.13.32.1 Brief description

This function reads from the hardware FIFO C, D0 or D1 and stores in the destination buffer.

8.13.32.2 Detailed description

Table 416:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
ed	inout	: Pointer to a physical Endpoint(ED) control block

Table 417:Return values

Name	Description
UX_ERROR	FIFO is not accessible.
UX_SYNERGY_DCD_FIFO_READ_OVER	FIFO read overflow.
UX_SYNERGY_DCD_FIFO_READ_SHORT	Short packet is received.
UX_SYNERGY_DCD_FIFO_READING	Continue reading FIFO.

8.13.33 ux_dcd_synergy_read_dma_set

```
ux_dcd_synergy_read_dma_set ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   ULONG fifo_sel )
```

8.13.33.1 Brief description

USBX DCD DMA read setup function. Call a subroutine for selected USB controller hardware.

8.13.33.2 Detailed description

Table 418:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
fifo_sel	in	FIFO select register

8.13.34 ux_dcd_synergy_read_dma_set_16bit

```
ux_dcd_synergy_read_dma_set_16bit ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   ULONG fifo_sel )
```

8.13.34.1 Brief description

USBX DCD DMA read setup function for USB hardwares with 16-bit FIFO.

8.13.34.2 Detailed description

Table 419:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
fifo_sel	in	FIFO select register

8.13.35 ux_dcd_synergy_fifo_read_dma_start

```
 ux_dcd_synergy_fifo_read_dma_start ( UX_DCD_SYNERGY  * dcd_synergy ,   UCHAR
* payload_buffer ,   VOID * p_fifo )
```

8.13.35.1 Brief description

USBX DCD FIFO read - DMA start function.

8.13.35.2 Detailed description

Table 420:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload_buffer	inout	Pointer to a payload buffer
p_fifo	in	FIFO register address

8.13.36 ux_dcd_synergy_fifo_read_software_copy

```
ux_dcd_synergy_fifo_read_software_copy ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   VOID * p_fifo ,
 ULONG fifo_sel )
```

8.13.36.1 Brief description

USBX DCD FIFO read by software copy. Call a subroutine for selected USB controller hardware.

8.13.36.2 Detailed description

Table 421:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address
fifo_sel	in	FIFO select register

8.13.37 ux_dcd_synergy_fifo_read_software_copy_16bit

```
ux_dcd_synergy_fifo_read_software_copy_16bit ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   VOID * p_fifo ,
 ULONG fifo_sel )
```

8.13.37.1 Brief description

USBX DCD FIFO read - Software copy for USB hardwares with 16-bit FIFO.

8.13.37.2 Detailed description

Table 422:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address
fifo_sel	in	FIFO select register

8.13.38 ux_dcd_synergy_fifo_read_last_bytes

```
 ux_dcd_synergy_fifo_read_last_bytes ( UX_DCD_SYNERGY_PAYLOAD_TRANSFER
* p_payload ,   VOID * p_fifo )
```

8.13.38.1 Brief description

USBX DCD FIFO read - Copy last bytes from FIFO by software if the rest bytes are less than FIFO access width.

8.13.38.2 Detailed description

Table 423:Parameters

Name	Direction	Description
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address

8.13.39 ux_dcd_synergy_fifoc_write

```
ux_dcd_synergy_fifoc_write ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.39.1 Brief description

This function writes a data from a buffer into USB FIFO using CPU.

8.13.39.2 Detailed description

Table 424:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
ed	inout	: Pointer to a physical Endpoint(ED) control block

Table 425:Return values

Name	Description
UX_ERROR	FIFO is not accessible.
UX_SYNERGY_DCD_FIFO_WRITE_END	Status for fifo write ends.
UX_SYNERGY_DCD_FIFO_WRITE_SHORT	Status for fifo short packet.
UX_SYNERGY_DCD_FIFO_WRITING	Status for fifo multiple writes.

8.13.40 ux_dcd_synergy_fifo_write_software_copy

```
ux_dcd_synergy_fifo_write_software_copy ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   VOID * p_fifo ,
 ULONG fifo_sel )
```

8.13.40.1 Brief description

USBX DCD FIFO write by software copy. Call a subroutine for selected USB controller hardware.

8.13.40.2 Detailed description

Table 426:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address
fifo_sel	in	FIFO select register

8.13.41 ux_dcd_synergy_fifo_write_software_copy_16bit

```
ux_dcd_synergy_fifo_write_software_copy_16bit ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   VOID * p_fifo ,
 ULONG fifo_sel )
```

8.13.41.1 Brief description

USBX DCD FIFO write - Software copy for USB hardwares with 16-bit FIFO.

8.13.41.2 Detailed description

Table 427:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address
fifo_sel	in	FIFO select register

8.13.42 ux_dcd_synergy_fifo_write_last_bytes

```
 ux_dcd_synergy_fifo_write_last_bytes ( UX_DCD_SYNERGY_PAYLOAD_TRANSFER
* p_payload ,   VOID * p_fifo ,   ULONG usb_base )
```

8.13.42.1 Brief description

USBX DCD FIFO write - Copy last bytes to FIFO by software if the rest bytes are less than FIFO access width.

8.13.42.2 Detailed description

Table 428:Parameters

Name	Direction	Description
p_payload	inout	Pointer to a payload transfer structure
p_fifo	in	FIFO register address
usb_base	in	USB controller hardware base address

8.13.43 ux_dcd_synergy_fifod_write

```
ux_dcd_synergy_fifod_write ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_ED  * ed )
```

8.13.43.1 Brief description

This function writes a buffer to FIFOD0 or FIFOD1.

8.13.43.2 Detailed description

Table 429:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
ed	inout	: Pointer to a physical Endpoint(ED) control block

Table 430:Return values

Name	Description
UX_ERROR	FIFO is not accessible.

Table 430:Return values (Continued)

Name	Description
UX_SYNERGY_DCD_FIFO_WRITE_END	Write ends of FIFO.
UX_SYNERGY_DCD_FIFO_WRITE_SHORT	Write short packet.
UX_SYNERGY_DCD_FIFO_WRITING	Continue multiple write to FIFO.

8.13.44 ux_dcd_synergy_write_dma_set

```
ux_dcd_synergy_write_dma_set ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   ULONG fifo_sel )
```

8.13.44.1 Brief description

USBX DCD DMA write setup function. Call a subroutine for selected USB controller hardware.

8.13.44.2 Detailed description

Table 431:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
fifo_sel	in	FIFO select register

8.13.45 ux_dcd_synergy_write_dma_set_16bit

```
ux_dcd_synergy_write_dma_set_16bit ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_DCD_SYNERGY_PAYLOAD_TRANSFER  * p_payload ,   ULONG fifo_sel )
```

8.13.45.1 Brief description

USBX DCD DMA write setup function for USB hardwares with 16-bit FIFO.

8.13.45.2 Detailed description

Table 432:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
p_payload	inout	Pointer to a payload transfer structure
fifo_sel	in	FIFO select register

8.13.46 ux_dcd_synergy_fifo_write_dma_start

```
 ux_dcd_synergy_fifo_write_dma_start ( UX_DCD_SYNERGY  * dcd_synergy ,   UCHAR
* payload_buffer ,   VOID * p_fifo )
```

8.13.46.1 Brief description

USBX DCD DMA FIFO write - DMA start function.

8.13.46.2 Detailed description

Table 433:Parameters

Name	Direction	Description
dcd_synergy	in	Pointer to the DCD control block
payload_buffer	inout	Pointer to a payload buffer
p_fifo	in	FIFO register address

8.13.47 ux_dcd_synergy_frame_number_get

```
 ux_dcd_synergy_frame_number_get ( UX_DCD_SYNERGY  * dcd_synergy ,   ULONG
* frame_number )
```

8.13.47.1 Brief description

This function will return the frame number currently used by the controller. This function is mostly used for isochronous purposes.

8.13.47.2 Detailed description

Table 434:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
frame_number	inout	: Pointer to a frame number in use

Table 435:Return values

Name	Description
UX_SUCCESS	In use frame number is returned successfully.

8.13.48 ux_dcd_synergy_function

```
 ux_dcd_synergy_function ( UX_SLAVE_DCD  * dcd ,   UINT function ,   VOID
* parameter )
```

8.13.48.1 Brief description

This function act as interface between upper layer USBX device stack and synergy controller.

8.13.48.2 Detailed description

Table 436:Parameters

Name	Direction	Description
dcd	inout	: Pointer to a USBX control block.
function	inout	: Function requested to be despatched.
parameter	inout	: Pointer to requested function parameters.

Table 437:Return values

Name	Description
UX_CONTROLLER_UNKNOWN	Desired controller is not specified.
UX_FUNCTION_NOT_SUPPORTED	DCD function is not supported by the controller.
UX_SUCCESS	DCD function despatched successfully.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_dcd_synergy_frame_number_get
- ux_dcd_synergy_transfer_request
- ux_dcd_synergy_endpoint_create
- ux_dcd_synergy_endpoint_destroy
- ux_dcd_synergy_endpoint_reset
- ux_dcd_synergy_endpoint_stall
- ux_dcd_synergy_endpoint_status

8.13.49 ux_dcd_synergy_initialize

```
ux_dcd_synergy_initialize (  ULONG dcd_io )
```

8.13.49.1 Brief description

This function initializes the USB slave controller for Renesas Synergy MCUs.

8.13.49.2 Detailed description

Table 438:Parameters

Name	Direction	Description
dcd_io	inout	: Address of DCD

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_dcd_synergy_initialize_common()

8.13.50 ux_dcd_synergy_initialize_transfer_support

```
ux_dcd_synergy_initialize_transfer_support (  ULONG dcd_io ,
UX_DCD_SYNERGY_TRANSFER  * p_transfer_instance )
```

8.13.50.1 Brief description

The function initializes the USB slave controller of the Renesas Synergy MCUs with associated DMA transfer modules.

8.13.50.2 Detailed description

Table 439:Parameters

Name	Direction	Description
dcd_io	in	Address of the USB controller.
p_transfer_instance	in	Pointer to Synergy Transfer module instances.

Table 440:Return values

Name	Description
UX_SUCCESS	Completed the USB controller initialization successfully.
UX_CONTROLLER_INIT_FAILED	Failed to initialize the USB controller.
UX_MEMORY_INSUFFICIENT	Memory was not allocated properly for the Synergy DCD instance.

8.13.51 ux_dcd_synergy_initialize_common

```
ux_dcd_synergy_initialize_common (  ULONG dcd_io )
```

8.13.51.1 Brief description

USBX DCD first half common part of the USB controller initialization.

8.13.51.2 Detailed description

Table 441:Parameters

Name	Direction	Description
dcd_io	in	Address of the USB controller.

Table 442:Return values

Name	Description
UX_SUCCESS	Completed the USB controller initialization successfully.
UX_CONTROLLER_INIT_FAILED	Failed to initialize the USB controller.
UX_MEMORY_INSUFFICIENT	Memory was not allocated properly for the Synergy DCD instance.

8.13.52 ux_dcd_synergy_initialize_common_complete

```
ux_dcd_synergy_initialize_common_complete (  void  )
```

8.13.52.1 Detailed description

USBX DCD bottom-half common part of the USB controller initialization.

8.13.53 ux_dcd_dma_complete_tx

```
ux_dcd_dma_complete_tx ( transfer_callback_args_t  * p_args )
```

8.13.53.1 Brief description

USBX DCD DMA write callback function.

8.13.53.2 Detailed description

Table 443:Parameters

Name	Direction	Description
p_args	in	Pointer to the argument of a Transfer module callback function.

8.13.54 ux_dcd_dma_complete_rx

```
ux_dcd_dma_complete_rx ( transfer_callback_args_t  * p_args )
```

8.13.54.1 Brief description

USBX DCD DMA read callback function.

8.13.54.2 Detailed description

Table 444:Parameters

Name	Direction	Description
p_args	in	Pointer to the argument of a Transfer module callback function.

8.13.55 ux_dcd_synergy_initialize_complete

```
ux_dcd_synergy_initialize_complete (  VOID  )
```

8.13.55.1 Brief description

This function completes the initialization of the USB slave controller for the Renesas Synergy MCUs.

8.13.55.2 Detailed description

Table 445:Return values

Name	Description
UX_SUCCESS	USB slave is initialized successfully.

8.13.56 ux_dcd_synergy_interrupt_handler

```
ux_dcd_synergy_interrupt_handler (  VOID  )
```

8.13.56.1 Brief description

This function is the interrupt handler for the synergy controller. The controller will trigger an interrupt when something happens on an endpoint whose mask has been set in the interrupt enable register.

8.13.57 ux_dcd_synergy_register_clear

```
ux_dcd_synergy_register_clear ( UX_DCD_SYNERGY  * dcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.57.1 Brief description

This function clears a bit in a register of the synergy.

8.13.57.2 Detailed description

Table 446:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
synergy_register	inout	: Register to clear
value	inout	: Value to clear

8.13.58 ux_dcd_synergy_register_read

```
ux_dcd_synergy_register_read ( UX_DCD_SYNERGY  * dcd_synergy ,
 ULONG synergy_register )
```

8.13.58.1 Brief description

This function reads a synergy USB register.

8.13.58.2 Detailed description

Table 447:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
synergy_register	inout	: Register to read

Table 448:Return values

Name	Description
dcd_reg	Value read from USB register.

8.13.59 ux_dcd_synergy_register_set

```
ux_dcd_synergy_register_set ( UX_DCD_SYNERGY  * dcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.59.1 Brief description

This function set a bit in synergy USB register.

8.13.59.2 Detailed description

Table 449:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
synergy_register	inout	: Register to set
value	inout	: Value to set

8.13.60 ux_dcd_synergy_register_write

```
ux_dcd_synergy_register_write ( UX_DCD_SYNERGY  * dcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.60.1 Brief description

This function writes a bit in synergy USB register.

8.13.60.2 Detailed description

Table 450:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
synergy_register	inout	: Register to write
value	inout	: Value to write

8.13.61 ux_dcd_synergy_transfer_callback

```
ux_dcd_synergy_transfer_callback ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_TRANSFER  * transfer_request ,   ULONG interrupt_status ,
 ULONG ctsq_mask )
```

8.13.61.1 Brief description

This function is invoked under ISR when an event happens on a specific endpoint.

8.13.61.2 Detailed description

Table 451:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
transfer_request	inout	: Pointer to USBX Device Transfer Request structure
interrupt_status	inout	: Check if we have SETUP condition or BRDY or BEMP interrupt.
ctsq_mask	inout	: Mask to isolate the CTSQ field.

Table 452:Return values

Name	Description
UX_SUCCESS	Function is invoked under ISR successfully.

8.13.62 ux_dcd_synergy_transfer_request

```
ux_dcd_synergy_transfer_request ( UX_DCD_SYNERGY  * dcd_synergy ,
UX_SLAVE_TRANSFER  * transfer_request )
```

8.13.62.1 Brief description

This function will initiate a transfer to a specific endpoint. If the endpoint is IN, the endpoint register will be set to accept the request. If the endpoint is IN, the endpoint FIFO will be filled with the buffer and the endpoint register set.

8.13.62.2 Detailed description

Table 453:Parameters

Name	Direction	Description
dcd_synergy	inout	: Pointer to a DCD control block
transfer_request	inout	: Pointer to transfer request

Table 454:Return values

Name	Description
UX_SUCCESS	Transfer to a specific endpoint is initiated successfully.
ux_slave_transfer_request_completion_code	Pointer to structure UX_SLAVE_TRANSFER.
UX_TRANSFER_ERROR	Transfer is completed with error.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_dcd_synergy_buffer_write

8.13.63 ux_hcd_synergy_asynch_queue_process

```
ux_hcd_synergy_asynch_queue_process ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.63.1 Brief description

This function process the asynchronous transactions. The function will identify the USB interrupts occurred associated with an endpoint and will process the interrupts.

8.13.63.2 Detailed description

Table 455:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

8.13.64 ux_hcd_synergy_asynch_queue_process_bemp

```
ux_hcd_synergy_asynch_queue_process_bemp ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.64.1 Brief description

This function process the BEMP(Buffer Empty) interrupt that occurred on a specific ED.

8.13.64.2 Detailed description

Table 456:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to an Endpoint control block

8.13.65 ux_hcd_synergy_asynch_queue_process_brdy

```
ux_hcd_synergy_asynch_queue_process_brdy ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.65.1 Brief description

This function process the BRDY(Buffer Ready)interrupt that occurred on a specific ED.

8.13.65.2 Detailed description

Table 457:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to an Endpoint control block

8.13.66 ux_hcd_synergy_asynch_queue_process_nrdy

```
ux_hcd_synergy_asynch_queue_process_nrdy ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.66.1 Brief description

This function process the NRDY(Not Ready) Interrupt that occurred on a specific ED.

8.13.66.2 Detailed description

Table 458:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to an Endpoint control block

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- UX_TRANSFER::ux_transfer_request_completion_function

8.13.67 ux_hcd_synergy_asynch_queue_process_sign

```
ux_hcd_synergy_asynch_queue_process_sign ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.67.1 Brief description

This function process the Setup transaction Error Interrupt.

8.13.67.2 Detailed description

Table 459:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to an Endpoint control block

8.13.68 ux_hcd_synergy_asynch_schedule

```
ux_hcd_synergy_asynch_schedule ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.68.1 Brief description

This function schedules new transfers from the control or bulk lists.

8.13.68.2 Detailed description

Table 460:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

8.13.69 ux_hcd_synergy_asynchronous_endpoint_create

```
ux_hcd_synergy_asynchronous_endpoint_create ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.69.1 Brief description

This function will create an asynchronous endpoint. The control and bulk endpoints fall into this category.

8.13.69.2 Detailed description

Table 461:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to endpoint

Table 462:Return values

Name	Description
UX_NO_ED_AVAILABLE	ED for new endpoint not available.
UX_NO_TD_AVAILABLE	Dummy TD not available for terminating the ED transfer chain.
UX_SUCCESS	Asynchronous endpoint created successfully.

8.13.70 ux_hcd_synergy_asynchronous_endpoint_destroy

```
ux_hcd_synergy_asynchronous_endpoint_destroy ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.70.1 Brief description

This function will destroy an asynchronous endpoint. The control and bulk endpoints fall into this category.

8.13.70.2 Detailed description

Table 463:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to endpoint

Table 464:Return values

Name	Description
UX_ENDPOINT_HANDLE_UNKNOWN	Physical endpoint has not been initialized properly.
UX_SUCCESS	Asynchronous endpoint destroyed successfully.

8.13.71 ux_hcd_synergy_buffer_empty_interrupt

```
ux_hcd_synergy_buffer_empty_interrupt ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.71.1 Brief description

This function enable or disable the BEMP(Buffer Empty) interrupt for the pipe.

8.13.71.2 Detailed description

Table 465:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to physical Endpoint(ED) control block
flag	in	: Check whether DCD synergy is enable or disable.

8.13.72 ux_hcd_synergy_buffer_notready_interrupt

```
ux_hcd_synergy_buffer_notready_interrupt ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.72.1 Brief description

This function enable or disable the NRDY(Not Ready) interrupt for the pipe.

8.13.72.2 Detailed description

Table 466:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to physical Endpoint(ED) control block
flag	in	: Check whether DCD synergy is enable or disable.

8.13.73 ux_hcd_synergy_buffer_read

```
ux_hcd_synergy_buffer_read ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.73.1 Brief description

This function reads from a specified pipe into a buffer.

8.13.73.2 Detailed description

Table 467:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to a physical Endpoint(ED) control block

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_fifo_read

8.13.74 ux_hcd_synergy_buffer_ready_interrupt

```
ux_hcd_synergy_buffer_ready_interrupt ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed ,   ULONG flag )
```

8.13.74.1 Brief description

This function enable or disable the BRDY(Ready) interrupt for the pipe.

8.13.74.2 Detailed description

Table 468:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to physical Endpoint(ED) control block
flag	in	: Check whether DCD synergy is enable or disable.

8.13.75 ux_hcd_synergy_buffer_write

```
 ux_hcd_synergy_buffer_write ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.75.1 Brief description

This function writes data to the selected FIFO of the endpoint.

8.13.75.2 Detailed description

Table 469:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	in	: Pointer to a physical Endpoint(ED) control block

Table 470:Return values

Name	Description
UX_SUCCESS	Buffer written to the specified PIPE successfully.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_fifod_write
- ux_hcd_synergy_fifoc_write

8.13.76 ux_hcd_synergy_bulk_endpoint_create

```
ux_hcd_synergy_bulk_endpoint_create ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.76.1 Brief description

This function will create a bulk endpoint.

8.13.76.2 Detailed description

Table 471:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to a USBX Endpoint Container structure

Table 472:Return values

Name	Description
UX_ERROR	PIPE is not available for bulk endpoint creation .
UX_SUCCESS	Bulk endpoints created successfully.
UX_NO_ED_AVAILABLE	ED for bulk endpoint is not available.
UX_NO_TD_AVAILABLE	Dummy TD for for terminating the ED transfer chain is not available.

8.13.77 ux_hcd_synergy_bulk_int_td_add

```
ux_hcd_synergy_bulk_int_td_add ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.77.1 Brief description

This function adds a transfer descriptor to an Bulk or INT ED.

8.13.77.2 Detailed description

Table 473:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to a Synergy ED structure

Table 474:Return values

Name	Description
UX_SUCCESS	Transfer descriptor added successfully.

8.13.78 ux_hcd_synergy_control_endpoint_create

```
ux_hcd_synergy_control_endpoint_create ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.78.1 Brief description

This function will create a control endpoint.

8.13.78.2 Detailed description

Table 475:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to an Endpoint control block

Table 476:Return values

Name	Description
UX_SUCCESS	Control endpoint created successfully.
UX_NO_ED_AVAILABLE	Failed to obtain an ED for control endpoint.
UX_NO_TD_AVAILABLE	Failed to obtain a TD for control endpoint.

8.13.79 ux_hcd_synergy_control_td_add

```
ux_hcd_synergy_control_td_add ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.79.1 Brief description

This function adds a transfer descriptor to an ED.

8.13.79.2 Detailed description

Table 477:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to Synergy ED structure

Table 478:Return values

Name	Description
UX_SUCCESS	Transfer descriptor added to an ED successfully.

8.13.80 ux_hcd_synergy_controller_disable

```
ux_hcd_synergy_controller_disable ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.80.1 Brief description

This function will disable the Synergy controller. The controller will release all its resources (memory, IO ...). After this, the controller will not send SOF any longer. All transactions should have been completed, all classes should have been closed.

8.13.80.2 Detailed description

Table 479:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 480:Return values

Name	Description
UX_SUCCESS	Synergy controller disabled successfully.

8.13.81 ux_hcd_synergy_current_endpoint_change

```
ux_hcd_synergy_current_endpoint_change ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed ,   ULONG direction )
```

8.13.81.1 Brief description

This function change the endpoint in the FIFO.

8.13.81.2 Detailed description

Table 481:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to Synergy ED structure
direction	in	: Direction to transfer

8.13.82 ux_hcd_synergy_data_buffer_size

```
ux_hcd_synergy_data_buffer_size ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.82.1 Brief description

This function returns the size of the buffer data.

8.13.82.2 Detailed description

Table 482:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to Synergy ED structure

Table 483:Return values

Name	Description
buffer_size	buffer size

8.13.83 ux_hcd_synergy_ed_obtain

```
UX_SYNERGY_ED ux_hcd_synergy_ed_obtain ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.83.1 Brief description

This function obtains a free ED from the ED list.

8.13.83.2 Detailed description

Table 484:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 485:Return values

Name	Description
UX_NULL	No available ED in the ED list.

ed Endpoint descriptor pointer address.

8.13.84 ux_hcd_synergy_ed_td_clean

```
ux_hcd_synergy_ed_td_clean ( UX_SYNERGY_ED  * ed )
```

8.13.84.1 Brief description

This function process cleans the ED of all tds except the last dummy TD.

8.13.84.2 Detailed description

Table 486:Parameters

Name	Direction	Description
ed	inout	: Pointer to Synergy ED structure

8.13.85 ux_hcd_synergy_endpoint_nak_set

```
ux_hcd_synergy_endpoint_nak_set ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed )
```

8.13.85.1 Brief description

This function sets a NAK(Not Acknowledged) to an endpoint.

8.13.85.2 Detailed description

Table 487:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block

Table 487:Parameters (Continued)

Name	Direction	Description
ed	in	: Pointer to Synergy ED structure

8.13.86 ux_hcd_synergy_endpoint_reset

```
 ux_hcd_synergy_endpoint_reset ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_ENDPOINT
* endpoint )
```

8.13.86.1 Brief description

This function will reset an endpoint.

8.13.86.2 Detailed description

Table 488:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
endpoint	inout	: Pointer to an Endpoint control block

Table 489:Return values

Name	Description
UX_SUCCESS	Endpoint reset successfully.

8.13.87 ux_hcd_synergy_entry

```
ux_hcd_synergy_entry ( UX_HCD  * hcd ,   UINT function ,   VOID * parameter )
```

8.13.87.1 Brief description

This function is the entry function to the USB driver from the USB stack.

8.13.87.2 Detailed description

Table 490:Parameters

Name	Direction	Description
hcd	in	: Pointer to USBX Host Controller structure.
function	in	: Function for driver to perform
parameter	in	: Pointer to function parameter(s)

Table 491:Return values

Name	Description
UX_SUCCESS	HCD function is dispatched successfully.
UX_CONTROLLER_UNKNOWN	Synergy controller is not known.
UX_FUNCTION_NOT_SUPPORTED	Function not supported.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_controller_disable
- ux_hcd_synergy_port_status_get
- ux_hcd_synergy_port_enable
- ux_hcd_synergy_port_disable
- ux_hcd_synergy_power_on_port
- ux_hcd_synergy_power_down_port
- ux_hcd_synergy_port_suspend
- ux_hcd_synergy_port_resume
- ux_hcd_synergy_port_reset
- ux_hcd_synergy_frame_number_get
- ux_hcd_synergy_request_transfer
- ux_hcd_synergy_transfer_abort
- ux_hcd_synergy_control_endpoint_create
- ux_hcd_synergy_bulk_endpoint_create
- ux_hcd_synergy_interrupt_endpoint_create

- ux_hcd_synergy_isochronous_endpoint_create
- ux_hcd_synergy_asynchronous_endpoint_destroy
- ux_hcd_synergy_periodic_endpoint_destroy
- ux_hcd_synergy_endpoint_reset

8.13.88 ux_hcd_synergy_fifo_port_change

```
ux_hcd_synergy_fifo_port_change ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_SYNERGY_ED  * ed ,   ULONG direction )
```

8.13.88.1 Brief description

This function change the port of the FIFO.

8.13.88.2 Detailed description

Table 492:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
ed	in	: Pointer to Synergy ED structure
direction	in	: Direction to transfer

Table 493:Return values

Name	Description
synergy_register	Content of FIFO control register.
UX_ERROR	Port not changed successfully or unable to access FIFO.

8.13.89 ux_hcd_synergy_fifo_read

```
ux_hcd_synergy_fifo_read ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.89.1 Brief description

This function read data from the FIFO configured for the PIPE(FIFO C, D0 or D1).

8.13.89.2 Detailed description

Table 494:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	in	: Pointer to Synergy ED structure

Table 495:Return values

Name	Description
UX_ERROR	Unable to access FIFO successfully.
UX_SYNERGY_HC_FIFO_READ_OVER	Status set to read overflow.
UX_SYNERGY_HC_FIFO_READ_SHORT	Short packet to read.
UX_SYNERGY_HC_FIFO_READING	Continue reading buffer.

8.13.90 ux_hcd_synergy_fifo_read_software_copy

```
ux_hcd_synergy_fifo_read_software_copy ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,  UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.90.1 Brief description

USBX HCD CPU FIFO read by software copy. Call a suitable subroutine for selected USB controller hardware.

8.13.90.2 Detailed description

Table 496:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload	in	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.91 ux_hcd_synergy_fifo_read_software_copy_16bit

```
ux_hcd_synergy_fifo_read_software_copy_16bit ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,  UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.91.1 Brief description

USBX HCD CPU FIFO read - Software copy with 16-bit FIFO access.

8.13.91.2 Detailed description

Table 497:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.92 ux_hcd_synergy_fifo_read_software_copy_8bit

```
ux_hcd_synergy_fifo_read_software_copy_8bit ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,  UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.92.1 Brief description

USBX HCD CPU FIFO read - Software copy with 8-bit FIFO access.

8.13.92.2 Detailed description

Table 498:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.93 ux_hcd_synergy_fifo_read_dma_start

```
ux_hcd_synergy_fifo_read_dma_start ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,  UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.93.1 Brief description

USBX HCD DMA read setup function.

8.13.93.2 Detailed description

Table 499:Parameters

Name	Direction	Description
hcd_synergy	inout	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.94 ux_hcd_synergy_fifo_read_dma_set_16bit_or_8bit

```
ux_hcd_synergy_fifo_read_dma_set_16bit_or_8bit ( UX_HCD_SYNERGY
* hcd_synergy ,  UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,
UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.94.1 Brief description

USBX HCD DMA FIFO read - DMA transfer with 16 or 8 bit FIFO access.

8.13.94.2 Detailed description

Table 500:Parameters

Name	Direction	Description
hcd_synergy	inout	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.95 ux_hcd_synergy_fifoc_write

```
ux_hcd_synergy_fifoc_write ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.95.1 Brief description

This function writes a buffer to FIFOC.

8.13.95.2 Detailed description

Table 501:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to Synergy ED structure

Table 502:Return values

Name	Description
UX_ERROR	Unable to access FIFO successfully.
UX_SYNERGY_HC_FIFO_WRITE_END	Writing at ends.
UX_SYNERGY_HC_FIFO_WRITE_SHORT	Writing short data.
UX_SYNERGY_HC_FIFO_WRITING	Doing multiple writes.

8.13.96 ux_hcd_synergy_fifo_write_software_copy

```
ux_hcd_synergy_fifo_write_software_copy ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG payload_length ,   UCHAR * payload_buffer ,   VOID * fifo_addr ,
 ULONG fifo_sel )
```

8.13.96.1 Brief description

USBX HCD CPU FIFO write by software copy. Call a suitable subroutine for selected USB controller hardware.

8.13.96.2 Detailed description

Table 503:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload_length	in	Payload length
payload_buffer	in	Payload buffer address
fifo_addr	in	FIFO register address
fifo_sel	in	FIFO select register

8.13.97 ux_hcd_synergy_fifo_write_software_copy_16bit

```
ux_hcd_synergy_fifo_write_software_copy_16bit ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG payload_length ,   UCHAR * payload_buffer ,   VOID * fifo_addr ,
 ULONG fifo_sel )
```

8.13.97.1 Brief description

USBX HCD CPU FIFO write - Software copy with 16-bit FIFO access.

8.13.97.2 Detailed description

Table 504:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload_length	inout	Payload length
payload_buffer	inout	Payload buffer address
fifo_addr	inout	FIFO register address
fifo_sel	in	FIFO select register

8.13.98 ux_hcd_synergy_fifo_write_software_copy_8bit

```
ux_hcd_synergy_fifo_write_software_copy_8bit ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG payload_length ,   UCHAR * payload_buffer ,   VOID * fifo_addr ,
 ULONG fifo_sel )
```

8.13.98.1 Brief description

USBX HCD CPU FIFO write - Software copy with 8-bit FIFO access.

8.13.98.2 Detailed description

Table 505:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block
payload_length	inout	Payload length
payload_buffer	inout	Payload buffer address
fifo_addr	inout	FIFO register address
fifo_sel	in	FIFO select register

8.13.99 ux_hcd_synergy_fifo_write_software_copy_remaining_bytes

```
 ux_hcd_synergy_fifo_write_software_copy_remaining_bytes ( UX_HCD_SYNERGY
* hcd_synergy ,   ULONG payload_length ,   UCHAR * payload_buffer ,   VOID
* fifo_addr )
```

8.13.99.1 Brief description

USBX HCD CPU FIFO write - Copy remaining bytes to FIFO by software if the rest bytes are less than FIFO access width.

8.13.99.2 Detailed description

Table 506:Parameters

Name	Direction	Description
hcd_synergy	in	Pointer to the HCD control block

Table 506:Parameters (Continued)

Name	Direction	Description
payload_length	inout	Payload length
payload_buffer	inout	Payload buffer address
fifo_addr	inout	FIFO register address

8.13.100 ux_hcd_synergy_fifod_write

```
 ux_hcd_synergy_fifod_write ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED
* ed )
```

8.13.100.1 Brief description

This function writes a buffer data to FIFOD0 or FIFOD1.

8.13.100.2 Detailed description

Table 507:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
ed	inout	: Pointer to Synergy ED structure

Table 508:Return values

Name	Description
UX_ERROR	Unable to access FIFO successfully.
UX_SYNERGY_HC_FIFO_WRITE_END	Writing at ends.
UX_SYNERGY_HC_FIFO_WRITE_SHORT	Writing short data.
UX_SYNERGY_HC_FIFO_WRITING	Doing multiple writes.

8.13.101 ux_hcd_synergy_fifod_write_dma_start

```
ux_hcd_synergy_fifod_write_dma_start ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,  UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.101.1 Brief description

USBX HCD DMA write setup function. Call a suitable subroutine for selected USB controller hardware.

8.13.101.2 Detailed description

Table 509:Parameters

Name	Direction	Description
hcd_synergy	inout	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	in	Pointer to the fifo block

8.13.102 ux_hcd_synergy_fifod_write_dma_set_16bit_or_8bit

```
ux_hcd_synergy_fifod_write_dma_set_16bit_or_8bit ( UX_HCD_SYNERGY
* hcd_synergy ,  UX_HCD_SYNERGY_PAYLOAD_TRANSFER  * payload ,
UX_HCD_SYNERGY_FIFO  * fifo )
```

8.13.102.1 Brief description

USBX HCD DMA FIFO write - DMA transfer with 16 or 8 bit FIFO access.

8.13.102.2 Detailed description

Table 510:Parameters

Name	Direction	Description
hcd_synergy	inout	Pointer to the HCD control block
payload	inout	Pointer to the Payload block
fifo	inout	Pointer to the fifo block

8.13.103 ux_hcd_synergy_frame_number_get

```
ux_hcd_synergy_frame_number_get ( UX_HCD_SYNERGY  * hcd_synergy ,   ULONG
* frame_number )
```

8.13.103.1 Brief description

This function will return the frame number currently used by the controller. This function is mostly used for isochronous purposes and for timing.

8.13.103.2 Detailed description

Table 511:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
frame_number	inout	: Frame number to set

Table 512:Return values

Name	Description
UX_SUCCESS	Frame number returned successfully.

8.13.104 ux_hcd_synergy_frame_number_set

```
ux_hcd_synergy_frame_number_set ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG frame_number )
```

8.13.104.1 Brief description

This function will set the current frame number to the one specified. This function is mostly used for isochronous purpos.es.

8.13.104.2 Detailed description

Table 513:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
frame_number	inout	: Frame number to set

8.13.105 ux_hcd_synergy_initialize

```
ux_hcd_synergy_initialize ( UX_HCD  * hcd )
```

8.13.105.1 Brief description

This function initializes the Synergy controller.

8.13.105.2 Detailed description

Table 514:Parameters

Name	Direction	Description
hcd	inout	: Pointer to USBX host controller structure.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_initialize_common()

8.13.106 ux_hcd_synergy_initialize_common

```
ux_hcd_synergy_initialize_common ( UX_HCD  * hcd )
```

8.13.106.1 Brief description

This function Initialize USB peripheral.

8.13.106.2 Detailed description

Table 515:Parameters

Name	Direction	Description
hcd	inout	: Pointer to USBX host controller structure.

Table 516:Return values

Name	Description
UX_SUCCESS	Initialize hcd transfer support successfully.
UX_CONTROLLER_INIT_FAILED	Failed in Transfer module setup, or Unsupported USB controller was specified.
UX_MEMORY_INSUFFICIENT	Failed in allocation memory.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_periodic_tree_create

8.13.107 ux_hcd_synergy_initialize_transfer_support

```
ux_hcd_synergy_initialize_transfer_support ( UX_HCD  * hcd ,
UX_HCD_SYNERGY_TRANSFER const * p_transfer_instance )
```

8.13.107.1 Brief description

USBX HCD Transfer Support with DMA support.

8.13.107.2 Detailed description

Table 517:Parameters

Name	Direction	Description
hcd	inout	Pointer to the USBX HCD control block.
p_transfer_instance	in	Pointer to Transfer module instances.

Table 518:Return values

Name	Description
UX_SUCCESS	Initialize hcd transfer support successfully.
UX_CONTROLLER_INIT_FAILED	Failed in Transfer module setup, or Unsupported USB controller was specified.
UX_SEMAPHORE_ERROR	Failed in creating a semaphore used for DMA transfer.
UX_MEMORY_INSUFFICIENT	Failed in allocation memory.

8.13.108 ux_hcd_synergy_initialize_common_complete

```
ux_hcd_synergy_initialize_common_complete ( UX_HCD  * hcd )
```

8.13.108.1 Brief description

USBX HCD final procedure for the initialization.

8.13.108.2 Detailed description

Table 519:Parameters

Name	Direction	Description
hcd	in	Pointer to the USBX HCD control block.

8.13.109 ux_hcd_dma_complete_tx

```
ux_hcd_dma_complete_tx ( transfer_callback_args_t  * p_args )
```

8.13.109.1 Brief description

USBX HCD DMA write callback function.

8.13.109.2 Detailed description

Table 520:Parameters

Name	Direction	Description
p_args	in	Pointer to the argument of a Transfer module callback function.

8.13.110 ux_hcd_dma_complete_rx

```
ux_hcd_dma_complete_rx ( transfer_callback_args_t  * p_args )
```

8.13.110.1 Brief description

USBX HCD DMA read callback function.

8.13.110.2 Detailed description

Table 521:Parameters

Name	Direction	Description
p_args	in	Pointer to the argument of a Transfer module callback function.

8.13.111 ux_hcd_synergy_interrupt_endpoint_create

```
ux_hcd_synergy_interrupt_endpoint_create ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.111.1 Brief description

This function will create an interrupt endpoint. The interrupt endpoint has an interval of operation from 1 to 255. The Synergy has no hardware scheduler but we still build an interrupt tree similar to the OHCI controller.

8.13.111.2 Detailed description

This routine will match the best interval for the Synergy hardware. It will also determine the best node to hook the endpoint based on the load that already exists on the horizontal ED chain.

The tricky part is to understand how the interrupt matrix is constructed. We have used eds with the skip bit on to build a frame of anchor eds. Each ED creates a node for an appropriate combination of interval frequency in the list.

After obtaining a pointer to the list with the lowest traffic, we traverse the list from the highest interval until we reach the interval required. At that node, we anchor our real ED to the node and link the ED that was attached to the node to our ED.

Table 522:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to an Endpoint control block

Table 523:Return values

Name	Description
UX_SUCCESS	Interrupt endpoint created successfully.
UX_ERROR	Available pipe is not found for interrupt endpoint.
UX_NO_ED_AVAILABLE	Failed to obtain an ED for new endpoint.
UX_NO_TD_AVAILABLE	Failed to obtain a TD for terminating the ED transfer chain.

8.13.112 ux_hcd_synergy_interrupt_handler

```
ux_hcd_synergy_interrupt_handler (  UINT hcd_index )
```

8.13.112.1 Brief description

This function is the interrupt handler for the Synergy USB HS controller. Normally an interrupt occurs from the controller when there is either a EOF signal and there has been transfers within the frame or when there is a change on one of the downstream ports.

8.13.112.2 Detailed description

All we need to do in the ISR is scan the controllers to find out which one has issued a IRQ. If there is work to do for this controller we need to wake up the corresponding thread to take care of the job.

Table 524:Parameters

Name	Direction	Description
hcd_index	in	: HCD number

8.13.113 ux_hcd_synergy_iso_queue_process

```
ux_hcd_synergy_iso_queue_process ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.113.1 Brief description

This function process the isochronous transactions that happened in the last frame. Present, this function doesn't support isochronous functionality.

8.13.113.2 Detailed description

Table 525:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block

8.13.114 ux_hcd_synergy_iso_schedule

```
ux_hcd_synergy_iso_schedule ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.114.1 Brief description

This function schedules new transfers from isochronous list. Present, this function doesn't support isochronous functionality.

8.13.114.2 Detailed description

Table 526:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block

8.13.115 ux_hcd_synergy_isochronous_endpoint_create

```
ux_hcd_synergy_isochronous_endpoint_create ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.115.1 Brief description

This function creates an isochronous endpoint.

8.13.115.2 Detailed description

Table 527:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to an Endpoint control block

Table 528:Return values

Name	Description
UX_SUCCESS	Isochronous endpoint is created successfully.
UX_NO_ED_AVAILABLE	Failed to obtain an ED terminating the ED transfer chain.
UX_NO_TD_AVAILABLE	Failed to obtain a TD for new endpoint.

8.13.116 ux_hcd_synergy_isochronous_td_obtain

```
UX_SYNERGY_ISO_TD ux_hcd_synergy_isochronous_td_obtain ( UX_HCD_SYNERGY
* hcd_synergy )
```

8.13.116.1 Brief description

This function obtains a free TD from the isochronous TD list.

8.13.116.2 Detailed description

Table 529:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 530:Return values

Name	Description
td	Pointer to Synergy ISO ED.
UX_NULL	TD not available in the TD list.

8.13.117 ux_hcd_synergy_least_traffic_list_get

```
UX_SYNERGY_ED ux_hcd_synergy_least_traffic_list_get ( UX_HCD_SYNERGY
* hcd_synergy )
```

8.13.117.1 Brief description

This function return a pointer to the first ED in the periodic tree that has the least traffic registered.

8.13.117.2 Detailed description

Table 531:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 532:Return values

Name	Description
min_bandwidth_ed	End descriptor interrupt Number(ed)

8.13.118 ux_hcd_synergy_periodic_endpoint_destroy

```
ux_hcd_synergy_periodic_endpoint_destroy ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_ENDPOINT  * endpoint )
```

8.13.118.1 Brief description

This function will destroy an isochronous endpoint.

8.13.118.2 Detailed description

Table 533:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
endpoint	inout	: Pointer to an Endpoint control block

Table 534:Return values

Name	Description
UX_SUCCESS	Isochronous endpoint is destroyed successfully.
UX_ENDPOINT_HANDLE_UNKNOWN	Physical endpoint has not been initialized properly.

8.13.119 ux_hcd_synergy_periodic_schedule

```
ux_hcd_synergy_periodic_schedule ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.119.1 Brief description

This function schedules new transfers from the periodic interrupt list.

8.13.119.2 Detailed description

Table 535:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

8.13.120 ux_hcd_synergy_periodic_tree_create

```
ux_hcd_synergy_periodic_tree_create ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.120.1 Brief description

This function creates the periodic static tree for the interrupt and isochronous eds.

8.13.120.2 Detailed description

Table 536:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 537:Return values

Name	Description
UX_SUCCESS	Periodic tree created successfully.
UX_NO_ED_AVAILABLE	Failed to obtain an ED.

8.13.121 ux_hcd_synergy_port_disable

```
ux_hcd_synergy_port_disable ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.121.1 Brief description

This function will disable a specific port attached to the root HUB.

8.13.121.2 Detailed description

Table 538:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 539:Return values

Name	Description
UX_SUCCESS	Port disabled successfully.

Table 539:Return values (Continued)

Name	Description
UX_PORT_INDEX_UNKNOWN	Invalid port .

8.13.122 ux_hcd_synergy_port_enable

```
ux_hcd_synergy_port_enable ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.122.1 Brief description

This function will enable a specific port attached to the root HUB.

8.13.122.2 Detailed description

Table 540:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
port_index	in	: Port Index

Table 541:Return values

Name	Description
UX_SUCCESS	Port enabled successfully.
UX_PORT_INDEX_UNKNOWN	Invalid port.
UX_NO_DEVICE_CONNECTED	Device not connected properly.

8.13.123 ux_hcd_synergy_port_reset

```
ux_hcd_synergy_port_reset ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.123.1 Brief description

This function will reset a specific port attached to the root HUB.

8.13.123.2 Detailed description

Table 542:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 543:Return values

Name	Description
UX_SUCCESS	Port reset successfully.
UX_PORT_INDEX_UNKNOWN	Invalid port.
UX_NO_DEVICE_CONNECTED	Device not connected properly.

8.13.124 ux_hcd_synergy_port_resume

```
ux_hcd_synergy_port_resume ( UX_HCD_SYNERGY  * hcd_synergy ,
 UINT port_index )
```

8.13.124.1 Brief description

This function will resume a specific port attached to the root HUB. Present, this function is not supported for resume port.

8.13.124.2 Detailed description

Table 544:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 545:Return values

Name	Description
UX_FUNCTION_NOT_SUPPORTED	Unsupported function.

8.13.125 ux_hcd_synergy_port_status_get

```
ux_hcd_synergy_port_status_get ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.125.1 Brief description

This function will return the status for each port attached to the root HUB.

8.13.125.2 Detailed description

Table 546:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
port_index	in	: Port Index

Table 547:Return values

Name	Description
UX_PORT_INDEX_UNKNOWN	Invalid port.
port_status	Synergy Port status

8.13.126 ux_hcd_synergy_port_suspend

```
ux_hcd_synergy_port_suspend ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.126.1 Brief description

This function will suspend a specific port attached to the root HUB. Present, this function is does not supported.

8.13.126.2 Detailed description

Table 548:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 549:Return values

Name	Description
UX_FUNCTION_NOT_SUPPORTED	Unsupported function.

8.13.127 ux_hcd_synergy_power_down_port

```
ux_hcd_synergy_power_down_port ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.127.1 Brief description

This function will power down a specific port attached to the root HUB. Present, this function is does not supported.

8.13.127.2 Detailed description

Table 550:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 551:Return values

Name	Description
UX_FUNCTION_NOT_SUPPORTED	Unsupported function.

8.13.128 ux_hcd_synergy_power_on_port

```
ux_hcd_synergy_power_on_port ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG port_index )
```

8.13.128.1 Brief description

This function will power a specific port attached to the root HUB. Present, this function is does not supported.

8.13.128.2 Detailed description

Table 552:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
port_index	in	: Port Index

Table 553:Return values

Name	Description
UX_FUNCTION_NOT_SUPPORTED	Unsupported function.

8.13.129 ux_hcd_synergy_power_root_hubs

```
ux_hcd_synergy_power_root_hubs ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.129.1 Brief description

This function will power the root HUB. Present, this function is does not supported.

8.13.129.2 Detailed description

Table 554:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block

8.13.130 ux_hcd_synergy_register_clear

```
ux_hcd_synergy_register_clear ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.130.1 Brief description

This function clears flags in a synergy USB register.

8.13.130.2 Detailed description

Table 555:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
synergy_register	in	: Register to write
value	in	: Value to clear

8.13.131 ux_hcd_synergy_register_read

```
ux_hcd_synergy_register_read ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG synergy_register )
```

8.13.131.1 Brief description

This function reads a data from synergy USB register.

8.13.131.2 Detailed description

Table 556:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
synergy_register	in	: Register to read

Table 557:Return values

Name	Description
hcd_reg	Value read from the specified register(ULONG value).

8.13.132 ux_hcd_synergy_register_set

```
ux_hcd_synergy_register_set ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.132.1 Brief description

This function sets flags in a synergy USB register.

8.13.132.2 Detailed description

Table 558:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
synergy_register	in	: Register to read
value	in	: Value to be set

8.13.133 ux_hcd_synergy_register_write

```
ux_hcd_synergy_register_write ( UX_HCD_SYNERGY  * hcd_synergy ,
 ULONG synergy_register ,   USHORT value )
```

8.13.133.1 Brief description

This function writes a data to a Synergy USB register.

8.13.133.2 Detailed description

Table 559:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block
synergy_register	in	: Register to write
value	in	: Value to write

8.13.134 ux_hcd_synergy_regular_td_obtain

```
UX_SYNERGY_TD ux_hcd_synergy_regular_td_obtain ( UX_HCD_SYNERGY  * hcd_synergy )
```

8.13.134.1 Brief description

This function obtains a free TD from the regular TD list.

8.13.134.2 Detailed description

Table 560:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 561:Return values

Name	Description
td	A pointer to Synergy TD.
UX_NULL	Null pointer.

8.13.135 ux_hcd_synergy_request_bulk_transfer

```
ux_hcd_synergy_request_bulk_transfer ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_TRANSFER  * transfer_request )
```

8.13.135.1 Brief description

This function performs a bulk transfer request. A bulk transfer can be larger than the size of the Synergy buffer so it may be required to chain multiple tds to accommodate this transfer request. A bulk transfer is non blocking, so we return before the transfer request is completed.

8.13.135.2 Detailed description

Table 562:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
transfer_request	inout	: Pointer to transfer request

Table 563:Return values

Name	Description
UX_SUCCESS	Bulk transfer happened successfully.
UX_NO_TD_AVAILABLE	Unavailable New TD.

8.13.136 ux_hcd_synergy_request_control_transfer

```
ux_hcd_synergy_request_control_transfer ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_TRANSFER  * transfer_request )
```

8.13.136.1 Brief description

This function performs a control transfer from a transfer request. The USB control transfer is in 3 phases (setup, data, status). This function will chain all phases of the control sequence before setting the Synergy endpoint as a candidate for transfer.

8.13.136.2 Detailed description

Table 564:Parameters

Name	Direction	Description
hcd_synergy	inout	: Pointer to a HCD control block

Table 564:Parameters (Continued)

Name	Direction	Description
transfer_request	inout	: Pointer to transfer request

Table 565:Return values

Name	Description
UX_MEMORY_INSUFFICIENT	Insufficient memory to build the SETUP request.
UX_NO_TD_AVAILABLE	Unavailable New TD.
ux_transfer_request_completion_code	Pointer to USBX transfer request structure(request completion code).

8.13.137 ux_hcd_synergy_request_interrupt_transfer

```
ux_hcd_synergy_request_interrupt_transfer ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_TRANSFER  * transfer_request )
```

8.13.137.1 Brief description

This function performs an interrupt transfer request. An interrupt transfer can only be as large as the MaxpacketField in the endpoint descriptor. This was verified at the USB layer and does not need to be reverified here.

8.13.137.2 Detailed description

Table 566:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
transfer_request	inout	: Pointer to transfer request

Table 567:Return values

Name	Description
UX_SUCCESS	Interrupt transfer request processed successfully.

Table 567:Return values (Continued)

Name	Description
UX_NO_TD_AVAILABLE	Unavailable new TD.

8.13.138 ux_hcd_synergy_request_isochronous_transfer

```
ux_hcd_synergy_request_isochronous_transfer ( UX_HCD_SYNERGY  * hcd_synergy ,
UX_TRANSFER  * transfer_request )
```

8.13.138.1 Brief description

This function performs an isochronous transfer request.

8.13.138.2 Detailed description

Table 568:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
transfer_request	inout	: Pointer to transfer request

Table 569:Return values

Name	Description
UX_SUCCESS	Isochronous transfer request processed successfully.
UX_NO_TD_AVAILABLE	Unavailable new TD.

8.13.139 ux_hcd_synergy_request_transfer

```
ux_hcd_synergy_request_transfer ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_TRANSFER
* transfer_request )
```

8.13.139.1 Brief description

This function is the handler for all the transactions on the USB. The transfer request passed as parameter contains the endpoint and the device descriptors in addition to the type of transaction de be executed. This function routes the transfer request to according to the type of transfer to be executed.

8.13.139.2 Detailed description

Table 570:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
transfer_request	inout	: Pointer to transfer request

Table 571:Return values

Name	Description
UX_ERROR	Error while Isolating the endpoint type and routing the transfer request.
UX_NO_DEVICE_CONNECTED	No device attached.

See Common Error Codes for other possible return codes or causes. This function calls:

- ux_hcd_synergy_request_control_transfer
- ux_hcd_synergy_request_bulk_transfer
- ux_hcd_synergy_request_interrupt_transfer
- ux_hcd_synergy_request_isochronous_transfer

8.13.140 ux_hcd_synergy_td_add

```
ux_hcd_synergy_td_add ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_SYNERGY_ED  * ed )
```

8.13.140.1 Brief description

This function add new TD for control, Bulk or Interrupt endpoint.

8.13.140.2 Detailed description

Table 572:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block

Table 572:Parameters (Continued)

Name	Direction	Description
ed	in	: Pointer to Synergy ED structure

Table 573:Return values

Name	Description
UX_SUCCESS	Transfer done successfully.

8.13.141 ux_hcd_synergy_transfer_abort

```
 ux_hcd_synergy_transfer_abort ( UX_HCD_SYNERGY  * hcd_synergy ,  UX_TRANSFER
* transfer_request )
```

8.13.141.1 Brief description

This function will abort transactions attached to a transfer request.

8.13.141.2 Detailed description

Table 574:Parameters

Name	Direction	Description
hcd_synergy	in	: Pointer to a HCD control block
transfer_request	inout	: Pointer to transfer request

Table 575:Return values

Name	Description
UX_SUCCESS	Transactions attached to transfer request are aborted successfully.
UX_ENDPOINT_HANDLE_UNKNOWN	Endpoint is not initialized properly.

8.13.142 API Data

8.13.142.1 SCHAR

```
typedef signed char SCHAR
```

8.13.142.2 SLONG

```
typedef long SLONG
```

8.14 USB Communication Framework

RTOS-integrated USBX CDC ACM device implementation.

8.14.1 Functions

- SF_EL_UX_COMMS_Open
- SF_EL_UX_COMMS_Close
- SF_EL_UX_COMMS_Read
- SF_EL_UX_COMMS_Write
- SF_EL_UX_COMMS_Lock
- SF_EL_UX_COMMS_Unlock
- SF_EL_UX_COMMS_VersionGet
- ux_cdc_instance_activate
- ux_cdc_instance_deactivate
- sf_el_ux_comms_read_leftover

8.14.2 Defines

- #define SF_EL_UX_COMMS_CODE_VERSION_MAJOR
  Initial value:`(1U)`
  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_EL_UX_COMMS_CODE_VERSION_MINOR
  Initial value:`(5U)`

8.14.3 SF_EL_UX_COMMS_Open

```
ssp_err_t SF_EL_UX_COMMS_Open ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_cfg_t  const *const p_cfg )
```

8.14.3.1 Brief description

Initializes a USB channel for CDC ACM mode.

8.14.3.2 Detailed description

Table 576:Return values

Name	Description
SSP_SUCCESS	Channel opened successfully
SSP_ERR_ASSERTION	Pointer to sf_el_ux_comms control block is NULL
SSP_ERR_INTERNAL	USBX initialize call returned an error.

NOTE: This function is reentrant.

8.14.3.3 Function steps

- Create the mutexes.
- Mark control block open by intializing it to "UXCM" in its ASCII equivalent .

8.14.4 SF_EL_UX_COMMS_Close

```
ssp_err_t SF_EL_UX_COMMS_Close ( sf_comms_ctrl_t  *const p_api_ctrl )
```

8.14.4.1 Detailed description

Finalize CDC ACM operation for a USB channel. and clear SCI device control block members.

Table 577:Return values

Name	Description
SSP_SUCCESS	Channel successfully closed
SSP_ERR_ASSERTION	Pointer to control block is NULL

Table 577:Return values (Continued)

Name	Description
SSP_ERR_INTERNAL	Deletion of mutex failed

NOTE: This function is reentrant.

8.14.4.2 Function steps

- Delete transmit mutex.
- Delete receive mutex.

8.14.5 SF_EL_UX_COMMS_Read

```
ssp_err_t SF_EL_UX_COMMS_Read ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   UINT const timeout )
```

8.14.5.1 Brief description

Read data from the USBX CDC ACM driver.

8.14.5.2 Detailed description

Table 578:Return values

Name	Description
SSP_SUCCESS	Data reception ends successfully.
SSP_ERR_TIMEOUT	Wait for the USBX instance to be created until the timeout occurs.
SSP_ERR_INTERNAL	USB device not enumerated or USBX read call returned an error.

NOTE: This API is reentrant.

8.14.5.3 Function steps

- Read data leftover from the last packet.
- Read from the CDC class.
- Buffer overflow occurred.

8.14.6 SF_EL_UX_COMMS_Write

```
ssp_err_t SF_EL_UX_COMMS_Write ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
const *const p_src ,   uint32_t const bytes ,   UINT const timeout )
```

8.14.6.1 Brief description

Write data to the USBX CDC ACM driver.

8.14.6.2 Detailed description

Table 579:Return values

Name	Description
SSP_SUCCESS	Data transmission finished successfully.
SSP_ERR_TIMEOUT	Wait for the USBX instance to be created until the timeout occurs.
SSP_ERR_ASSERTION	Pointer to sf_el_ux_comms control block is NULL.
SSP_ERR_INTERNAL	USB device not enumerated or USBX write call returned an error.

NOTE: This function is reentrant.

8.14.6.3 Function steps

- Wait for the USB to be plugged in.

8.14.7 SF_EL_UX_COMMS_Lock

```
ssp_err_t SF_EL_UX_COMMS_Lock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type ,   UINT timeout )
```

8.14.7.1 Brief description

Locks the communication to a thread by acquiring a mutex before the timeout occurs.

8.14.7.2 Detailed description

Table 580:Return values

Name	Description
SSP_SUCCESS	Locking sf_el_ux_comms resource successful.
SSP_ERR_ASSERTION	Pointer to sf_el_ux_comms control block is NULL.
SSP_ERR_TIMEOUT	Mutex not available in timeout.
SSP_ERR_NOT_OPEN	The framework is not opened. Open the framework first before calling this API.

8.14.8 SF_EL_UX_COMMS_Unlock

```
ssp_err_t SF_EL_UX_COMMS_Unlock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type )
```

8.14.8.1 Brief description

Unlocks the communication to a thread by releasing the mutex for write or read or both.

8.14.8.2 Detailed description

Table 581:Return values

Name	Description
SSP_SUCCESS	Unlocks the sf_el_ux_comms communication resource successful.
SSP_ERR_ASSERTION	Pointer to sf_el_ux_comms control block is NULL.
SSP_ERR_NOT_OPEN	The framework is not opened. Open the framework first before calling this API.

8.14.9 SF_EL_UX_COMMS_VersionGet

```
ssp_err_t SF_EL_UX_COMMS_VersionGet ( ssp_version_t  *const p_version )
```

8.14.9.1 Detailed description

Get driver version

Table 582:Parameters

Name	Direction	Description
p_version	out	Version will be stored here.

Table 583:Return values

Name	Description
SSP_SUCCESS	Version stored in provided pointer.
SSP_ERR_ASSERTION	p_version was null.

NOTE: This function is reentrant.

8.14.10 ux_cdc_instance_activate

```
ux_cdc_instance_activate (  VOID * cdc_instance )
```

8.14.11 ux_cdc_instance_deactivate

```
ux_cdc_instance_deactivate (  VOID * cdc_instance )
```

8.14.12 sf_el_ux_comms_read_leftover

```
ssp_err_t sf_el_ux_comms_read_leftover ( sf_el_ux_comms_instance_ctrl_t
* p_ux_comms_ctrl ,   uint8_t *const p_destination ,   uint32_t
* p_remaining_bytes ,   UINT time )
```

8.14.12.1 Brief description

Subroutine for SF_EL_UX_COMMS_Read API which will be called to read the leftover data from last packet.

8.14.12.2 Detailed description

Table 584:Parameters

Name	Direction	Description
p_ux_comms_ctrl	in	Pointer to the sf_el_ux_comms control block.
p_destination	out	Pointer to the destination buffer.
p_remaining_bytes	inout	Pointer to remaining bytes.
time	in	User specified timeout value.

Table 585:Return values

Name	Description
SSP_ERR_TIMEOUT	Wait for the USBX instance to be created until the timeout occurs.
SSP_SUCCESS	Read the leftover data from previous packet successful.

8.14.12.3 Function steps

- Wait for the USB to be plugged in.
- If there is data leftover from the last packet, use it.

8.14.13 Extensions

8.14.13.1 sf_el_ux_comms_instance_ctrl_t

sf_el_ux_comms_instance_ctrl_t

Detailed description

USBX CDC ACM device communications instance control structure. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open
- TX_MUTEX mutex[2]
- UX_SLAVE_CLASS_CDC_ACM * p_cdc

- uint32_t leftover_length
- uint32_t index
- uint8_t memory[SF_EL_UX_COMMS_CFG_USB_MEMORY_SIZE_BYTES]
- uint8_t rx_memory[SF_EL_UX_COMMS_CFG_BUFFER_MAX_LENGTH]
- TX_SEMAPHORE semaphore

8.14.14 Modules

- Build Time Configurations

8.14.14.1 Build Time Configurations

Defines

- #define SF_EL_UX_COMMS_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking
- #define SF_EL_UX_COMMS_CFG_USB_MEMORY_SIZE_BYTES

  Initial value:`(65536)`

  Specify the size of memory to allocate for USBX. Size of 65536 is recommended for device applications.
- #define SF_EL_UX_COMMS_CFG_BUFFER_MAX_LENGTH

  Initial value:`(128)`

  Specify the size of memory to allocate for a read input buffer.

8.15 USB Communication Framework V2

RTOS-integrated USBX CDC ACM device implementation.

8.15.1 Functions

- SF_EL_UX_COMMS_Open
- SF_EL_UX_COMMS_Close
- SF_EL_UX_COMMS_Read
- SF_EL_UX_COMMS_Write
- SF_EL_UX_COMMS_Lock

- SF_EL_UX_COMMS_Unlock
- SF_EL_UX_COMMS_VersionGet

8.15.2 Defines

- #define SF_EL_UX_COMMS_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_EL_UX_COMMS_CODE_VERSION_MINOR

  Initial value:`(5U)`

8.15.3 SF_EL_UX_COMMS_Open

```
ssp_err_t SF_EL_UX_COMMS_Open ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_cfg_t  const *const p_cfg )
```

8.15.3.1 Brief description

Initializes a USB channel for CDC ACM mode.

8.15.3.2 Detailed description

Table 586:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to control structure block.
p_cfg	in	Pointer to configuration structure block. This parameter is not used in the framework hence the NULL parameter check not implemented.

Table 587:Return values

Name	Description
SSP_SUCCESS	Channel opened successfully
SSP_ERR_ASSERTION	p_api_ctrl pointer parameter to control block is NULL

Table 587:Return values (Continued)

Name	Description
SSP_ERR_TIMEOUT	Mutex not available in timeout.
SSP_ERR_IN_USE	Channel/peripheral is running/busy.

NOTE: This function is reentrant.

8.15.3.3 Function steps

- Suspend here until a USBX CDC instance is created by the USBX CDC for this module.
- Create the mutexes.
- Mark control block open.

8.15.4 SF_EL_UX_COMMS_Close

```
ssp_err_t SF_EL_UX_COMMS_Close ( sf_comms_ctrl_t  *const p_api_ctrl )
```

8.15.4.1 Brief description

Releases all the ThreadX Resources.

8.15.4.2 Detailed description

Table 588:Return values

Name	Description
SSP_SUCCESS	Channel successfully closed
SSP_ERR_ASSERTION	Pointer to control block is NULL
SSP_ERR_INTERNAL	Deletion of mutex failed

NOTE: This function is reentrant. No actual USB COM port close operation being held.

8.15.4.3 Function steps

- Delete transmit mutex.
- Delete receive mutex.

8.15.5 SF_EL_UX_COMMS_Read

```
ssp_err_t SF_EL_UX_COMMS_Read ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   UINT const timeout )
```

8.15.5.1 Brief description

Read data from the USBX CDC-ACM driver.

8.15.5.2 Detailed description

Read data from the USBX CDC ACM driver.

Table 589:Return values

Name	Description
SSP_SUCCESS	Data reception ends successfully.
SSP_ERR_INTERNAL	USB device not enumerated or USBX read call returned an error.
SSP_ERR_ASSERTION	Either of pointer type argument is NULL.
SSP_ERR_NOT_OPEN	Module is not opened.

NOTE: This API is reentrant.

8.15.5.3 Function steps

- Check if module has been opened.
- Check for the USB to be plugged in.
- If there is data leftover from the last packet, use it.
- Read from the CDC class.
- Buffer overflow occurred.

8.15.6 SF_EL_UX_COMMS_Write

```
ssp_err_t SF_EL_UX_COMMS_Write ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
const *const p_src ,   uint32_t const bytes ,   UINT const timeout )
```

8.15.6.1 Brief description

Write data to the USBX CDC-ACM framework.

8.15.6.2 Detailed description

Write data to the USBX CDC ACM driver.

Table 590:Return values

Name	Description
SSP_SUCCESS	Data transmission finished successfully.
SSP_ERR_ASSERTION	Pointer to control block is NULL
SSP_ERR_NOT_OPEN	Module is not opened.
SSP_ERR_INTERNAL	USB device not enumerated or USBX write call returned an error.

NOTE: This function is reentrant.

8.15.6.3 Function steps

- Check if module has been opened.
- Check for the USB to be plugged in.

8.15.7 SF_EL_UX_COMMS_Lock

```
ssp_err_t SF_EL_UX_COMMS_Lock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type ,   UINT timeout )
```

8.15.7.1 Brief description

Lock the USB COM resource.

8.15.7.2 Detailed description

Locks the communication to a thread by acquiring a mutex before the timeout occurs.

Table 591:Return values

Name	Description
SSP_SUCCESS	Locking a USB COM resource successful.
SSP_ERR_ASSERTION	Pointer to control block is NULL.

Table 591:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Module is not opened.
SSP_ERR_TIMEOUT	Mutex not available in timeout.

8.15.8 SF_EL_UX_COMMS_Unlock

```
ssp_err_t SF_EL_UX_COMMS_Unlock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type )
```

8.15.8.1 Brief description

Unlock the USB COM resource.

8.15.8.2 Detailed description

Unlocks the communication to a thread by releasing the mutex for write or read or both.

Table 592:Return values

Name	Description
SSP_SUCCESS	Unlocking a USB COM resource successful.
SSP_ERR_ASSERTION	Pointer to control block is NULL.
SSP_ERR_NOT_OPEN	Module is not opened.
SSP_ERR_TIMEOUT	An internal ThreadX error has occurred. This is typically a failure to use a mutex.

8.15.9 SF_EL_UX_COMMS_VersionGet

```
ssp_err_t SF_EL_UX_COMMS_VersionGet ( ssp_version_t  *const p_version )
```

8.15.9.1 Brief description

Get driver version.

8.15.9.2 Detailed description

Table 593:Parameters

Name	Direction	Description
p_version	out	Version will be stored here.

NOTE: This function is reentrant.

8.15.10 Extensions

8.15.10.1 sf_el_ux_comms_instance_ctrl_t

sf_el_ux_comms_instance_ctrl_t

Detailed description

USBX CDC ACM device communications instance control structure. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open
- TX_MUTEX mutex[2]
- UX_SLAVE_CLASS_CDC_ACM * p_cdc
- uint32_t leftover_length
- uint32_t index
- uint8_t memory[SF_EL_UX_COMMS_CFG_USB_MEMORY_SIZE_BYTES]
- uint8_t rx_memory[SF_EL_UX_COMMS_CFG_BUFFER_MAX_LENGTH]
- TX_SEMAPHORE semaphore

8.16 External IRQ Framework

RTOS-integrated external IRQ Framework.

8.16.1 Summary

This module is a ThreadX-aware external IRQ Framework for external inputs such as switches or other binary signals.

8.16.2 Functions

- SF_EXTERNAL_IRQ_Open
- SF_EXTERNAL_IRQ_Wait
- SF_EXTERNAL_IRQ_VersionGet
- SF_EXTERNAL_IRQ_Close

8.16.3 Defines

- #define SF_EXTERNAL_IRQ_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_EXTERNAL_IRQ_CODE_VERSION_MINOR

  Initial value:`(5U)`

8.16.4 SF_EXTERNAL_IRQ_Open

```
ssp_err_t SF_EXTERNAL_IRQ_Open ( sf_external_irq_ctrl_t  *const p_api_ctrl ,
 sf_external_irq_cfg_t  const *const p_cfg )
```

8.16.4.1 Brief description

Configure external IRQ and optionally enable external IRQ callbacks. Implements open.

8.16.4.2 Detailed description

The SF_EXTERNAL_IRQ_Open function acquires a mutex for the external IRQ channel used, then calls the HAL driver open function. After successful initialization, the external IRQ is ready for use.

Table 594:Return values

Name	Description
SSP_SUCCESS	Initialization was successful and external interrupt has started.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg, p_api, or p_api->open. See HAL driver for other possible causes.
SSP_ERR_IN_USE	This channel is already open.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls:

- open

NOTE: This function is reentrant for any channel.

8.16.4.3 Function steps

- Save driver structure for use in other framework layer functions
- Create semaphore for use with wait function
- Get mutex before calling lower layer, which will access HW registers.
- Prepare configuration for lower layer
- Open lower layer
- Post the mutex
- If low level initialization failed, delete the mutex and exit the function with the error code
- Delete RTOS services used and log the error
- log the error and return the error
- If there was any mutex error, return it
- Mark control block open so other tasks know it is valid

8.16.5 SF_EXTERNAL_IRQ_Wait

```
ssp_err_t SF_EXTERNAL_IRQ_Wait ( sf_external_irq_ctrl_t  *const p_api_ctrl ,
  ULONG const timeout )
```

8.16.5.1 Brief description

Get mutex, wait for external interrupt to expire, and release mutex. Implements wait.

8.16.5.2 Detailed description

Table 595:Return values

Name	Description
SSP_SUCCESS	External interrupt stopped successfully.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_EXTERNAL_IRQ_Open to configure.
SSP_ERR_TIMEOUT	Time out happens while waiting a semaphore.

Table 595:Return values (Continued)

Name	Description
SSP_ERR_WAIT_ABORTED	Suspension was aborted by another thread.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.16.5.3 Function steps

- Wait for semaphore post from ISR

8.16.6 SF_EXTERNAL_IRQ_VersionGet

```
ssp_err_t SF_EXTERNAL_IRQ_VersionGet ( ssp_version_t  *const p_version )
```

8.16.6.1 Brief description

Get version and store it in provided pointer p_version. Implements versionGet.

8.16.6.2 Detailed description

Table 596:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.16.7 SF_EXTERNAL_IRQ_Close

```
ssp_err_t SF_EXTERNAL_IRQ_Close ( sf_external_irq_ctrl_t  *const p_api_ctrl )
```

8.16.7.1 Brief description

Release channel mutex and close channel at HAL layer. Implements close.

8.16.7.2 Detailed description

Table 597:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter ctrl is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_IN_USE	This channel is already open.
SSP_ERR_UNSUPPORTED	Unsupported operation.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls:

- close

8.16.7.3 Function steps

- Get mutex since this will access hardware registers
- Close channel in HAL layer
- Post the mutex
- Clear information from control block so other functions know this block is closed
- Clear information from control block so other functions know this instance is closed
- Delete RTOS services used

8.16.8 Extensions

8.16.8.1 sf_external_irq_instance_ctrl_t

sf_external_irq_instance_ctrl_t

Detailed description

Instance control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open

  Used by driver to check if control block is valid.

- TX_MUTEX mutex

  Mutex used to protect access to lower level driver hardware registers.
- TX_SEMAPHORE semaphore

  Semaphore used for SF_EXTERNAL_IRQ_Wait.
- external_irq_instance_t const * p_lower_lvl_irq

  Pointer to lower level driver instance.
- bool callback_used

  Used by driver to check if wait can be used.

8.17 I2C Framework

RTOS-integrated I2C Framework.

SSP I2C framework driver API .

8.17.1 Summary

This is a ThreadX-aware I2C driver API. The API implements the I2C Framework interface and can access several hardware peripherals at the HAL layer. The connection to the HAL layer is established by passing in a driver structure in SF_I2C_Open.

8.17.2 Interface Used

I2C Framework

8.17.3 Functions

- sf_i2c_callback
- sf_i2c_common_start
- sf_i2c_common_wait
- sf_i2c_common_finish
- sf_i2c_check_lower_lvl_driver_parameters
- sf_i2c_check_common_parameters
- sf_i2c_reconfigure_device
- SF_I2C_Open
- SF_I2C_Read
- SF_I2C_Write
- SF_I2C_Reset

- SF_I2C_Close
- SF_I2C_Lock
- SF_I2C_Unlock
- SF_I2C_VersionGet

8.17.4 Variables

- g_sf_i2c_version
- g_sf_i2c_on_sf_i2c

8.17.5 Defines

- #define SF_I2C_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_I2C_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_I2C_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((expression), (error), &g_module_name[0], &g_sf_i2c_version)`

8.17.6 sf_i2c_callback

```
sf_i2c_callback ( i2c_callback_args_t  * parg )
```

8.17.6.1 Brief description

I2C SSP framework level callback.

8.17.6.2 Detailed description

Table 598:Parameters

Name	Direction	Description
p_args	in	Pointer to callback parameters

Table 599:Return values

Name	Description
void	

8.17.7 sf_i2c_common_start

```
ssp_err_t sf_i2c_common_start ( sf_i2c_instance_ctrl_t  *const p_ctrl ,
  uint32_t const timeout )
```

8.17.7.1 Brief description

Common I2C transfer start function. Used in all framework read write calls. This function checks is there any need for reconfiguration and gets the mutex and reconfigures if required.

8.17.7.2 Detailed description

(end addtogroup SF_I2C)

Table 600:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for I2C framework driver context for a device.
timeout	in	ThreadX timeout.

Table 601:Return values

Name	Description
SSP_SUCCESS	Transfer started successfully.
SSP_ERR_INTERNAL	Internal error occurred.

NOTE: This function is reentrant for any device.

8.17.7.3 Function steps

- Reconfigure the device address, if necessary

8.17.8 sf_i2c_common_wait

```
ssp_err_t sf_i2c_common_wait ( sf_i2c_instance_ctrl_t  *const p_ctrl ,
 i2c_event_t  event ,   uint32_t const timeout )
```

8.17.8.1 Brief description

Common I2C wait. Waits for an operation to finish.

8.17.8.2 Detailed description

Table 602:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for I2C framework driver context for a device
event	in	Event
timeout	in	ThreadX timeout

Table 603:Return values

Name	Description
SSP_SUCCESS	Data transmitted successfully.
SSP_ERR_TRANSFER_ABORTED	Transfer aborted.
SSP_ERR_INTERNAL	Internal error occurred.

NOTE: This function is reentrant for any device.

8.17.9 sf_i2c_common_finish

```
ssp_err_t sf_i2c_common_finish ( sf_i2c_instance_ctrl_t  *const p_ctrl )
```

8.17.9.1 Brief description

Common I2C finish. Release mutex if the bus is not locked and restart not issued..

8.17.9.2 Detailed description

Table 604:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for I2C framework driver context for a device

Table 605:Return values

Name	Description
SSP_SUCCESS	Transfer finished successfully.
SSP_ERR_INTERNAL	Internal error occurred.

NOTE: This function is reentrant for any device.

8.17.10 sf_i2c_check_lower_lvl_driver_parameters

```
sf_i2c_check_lower_lvl_driver_parameters ( sf_i2c_cfg_t  const *const p_cfg )
```

8.17.10.1 Brief description

Checks whether lower level I2C module and bus are configured.

8.17.10.2 Detailed description

Table 606:Parameters

Name	Direction	Description
p_cfg	in	Pointer to I2C framework Configuration Structure

Table 607:Return values

Name	Description
true	Lower level I2C driver, and I2C bus are configured.

Table 607:Return values (Continued)

Name	Description
false	Lower level I2C driver, and I2C bus are not configured.

8.17.11 sf_i2c_check_common_parameters

```
 sf_i2c_check_common_parameters ( sf_i2c_ctrl_t  *const p_ctrl ,   uint32_t
const bytes )
```

8.17.11.1 Brief description

Check if I2C framework control block address and number of bytes are not NULL.

8.17.11.2 Detailed description

Table 608:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to I2C framework control block
bytes	in	Number of bytes of data to be transferred

Table 609:Return values

Name	Description
true	I2C framework control block address, number of bytes are not NULL.
false	I2C framework control block address, number of bytes are NULL.

8.17.12 sf_i2c_reconfigure_device

```
ssp_err_t sf_i2c_reconfigure_device ( sf_i2c_instance_ctrl_t  *const p_ctrl )
```

8.17.12.1 Brief description

Assign a slave address of new device to current device on the bus.

8.17.12.2 Detailed description

Table 610:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for I2C framework context for a device

Table 611:Return values

Name	Description
SSP_SUCCESS	New device slave address assigned to current device

See Common Error Codes and lower level drivers for other possible return codes. These driver functions are:

- slaveAddressSet

8.17.12.3 Function steps

- Reconfigure the device by changing the slave address.
- Change the slave address and addressing mode.
- Assign this device to current.

8.17.13 SF_I2C_Open

```
ssp_err_t SF_I2C_Open ( sf_i2c_ctrl_t  *const p_api_ctrl ,  sf_i2c_cfg_t
const *const p_cfg )
```

8.17.13.1 Brief description

Initialize a I2C bus and open low level I2C driver.

8.17.13.2 Detailed description

Table 612:Return values

Name	Description
SSP_SUCCESS	I2C device is successfully opened.

Table 612:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the following parameters is NULL: p_api_ctrl, p_cfg, Pointer to Open, Close, Read, Write, or reset API interfaces,p_cfg->p_bus.
SSP_ERR_INTERNAL	Internal error occurred.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is

- open

NOTE: This function is reentrant for any channel.

NOTE: Control handle must be cleared by caller before calling this function.

8.17.13.3 Function steps

- Copy bus to control
- Initialize bus lock to false
- Set framework level callback function.
- Save context for use in ISRs.
- Create mutex for this bus.
- Create Event flag.
- Open low level.
- Save device configuration for reconfiguration.
- Set device state as Opened.
- Initialize restarted flag to false.
- Increment device count.

8.17.14 SF_I2C_Read

```
ssp_err_t SF_I2C_Read ( sf_i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   bool const restart ,   uint32_t
const timeout )
```

8.17.14.1 Brief description

Start the transfer process and receive data from I2C device.

8.17.14.2 Detailed description

Table 613:Return values

Name	Description
SSP_SUCCESS	Data received successfully.
SSP_ERR_NOT_OPEN	Device instance not opened.
SSP_ERR_ASSERTION	One of the following parameters is NULL: p_api_ctrl, p_dest, bytes, timeout.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- read

8.17.14.3 Function steps

- Check whether device is opened or not.
- Start transfer process - check lock, check reconfiguration, get Mutex.
- Get the low level control in use.
- Perform read.
- Wait for callback to set event flag.
- Finish transfer.
- Finish transfer.

8.17.15 SF_I2C_Write

```
ssp_err_t SF_I2C_Write ( sf_i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_src ,   uint32_t const bytes ,   bool const restart ,   uint32_t
const timeout )
```

8.17.15.1 Brief description

Start the transfer process and send data on I2C bus.

8.17.15.2 Detailed description

Table 614:Return values

Name	Description
SSP_SUCCESS	Data written successfully.
SSP_ERR_NOT_OPEN	Device instance not opened.
SSP_ERR_ASSERTION	One of the following parameters is NULL: p_api_ctrl, p_src, bytes.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- write

8.17.15.3 Function steps

- Check whether device is opened or not.
- Start transfer process - check lock, check reconfiguration, get Mutex.
- Get the low level control in use.
- Wait for callback to set event flag.

8.17.16 SF_I2C_Reset

```
ssp_err_t SF_I2C_Reset ( sf_i2c_ctrl_t  *const p_api_ctrl ,   uint32_t
const timeout )
```

8.17.16.1 Brief description

Abort any in-progress transfer.

8.17.16.2 Detailed description

Table 615:Return values

Name	Description
SSP_SUCCESS	Channel was reseted without issue.
SSP_ERR_NOT_OPEN	Device was not even opened.

Table 615:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	In-use error.
SSP_ERR_INTERNAL	Internal error occurred.
SSP_ERR_ASSERTION	Following parameters is NULL: p_api_ctrl.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- reset

8.17.16.3 Function steps

- Check whether device is opened or not.
- Get the low level control in use.
- Get mutex since this will access hardware registers.

8.17.17 SF_I2C_Close

```
ssp_err_t SF_I2C_Close ( sf_i2c_ctrl_t  *const p_api_ctrl )
```

8.17.17.1 Brief description

Close the I2C device designated by the control handle and close the RTOS services used by the bus if no device is connected to the bus.

8.17.17.2 Detailed description

Table 616:Return values

Name	Description
SSP_SUCCESS	Device is successfully closed.
SSP_ERR_IN_USE	In-use error.
SSP_ERR_NOT_OPEN	Device was not even opened.*
SSP_ERR_INTERNAL	Internal error occurred.
SSP_ERR_ASSERTION	Following parameters is NULL: p_api_ctrl.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- close

NOTE: This function is reentrant for any device.

8.17.17.3 Function steps

- Check whether device is opened or not.
- See if the device need to close through close. close only if this is the current device, else it might have closed during reconfiguration, or not opened through the open().
- Get mutex since this will access hardware registers.
- Close low level driver.
- Release mutex.
- Decrement device count.
- Check whether any device is still using the bus.
- Delete RTOS services used by bus.
- Set device to closed state.

8.17.18 SF_I2C_Lock

```
ssp_err_t SF_I2C_Lock ( sf_i2c_ctrl_t  *const p_api_ctrl )
```

8.17.18.1 Brief description

Lock the bus for a device. Once bus is locked by a device it can not be used by other devices.

8.17.18.2 Detailed description

Table 617:Return values

Name	Description
SSP_SUCCESS	I2C channel is successfully locked.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_IN_USE	In-use error.
SSP_ERR_ASSERTION	Following parameters is NULL: p_api_ctrl.

NOTE: This function is reentrant for any device.

8.17.18.3 Function steps

- Check whether device is opened or not.
- Check whether lock is already taken.
- Get the mutex for this device.
- Reconfigure the device address, if necessary
- Set lock flag.

8.17.19 SF_I2C_Unlock

```
ssp_err_t SF_I2C_Unlock ( sf_i2c_ctrl_t  *const p_api_ctrl )
```

8.17.19.1 Brief description

Unlock the locked bus and make the bus usable for other devices.

8.17.19.2 Detailed description

Table 618:Return values

Name	Description
SSP_SUCCESS	I2C bus is successfully unlocked.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_IN_USE	In-use error.
SSP_ERR_ASSERTION	Following parameters is NULL: p_api_ctrl.

NOTE: This function is reentrant for any device.

8.17.19.3 Function steps

- Check whether device is opened or not.
- Clear lock flag.
- Release the mutex so that others can use the bus.

8.17.20 SF_I2C_VersionGet

```
ssp_err_t SF_I2C_VersionGet ( ssp_version_t  *const p_version )
```

8.17.20.1 Brief description

Get the version information of the framework.

8.17.20.2 Detailed description

Table 619:Return values

Name	Description
SSP_SUCCESS	Got version number successfully.
SSP_ERR_ASSERTION	Following parameters is NULL: p_version.

8.17.20.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.

8.17.21 g_sf_i2c_version

`ssp_version_t::g_sf_i2c_version`

8.17.21.1 Detailed description

Version data structure used by error logger macro.

8.17.21.2 Initialized as

```
g_sf_i2c_version=
{
.api_version_minor  = SF_I2C_API_VERSION_MINOR,
    .api_version_major  = SF_I2C_API_VERSION_MAJOR,
.code_version_minor = SF_I2C_CODE_VERSION_MINOR,
    .code_version_major = SF_I2C_CODE_VERSION_MAJOR,
}
```

8.17.22 g_sf_i2c_on_sf_i2c

`sf_i2c_api_t::g_sf_i2c_on_sf_i2c`

8.17.22.1 Initialized as

```
g_sf_i2c_on_sf_i2c=
{
```

```
    .open    = SF_I2C_Open,
    .read    = SF_I2C_Read,
    .write   = SF_I2C_Write,
    .reset   = SF_I2C_Reset,
    .close   = SF_I2C_Close,
    .lock    = SF_I2C_Lock,
    .unlock  = SF_I2C_Unlock,
    .version = SF_I2C_VersionGet
}
```

8.17.23 Extensions

8.17.23.1 sf_i2c_instance_ctrl_t

sf_i2c_instance_ctrl_t

Detailed description

I2C instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- sf_i2c_bus_t * p_bus

  Bus using this device. Copy from configuration structure.
- i2c_master_instance_t const * p_lower_lvl_i2c

  I2C instance.
- i2c_cfg_t lower_lvl_cfg

  Used to reconfigure I2C driver.
- i2c_ctrl_t * p_lower_lvl_ctrl

  I2C peripheral control block.
- sf_i2c_dev_state_t dev_state

  Device status.
- bool locked

  Lock and unlock bus for a device.
- bool restarted

  Indicates whether device issued a restart.

8.18 JPEG Framework

RTOS-integrated JPEG Framework.

8.18.1 Functions

- sf_jpeg_initialize
- sf_jpeg_callback_function
- SF_JPEG_Decode_Open
- SF_JPEG_Decode_InputBufferSet
- SF_JPEG_Decode_LinesDecodedGet
- SF_JPEG_Decode_HorizontalStrideSet
- SF_JPEG_Decode_ImageSubsampleSet
- SF_JPEG_Decode_OutputBufferSet
- SF_JPEG_Decode_Wait
- SF_JPEG_Decode_StatusGet
- SF_JPEG_Decode_PixelFormatGet
- SF_JPEG_Decode_ImageSizeGet
- SF_JPEG_Decode_Close
- SF_JPEG_Decode_VersionGet

8.18.2 Variables

- s_sf_jpeg_version
- p_event_flags
- g_sf_jpeg_decode_on_sf_jpeg_decode

8.18.3 Defines

- #define SF_JPEG_DECODE_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file
- #define SF_JPEG_DECODE_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_JPEG_DECODE_OPEN

  Initial value:`(0x4A504547U)`

  "JPEG" in ASCII, used to identify general JPEG control block

- #define SF_JPEG_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &s_sf_jpeg_version)`

  Macro for error logger.
- #define JPEG_ALL_EVENTS

  Initial value:`((ULONG) 0xFFFFFFFF)`

8.18.4 sf_jpeg_initialize

```
ssp_err_t sf_jpeg_initialize ( sf_jpeg_decode_instance_ctrl_t  *const p_ctrl ,
 sf_jpeg_decode_cfg_t  const *const p_cfg )
```

8.18.4.1 Brief description

Acquires mutex, then handles driver initialization at the HAL layer. This function releases the mutex before returns to the caller.

8.18.4.2 Detailed description

Table 620:Return values

Name	Description
SSP_SUCCESS	JPEG Decode driver is successfully opened.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- open

8.18.5 sf_jpeg_callback_function

```
sf_jpeg_callback_function ( jpeg_decode_callback_args_t  * p_args )
```

8.18.6 SF_JPEG_Decode_Open

```
ssp_err_t SF_JPEG_Decode_Open ( sf_jpeg_decode_ctrl_t  *const p_api_ctrl ,
 sf_jpeg_decode_cfg_t  const *const p_cfg )
```

8.18.6.1 Brief description

Parameter checking and initialize JPEG decode with sf_jpeg_initialize helper function and marking the open flag in control block.

8.18.6.2 Detailed description

Table 621:Return values

Name	Description
SSP_SUCCESS	JPEG Decode framework is successfully opened.
SSP_ERR_ASSERTION	One of the following parameters may be null: p_ctrl or p_cfg.
SSP_ERR_ALREADY_OPEN	JPEG Decode framework is already open.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.18.7 SF_JPEG_Decode_InputBufferSet

```
ssp_err_t SF_JPEG_Decode_InputBufferSet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,   void *const p_buffer ,   uint32_t const buffer_size )
```

8.18.7.1 Brief description

Configures JPEG coded input data.

8.18.7.2 Detailed description

This API configures the decode input buffer address register. After the input buffer address is set, the driver checks whether the output buffer address is set, and verifies that the output buffer size is large enough to hold at least eight output lines of data. If both the input buffer and output buffer are set properly, the driver automatically starts the decode process.

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 622:Return values

Name	Description
SSP_SUCCESS	Decode input buffer is successfully configured.

Table 622:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- inputBufferSet

8.18.7.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Call the HAL driver layer inputBufferSet routine.
- Release mutex

8.18.8 SF_JPEG_Decode_LinesDecodedGet

```
ssp_err_t SF_JPEG_Decode_LinesDecodedGet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,   uint32_t *const p_lines )
```

8.18.8.1 Brief description

Obtain number of lines decoded by the codec.

8.18.8.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 623:Return values

Name	Description
SSP_SUCCESS	Lines decoded value is successfully obtained.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.

Table 623:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- linesDecodedGet

8.18.8.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.9 SF_JPEG_Decode_HorizontalStrideSet

```
ssp_err_t SF_JPEG_Decode_HorizontalStrideSet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,   uint32_t horizontal_stride )
```

8.18.9.1 Brief description

Configure the horizontal stride value.

8.18.9.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 624:Return values

Name	Description
SSP_SUCCESS	Horizontal Stride value is successfully configured.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- horizontalStrideSet

8.18.9.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.10 SF_JPEG_Decode_ImageSubsampleSet

```
ssp_err_t SF_JPEG_Decode_ImageSubsampleSet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,  jpeg_decode_subsample_t  horizontal_subsample ,
 jpeg_decode_subsample_t  vertical_subsample )
```

8.18.10.1 Brief description

Configure the horizontal and vertical subsample values. This allows an application to reduce the size of the decoded image at runtime.

8.18.10.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 625:Return values

Name	Description
SSP_SUCCESS	Image subsample values are successfully configured.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- imageSubsampleSet

8.18.10.3 Function steps

- Obtain mutex before making HAL-level driver call.

- Release mutex

8.18.11 SF_JPEG_Decode_OutputBufferSet

```
ssp_err_t SF_JPEG_Decode_OutputBufferSet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,   void * p_buffer ,   uint32_t buffer_size )
```

8.18.11.1 Brief description

Configure the decode output buffer.

8.18.11.2 Detailed description

This API configures the decode output buffer address register. After the output buffer address is set, the driver computers the number of output lines the buffer is able to hold. The hardware requires the number out output lines to decode at a time is multiple of eight. If both the input buffer and output buffer are set properly, the driver automatically starts the decode process.

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 626:Return values

Name	Description
SSP_SUCCESS	Output buffer is successfully configured.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- outputBufferSet

8.18.11.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.12 SF_JPEG_Decode_Wait

```
ssp_err_t SF_JPEG_Decode_Wait ( sf_jpeg_decode_ctrl_t  *const p_api_ctrl ,
 jpeg_decode_status_t  *const p_status ,   uint32_t timeout )
```

8.18.12.1 Brief description

Wait for current JPEG codec operation to finish.

8.18.12.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 627:Return values

Name	Description
SSP_SUCCESS	The wait function returns successfully.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_TIMEOUT	The wait operation timed out, the underlying driver did not response in time.
SSP_ERR_WAIT_ABORTED	System internal error occurred.

8.18.12.3 Function steps

- Obtain mutex before making HAL-level driver call.

8.18.13 SF_JPEG_Decode_StatusGet

```
ssp_err_t SF_JPEG_Decode_StatusGet ( sf_jpeg_decode_ctrl_t  *const p_api_ctrl ,
 jpeg_decode_status_t  *const p_status )
```

8.18.13.1 Brief description

Obtain JPEG codec status. This function can be used to poll the device instead of using SF_JPEG_Decode_Wait to block on JPEG operations.

8.18.13.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 628:Return values

Name	Description
SSP_SUCCESS	The JPEG status information is obtained.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- statusGet

8.18.13.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.14 SF_JPEG_Decode_PixelFormatGet

```
ssp_err_t SF_JPEG_Decode_PixelFormatGet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,  jpeg_decode_color_space_t  *const p_color_space )
```

8.18.14.1 Brief description

Obtain the format of the image. This function is only useful for decoding a JPEG image.

8.18.14.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 629:Return values

Name	Description
SSP_SUCCESS	The JPEG image size is obtained.
SSP_ERR_ASSERTION	p_ctrl is null.

Table 629:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- pixelFormatGet

8.18.14.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.15 SF_JPEG_Decode_ImageSizeGet

```
ssp_err_t SF_JPEG_Decode_ImageSizeGet ( sf_jpeg_decode_ctrl_t
*const p_api_ctrl ,   uint16_t * p_horizontal_size ,   uint16_t
* p_vertical_size )
```

8.18.15.1 Brief description

Obtain the size of the image. This function is only useful for decoding a JPEG image.

8.18.15.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the JPEG codec block before using this function.

Table 630:Return values

Name	Description
SSP_SUCCESS	The JPEG image size is obtained.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.

Table 630:Return values (Continued)

Name	Description
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- imageSizeGet

8.18.15.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex

8.18.16 SF_JPEG_Decode_Close

```
ssp_err_t SF_JPEG_Decode_Close ( sf_jpeg_decode_ctrl_t  *const p_api_ctrl )
```

8.18.16.1 Brief description

Close JPEG codec device. Un-finished codec operation is interrupted, and output data are discarded.

8.18.16.2 Detailed description

NOTE: Call SF_JPEG_Decode_Open to configure the timer before using this function.

Table 631:Return values

Name	Description
SSP_SUCCESS	The JPEG decode device is successfully closed.
SSP_ERR_ASSERTION	p_ctrl is null.
SSP_ERR_NOT_OPEN	JPEG Decode Framework module is not yet initialized.
SSP_ERR_IN_USE	The mutex may be unavailable for the the device. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes. This function calls

- close

8.18.16.3 Function steps

- Obtain mutex before making HAL-level driver call.
- Release mutex
- Clear information from control block so other functions know this block is closed
- Delete RTOS services allocated during the open call.

8.18.17 SF_JPEG_Decode_VersionGet

```
ssp_err_t SF_JPEG_Decode_VersionGet ( ssp_version_t  *const p_version )
```

8.18.17.1 Brief description

Get version and store it in provided pointer p_version.

8.18.17.2 Detailed description

Table 632:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version is null.

8.18.18 g_sf_jpeg_decode_on_sf_jpeg_decode

```
sf_jpeg_decode_api_t::g_sf_jpeg_decode_on_sf_jpeg_decode
```

8.18.18.1 Initialized as

```
g_sf_jpeg_decode_on_sf_jpeg_decode=
{
    .open                   = SF_JPEG_Decode_Open,
    .inputBufferSet         = SF_JPEG_Decode_InputBufferSet,
    .outputBufferSet        = SF_JPEG_Decode_OutputBufferSet,
    .linesDecodedGet        = SF_JPEG_Decode_LinesDecodedGet,
    .horizontalStrideSet    = SF_JPEG_Decode_HorizontalStrideSet,
    .imageSubsampleSet      = SF_JPEG_Decode_ImageSubsampleSet,
    .wait                   = SF_JPEG_Decode_Wait,
    .statusGet              = SF_JPEG_Decode_StatusGet,
    .imageSizeGet           = SF_JPEG_Decode_ImageSizeGet,
    .pixelFormatGet         = SF_JPEG_Decode_PixelFormatGet,
```

```
    .close                  = SF_JPEG_Decode_Close,
    .versionGet             = SF_JPEG_Decode_VersionGet
}
```

8.18.19 Extensions

8.18.19.1 sf_jpeg_decode_instance_ctrl_t

sf_jpeg_decode_instance_ctrl_t

Detailed description

JPEG framework instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- uint32_t open

  Indicate whether the driver is open.

- uint32_t state

  Used by driver to check if pointer to control block is valid.

- TX_MUTEX mutex

  Mutex used to protect access to lower level driver hardware.

- TX_EVENT_FLAGS_GROUP events

  Event flags used by the HAL driver to notify the framework driver of.

- jpeg_decode_instance_t const * p_lower_lvl_jpeg_decode

  Pointer to lower level instance.

8.19 Messaging Framework

RTOS-integrated Messaging Framework implementation.

8.19.1 Functions

- SF_MESSAGE_Open
- SF_MESSAGE_Close
- SF_MESSAGE_BufferAcquire
- SF_MESSAGE_BufferRelease
- SF_MESSAGE_Post
- SF_MESSAGE_Pend
- SF_MESSAGE_VersionGet

8.19.2 Defines

- #define SF_MESSAGE_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_MESSAGE_CODE_VERSION_MINOR

  Initial value:`(5U)`

- #define SF_MESSAGE_QUEUE_MESSAGE_WORDS

  Initial value:`(1)`

  The size of a message queue in words

8.19.3 SF_MESSAGE_Open

```
ssp_err_t SF_MESSAGE_Open ( sf_message_ctrl_t  *const p_api_ctrl ,
 sf_message_cfg_t  const *const p_cfg )
```

8.19.3.1 Brief description

Initialize message framework. This function initiates the messaging framework control block, configures the work memory corresponding to the configuration parameters.

8.19.3.2 Detailed description

Table 633:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	p_ctrl, p_cfg or p_cfg->p_work_memory_start is NULL.
SSP_ERR_INTERNAL	OS service call fails.
SSP_ERR_IN_USE	The Messaging Framework is in use.
SSP_ERR_INVALID_WORKBUFFER_SIZE	Invalid work buffer size.
SSP_ERR_INVALID_MSG_BUFFER_SIZE	Message buffer size is invalid.
SSP_ERR_ILLEGAL_SUBSCRIBER_LISTS	Message subscriber lists is illegal.

NOTE: This API function is allowed to be called once per instance. The behavior if called twice is undefined.

NOTE: This API function only allows to be called from thread context.

8.19.3.3 Function steps

- Creates the memory pools in the work memory area
- Registers subscriber lists
- Changes the messaging framework status from CLOSED to OPENED

8.19.4 SF_MESSAGE_Close

```
ssp_err_t SF_MESSAGE_Close ( sf_message_ctrl_t  *const p_api_ctrl )
```

8.19.4.1 Brief description

Closes message framework.

8.19.4.2 Detailed description

Table 634:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.
SSP_ERR_ILLEGAL_SUBSCRIBER_LISTS	Message subscriber lists is illegal.
SSP_ERR_INTERNAL	OS service call fails

NOTE: This API function only allows to be called from thread context.

8.19.4.3 Function steps

- Finds subscribers and flushes their queues
- Deletes memory pools allocated in the work memory

8.19.5 SF_MESSAGE_BufferAcquire

```
ssp_err_t SF_MESSAGE_BufferAcquire ( sf_message_ctrl_t  const
*const p_api_ctrl ,  sf_message_header_t  ** pp_buffer ,
```

```
 sf_message_acquire_cfg_t  const *const p_acquire_cfg ,   uint32_t
const wait_option )
```

8.19.5.1 Brief description

Acquire buffer for message passing from the block.

8.19.5.2 Detailed description

Table 635:Return values

Name	Description
SSP_SUCCESS	Buffer acquisition was successful.
SSP_ERR_ASSERTION	p_ctrl, p_acquire_cfg or pp_buffer is NULL.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.
SSP_ERR_NO_MORE_BUFFER	No more buffer found in the memory block pool.
SSP_ERR_TIMEOUT	OS service call returns timeout.
SSP_ERR_INTERNAL	OS service call fails.

NOTE: This API function allows to be called from not only thread but also ISR.

8.19.5.3 Function steps

- Allocates buffer in the block memory pool.
- Clears buffer control block
- Sets the buffer in-use flag
- Sets the address of the allocated buffer to 'pp_buffer'
- Clears the event class and event code in the buffer. This is because the initial value in the buffer control block is unknown and it would not be safe.
- Sets the 'buffer_keep' flag in the buffer control block if SF_MESSAGE_ACQUIRE_OPTION_KEEP is set to the 'option' argument

8.19.6 SF_MESSAGE_BufferRelease

```
ssp_err_t SF_MESSAGE_BufferRelease ( sf_message_ctrl_t  *const p_api_ctrl ,
 sf_message_header_t  *const p_buffer ,  sf_message_release_option_t
const option )
```

8.19.6.1 Brief description

Release buffer obtained by SF_MESSAGE_BufferAcquire.

8.19.6.2 Detailed description

Table 636:Return values

Name	Description
SSP_SUCCESS	Buffer release was successful.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.
SSP_ERR_ASSERTION	p_ctrl or p_buffer is NULL.
SSP_ERR_ILLEGAL_BUFFER_ADDRESS	If buffer address is not aligned or p_buffer is not in the block pool range.
SSP_ERR_BUFFER_RELEASED	Buffer has been released.

NOTE: This API function allows to be called from thread but also from ISR.

8.19.6.3 Function steps

- Calculates the address of the buffer control block
- Release buffer in the condition below. (1) The counting semaphore is zero and the buffer keep option is not specified. (2) 'option' is set to SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE.
- Clears the flags in the buffer control block.
- Set back the backed up interrupt mask level.
- Invokes an user callback function if it is registered in the condition below. (1) The counting semaphore is zero. (2) 'option' is set to SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE.
- Sets SF_MESSAGE_CALLBACK_EVENT_NAK if any subscribers for the message have responded NAK
- Sets SF_MESSAGE_CALLBACK_EVENT_ACK if all subscribers for the message have responded ACK
- Sets the pointer to the context to kept in the buffer control block
- Invokes the registered user callback function.

8.19.7 SF_MESSAGE_Post

```
ssp_err_t SF_MESSAGE_Post ( sf_message_ctrl_t  *const p_api_ctrl ,
 sf_message_header_t  const *const p_buffer ,  sf_message_post_cfg_t  const
*const p_post_cfg ,  sf_message_post_err_t  *const p_post_err ,   uint32_t
const wait_option )
```

8.19.7.1 Brief description

Post a message to the subscribers.

8.19.7.2 Detailed description

Table 637:Return values

Name	Description
SSP_SUCCESS	Message posting was successful.
SSP_ERR_ASSERTION	p_ctrl or p_buffer is NULL.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.
SSP_ERR_NO_SUBSCRIBER_FOUND	No subscriber found.
SSP_ERR_BUFFER_RELEASED	Buffer has been released.
SSP_ERR_MESSAGE_QUEUE_FULL	Queue is full (Timeout occurs before sending a message if timeout option is specified)
SSP_ERR_ILLEGAL_BUFFER_ADDRESS	If buffer address is not aligned or p_buffer is not in the block pool range.
SSP_ERR_INTERNAL	OS service call fails

NOTE: This API function allows to be called from not only thread but also ISR(if wait_option is TX_NO_WAIT).

NOTE: Another buffer writing to the buffer before the message read by message consumers results data overwriting.

8.19.7.3 Function steps

- Checks the number of the subscribers of specified event class
- Calculates the address of the buffer control block
- Counts up the counting semaphore in the buffer control block
- Registers user callback function and context passed from user

8.19.8 SF_MESSAGE_Pend

```
ssp_err_t SF_MESSAGE_Pend ( sf_message_ctrl_t  const *const p_api_ctrl ,
 TX_QUEUE  const *const p_queue ,  sf_message_header_t  ** pp_buffer ,   uint32_t
const wait_option )
```

8.19.8.1 Brief description

Pend on a message.

8.19.8.2 Detailed description

Table 638:Return values

Name	Description
SSP_SUCCESS	Message pending was successful.
SSP_ERR_ASSERTION	p_ctrl, pp_buffer or p_queue is NULL.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.
SSP_ERR_MESSAGE_QUEUE_EMPTY	Queue is empty.
SSP_ERR_TIMEOUT	OS service call returns timeout.
SSP_ERR_INTERNAL	OS service call fails.

NOTE: This API function allows to be called from not only thread but also ISR(if wait_option is TX_NO_WAIT).

8.19.8.3 Function steps

- Receiving message here. Receiving data is not message itself but the pointer to the buffer
- If there is no data in the message queue and TX_NO_WAIT is specified to wait_option, return immediately with SSP_ERR_MESSAGE_QUEUE_EMPTY error code

8.19.9 SF_MESSAGE_VersionGet

```
ssp_err_t SF_MESSAGE_VersionGet ( ssp_version_t  *const p_version )
```

8.19.9.1 Brief description

Get the version of the messaging framework. Stores version information in provided pointer.

8.19.9.2 Detailed description

Table 639:Return values

Name	Description
SSP_SUCCESS	Got version number successfully.
SSP_ERR_ASSERTION	p_version is NULL.

8.19.10 Extensions

8.19.10.1 sf_message_instance_ctrl_t

sf_message_instance_ctrl_t

Detailed description

Messaging framework instance control block structure

Variables

- TX_BLOCK_POOL block_pool

  Pointer to the memory block pool control.
- sf_message_subscriber_list_t ** pp_subscriber_lists

  Pointer array to the subscriber lists.
- uint32_t buffer_size

  Bytes of the message buffer.
- uint32_t number_of_buffers

  The number of allocated buffers.
- uint16_t number_of_subscriber_groups

  The number of subscriber groups.
- sf_message_state_t state

  Status of the message framework.

8.20 Power Profiles Framework

Power Profiles Framework.

8.20.1 Functions

- SF_POWER_PROFILES_Open
- SF_POWER_PROFILES_Sleep
- SF_POWER_PROFILES_Close
- SF_POWER_PROFILES_VersionGet

8.20.2 Defines

- #define SF_POWER_PROFILES_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_POWER_PROFILES_CODE_VERSION_MINOR

  Initial value:`(4U)`

8.20.3 SF_POWER_PROFILES_Open

```
ssp_err_t SF_POWER_PROFILES_Open ( sf_power_profiles_ctrl_t  *const p_ctrl ,
 sf_power_profiles_cfg_t  const *const p_cfg )
```

8.20.3.1 Brief description

Configures the Power Profiles framework and opens any required HAL layer drivers that will be used.

8.20.3.2 Detailed description

The SF_POWER_PROFILES_Open function acquires a mutex for the Power Profile module, it then calls the open/init functions for the necessary lower level drivers (RTC, LPM) and finally calls the driver .open function for the Power Profiles module. The mutex is released following the driver layer open functions.

Table 640:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See HAL driver for other possible causes.
SSP_ERR_IN_USE	The mutex may be unavailable for the unit requested. See HAL driver for other possible causes.

Table 640:Return values (Continued)

Name	Description
SSP_ERR_INVALID_HW_CONDITION	Incompatible system clock configuration.
SSP_ERR_INTERNAL	Creation of mutex failed

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

8.20.3.3 Function steps

- Save driver structure pointer for use in other framework layer functions
- Keep the control pointer that is associated with the already open RTC module
- Keep the pointer to the callback function
- Keep the pointer to the sleep and awake ioport tables
- Make a local copy of the LPM configuration structure to modify it
- Initialize the LPM HAL driver
- Mark driver as open by initializing it to "PPMG" in its ASCII equivalent

8.20.4 SF_POWER_PROFILES_Sleep

```
ssp_err_t SF_POWER_PROFILES_Sleep ( sf_power_profiles_ctrl_t  *const p_ctrl )
```

8.20.4.1 Brief description

Places the MCU in Software Standby mode.

8.20.4.2 Detailed description

Table 641:Return values

Name	Description
SSP_SUCCESS	Entered Sleep and then Awoke, or called with mode == SF_POWER_PROFILES_MODE_RUN.
SSP_ERR_ASSERTION	p_ctrl or p_ctrl->p_api is NULL.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_POWER_PROFILES_Open to configure.

Table 641:Return values (Continued)

Name	Description
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (p_ctrl->p_api->start is NULL).

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

8.20.4.3 Function steps

- PREPARE FOR SLEEP MODE
- Set the ioport pins per the supplied sleep configuration
- Check to see if LOCO and/or Subclk are currently running
- Get the current WUPEN bits for restore later
- Get some info about the RTC
- Set the WUPEN bits such the RTC Periodic and Alarm are capable of interrupting and waking them MCU
- Stop the LOCO
- Stop the Sub-Clock
- Set the WUPEN bits such the RTC Periodic and Alarm are capable of interrupting and waking them MCU
- These clocks are allowed to be on during Software Standby mode. If the user is not using RTC mode then we will turn these off.
- Stop the LOCO
- Stop the Sub-Clock
- S7G2
- Some peripherals can be set to run even in Software Standby mode. To insure lowest power usage we will turn those off before going to sleep, if they are on and restore them on Wakeup These include: SCI0 (MSTPCRB b31) IIC0 (MSTPCRB b09)
- MSTPCRB 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
- Above are our desired bit settings for MSTPCRB, our MSTBCRB mask is therefore 0x7FFFFDFF
- D/A Convertor 0 (MSTPCRD b20), AGT1 (MSTPCRD b1) AGT0 (MSTPCRD b3) Comparator-OC5 (MSTPCRD b23) Comparator-OC4 (MSTPCRD b24) Comparator-OC3 (MSTPCRD b25) Comparator-OC2 (MSTPCRD b26) Comparator-OC1 (MSTPCRD b27) Comparator-OC0 (MSTPCRD b28)
- MSTPCRD 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1
- Above are our desired bit settings for MSTPCRD, our MSTBCRD mask is therefore 0xE06FFFF3
- S3A7

- Some peripherals can be set to run even in Software Standby mode. To insure lowest power usage we will turn those off before going to sleep, if they are on and restore them on Wakeup These include: SCI0 (MSTPCRB b31) IIC0 (MSTPCRB b09)
- MSTPCRB 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
- Above are our desired bit settings for MSTPCRB, our MSTBCRB mask is therefore 0x7FFFFDFF
- D/A Convertor 0 (MSTPCRD b20), AGT1 (MSTPCRD b2) AGT0 (MSTPCRD b3) Comparator-OC0 (MSTPCRD b28) Low Power Comparator (MSTPCRD b29)
- MSTPCRD 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1
- Above are our desired bit settings for MSTPCRD, our MSTBCRD mask is therefore 0xCFEFFFF3
- < Disable the LCD
- issue the Pre-Sleep Callback
- Populate the callback argument fields
- Call the user callback
- Setup to enter the low power mode
- ENTER STANDBY MODE
- AWAKE
- Restore any clocks we stopped
- Restore the original mstp mode
- < Reenable the LCD
- Set the ioport pins per the supplied wake configuration
- issue the Post-Sleep Callback
- Populate the callback argument fields
- Call the user callback
- Restore the WUPEN contents to what they were before we entered Software Standby mode

8.20.5 SF_POWER_PROFILES_Close

```
ssp_err_t SF_POWER_PROFILES_Close ( sf_power_profiles_ctrl_t  *const p_ctrl )
```

8.20.5.1 Brief description

The close function acquires the unit's mutex, calls the driver close function, and releases the mutex.

8.20.5.2 Detailed description

Table 642:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl or p_ctrl->p_api is NULL.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_POWER_PROFILES_Open to configure.
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (p_ctrl->p_api->close is NULL).
SSP_ERR_INTERNAL	Deletion of mutex failed.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: - For ThreadX specific API details, refer ThreadX user manual.

8.20.5.3 Function steps

- Clear information from control block so other functions know this block is closed

8.20.6 SF_POWER_PROFILES_VersionGet

```
ssp_err_t SF_POWER_PROFILES_VersionGet ( ssp_version_t  *const p_version )
```

8.20.6.1 Brief description

Gets version and stores it in provided pointer p_version.

8.20.6.2 Detailed description

Table 643:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.21 Power Profiles V2 Framework

8.21.1 Functions

- SF_POWER_PROFILES_V2_Open
- SF_POWER_PROFILES_V2_RunApply
- SF_POWER_PROFILES_V2_LowPowerApply
- SF_POWER_PROFILES_V2_Close
- SF_POWER_PROFILES_V2_VersionGet
- clock_config_apply
- low_power_config_apply
- ioport_cfg_apply
- pre_low_power_callback_handle
- post_low_power_callback_handle

8.21.2 SF_POWER_PROFILES_V2_Open

```
ssp_err_t SF_POWER_PROFILES_V2_Open (  sf_power_profiles_v2_ctrl_t
*const p_ctrl ,   sf_power_profiles_v2_cfg_t const *const p_cfg )
```

8.21.2.1 Brief description

Configures the Power Profiles framework and opens any required HAL layer drivers that will be used.

8.21.2.2 Detailed description

The SF_POWER_PROFILES_V2_Open function initializes the critical data structures and variables.

Table 644:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See HAL driver for other possible causes.
SSP_ERR_IN_USE	Power profiles framework is already open.
SSP_ERR_INTERNAL	Unable to obtain mutex. Unable to release mutex.

Table 644:Return values (Continued)

Name	Description
See	Common Error Codes for other possible return codes or causes.

8.21.3 SF_POWER_PROFILES_V2_RunApply

```
ssp_err_t SF_POWER_PROFILES_V2_RunApply (  sf_power_profiles_v2_ctrl_t
*const p_ctrl ,   sf_power_profiles_v2_run_cfg_t const *const p_cfg )
```

8.21.3.1 Brief description

Applies a Run profile.

8.21.3.2 Detailed description

The SF_POWER_PROFILES_V2_RunApply function will:

- Apply an IO port configuration, if supplied
- Apply a clock configuration

Table 645:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See HAL driver for other possible causes.
SSP_ERR_INVALID_ARGUMENT	Clock configuration is invalid.
SSP_ERR_NOT_OPEN	Power profiles framework is not open.
SSP_ERR_IN_USE	Unable to obtain mutex.
SSP_ERR_INTERNAL	Unable to release mutex.
SSP_ERR_INVALID_HW_CONDITION	Incompatible system clock configuration.
See	Common Error Codes, r_ioport, or r_cgc driver for other possible return codes or causes.

8.21.4 SF_POWER_PROFILES_V2_LowPowerApply

```
ssp_err_t SF_POWER_PROFILES_V2_LowPowerApply (  sf_power_profiles_v2_ctrl_t
*const p_ctrl ,   sf_power_profiles_v2_low_power_cfg_t const *const p_cfg )
```

8.21.4.1 Brief description

Applies a Low Power profile.

8.21.4.2 Detailed description

The SF_POWER_PROFILES_V2_LowPowerApply function will:

- Apply a LPMv2 configuration to prepare for low power
- Apply an IO port configuration to prepare for low power, if supplied
- Notify application that low power is about to be entered
- Enter low power mode
- When low power mode is exited, apply an IO port configuration for wake up, if supplied
- Notify application that wake up has occurred

Table 646:Return values

Name	Description
SSP_SUCCESS	Entered and exited low power mode successfully.
SSP_ERR_ASSERTION	p_ctrl or p_ctrl->p_api is NULL.
SSP_ERR_NOT_OPEN	Power profiles framework is not open.
SSP_ERR_UNSUPPORTED	This function is not supported by one of the HAL drivers, r_lpmv2, r_ioport.
SSP_ERR_INVALID_MODE	r_lpmv2 mode is not LPMV2_LOW_POWER_MODE_SLEEP but r_lmv2 p_extend is NULL.
SSP_ERR_IN_USE	Unable to obtain mutex.
SSP_ERR_INTERNAL	Unable to release mutex.
See	Common Error Codes, r_ioport, or r_lpmv2 drivers for other possible return codes or causes.

8.21.5 SF_POWER_PROFILES_V2_Close

```
ssp_err_t SF_POWER_PROFILES_V2_Close (  sf_power_profiles_v2_ctrl_t
*const p_ctrl )
```

8.21.5.1 Brief description

Closes the framework.

8.21.5.2 Detailed description

Table 647:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Power profiles framework is not open.
SSP_ERR_IN_USE	Unable to obtain mutex.
SSP_ERR_INTERNAL	Unable to release mutex.

8.21.6 SF_POWER_PROFILES_V2_VersionGet

```
ssp_err_t SF_POWER_PROFILES_V2_VersionGet ( ssp_version_t  *const p_version )
```

8.21.6.1 Brief description

Gets version and stores it in provided pointer p_version.

8.21.6.2 Detailed description

Table 648:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.21.7 clock_config_apply

```
ssp_err_t clock_config_apply ( cgc_clocks_cfg_t  const *const p_clock_cfg )
```

8.21.7.1 Brief description

Internal function that checks and applies the clock configuration.

8.21.7.2 Detailed description

Table 649:Return values

Name	Description
SSP_SUCCESS	Clock configuration applied successfully.
See	Common Error Codes or r_cgc driver for other possible return codes or causes.

8.21.8 low_power_config_apply

```
ssp_err_t low_power_config_apply (  sf_power_profiles_v2_ctrl_t *const p_ctrl ,
  sf_power_profiles_v2_low_power_cfg_t const *const p_cfg )
```

8.21.8.1 Brief description

Apply the low power configuration using the lpm driver. Internal function, do not use directly. Returns mutex if an error occurs.

8.21.8.2 Detailed description

Table 650:Parameters

Name	Direction	Description
p_ctrl	in	PPM V2 control structure
p_cfg	in	PPM V2 config structure

Table 651:Return values

Name	Description
SSP_SUCCESS	LPM configuration was successful.
See	Common Error Codes or r_lpmv2 driver for other possible return codes or causes.

8.21.9 ioport_cfg_apply

```
ssp_err_t ioport_cfg_apply (  sf_power_profiles_v2_ctrl_t *const p_ctrl ,
 ioport_cfg_t  const * p_ioport_pin_tbl )
```

8.21.9.1 Brief description

Apply the IO Port configuration using the ioport driver. Internal function, do not use directly. Returns mutex if an error occurs.

8.21.9.2 Detailed description

Table 652:Parameters

Name	Direction	Description
p_ctrl	in	PPM V2 control structure
p_ioport_pin_tbl	in	Pointer to ioport settings

Table 653:Return values

Name	Description
SSP_SUCCESS	IO Port configuration was successful.
See	Common Error Codes or r_ioport driver for other possible return codes or causes.

8.21.10 pre_low_power_callback_handle

```
 pre_low_power_callback_handle (  sf_power_profiles_v2_low_power_cfg_t const
*const p_cfg )
```

8.21.10.1 Brief description

Notify application using pre-low power callback. Internal function, do not use directly.

8.21.10.2 Detailed description

Table 654:Parameters

Name	Direction	Description
p_cfg	in	PPM V2 config structure

Table 655:Return values

Name	Description
none	

8.21.11 post_low_power_callback_handle

```
 post_low_power_callback_handle (  sf_power_profiles_v2_low_power_cfg_t const
*const p_cfg )
```

8.21.11.1 Brief description

Notify application using post-low power callback. Internal function, do not use directly.

8.21.11.2 Detailed description

Table 656:Parameters

Name	Direction	Description
p_cfg	in	PPM V2 config structure

Table 657:Return values

Name	Description
none	

8.22 SPI Framework

RTOS-integrated SPI Framework.

8.22.1 Functions

- sf_spi_callback
- sf_spi_common_start
- sf_spi_common_wait
- sf_spi_common_finish
- sf_spi_bus_device_check
- sf_spi_chipselect_assert
- sf_spi_chipselect_deassert
- sf_spi_check_lower_lvl_driver_parameters
- sf_spi_reconfigure_device
- sf_spi_check_common_parameters
- SF_SPI_Open
- SF_SPI_Read
- SF_SPI_Write
- SF_SPI_WriteRead
- SF_SPI_Close
- SF_SPI_Lock
- SF_SPI_Unlock
- SF_SPI_VersionGet

8.22.2 Variables

- g_sf_spi_version
- g_sf_spi_on_sf_spi

8.22.3 Defines

- #define SF_SPI_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define SF_SPI_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_SPI_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((expression), (error), &g_module_name[0], &g_sf_spi_version)`

8.22.4 sf_spi_callback

```
sf_spi_callback ( spi_callback_args_t  * parg )
```

8.22.4.1 Brief description

SPI SSP framework level callback.

8.22.4.2 Detailed description

Table 658:Parameters

Name	Direction	Description
pcb_arg	in	Pointer to callback parameters

Table 659:Return values

Name	Description
void	

8.22.4.3 Function steps

- Event occurs wake up the suspended thread.
- < Points location for event flag for reception
- Set flag to trigger waiting thread.

8.22.5 sf_spi_common_start

```
ssp_err_t sf_spi_common_start ( sf_spi_instance_ctrl_t  *const p_ctrl ,
  uint32_t const timeout )
```

8.22.5.1 Brief description

Common SPI transfer start function. Used in all framework read write calls. This function checks whether there is any need for reconfiguration and if yes then it checks whether device is supported by the bus. If bus supports the device it acquires the mutex, reconfigures the bus and then enables the slave.

8.22.5.2 Detailed description

Table 660:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for SPI framework driver context for a device
timeout	in	ThreadX timeout

Table 661:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully closed
SSP_ERR_INTERNAL	Internal error

See Common Error Codes and lower level driver function for other possible return codes. These driver functions are:

- close
- open

NOTE: This function is reentrant for any device.

8.22.5.3 Function steps

- See if SPI bus needs to be reconfigured.
- Different device is using bus than last time. Check that bus supports device.
- Get mutex for this bus.
- Need to reconfigure.
- Enable slave.

8.22.6 sf_spi_common_wait

```
ssp_err_t sf_spi_common_wait ( sf_spi_instance_ctrl_t  *const p_ctrl ,
  uint32_t const timeout )
```

8.22.6.1 Brief description

Common SPI wait. Waits for an operation to finish.

8.22.6.2 Detailed description

Table 662:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for SPI framework driver context for a device
timeout	in	ThreadX timeout.

Table 663:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully closed.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TRANSFER_ABORTED	The data transfer was aborted.
SSP_ERR_UNDERFLOW	Read underflow occurs.
SSP_ERR_MODE_FAULT	Mode fault error occurs.
SSP_ERR_READ_OVERFLOW	Read overflow occurs.
SSP_ERR_PARITY	Parity error occurs.
SSP_ERR_OVERRUN	Overrun error occurs.

NOTE: This function is reentrant for any device.

8.22.7 sf_spi_common_finish

```
ssp_err_t sf_spi_common_finish ( sf_spi_instance_ctrl_t  *const p_ctrl )
```

8.22.7.1 Brief description

Common SPI finish. Release mutex and deassert chip select.

8.22.7.2 Detailed description

Table 664:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for SPI framework driver context for a device

Table 665:Return values

Name	Description
SSP_SUCCESS	Chip select deactivated and mutex released.
SSP_ERR_INTERNAL	Internal error.

NOTE: This function is reentrant for any device.

8.22.8 sf_spi_bus_device_check

```
ssp_err_t sf_spi_bus_device_check ( sf_spi_instance_ctrl_t  *const p_ctrl )
```

8.22.8.1 Brief description

SPI device check, checks device compatibility on the bus.

8.22.8.2 Detailed description

Table 666:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for SPI framework driver context for a device

Table 667:Return values

Name	Description
SSP_SUCCESS	Selected device supported by the SPI bus
SSP_ERR_UNSUPPORTED	Selected device not supported by the SPI bus

NOTE: This function is reentrant for any device.

8.22.9 sf_spi_chipselect_assert

```
sf_spi_chipselect_assert ( ioport_port_pin_t  chip_select ,
ioport_level_t  active_level )
```

8.22.9.1 Brief description

SPI SSP framework level chip select utility function.

8.22.9.2 Detailed description

Table 668:Parameters

Name	Direction	Description
chip_select	in	Chip select pin
active_level	in	Active high or active low

Table 669:Return values

Name	Description
void	

8.22.10 sf_spi_chipselect_deassert

```
sf_spi_chipselect_deassert ( ioport_port_pin_t  chip_select ,
ioport_level_t  active_level )
```

8.22.10.1 Brief description

SPI SSP framework level chip select utility function.

8.22.10.2 Detailed description

Table 670:Parameters

Name	Direction	Description
chip_select	in	Chip select pin
active_level	in	Active high or active low

Table 671:Return values

Name	Description
void	

8.22.11 sf_spi_check_lower_lvl_driver_parameters

```
sf_spi_check_lower_lvl_driver_parameters ( sf_spi_cfg_t  const *const p_cfg )
```

8.22.11.1 Brief description

Checks whether lower level SPI module and bus are defined.

8.22.11.2 Detailed description

Table 672:Parameters

Name	Direction	Description
p_cfg	in	Pointer to SPI framework Configuration Structure

Table 673:Return values

Name	Description
true	Lower level SPI driver, and SPI bus are configured
false	Lower level SPI driver, and SPI bus are not configured

8.22.12 sf_spi_reconfigure_device

```
ssp_err_t sf_spi_reconfigure_device ( sf_spi_instance_ctrl_t  *const p_ctrl )
```

8.22.12.1 Brief description

Assign a new device address to current device.

8.22.12.2 Detailed description

Table 674:Parameters

Name	Direction	Description
p_ctrl	in	Control handle for SPI framework context for a device

Table 675:Return values

Name	Description
SSP_SUCCESS	New device address assigned to current device

See Common Error Codes and lower level drivers for other possible return codes. These driver functions are:

- close
- open

8.22.12.3 Function steps

- < Get the current device.
- Close the device currently using the bus.
- Assign bus for the new device to use.

- Assign this device to current.

8.22.13 sf_spi_check_common_parameters

```
 sf_spi_check_common_parameters ( sf_spi_ctrl_t  *const p_ctrl ,   uint32_t
const length ,   uint32_t const timeout )
```

8.22.13.1 Brief description

Checks if SPI framework control block address, length and timeout are NULL.

8.22.13.2 Detailed description

Table 676:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to SPI framework control block
length	in	Number of bytes of data to be transferred
timeout	in	Timeout in ThreadX tick counts

Table 677:Return values

Name	Description
true	SPI framework control block address, length and timeout are not NULL.
false	SPI framework control block address, length and timeout are NULL.

8.22.14 SF_SPI_Open

```
ssp_err_t SF_SPI_Open ( sf_spi_ctrl_t  *const p_api_ctrl ,  sf_spi_cfg_t  const
*const p_cfg )
```

8.22.14.1 Brief description

Initialize a SPI bus and open low level SPI driver.

8.22.14.2 Detailed description

Table 678:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully opened.
SSP_ERR_ASSERTION	One of the following parameters is NULL: p_api_ctrl, p_cfg, Pointer to Open, Close, Read, Write, or Writeread API interfaces, p_cfg->p_bus or p_cfg->p_lower_lvl_cfg.
SSP_ERR_INTERNAL	Internal error occurred.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is

- open

NOTE: This function is reentrant for any channel.

NOTE: Control block must be cleared by caller before calling this function.

8.22.14.3 Function steps

- Copy bus to control
- Copy chip_select to control
- Copy chip_select level to control
- Initialize bus lock to false
- Set framework level callback function.
- Save context for use in ISRs.
- Use bus channel in device open.
- Open only for the first device on the bus. If the device requires a bus reconfiguration then that will happen when a later read/write occurs.
- Create mutex for this bus.
- Create Event flag.
- Open the low level SPI module.
- Assign last used device ctrl on this bus.
- Save device configuration for reconfiguration.
- Set device state as Opened.
- Increment device count.
- Initialize chip select.

8.22.15 SF_SPI_Read

```
ssp_err_t SF_SPI_Read ( sf_spi_ctrl_t  *const p_api_ctrl ,  void *const p_dest ,
  uint32_t const length ,  spi_bit_width_t  const bit_width ,  uint32_t
const timeout )
```

8.22.15.1 Brief description

Starts the transfer process and receives data from SPI device.

8.22.15.2 Detailed description

Table 679:Return values

Name	Description
SSP_SUCCESS	Data read completed successfully.
SSP_ERR_ASSERTION	One of the following parameters is NULL: p_api_ctrl, p_dest, length, timeout.
SSP_ERR_NOT_OPEN	Device not opened.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- read

8.22.15.3 Function steps

- Check whether device is open.
- Start transfer process - check lock, check reconfiguration, check bus compatibility, enable chip select.
- Perform write read.
- Finish transfer.

8.22.16 SF_SPI_Write

```
ssp_err_t SF_SPI_Write ( sf_spi_ctrl_t  *const p_api_ctrl ,  void *const p_src ,
  uint32_t const length ,  spi_bit_width_t  const bit_width ,  uint32_t
const timeout )
```

8.22.16.1 Brief description

Starts the transfer process and writes data to SPI device.

8.22.16.2 Detailed description

Table 680:Return values

Name	Description
SSP_SUCCESS	Data write completed successfully.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_api_ctrl, p_src, length, timeout.
SSP_ERR_NOT_OPEN	Device not opened.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- write

8.22.16.3 Function steps

- Check that device is open.
- Start transfer process - check lock, check reconfiguration, check bus compatibility, enable chip select.
- Perform write read.
- Finish transfer.

8.22.17 SF_SPI_WriteRead

```
ssp_err_t SF_SPI_WriteRead ( sf_spi_ctrl_t  *const p_api_ctrl ,   void
*const p_src ,   void *const p_dest ,   uint32_t const length ,  spi_bit_width_t
const bit_width ,   uint32_t const timeout )
```

8.22.17.1 Brief description

Simultaneously transmit data to SPI device while receiving data from SPI device(full duplex).

8.22.17.2 Detailed description

Table 681:Return values

Name	Description
SSP_SUCCESS	Data write completed successfully.

Table 681:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_api_ctrl, p_src, p_dest, length, timeout.
SSP_ERR_NOT_OPEN	Device not opened.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- writeRead

8.22.17.3 Function steps

- Check whether device is open.
- Start transfer process - check lock, check reconfiguration, check bus compatibility, enable chip select.
- Perform write read.
- Finish transfer.

8.22.18 SF_SPI_Close

```
ssp_err_t SF_SPI_Close ( sf_spi_ctrl_t  *const p_api_ctrl )
```

8.22.18.1 Brief description

Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus.

8.22.18.2 Detailed description

Table 682:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully closed.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_IN_USE	In-use error.
SSP_ERR_INTERNAL	Internal error occurred.

See Common Error Codes and lower level driver function for other possible return codes. This driver function is:

- close

NOTE: This function is reentrant for any device.

8.22.18.3 Function steps

- Check whether device is open.
- Check whether the device must be closed through close. Close only if this is the current device. Otherwise the device might have been closed during reconfiguration or might not have been opened through open().
- Get mutex since this will access hardware registers.
- Close low level driver.
- Release mutex.
- Decrement device count.
- < Check is there any device still using the bus.
- Delete RTOS services used by bus.
- Set device to closed state

8.22.19 SF_SPI_Lock

```
ssp_err_t SF_SPI_Lock ( sf_spi_ctrl_t  *const p_api_ctrl )
```

8.22.19.1 Brief description

Lock the bus for a device.

8.22.19.2 Detailed description

Table 683:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully closed.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_IN_USE	In-use error.

NOTE: This function is reentrant for any device.

8.22.19.3 Function steps

- Check whether device is open.
- Check whether device is already locked.
- Get the mutex for this device.
- Set lock flag.
- Enable slave.

8.22.20 SF_SPI_Unlock

```
ssp_err_t SF_SPI_Unlock ( sf_spi_ctrl_t  *const p_api_ctrl )
```

8.22.20.1 Brief description

Unlock the bus for a particular device and make the bus usable for other devices.

8.22.20.2 Detailed description

Table 684:Return values

Name	Description
SSP_SUCCESS	SPI channel is successfully closed.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_IN_USE	In-use error.

NOTE: This function is reentrant for any device.

8.22.20.3 Function steps

- Check whether device is open.
- Clear lock flag.
- Release the mutex so that others can use the bus.
- Disable slave.

8.22.21 SF_SPI_VersionGet

```
ssp_err_t SF_SPI_VersionGet ( ssp_version_t  *const p_version )
```

8.22.21.1 Brief description

Get the version information of the framework.

8.22.21.2 Detailed description

Table 685:Return values

Name	Description
SSP_ERR_ASSERTION	p_version is NULL.
SSP_SUCCESS	Successful return.

8.22.21.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.

8.22.22 g_sf_spi_version

`ssp_version_t::g_sf_spi_version`

8.22.22.1 Detailed description

Version data structure used by error logger macro.

8.22.22.2 Initialized as

```
g_sf_spi_version=
{
    .api_version_minor  = SF_SPI_API_VERSION_MINOR,
    .api_version_major  = SF_SPI_API_VERSION_MAJOR,
    .code_version_minor = SF_SPI_CODE_VERSION_MINOR,
    .code_version_major = SF_SPI_CODE_VERSION_MAJOR,
}
```

8.22.23 g_sf_spi_on_sf_spi

`sf_spi_api_t::g_sf_spi_on_sf_spi`

8.22.23.1 Initialized as

```
g_sf_spi_on_sf_spi=
{
```

```
    .open      = SF_SPI_Open,
    .read      = SF_SPI_Read,
    .write     = SF_SPI_Write,
    .writeRead = SF_SPI_WriteRead,
    .close     = SF_SPI_Close,
    .lock      = SF_SPI_Lock,
    .unlock    = SF_SPI_Unlock,
    .version   = SF_SPI_VersionGet
}
```

8.22.24 Extensions

8.22.24.1 sf_spi_instance_ctrl_t

sf_spi_instance_ctrl_t

Detailed description

SPI device context. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- sf_spi_bus_t * p_bus

  Bus using this device (copy from cfg)

- ioport_port_pin_t chip_select

  Chip select for this device (copy from cfg)

- ioport_level_t chip_select_level_active

  Polarity of CS, active High or Low (copy from cfg)

- spi_cfg_t lower_lvl_cfg

  SPI peripheral configuration, use for bus reconfiguration.

- spi_ctrl_t * p_lower_lvl_ctrl

  SPI peripheral control block.

- sf_spi_dev_state_t dev_state

  Device status.

- bool locked

  Lock and unlock bus for a device.

8.23 Thread Monitor Framework

Framework module providing monitoring of threads.

Any misbehaving threads result in the device being reset. Both the WDT and IWDT HAL modules are supported by this framework module.

8.23.1 Interface Used

WDT Interface

8.23.2 Functions

- SF_THREAD_MONITOR_Thread
- SF_THREAD_MONITOR_Open
- SF_THREAD_MONITOR_Close
- SF_THREAD_MONITOR_ThreadRegister
- SF_THREAD_MONITOR_ThreadUnregister
- SF_THREAD_MONITOR_CountIncrement
- SF_THREAD_MONITOR_VersionGet

8.23.3 Defines

- #define SF_THREAD_MONITOR_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_THREAD_MONITOR_CODE_VERSION_MINOR

  Initial value:`(4U)`

8.23.4 SF_THREAD_MONITOR_Thread

```
SF_THREAD_MONITOR_Thread (  ULONG thread_input )
```

8.23.4.1 Brief description

The SF_THREAD_MONITOR_thread() is the main thread monitor thread which runs periodically.

8.23.4.2 Detailed description

The period is determined from the timeout period of the Watchdog hardware. This thread's period should be half of that of the watchdog hardware so the watchdog is refreshed at 50% of the count value.

Table 686:Parameters

Name	Direction	Description
thread_input	in	This is the address of the control block for the thread monitor module. This allows this thread to be created multiple times with the data used and hardware interfaced with governed by the control structure.

As this is a ThreadX thread this function does not return.

NOTE: *This function is not reentrant.****NOTE:*** *If there are no threads registered to be monitored the monitoring thread will refresh/tickle/kick the watchdog and so prevent the watchdog from reseting the device.*

8.23.5 SF_THREAD_MONITOR_Open

```
ssp_err_t SF_THREAD_MONITOR_Open ( sf_thread_monitor_ctrl_t  *const p_api_ctrl ,
 sf_thread_monitor_cfg_t  const *const p_cfg )
```

8.23.5.1 Brief description

Calls the driver .open function in the p_lower_lvl_wdt parameter.

8.23.5.2 Detailed description

After successively opening and configuring the HAL driver, the SF_THREAD_MONITOR_Open function starts the thread monitoring thread. This thread is responsible for checking thread execution through thread count variables and for refreshing the implemented watchdog timer peripheral.

Table 687:Return values

Name	Description
SSP_SUCCESS	Initialization was successful and watchdog monitoring thread has started.
SSP_ERR_ASSERTION	p_ctrl, p_cfg, p_cfg->p_lower_lvl_wdt pointers and/or p_cfg->p_lower_lvl_wdt are NULL.
SSP_ERR_IN_USE	Thread monitor has already been opened.
SSP_ERR_INVALID_MODE	Low level watchdog peripheral returned an error when opened.
SSP_ERR_UNSUPPORTED	Data structure could not be allocated.

Table 687:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid for the low level driver.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: This function is not reentrant.

NOTE: If there are no threads registered to be monitored the monitoring thread refreshes the watchdog and prevents the watchdog from reseting the device.

NOTE: The Thread Monitor can be closed with SF_THREAD_MONITOR_Close and then SF_THREAD_MONITOR_Open can be called again. However, if the underlying watchdog timer hardware such as WDT or IWDT cannot be closed or stopped then there are likely to be undesirable results.

8.23.6 SF_THREAD_MONITOR_Close

```
ssp_err_t SF_THREAD_MONITOR_Close ( sf_thread_monitor_ctrl_t
*const p_api_ctrl )
```

8.23.6.1 Brief description

Stop the thread monitoring thread. All threads are unregistered and no longer monitored.

8.23.6.2 Detailed description

Table 688:Return values

Name	Description
SSP_SUCCESS	Close was successful and watchdog monitoring thread has stopped.
SSP_ERR_ASSERTION	p_ctrl pointer is NULL.
SSP_ERR_NOT_OPEN	SF_THREAD_MONITOR_Open has either not been called or it was not called successfully.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: This function is not reentrant.

NOTE: This function DOES NOT stop the watchdog timer (e.g. WDT or IWDT). It does however stop the thread which refreshes the these timers. To prevent the hardware watchdog from reseting the device the watchdog must be refreshed elsewhere.

NOTE: The Thread Monitor can be closed with SF_THREAD_MONITOR_Close. However, if the underlying watchdog timer hardware such as WDT or IWDT cannot be closed or stopped then there are likely to be undesirable results.

8.23.7 SF_THREAD_MONITOR_ThreadRegister

```
ssp_err_t SF_THREAD_MONITOR_ThreadRegister ( sf_thread_monitor_ctrl_t
*const p_api_ctrl ,  sf_thread_monitor_counter_min_max_t  const
* p_counter_min_max )
```

8.23.7.1 Brief description

Register a thread to be monitored by the watchdog monitoring thread.

8.23.7.2 Detailed description

This thread must supply the minimum and maximum expected values for the thread. The mininimum and maximum values can be determined by using monitoring mode.

Table 689:Return values

Name	Description
SSP_SUCCESS	Thread successfully registered.
SSP_ERR_ASSERTION	p_ctrl or p_counter_min_max pointers are NULL.
SSP_ERR_INSUFFICIENT_SPACE	Not enough entries in the threads to be monitored array to add this thread. Increase the value of THREAD_MONITOR_CFG_MAX_NUMBER_OF_THREADS in sf_thread_moinitor_cfg.h
SSP_ERR_NOT_OPEN	SF_THREAD_MONITOR_Open has either not been called or it was not called successfully.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: This function is reentrant.

8.23.8 SF_THREAD_MONITOR_ThreadUnregister

```
ssp_err_t SF_THREAD_MONITOR_ThreadUnregister ( sf_thread_monitor_ctrl_t
*const p_api_ctrl )
```

8.23.8.1 Brief description

Remove a thread from being monitored by the Watchdog monitoring thread.

8.23.8.2 Detailed description

Table 690:Return values

Name	Description
SSP_SUCCESS	Thread successfully unregistered.
SSP_ERR_ASSERTION	p_ctrl pointer is NULL.
SSP_ERR_NOT_OPEN	SF_THREAD_MONITOR_Open has either not been called or it was not called successfully.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: This function is reentrant.

8.23.9 SF_THREAD_MONITOR_CountIncrement

```
ssp_err_t SF_THREAD_MONITOR_CountIncrement ( sf_thread_monitor_ctrl_t
*const p_api_ctrl )
```

8.23.9.1 Brief description

Safely increment the counter of a thread. A mutex is used to ensure this increment occurs without corruption.

8.23.9.2 Detailed description

Table 691:Return values

Name	Description
SSP_SUCCESS	Thread counter successfully incremented.
SSP_ERR_ASSERTION	p_ctrl pointer is NULL.
SSP_ERR_INSUFFICIENT_SPACE	Not enough entries in the threads to be monitored array to add this thread.

Table 691:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	SF_THREAD_MONITOR_Open has either not been called or it was not called successfully.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or HAL driver for other possible return codes or causes.

NOTE: This function is reentrant.

8.23.10 SF_THREAD_MONITOR_VersionGet

```
ssp_err_t SF_THREAD_MONITOR_VersionGet ( ssp_version_t  *const p_version )
```

8.23.10.1 Brief description

Get version and store it in provided pointer p_version. Implements versionGet.

8.23.10.2 Detailed description

Table 692:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.23.11 Extensions

8.23.11.1 sf_thread_monitor_instance_ctrl_t

sf_thread_monitor_instance_ctrl_t

Detailed description

Thread monitor control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- uint32_t open

  Used by driver to check if the control structure is valid

- wdt_instance_t const * p_lower_lvl_wdt

  Pointer to interface structure for the watchdog peripheral
- uint32_t timeout_period_msec

  Time in milliseconds of the watchdog timeout period. Used to calculate the period of the monitoring thread.
- uint32_t timeout_period_watchdog_clocks

  Maximum count value of the watchdog. Used to synchronise to the count.
- bool profiling_mode_enabled

  Used by the driver to check if profiling mode is enabled.
- TX_MUTEX mutex

  Mutex to protect access to the thread counters.
- uint32_t profiling_mode_check

  Value used to verify profiling mode is enabled when prfiling_mode_enabled == true.
- sf_thread_monitor_thread_counter_t thread_counters[THREAD_MONITOR_CFG_MAX_NUMBER_OF_THREADS]

  Data storage for the thread counter information.
- TX_THREAD thread

  Thread monitor thread.
- void const * p_extend

  Extended configuration data.
- uint8_t stack[THREAD_MONITOR_THREAD_STACK_SIZE]

  Stack for thread monitor thread.

8.24 CTSU Framework

RTOS-integrated CTSU Framework.

8.24.1 Functions

- SF_TOUCH_CTSU_Open
- SF_TOUCH_CTSU_Read
- SF_TOUCH_CTSU_Close
- SF_TOUCH_CTSU_VersionGet

8.24.2 Defines

- #define SF_TOUCH_CTSU_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_TOUCH_CTSU_CODE_VERSION_MINOR

  Initial value:`(3U)`

8.24.3 SF_TOUCH_CTSU_Open

```
ssp_err_t SF_TOUCH_CTSU_Open ( sf_touch_ctsu_ctrl_t  *const p_api_ctrl ,
 sf_touch_ctsu_cfg_t  const *const p_cfg )
```

8.24.3.1 Brief description

Configures Touch CTSU framework.

8.24.3.2 Detailed description

The SF_TOUCH_CTSU_Open function checks to see if it was already initialized and if not, then initializes the CTSU HAL layer, saves the callback for the calling layer, and creates the internal thread. If the framework was already initialized by the same widget layer, then it will return an error. If the calling widget is a first time caller, then the callback and context are stored internally.

Table 693:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See HAL driver for other possible causes.
SSP_ERR_IN_USE	The framework has already been initialized by this particular widget.
SSP_ERR_INTERNAL	The hardware initialization or thread/semaphore failed.
SSP_ERR_INVALID_ARGUMENT	An input parameter was outside the supported bounds.

NOTE: Refer to the lower level driver for additional return value descriptions.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.24.3.3 Function steps

- Initialize the elements to NULL if this is the first time the framework is being opened
- If the framework has already been opened by at least one widget layer; this is verified by checking to see if the callback from the widget layer is the same as any of the already stored ones
- Check if the table already contains the callback and the passed callback is not NULL. If true, then the same widget layer is calling the initialization function again in error. If the passed callback is NULL, this means that the widget layer does not want a callback.
- Save the widget callback and context into the framework callback array
- Verify there was room to store the callback and/or context
- If the framework has not been opened by at least one widget layer then initialize the lower layer and create the internal thread
- Create semaphore to control access to the lower level driver
- Delete the semaphore_ctsu_scan_complete semaphore
- Return error
- Assign the instance for the lower level into the control structure
- Create a local version of the CTSU HAL driver so that we can modify the HAL callback to use the callback defined in this file.
- Change the callback function to private function in this file.
- Open the CTSU for operation
- Save configuration and lower level API info
- Close the lower level driver to clean up hardware locks and memory
- Clear the callback
- Clear the callback
- Delete the semaphores
- Return the error
- Create internal CTSU thread to drive scans.
- Delete the semaphore
- If this framework has already been initialized by a previous call, then update the p_ctrl for the current calling layer
- Save the locally saved pointer to the lower level control to the current calling frameworks lower level control so that the current framework can close the lower level
- Save the callback index
- Mark control block open so other tasks know it is valid
- Mark the local flag as initialized

8.24.4 SF_TOUCH_CTSU_Read

```
ssp_err_t SF_TOUCH_CTSU_Read ( sf_touch_ctsu_ctrl_t  *const p_api_ctrl ,   void
* p_dest ,  ctsu_read_t  read_options ,  ctsu_channel_pair_t const * channels ,
  const uint16_t count )
```

8.24.4.1 Brief description

Reads data from Touch CTSU framework.

8.24.4.2 Detailed description

The SF_TOUCH_CTSU_Read function allows the user to read back data from the lower level layer. Access to the lower level data is controlled via semaphore to prevent data reading when a scan or processing is ongoing.

Table 694:Return values

Name	Description
SSP_SUCCESS	Read completed successfully.
SSP_ERR_ASSERTION	p_ctrl or p_ctrl->p_api is NULL.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_TOUCH_CTSU_Open to configure.
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (p_ctrl->p_lower_lvl_api->read is NULL).

See Common Error Codes or HAL driver for other possible return codes or causes.

8.24.4.3 Function steps

- Hold the TX status
- Wait for a scan to complete if the CTSU is scanning.
- Read the data from the lower layer driver
- Return the semaphore
- Read the converted data from the lower layer driver. Reading this data does not require a semaphore since this data is not actively updated by an ongoing scan.
- If there is a ThreadX error, return it, else return the lower level driver return code

8.24.5 SF_TOUCH_CTSU_Close

```
ssp_err_t SF_TOUCH_CTSU_Close ( sf_touch_ctsu_ctrl_t  *const p_api_ctrl )
```

8.24.5.1 Brief description

The close function checks to see if there are any other registered callbacks for other widgets. If there are then this function simply sets the callback for this widget to NULL and returns. If there are no other registered callbacks then this function terminates the internal scanning thread and calls the lower level close function.

8.24.5.2 Detailed description

Table 695:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_ctrl or p_ctrl->p_api is NULL.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_TOUCH_CTSU_Open to configure.
SSP_ERR_INTERNAL	ThreadX call failed.
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (p_ctrl->p_lower_lvl_api->close is NULL).

See Common Error Codes or HAL driver for other possible return codes or causes.

8.24.5.3 Function steps

- Hold the status
- Set the callback for this widget to NULL
- Clear information from control block so other functions know this block is closed
- Check if the table contains any non-NULL callbacks which would indicate that some other widget layer is also using this framework. If no non-NULL callbacks are in the table then the internal thread can be terminated and the HAL closed
- Return success
- Grab the semaphore first before deleting the Thread. May be unnecessary depending on TX Implementation
- Terminate the internal scanning thread first
- Delete the internal scanning thread once it has been terminated
- Close the lower layer driver
- Delete the semaphore
- Clear the local flag to indicate that the framework module is closed
- If there is a ThreadX error, return it, else return the lower level driver return code

8.24.6 SF_TOUCH_CTSU_VersionGet

```
ssp_err_t SF_TOUCH_CTSU_VersionGet ( ssp_version_t  *const p_version )
```

8.24.6.1 Brief description

Gets version and stores it in provided pointer p_version.

8.24.6.2 Detailed description

Table 696:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.24.7 Extensions

8.24.7.1 sf_touch_ctsu_instance_ctrl_t

sf_touch_ctsu_instance_ctrl_t

Detailed description

Instance control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open

  Used by driver to check if control block is valid.
- uint16_t update_hz

  The frequency to run the scans at.
- ctsu_instance_t * p_lower_lvl_instance

  Pointer to CTSU instance.
- uint32_t cb_index

  Indicates the index in callback registry table.

8.25 CTSU Button Framework

RTOS-integrated CTSU Button Framework.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

8.25.1 Functions

- SF_TOUCH_CTSU_Button_Open
- SF_TOUCH_CTSU_Button_Close
- SF_TOUCH_CTSU_Button_Enable
- SF_TOUCH_CTSU_Button_Disable
- SF_TOUCH_CTSU_Button_VersionGet

8.25.2 Defines

- #define SF_TOUCH_CTSU_BUTTON_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Version of code that implements the API defined in this file
- #define SF_TOUCH_CTSU_BUTTON_CODE_VERSION_MINOR

  Initial value:`(2U)`
- #define SF_TOUCH_CTSU_BUTTON_MAX_CHANNELS

  Initial value:`CTSU_MAX_CHANNELS`

  Maximum supported button channel
- #define SF_TOUCH_CTSU_BUTTON_BIT_MASK_ARRAY_SIZE

  Initial value:`6`

8.25.3 SF_TOUCH_CTSU_Button_Open

```
ssp_err_t SF_TOUCH_CTSU_Button_Open ( sf_touch_ctsu_button_ctrl_t
*const p_api_ctrl ,  sf_touch_ctsu_button_cfg_t  const *const p_cfg )
```

8.25.3.1 Brief description

Configures Touch CTSU Button framework.

8.25.3.2 Detailed description

The SF_TOUCH_CTSU_Button_Open function initializes all the buttons in the configuration structure and also initializes the lower level framework module.

Table 697:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See lower driver for other possible causes.
SSP_ERR_IN_USE	The framework has already been initialized by this particular widget.

See HAL driver for other possible return codes or causes.

8.25.4 SF_TOUCH_CTSU_Button_Close

```
ssp_err_t SF_TOUCH_CTSU_Button_Close ( sf_touch_ctsu_button_ctrl_t
* p_api_ctrl )
```

8.25.4.1 Brief description

Closes the Touch CTSU Button framework.

8.25.4.2 Detailed description

The SF_TOUCH_CTSU_Button_Close closes the button framework.

Table 698:Return values

Name	Description
SSP_SUCCESS	Close was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_TOUCH_CTSU_Button_Open to configure.
SSP_ERR_INTERNAL	The hardware initialization or thread/mutex/semaphore failed.

See HAL driver for other possible return codes or causes.

8.25.4.3 Function steps

- Close the lower level module
- Mark the module as uninitialized

8.25.5 SF_TOUCH_CTSU_Button_Enable

```
ssp_err_t SF_TOUCH_CTSU_Button_Enable ( sf_touch_ctsu_button_ctrl_t
* p_api_ctrl ,  sf_touch_ctsu_button_id  id )
```

8.25.5.1 Brief description

Enables events for a specific button.

8.25.5.2 Detailed description

The SF_TOUCH_CTSU_Button_Enable function resets a buttons state and enables events to be generated from actions on that button.

Table 699:Return values

Name	Description
SSP_SUCCESS	Button Enable was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_TOUCH_CTSU_Button_Open to configure.
SSP_ERR_INTERNAL	The hardware initialization or thread/mutex/semaphore failed.

See HAL driver for other possible return codes or causes.

8.25.5.3 Function steps

- Iterate through the list of button identifiers to find the button and then enable the button

8.25.6 SF_TOUCH_CTSU_Button_Disable

```
ssp_err_t SF_TOUCH_CTSU_Button_Disable ( sf_touch_ctsu_button_ctrl_t
* p_api_ctrl ,  sf_touch_ctsu_button_id  id )
```

8.25.6.1 Brief description

Disables a specific button.

8.25.6.2 Detailed description

The SF_TOUCH_CTSU_Button_Disable function disables events from a particular button.

Table 700:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl.
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_TOUCH_CTSU_Button_Open to configure.

See HAL driver for other possible return codes or causes.

8.25.6.3 Function steps

- Iterate through the list of button identifiers to find the button and then disable the button

8.25.7 SF_TOUCH_CTSU_Button_VersionGet

```
ssp_err_t SF_TOUCH_CTSU_Button_VersionGet ( ssp_version_t  *const p_version )
```

8.25.7.1 Brief description

Gets version and stores it in provided pointer p_version.

8.25.7.2 Detailed description

Table 701:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.25.8 Extensions

8.25.8.1 sf_touch_ctsu_button_instance_ctrl_t

sf_touch_ctsu_button_instance_ctrl_t

Detailed description

Control structure for this Interface

Variables

- uint32_t opened

  Save the initialization state.

- sf_touch_ctsu_button_hdl * p_button_hdl

  Pointer to the button handle.

- uint32_t button_count

  Button Count.

- sf_touch_ctsu_instance_t const * p_lower_lvl_ctsu

  Pointer to the lower level instance.

- void const * p_context

  Placeholder for user data.

- void(* p_callback)(*p_args)

  Callback function.

8.26 CTSU Slider Framework

RTOS-integrated CTSU Slider Framework.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

8.26.1 Functions

- SF_TOUCH_CTSU_Slider_VersionGet
- SF_TOUCH_CTSU_Slider_Open
- SF_TOUCH_CTSU_Slider_Close
- SF_TOUCH_CTSU_Slider_Enable
- SF_TOUCH_CTSU_Slider_Disable

8.26.2 Defines

- #define SF_TOUCH_CTSU_SLIDER_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Version of code that implements the API defined in this file
- #define SF_TOUCH_CTSU_SLIDER_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define SF_TOUCH_CTSU_SLIDER_MAX_CHANNELS

  Initial value:`CTSU_MAX_CHANNELS`

  Maximum supported CTSU channels
- #define SF_TOUCH_CTSU_SLIDER_MAX_CHANNELS_PER_SLIDER

  Initial value:`10`

  Maximum supported channels per slider based on Workbench6 as of February 2016
- #define SF_TOUCH_CTSU_SLIDER_BIT_MASK_ARRAY_SIZE

  Initial value:`6`

8.26.3 SF_TOUCH_CTSU_Slider_VersionGet

```
ssp_err_t SF_TOUCH_CTSU_Slider_VersionGet ( ssp_version_t  *const p_version )
```

8.26.3.1 Brief description

Gets version and stores it in provided pointer p_version.

8.26.3.2 Detailed description

Table 702:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.26.4 SF_TOUCH_CTSU_Slider_Open

```
ssp_err_t SF_TOUCH_CTSU_Slider_Open ( sf_touch_ctsu_slider_ctrl_t
*const p_api_ctrl ,  sf_touch_ctsu_slider_cfg_t  const *const p_cfg )
```

8.26.4.1 Brief description

Configures Touch CTSU Slider framework.

8.26.4.2 Detailed description

The SF_TOUCH_CTSU_Slider_Open function initializes all the sliders/wheels in the configuration structure and initializes the lower level framework module.

Table 703:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See lower driver for other possible causes.
SSP_ERR_IN_USE	The framework has already been initialized by this particular widget.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.26.4.3 Function steps

- Discover active channels

8.26.5 SF_TOUCH_CTSU_Slider_Close

```
ssp_err_t SF_TOUCH_CTSU_Slider_Close ( sf_touch_ctsu_slider_ctrl_t
*const p_api_ctrl )
```

8.26.5.1 Brief description

Closes the Touch CTSU Slider framework.

8.26.5.2 Detailed description

The SF_TOUCH_CTSU_Slider_Close

Table 704:Return values

Name	Description
SSP_SUCCESS	Initialization was successful.

Table 704:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See lower driver for other possible causes.
SSP_ERR_NOT_OPEN	The framework has not been initialized by this particular widget.
SSP_ERR_UNSUPPORTED	p_ctrl->p_lower_lvl_ctsu->p_api or p_ctrl->p_lower_lvl_ctsu->p_api->close was NULL

See Common Error Codes or HAL driver for other possible return codes or causes.

8.26.5.3 Function steps

- Close the lower level module
- Mark the module as uninitialized

8.26.6 SF_TOUCH_CTSU_Slider_Enable

```
ssp_err_t SF_TOUCH_CTSU_Slider_Enable ( sf_touch_ctsu_slider_ctrl_t
*const p_api_ctrl ,  sf_touch_ctsu_slider_id_t  id )
```

8.26.6.1 Brief description

The SF_TOUCH_CTSU_Slider_Enable.

8.26.6.2 Detailed description

Table 705:Return values

Name	Description
SSP_SUCCESS	Enable was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See lower driver for other possible causes.
SSP_ERR_NOT_OPEN	The framework has not been initialized by this particular widget.
SSP_ERR_INVALID_ARGUMENT	Invalid slider identifier.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.26.6.3 Function steps

- Iterate through the list of slider identifiers to find the slider and then enable the slider

8.26.7 SF_TOUCH_CTSU_Slider_Disable

```
ssp_err_t SF_TOUCH_CTSU_Slider_Disable ( sf_touch_ctsu_slider_ctrl_t
*const p_api_ctrl ,  sf_touch_ctsu_slider_id_t  id )
```

8.26.7.1 Brief description

The SF_TOUCH_CTSU_Slider_Disable.

8.26.7.2 Detailed description

Table 706:Return values

Name	Description
SSP_SUCCESS	Disable was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_ctrl, p_api, or p_cfg. See lower driver for other possible causes.
SSP_ERR_NOT_OPEN	The framework has not been initialized by this particular widget.
SSP_ERR_IN_USE	The framework has already been initialized by this particular widget.

See Common Error Codes or HAL driver for other possible return codes or causes.

8.26.7.3 Function steps

- Iterate through the list of slider identifiers to find the slider and then disable the slider

8.26.8 Extensions

8.26.8.1 sf_touch_ctsu_slider_instance_ctrl_t

sf_touch_ctsu_slider_instance_ctrl_t

Detailed description

Instance control structure for this module

Variables

- uint32_t opened

  Save the initialization state of the framework.

- uint32_t slider_count

  Slider Count.

- sf_touch_ctsu_instance_t const * p_lower_lvl_ctsu

  Pointer to the lower level instance.

- void const * p_context

  Placeholder for user data.

- void(* p_callback)(*p_args)

  Function to call when an event occurs

8.26.8.2 sf_slider_on_ctsu_cfg_t

sf_slider_on_ctsu_cfg_t

Detailed description

Individual slider structure to be populated by Workbench

Variables

- sf_slider_type_tconst type

  Linear or circular (wheel)

- uint32_t num_slider_channels
- ctsu_channel_pair_t const * p_slider_channels

  Define the channel/channel pairs which make up a slider.

- int32_t const * p_normalization
- int32_t * p_channel_average
- uint32_t * p_offset
- uint16_t *const p_slider_scount
- uint16_t *const p_slider_baseline
- int32_t *const p_slider_delta
- sf_touch_ctsu_slider_id_t id

  Unique identifier for slider

- int32_t max_slider_value

  Maximum slider value, must be greater than 0; Minimum is always 0

- const uint16_t slider_norm_max

  Individual slider settings that can be modified by the user. Should be the same as in st_sf_touch_ctsu_slider_h-dlTOUCH_SLIDER_CFG_NORM_MAX

- const int32_t slider_threshold

  Touch threshold. Ensure that value is greater than 0

- const int32_t channel_average_weight

  Weight of running average of counts for each channel

- const int32_t prev_sum_weight

  Weight of running average of previous sum in position calculation, must be greater than 0

- const int32_t cutoff

  Defines how far below prev_sum running average to get before "SF_TOUCH_SLIDER_STATE_RELEASED" detected

8.26.8.3 st_sf_touch_ctsu_slider_hdl

st_sf_touch_ctsu_slider_hdl

Detailed description

Handle for each slider

Needed for forward reference to sf_touch_ctsu_slider_hdl_t

Variables

- uint32_t open

  Indicate that slider has been opened

- sf_touch_ctsu_slider_id_t id

  Unique identifier for slider.

- sf_touch_ctsu_slider_state_t state

  Represents the current state of the slider.

- sf_slider_type_t type

  Linear or circular (wheel)

- uint32_t num_slider_channels
- ctsu_channel_pair_t const * p_slider_channels

  Define the channel/channel pairs which make up a slider.

- int32_t const * p_normalization
- int32_t * p_channel_average
- uint32_t * p_offset
- uint16_t * p_slider_scount
- uint16_t * p_slider_baseline

- int32_t * p_slider_delta
- uint32_t position

  Calculated position.
- int32_t prev_sum

  Used to store the running average of previous sum in position calculation
- int32_t max_slider_value

  Maximum slider value, must be greater than 0; Minimum is always 0
- ssp_err_t(* p_update)(*const hdl, const *const p_lower_lvl_touch_framework, uint32_t *p_pos, *const p_state)

  Function to calculate position of touch.
- uint64_t bit_mask[SF_TOUCH_CTSU_SLIDER_BIT_MASK_ARRAY_SIZE]

  Bit mask to be used in update function
- uint16_t slider_norm_max

  Individual slider settings that can be modified by the user. Should be the same as in sf_slider_on_ctsu_cfg_t-TOUCH_SLIDER_CFG_NORM_MAX
- int32_t slider_threshold

  Touch threshold. Ensure that value is greater than 0
- int32_t channel_average_weight

  Weight of running average of counts for each channel
- int32_t prev_sum_weight

  Weight of running average of previous sum in position calculation, must be greater than 0
- int32_t cutoff

  Defines how far below prev_sum running average to get before "SF_TOUCH_SLIDER_STATE_RELEASED" detected

8.27 I2C Touch Panel Framework

RTOS-integrated touch panel Framework implementation for external I2C touch chips.

8.27.1 Summary

This is a ThreadX touch panel framework implemented for external I2C touch controllers with IRQ pins used to notify the application when new data is available.

8.27.2 Functions

- sf_touch_panel_i2c_thread

- sf_touch_panel_i2c_check_event_to_be_sent
- sf_touch_panel_i2c_touch_event_post
- SF_TOUCH_PANEL_I2C_Open
- SF_TOUCH_PANEL_I2C_Calibrate
- SF_TOUCH_PANEL_I2C_Start
- SF_TOUCH_PANEL_I2C_Stop
- SF_TOUCH_PANEL_I2C_Reset
- SF_TOUCH_PANEL_I2C_Close
- SF_TOUCH_PANEL_I2C_VersionGet

8.27.3 Defines

- #define SF_TOUCH_PANEL_I2C_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SF_TOUCH_PANEL_I2C_CODE_VERSION_MINOR

  Initial value:`(5U)`
- #define SF_TOUCH_PANEL_I2C_RESET_PIN_UNUSED

  Initial value:`(0xFFFF)`
- #define SF_TOUCH_PANEL_I2C_STACK_SIZE

  Initial value:`(1024)`

  Stack size for I2C touch panel thread.
- #define SF_TOUCH_PANEL_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &g_version)`

  Macro for error logger.
- #define SF_TOUCH_PANEL_PRV_RTOS_TICKS_SEC

  Initial value:`(100U)`
- #define OPEN

  Initial value:`(0x54493243U)`

  "TI2C" in ASCII, used to identify I2C touch panel handle

8.27.4 sf_touch_panel_i2c_thread

```
sf_touch_panel_i2c_thread (  ULONG thread_input )
```

8.27.4.1 Brief description

Publishes touch input messages.

8.27.4.2 Detailed description

(end defgroup SF_TOUCH_PANEL_I2C)

Table 707:Parameters

Name	Direction	Description
thread_input	in	Pointer to touch panel control structure (sf_touch_panel_i2c_instance_ctrl_t).

8.27.4.3 Function steps

- Get a buffer from the messaging framework to store touch data.
- Set message payload event class and event.
- Check for touch control stop flag. If it is received, wait for touch control start flag. After start is received, reset touch chip to ensure old buffered data is flushed, then wait for the next pin interrupt from the touch controller.
- Get the touch event data from the touch driver
- Check the touch coordinate got from the touch driver
- Check if new touch event is to be sent.
- Store time, used to prevent too many events from being generated
- Post a touch event to the message queue(s) of touch event subscribers.

8.27.5 sf_touch_panel_i2c_check_event_to_be_sent

```
 sf_touch_panel_i2c_check_event_to_be_sent ( sf_touch_panel_i2c_instance_ctrl_t
* p_ctrl ,  sf_touch_panel_event_t  event ,   ULONG last_updated_time )
```

8.27.5.1 Brief description

sf_touch_panel_i2c_check_event_to_be_sent : Check a touch event to be sent or not.

8.27.5.2 Detailed description

This function checks if new touch event is to be sent or not. DOWN and UP will always be sent. HOLD, MOVE and INVALID will be sent if user specified dormant time elapsed after previous event was sent.

Table 708:Parameters

Name	Direction	Description
p_ctrl	in	: Pointer to SF_TOUCH_PANEL_I2C instance structure
event	in	: Touch event
last_updated_time	in	: Last time when touch event was updated.

Table 709:Return values

Name	Description
true	- Send the event.
false	- Do not send the event.

8.27.5.3 Function steps

- Only send repeat events as requested by user.

8.27.6 sf_touch_panel_i2c_touch_event_post

```
 sf_touch_panel_i2c_touch_event_post ( sf_touch_panel_i2c_instance_ctrl_t
* p_ctrl )
```

8.27.6.1 Brief description

sf_touch_panel_i2c_touch_event_post : Post new touch event message.

8.27.6.2 Detailed description

This function posts a touch event message to the message subscribers.

Table 710:Parameters

Name	Direction	Description
p_ctrl	in	: Pointer to SF_TOUCH_PANEL_I2C instance structure

8.27.6.3 Function steps

- Rotate touch coordinate clockwise.
- Post an event to message queue.

8.27.7 SF_TOUCH_PANEL_I2C_Open

```
ssp_err_t SF_TOUCH_PANEL_I2C_Open ( sf_touch_panel_ctrl_t  *const p_api_ctrl ,
 sf_touch_panel_cfg_t  const *const p_cfg )
```

8.27.7.1 Brief description

Implements open.

8.27.7.2 Detailed description

Table 711:Return values

Name	Description
SSP_SUCCESS	Touch panel thread created and lower level drivers opened successfully.
SSP_ERR_ASSERTION	A pointer parameter was NULL, or a lower level driver reported this error.
SSP_ERR_INTERNAL	The touch panel thread or event flags could not be created, or a lower level driver reported this error.
SSP_ERR_IN_USE	Mutex was not available, or a lower level driver reported this error.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- open
- reset
- open

NOTE: This function is reentrant for any panel.

8.27.7.3 Function steps

- Store API's.
- Store user configurations in control block.
- Initialize previous event.

- Get mutex before calling lower layer, which will access HW registers.
- Open the lower level I2C driver.
- Reset the touch chip.
- Open the lower level external IRQ driver.
- Return mutex
- Create event flags for internal communication.
- Create main touch panel thread.
- Mark control block open so other tasks know it is valid

8.27.8 SF_TOUCH_PANEL_I2C_Calibrate

```
ssp_err_t SF_TOUCH_PANEL_I2C_Calibrate ( sf_touch_panel_ctrl_t
*const p_api_ctrl ,  sf_touch_panel_calibrate_t  const *const p_expected ,
 sf_touch_panel_payload_t  const *const p_actual ,   ULONG timeout )
```

8.27.8.1 Brief description

Implements calibrate.

8.27.8.2 Detailed description

Table 712:Return values

Name	Description
SSP_SUCCESS	Touch panel calibrated successfully.
SSP_ERR_ASSERTION	A pointer parameter was NULL, or a lower level driver reported this error.
SSP_ERR_CALIBRATE_FAILED	Actual touch value was not in expected range.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any panel.

8.27.8.3 Function steps

- Timeout not used in this implementation.
- Determine if touch message data matches expected value.

8.27.9 SF_TOUCH_PANEL_I2C_Start

```
ssp_err_t SF_TOUCH_PANEL_I2C_Start ( sf_touch_panel_ctrl_t  *const p_api_ctrl )
```

8.27.9.1 Brief description

Implements start.

8.27.9.2 Detailed description

Table 713:Return values

Name	Description
SSP_SUCCESS	Touch panel start flag posted to touch thread.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any panel.

8.27.9.3 Function steps

- Set start flag.

8.27.10 SF_TOUCH_PANEL_I2C_Stop

```
ssp_err_t SF_TOUCH_PANEL_I2C_Stop ( sf_touch_panel_ctrl_t  *const p_api_ctrl )
```

8.27.10.1 Brief description

Implements stop.

8.27.10.2 Detailed description

Table 714:Return values

Name	Description
SSP_SUCCESS	Touch panel will not respond to touch events.
SSP_ERR_ASSERTION	Parameter p_ctrl was NULL, or a lower level driver reported this error.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred.

See Common Error Codes or lower level drivers for other possible return codes.

NOTE: This function is reentrant for any panel.

8.27.10.3 Function steps

- Set stop flag.

8.27.11 SF_TOUCH_PANEL_I2C_Reset

```
ssp_err_t SF_TOUCH_PANEL_I2C_Reset ( sf_touch_panel_ctrl_t  *const p_api_ctrl )
```

8.27.11.1 Brief description

Implements reset.

8.27.11.2 Detailed description

Table 715:Return values

Name	Description
SSP_SUCCESS	Touch panel and lower level I2C driver were reset successfully.
SSP_ERR_ASSERTION	Parameter p_ctrl was NULL, or a lower level driver reported this error.

Table 715:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	Mutex was not available, or a lower level driver reported this error.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Use Open API to configure.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- reset

NOTE: This function is reentrant for any panel.

8.27.11.3 Function steps

- Obtain mutex since this accesses shared resources.
- Call hardware specific reset function.
- Release mutex.

8.27.12 SF_TOUCH_PANEL_I2C_Close

```
ssp_err_t SF_TOUCH_PANEL_I2C_Close ( sf_touch_panel_ctrl_t  *const p_api_ctrl )
```

8.27.12.1 Brief description

Implements close.

8.27.12.2 Detailed description

Table 716:Return values

Name	Description
SSP_SUCCESS	Touch panel instance successfully closed.
SSP_ERR_ASSERTION	Parameter p_ctrl was NULL, or a lower level driver reported this error.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.

See Common Error Codes or lower level drivers for other possible return codes. This function calls:

- close

- close
- bufferRelease

NOTE: This function is reentrant.

8.27.12.3 Function steps

- Close the lower level external IRQ driver.
- Close the lower level I2C driver.
- If I2C Abort returned from previous close, one more try to close I2C driver.
- Suspend internal thread.
- If a messaging framework buffer is taken but a touch message is not yet posted to the subscribers, release it.
- Delete RTOS services used
- Mark control block close

8.27.13 SF_TOUCH_PANEL_I2C_VersionGet

```
ssp_err_t SF_TOUCH_PANEL_I2C_VersionGet ( ssp_version_t  *const p_version )
```

8.27.13.1 Brief description

Implements versionGet.

8.27.13.2 Detailed description

Table 717:Return values

Name	Description
SSP_SUCCESS	Version returned successfully.
SSP_ERR_ASSERTION	Parameter p_version was null.

8.27.14 API Data

8.27.14.1 sf_touch_panel_prv_flags_t

```
sf_touch_panel_prv_flags_t
```

Enumerated values

Name	Description
SF_TOUCH_PANEL_PRV_FLAGS_STOP	
SF_TOUCH_PANEL_PRV_FLAGS_START	

8.27.15 Extensions

8.27.15.1 sf_touch_panel_i2c_chip_t

sf_touch_panel_i2c_chip_t

Detailed description

Chip definition.

Variables

- ssp_err_t(* payloadGet)(*const p_ctrl, *const p_payload)

  Reads the touch chip and fills in the touch payload data. p_ctrlPointer to a structure allocated by user. This control structure is initialized in this function. p_payloadPointer to the payload to data structure. Touch data provided should be processed to logical pixel values.

- ssp_err_t(* reset)(*const p_ctrl)

  Resets the touch chip. p_ctrlPointer to a structure allocated by user. This control structure is initialized in this function.

8.27.15.2 sf_touch_panel_i2c_instance_ctrl_t

sf_touch_panel_i2c_instance_ctrl_t

Detailed description

Instance control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t open

  Used by driver to check if control block is valid.

- uint16_t hsize_pixels

  Horizontal size of screen in pixels.

- uint16_t vsize_pixels

  Vertical size of screen in pixels.

- sf_message_instance_t const * p_message

  Pointer to messaging framework control block.

- uint8_t event_class_instance

  Event class instance number for posting touch event class messages.
- sf_touch_panel_payload_t * p_payload

  Pointer to buffer used to store payload.
- sf_touch_panel_payload_t last_payload

  Stores most recent payload put on queue for comparison.
- TX_MUTEX mutex

  Mutex used to protect access to shared resources.
- TX_EVENT_FLAGS_GROUP flags

  Event flags for internal communication.
- TX_THREAD thread

  Main touch panel thread.
- ioport_port_pin_t pin

  Reset pin.
- i2c_master_instance_t const * p_lower_lvl_i2c

  Lower level I2C.
- sf_external_irq_instance_t const * p_lower_lvl_irq

  Lower level external IRQ.
- uint8_t stack[SF_TOUCH_PANEL_I2C_STACK_SIZE]

  Stack for touch panel thread.
- sf_touch_panel_i2c_chip_t const * p_chip

  Chip specific functions and definitions.
- uint16_t update_hz

  The frequency to report repeat (SF_TOUCH_PANEL_EVENT_DOWN or SF_TOUCH_PANEL_EVENT_HOLD) touch events in Hertz. This will be converted to RTOS ticks in the driver and rounded up to the nearest integer value of RTOS ticks.
- uint16_t rotation_angle

  Touch coordinate rotation angle(0/90/180/270)

8.27.15.3 sf_touch_panel_i2c_cfg_t

sf_touch_panel_i2c_cfg_t

Detailed description

Configuration for RTOS touch panel driver.

Variables

- i2c_master_instance_t const * p_lower_lvl_i2c

  Pointer to lower level I2C.

- sf_external_irq_instance_t const * p_lower_lvl_irq

  Pointer to lower level external IRQ.

- ioport_port_pin_t pin

  Port pin connected to reset line on touch controller chip. Set to SF_TOUCH_PANEL_I2C_RESET_PIN_UNUSED if unused.

- sf_touch_panel_i2c_chip_t const *const p_chip

  Selected touch controller chip.

8.28 UART Framework Instance

RTOS-integrated Communications Framework UART implementation.

8.28.1 Functions

- SF_UART_COMMS_Open
- SF_UART_COMMS_Close
- SF_UART_COMMS_Read
- SF_UART_COMMS_Write
- SF_UART_COMMS_Lock
- SF_UART_COMMS_Unlock
- SF_UART_COMMS_VersionGet

8.28.2 Defines

- #define SF_UART_COMMS_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of code that implements the API defined in this file

- #define SF_UART_COMMS_CODE_VERSION_MINOR

  Initial value:`(6U)`

8.28.3 SF_UART_COMMS_Open

```
ssp_err_t SF_UART_COMMS_Open ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_cfg_t  const *const p_cfg )
```

8.28.3.1 Brief description

Open the UART for communication.

8.28.3.2 Detailed description

Table 718:Parameters

Name	Direction	Description
p_api_ctrl	inout	Pointer to the UART control block
p_cfg	in	Pointer to the configuration structure

Table 719:Return values

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Parameter check failed for one of the following: • Pointer p_api_ctrl is NULL. • Pointer p_cfg is NULL • Pointer p_cfg->p_extend is NULL • Pointer p_cfg_extend->p_lower_lvl_uart is NULL • Pointer p_cfg_extend->p_lower_lvl_uart->p_api->open is NULL • Pointer p_cfg_extend->p_lower_lvl_uart->p_cfg is NULL
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use thread mutex or failure create/use event flags or queue.

See Common Error Codes or functions called by this function for other possible codes. This function calls:

- open

NOTE: This function is reentrant for any channel. Handle must be cleared by caller before calling this function.

8.28.3.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.
- Initialize and start ThreadX resources
- Calls open function of UART HAL driver

8.28.4 SF_UART_COMMS_Close

```
ssp_err_t SF_UART_COMMS_Close ( sf_comms_ctrl_t  *const p_api_ctrl )
```

8.28.4.1 Brief description

Close the UART Channel and clean up the resources.

8.28.4.2 Detailed description

Table 720:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to the UART control block

Table 721:Return values

Name	Description
SSP_SUCCESS	UART channel is successfully closed.
SSP_ERR_ASSERTION	Parameter check failed for one of the following: • Pointer p_api_ctrl is NULL. • Pointer p_ctrl->p_lower_lvl_uart is NULL • Pointer p_ctrl->p_lower_lvl_uart->p_api is NULL • Pointer p_ctrl->p_lower_lvl_uart->p_api->close is NULL
SSP_ERR_NOT_OPEN	Channel is not opened.

Table 721:Return values (Continued)

Name	Description
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to delete thread mutex , event flags or queue.

See Common Error Codes or functions called by this function for other possible codes. This function calls:

- close

NOTE: This function is reentrant for any channel.

8.28.4.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.
- Calls close function of UART HAL driver
- Release ThreadX resources

8.28.5 SF_UART_COMMS_Read

```
ssp_err_t SF_UART_COMMS_Read ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   UINT const timeout )
```

8.28.5.1 Brief description

Read user specified number of bytes into destination buffer pointer.

8.28.5.2 Detailed description

Table 722:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to the UART control block
bytes	in	No.of bytes to be read
timeout	in	timeout for read
p_dest	out	Destination buffer

Table 723:Return values

Name	Description
SSP_SUCCESS	Data reception ends successfully.
SSP_ERR_NOT_OPEN	Channel is not opened.
SSP_ERR_HW_LOCKED	Channel is locked.
SSP_ERR_ASSERTION	Pointer to UART control block/pointer to destination address is NULL.
SSP_ERR_INVALID_MODE	Channel is used for non-UART mode.
SSP_ERR_INSUFFICIENT_DATA	Not enough data in receive circular buffer.
SSP_ERR_RXBUF_OVERFLOW	Receive queue overflow.
SSP_ERR_OVERFLOW	Hardware overflow.
SSP_ERR_FRAMING	Framing error.
SSP_ERR_PARITY	Parity error.
SSP_ERR_BREAK_DETECT	Break signal detected.
SSP_ERR_TIMEOUT	One of the following operation timed out. • 'Event flags get' timed out • 'Receive mutex get' timed out
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use thread mutex or failure create/use event flags or queue.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- read

8.28.5.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.
- Locks the UART hardware resource
- Unlock the UART hardware resource

8.28.6 SF_UART_COMMS_Write

```
ssp_err_t SF_UART_COMMS_Write ( sf_comms_ctrl_t  *const p_api_ctrl ,   uint8_t
const *const p_src ,   uint32_t const bytes ,   UINT const timeout )
```

8.28.6.1 Brief description

Write user specified number of bytes from the source buffer.

8.28.6.2 Detailed description

Table 724:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to the UART control block
bytes	in	No.of bytes to to be written.
timeout	in	timeout for read
p_src	out	Source buffer

Table 725:Return values

Name	Description
SSP_SUCCESS	Data transmission finished successfully.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL. Pointer to source buffer is NULL.
SSP_ERR_NOT_OPEN	Channel is not opened.
SSP_ERR_INVALID_MODE	Channel is used for non-UART mode or illegal mode is set in handle.
SSP_ERR_INSUFFICIENT_SPACE	Not enough space in transmission circular buffer.
SSP_ERR_HW_LOCKED	Could not lock hardware.
SSP_ERR_TIMEOUT	'Transmit mutex get' timed out
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use thread mutex or failure create/use event flags or queue.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- write

8.28.6.3 Function steps

- Checks error. Further parameter checking can be done at the driver layer.
- Locks an UART hardware resource
- Clear the event flag.
- Calls write function of UART HAL driver
- Wait until write operation is completed. Event is signaled in event flag object
- Unlock the UART hardware resource

8.28.7 SF_UART_COMMS_Lock

```
ssp_err_t SF_UART_COMMS_Lock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type ,   UINT timeout )
```

8.28.7.1 Brief description

Lock the UART resource.

8.28.7.2 Detailed description

Table 726:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to the UART control block
lock_type	in	Type of Lock.
timeout	in	timeout for lock.

Table 727:Return values

Name	Description
SSP_SUCCESS	Locking UART resource successful.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL.

Table 727:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Channel is not opened.
SSP_ERR_TIMEOUT	Timeout Error. 'Receive mutex get' timed out 'Transmit mutex get' timed out

8.28.8 SF_UART_COMMS_Unlock

```
ssp_err_t SF_UART_COMMS_Unlock ( sf_comms_ctrl_t  *const p_api_ctrl ,
 sf_comms_lock_t  lock_type )
```

8.28.8.1 Brief description

UnLock the UART resource.

8.28.8.2 Detailed description

Table 728:Parameters

Name	Direction	Description
p_api_ctrl	in	Pointer to the UART control block
lock_type	in	Type of Lock.

Table 729:Return values

Name	Description
SSP_SUCCESS	Unlocking UART resource successful.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL.
SSP_ERR_NOT_OPEN	Channel is not opened.
SSP_ERR_TIMEOUT	Timeout Error. 'Receive mutex put' timed out 'Transmit mutex put' timed out

8.28.9 SF_UART_COMMS_VersionGet

```
ssp_err_t SF_UART_COMMS_VersionGet ( ssp_version_t  *const p_version )
```

8.28.9.1 Detailed description

Table 730:Return values

Name	Description
SSP_SUCCESS	Version number obtained successfully.
SSP_ERR_ASSERTION	

8.28.10 API Data

8.28.10.1 sf_uart_comms_state_t

`sf_uart_comms_state_t`

Detailed description

Framework UART state

Enumerated values

Name	Description
SF_UART_COMMS_STATE_CLOSED	UART port is closed.
SF_UART_COMMS_STATE_OPENED	UART port is opened.
SF_UART_COMMS_STATE_READING	UART port is on data reception.
SF_UART_COMMS_STATE_WRITING	UART port is on data transmission.

8.28.11 Extensions

8.28.11.1 sf_uart_comms_instance_ctrl_t

sf_uart_comms_instance_ctrl_t

Detailed description

UART communications instance control structure. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- uint32_t state

  UART status.
- uart_instance_t const * p_lower_lvl_uart

  Pointer to UART interface (copied from cfg)
- TX_MUTEX mutex[2]

  Pointer to the mutex object for UART resource mutual exclusion.
- TX_EVENT_FLAGS_GROUP eventflag[2]

  Pointer to the event flag object for UART data transfer.
- TX_QUEUE queue

  Queue for reading.
- uint32_t queue_mem[SF_UART_COMMS_CFG_QUEUE_SIZE_WORDS]

  Queue memory.

8.28.11.2 sf_uart_comms_cfg_t

sf_uart_comms_cfg_t

Detailed description

Configuration for RTOS integrated UART driver

Variables

- uart_instance_t const * p_lower_lvl_uart

  Pointer to UART Driver instance

8.29 NetX Synergy Port

Name of module used by error logger macro

8.29.1 Functions

- edmac_eint_isr
- nx_ms_to_ticks
- nx_synergy_ethernet_init
- nx_driver_timer_handler
- nx_driver_event_handler
- enet_hw_enable_interrupt
- nx_ether_driver_eth1

- nx_ether_driver_eth0
- nx_ether_driver_support_mac_config_eth0
- nx_ether_driver_support_mac_config_eth1
- nx_ether_driver
- nx_ether_interrupt
- nx_ether_send
- nx_ether_write
- nx_storage_init
- nx_tx_interrupt
- nx_rx_interrupt
- nx_receive
- nx_driver_deferred_processing
- nx_ethernet_version_get

8.29.2 edmac_eint_isr

```
edmac_eint_isr (  void  )
```

8.29.2.1 Brief description

edmac_eint_isr

8.29.2.2 Detailed description

This is the Ethernet interrupt service routine. It clears the interrupt flag and calls nx_ether_interrupt to process the interrupt.

8.29.2.3 Function steps

- Save context if RTOS is used
- Process the interrupt.
- Clear pending interrupt flag to make sure it doesn't fire again after exiting.
- Restore context if RTOS is used

8.29.3 nx_ms_to_ticks

```
nx_ms_to_ticks (  ULONG time_ms )
```

8.29.4 nx_synergy_ethernet_init

```
nx_synergy_ethernet_init ( NX_REC  * nx_rec_ptr )
```

8.29.4.1 Brief description

nx_synergy_ethernet_init

8.29.4.2 Detailed description

This function initializes the specified Ethernet port.

Table 731:Parameters

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure

Table 732:Return values

Name	Description
SSP_SUCCESS	: Call successful.
NX_NOT_SUCCESSFUL	: Call not successful.

8.29.4.3 Function steps

- Configure the Ethernet interrupt.
- Enable clock
- Reset PHY
- Initialize & reset EDMAC and ETHERC
- Wait at least 64 cycles of PCLKA to reset the EDMAC and ETHERC. PCLKA must be at least 12.5 MHz to use Ethernet, so wait at least 5.12 us.
- Set ETHERC default modes 100 Mbps, Full duplex
- Set to little Endian.
- Initialize controller
- Set up descriptor addresses
- Configure FIFO
- Enable EDMAC receive

- Enable receive, transmit, ECI interrupts.
- Initialize PHY.
- Start PHY auto negotiation
- Create a timer to poll for completion of autonegotiation.

8.29.5 nx_driver_timer_handler

```
nx_driver_timer_handler (  ULONG channel )
```

8.29.5.1 Brief description

nx_driver_timer_handler

8.29.5.2 Detailed description

This function is called from the timer expiration event. On every time event, this routine register a deferred event in IP helper thread

Table 733:Parameters

Name	Direction	Description
channel	in	: Ethernet channel number.

8.29.5.3 Function steps

- Mark the polling flag.
- Call NetX to register a deferred event.

8.29.6 nx_driver_event_handler

```
nx_driver_event_handler ( NX_REC  * nx_rec_ptr )
```

8.29.6.1 Brief description

nx_driver_event_handler

8.29.6.2 Detailed description

This function is called from the IP thread deferred event. On every deferred event, this routine checks for Phy link status and handles link up/down and link change. During initialization this routine is responsible for checking for autonegotiation.

Table 734:Parameters

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure.

8.29.6.3 Function steps

- Determine previous link status.
- Save link status and changed polling interval.
- Check PHY link status.
- Save link status and changed polling interval.

8.29.7 enet_hw_enable_interrupt

```
enet_hw_enable_interrupt ( NX_REC  * nx_rec )
```

8.29.7.1 Brief description

enet_hw_enable_interrupt

8.29.7.2 Detailed description

This function enables interrupts for the given Ethernet port.

Table 735:Parameters

Name	Direction	Description
nx_rec	in	: Pointer to Ethernet record structure.

NOTE: The pointer parameter passed to this function is already validated at the higher level.

8.29.7.3 Function steps

- Enable interrupts at the NVIC.
- Enable interrupts at the Ethernet controller

8.29.8 nx_ether_driver_eth1

```
ssp_err_t nx_ether_driver_eth1 ( NX_IP_DRIVER  * driver_req_ptr )
```

8.29.8.1 Brief description

nx_ether_driver_eth1

8.29.8.2 Detailed description

Driver entry function for Ethernet Port 1

Table 736:Parameters

Name	Direction	Description
driver_req_ptr	in	: Pointer to driver request structure

Table 737:Return values

Name	Description
SSP_SUCCESS	: Call successful.
SSP_ERR_ASSERTION	: One or more pointers point to NULL.

8.29.9 nx_ether_driver_eth0

```
ssp_err_t nx_ether_driver_eth0 ( NX_IP_DRIVER  * driver_req_ptr )
```

8.29.9.1 Brief description

nx_ether_driver_eth0

8.29.9.2 Detailed description

Driver entry function for Ethernet Port 0

Table 738:Parameters

Name	Direction	Description
driver_req_ptr	in	: Pointer to driver request structure

Table 739:Return values

Name	Description
SSP_SUCCESS	: Call successful.
SSP_ERR_ASSERTION	: One or more pointers point to NULL.

8.29.10 nx_ether_driver_support_mac_config_eth0

```
ssp_err_t nx_ether_driver_support_mac_config_eth0 ( NX_IP_DRIVER
* driver_req_ptr ,  nx_mac_address_t void(*)( *) p_callback )
```

8.29.10.1 Brief description

nx_ether_driver_support_mac_config_eth0

8.29.10.2 Detailed description

Driver entry function for Ethernet Port 0 with runtime MAC address support.

Table 740:Parameters

Name	Direction	Description
driver_req_ptr	in	: Pointer to driver request structure
p_callback	in	: Callback to set MAC address at runtime

Table 741:Return values

Name	Description
SSP_SUCCESS	: Call successful.
SSP_ERR_ASSERTION	: One or more pointers point to NULL.

8.29.11 nx_ether_driver_support_mac_config_eth1

```
ssp_err_t nx_ether_driver_support_mac_config_eth1 ( NX_IP_DRIVER
* driver_req_ptr ,  nx_mac_address_t void(*)( *) p_callback )
```

8.29.11.1 Brief description

nx_ether_driver_support_mac_config_eth1

8.29.11.2 Detailed description

Driver entry function for Ethernet Port 1 with runtime MAC address support.

Table 742:Parameters

Name	Direction	Description
driver_req_ptr	in	: Pointer to driver request structure
p_callback	in	: Callback to set MAC address at runtime

Table 743:Return values

Name	Description
SSP_SUCCESS	: Call successful.
SSP_ERR_ASSERTION	: One or more pointers point to NULL.

8.29.12 nx_ether_driver

```
ssp_err_t nx_ether_driver ( NX_IP_DRIVER  * driver_req_ptr ,  NX_REC
* nx_rec_ptr ,  nx_mac_address_t void(*)( *) p_callback )
```

8.29.12.1 Brief description

nx_ether_driver

8.29.12.2 Detailed description

Ethernet driver function for Renesas Synergy.

Table 744:Parameters

Name	Direction	Description
driver_req_ptr	in	: Pointer to driver request structure

Table 744:Parameters (Continued)

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure
p_callback	in	: Callback to set MAC address at runtime

Table 745:Return values

Name	Description
SSP_SUCCESS	: Call successful.
SSP_ERR_ASSERTION	: One or more pointers point to NULL.

8.29.12.3 Function steps

- Perform parameter checking
- Setup the IP pointer from the driver request.
- Default to successful return.
- Extract driver command.
- Process the driver request.
- Record the interface structure in the driver record.
- This command is used in multi homed devices - return NX_SUCCESS by default.
- Process driver initialization.
- Initialize BDs.
- Record the interface structure in the driver record.
- Configure mac address.
- Get MAC address from user.
- Setup the physical address of this IP instance.
- Initialize the ETHERC and E-DMAC hardware.
- Setup the link maximum IP layer transfer unit.
- See if we can honor the NX_LINK_ENABLE request.
- Enable the interrupts, at the interrupt controller and Ethernet controller.
- Make sure we are in the right state to do the NX_LINK_DISABLE.
- Disable receive and transmit.
- Clear link enabled flag.

- Clear the enabled flag since there is no-one else.
- Free the packet that we will not send.
- Process driver send packet. Place the Ethernet frame at the front of the packet.
- Adjust packet length.
- Setup the prepend pointer in order to build the Ethernet frame. Backup another 2 bytes to get 32-bit word alignment.
- Build the actual Ethernet frame.
- Use MAC broadcast for this frame.
- Build the sender's MAC access.
- Set the Ethernet frame type.
- Endian swapping if necessary.
- Put the frame on the wire.
- Enable multicast.
- The device automatically receives multicast packets. Nothing needs to be done.
- Process driver deferred requests. Process a device driver function on behalf of the IP thread.
- Return the unhandled command status.
- Return NULL in the supplied return pointer.

8.29.13 nx_ether_interrupt

```
nx_ether_interrupt ( NX_REC  * nx_rec_ptr )
```

8.29.13.1 Brief description

nx_ether_interrupt

8.29.13.2 Detailed description

This function is the main Renesas Ethernet interrupt handler.

Table 746:Parameters

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure

8.29.13.3 Function steps

- Read EDMAC and EtherC status register.
- Clear all interrupts.

- Frame transmit completed interrupt The MAC sets Transmit Complete flag only if all frames are transmitted. This creates a delay in processing transmitted frames. The work around: this driver process transmitted packets when frame receive interrupt occurs. This way the transmitted packets can be released before processing received packets, reducing the delay.
- Frame received.
- Special case for ECI interrupt + link change.
- To report link status changes to the application, add a semaphore put or event flag set to the statement below.
- Link present.

8.29.14 nx_ether_send

```
nx_ether_send ( NX_PACKET  * packet ,  NX_REC  * nx_rec_ptr )
```

8.29.14.1 Brief description

nx_ether_send

8.29.14.2 Detailed description

This function decides whether to queue or send the packet

Table 747:Parameters

Name	Direction	Description
packet	in	: Pointer to the data packet
nx_rec_ptr	in	: Pointer to Ethernet record structure

NX_SUCCESS : Success NX_TX_QUEUE_DEPTH : Queue depth NX_NOT_ENABLED : Link is not enabled See NX user manual for all possible return values

8.29.14.3 Function steps

- Check if the link is enabled.
- Call write function to send the packet, then check the result.
- Transmit failed, drop the packet.

8.29.15 nx_ether_write

```
nx_ether_write ( NX_PACKET  * pkt ,  NX_REC  * nx_rec_ptr )
```

8.29.15.1 Brief description

nx_ether_write

8.29.15.2 Detailed description

This function writes the data to the TX FIFO buffer descriptor. This function can only be called when there is no active transmission.

Table 748:Parameters

Name	Direction	Description
packet	in	: Pointer to the data packet
nx_rec_ptr	in	: Pointer to Ethernet record structure

NX_SUCCESS : Success NX_NO_MORE_ENTRIES : No more entries(BD's) allowed See NX user manual for all possible return values

NOTE: The pointer parameters passed to this function are already validated at higher level.

8.29.15.3 Function steps

- Check if the BD is free.
- Set up a transfer descriptor for each NX_PACKET in the packet chain.
- Special case: alignment requirement to 5 bits for data 16 bytes or less. To accommodate this special case, NX_PACKETs need to be minimum 32+16 in size.
- Check alignment for packets 16 bytes or less.
- Loop from back to front to copy data in the NX_PACKET.
- Set new prepend and append pointers.
- Set up the transmit BD.
- This was the last packet in the chain. Store the original packet pointer in the last BD.
- Get next NetX packet in the chain.
- Move to next BD.
- Disable interrupts.
- Restore interrupts.
- Set active flag to indicate BDs are ready.
- Set active flag in reverse order, move to previous BD.
- Set active flag.
- If the transmission is suspended, resume transmission.

- Start transmission.

8.29.16 nx_storage_init

```
nx_storage_init ( NX_IP  * ip_ptr ,  NX_REC  * nx_rec_ptr )
```

8.29.16.1 Brief description

nx_storage_init

8.29.16.2 Detailed description

This function initializes byte pool, packet pool, list of descriptors and Tx & Rx FIFO.

Table 749:Parameters

Name	Direction	Description
ip_ptr	in	Pointer to IP protocol block
nx_rec_ptr	in	Pointer to Ethernet record structure

NX_SUCCESS Successful send NX_NO_PACKET No packet available NX_WAIT_ABORTED Requested suspension was aborted by a call to tx_thread_wait_abort. See NX user manual for all possible return values

NOTE: The pointer parameters passed to this function are already validated at higher level.

8.29.16.3 Function steps

- All memory is assumed to be allocated at link time in the .bss section to avoid memory allocation errors at runtime.
- 16 byte alignment for BDs.
- Initialize receive buffer descriptors.
- Allocate packets from the pool to be used for reception.
- Check if the allocation was successful.
- Release allocated packets.
- Set buffer length, payload address (32byte align), initial status.
- Length has to be multiple of 32.
- Set DL bit in last descriptor.

8.29.17 nx_tx_interrupt

```
nx_tx_interrupt ( NX_REC  * nx_rec_ptr )
```

8.29.17.1 Brief description

nx_tx_interrupt

8.29.17.2 Detailed description

This function handles the Tx done interrupt.

Table 750:Parameters

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure

NOTE: The pointer parameter passed to this function is already validated at the higher level.

8.29.17.3 Function steps

- Loop through buffers in use.
- Check if the BD is in non active state.
- Check if this is the last BD in a chain.
- Release packet transmitted.
- Move to next BD.

8.29.18 nx_rx_interrupt

```
nx_rx_interrupt ( NX_REC  * nx_rec_ptr )
```

8.29.18.1 Brief description

nx_rx_interrupt

8.29.18.2 Detailed description

This function handles the receive interrupt.

Table 751:Parameters

Name	Direction	Description
nx_rec_ptr	in	: Pointer to Ethernet record structure

NOTE: The pointer parameter passed to this function is already validated at the higher level.

8.29.18.3 Function steps

- Do for every complete frame in the receive buffer.
- Each frame consists of 1 or more BDs.
- The driver should be able to process at least 1 completed frame when it gets here.
- Get the number of BDs in this receive chain: Last BD has FRAME_POSITION_END set, BDs preceding should have DESCRIPTOR_FLAG_ACTIVE bit cleared.
- This frame has a BD that is still active and is not fully received, break out here.
- Move to next BD.
- Get first packet from first BD.
- Store the total frame length from the last BD + the padding in the NetX packet.
- Get the last packet in the chain, we need this to build NetX packet chain.
- For each packet in the chain, except the last one:
- Adjust the append pointer according to the size in BD.
- Update remaining_bytes.
- Chain packet (set _next pointer).
- Set nx_packet_last pointer to point to last packet in chain.
- Insert a new NetX packet into the BD.
- Set buffer length and address and align the address to 32 bytes boundary.
- Make the size to be multiple of 32.
- Check if we had some successful allocation.
- Release the packet.
- Set the BDs to active.
- Set buffer length and address. Align to 32 byte address
- Make the size to be multiple of 32.
- Check if we had allocation failures.
- Advance the index for next reception.
- Last BD or packet.
- Set append pointer for the last packet.
- Clear _next pointer.
- Insert a new NetX packet into the BD.
- Send the packet up the stack.
- Allocation failed. Assign last packet to the BD.
- Prepare the BD for receiving data.
- Set buffer length and address align to 32 byte address.

- Make the size to be multiple of 32.
- advance the index for next reception
- If reception is in suspended state, resume it.
- Resume reception.

8.29.19 nx_receive

```
nx_receive ( NX_IP  * ip_ptr ,  NX_PACKET  * packet_ptr )
```

8.29.19.1 Brief description

nx_receive

8.29.19.2 Detailed description

nx_rx_interrupt() This function processing incoming packets. This routine would be called from the receive packet ISR.

Table 752:Parameters

Name	Direction	Description
ip_ptr	in	: Pointer to IP protocol block
packet_ptr	in	: Packet pointer

NOTE: The pointer parameters passed to this function are already validated at higher level.

8.29.19.3 Function steps

- Pickup the packet header to determine where the packet needs to be sent.
- Clean off the Ethernet header.
- Adjust the packet length.
- Route the incoming packet according to its Ethernet type.
- If Ethernet header id invalid, release the packet.

8.29.20 nx_driver_deferred_processing

```
nx_driver_deferred_processing ( NX_IP_DRIVER  * driver_req_ptr ,  NX_REC
* nx_rec_ptr )
```

8.29.20.1 Brief description

nx_driver_deferred_processing

8.29.20.2 Detailed description

This function processing the deferred action within the context of the IP thread.

Table 753:Parameters

Name	Direction	Description
driver_req_ptr	in	: Driver command from the IP thread
nx_rec_ptr	in	: Pointer to Ethernet record structure

8.29.20.3 Function steps

- Check if we need to poll PHY status.
- If so, clear the polling flag.
- Mark request as successful.

8.29.21 nx_ethernet_version_get

```
ssp_err_t nx_ethernet_version_get ( ssp_version_t  *const p_version )
```

8.29.21.1 Brief description

Retrieve the API version number.

8.29.21.2 Detailed description

Table 754:Return values

Name	Description
SSP_SUCCESS	Successful return.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

8.29.21.3 Function steps

- Return the version number

8.30 NetX Synergy Port PHY Driver

8.30.1 Functions

- Phy_Init
- Phy_Start_Autonegotiate
- Phy_Get_Autonegotiate
- Phy_GetLinkStatus

8.30.2 Phy_Init

```
Phy_Init (  uint32_t channel )
```

8.30.2.1 Brief description

Phy_Init - Initialize Ethernet PHY device.

8.30.2.2 Detailed description

Table 755:Parameters

Name	Direction	Description
channel	in	Ethernet channel number

Table 756:Return values

Name	Description
R_PHY_OK	PHY device is initialized successfully
R_PHY_ERROR	PHY device is not initialized successfully

NOTE: Refer HW manual for valid channels for a target MCU

8.30.2.3 Function steps

- Resets PHY device
- Waits the reset completion, PHY_CONTROL_RESET bit is self-cleared after '1' is written to it.

- When MICREL_KSZ8091RNB of the Micrel, Inc. is used, the pin that outputs the state of LINK is used combinedly with ACTIVITY in default. The setting of the pin is changed so that only the state of LINK is output. Set Clock Mode to 50MHz
- Sets Duplex Mode as Full-duplex

8.30.3 Phy_Start_Autonegotiate

```
Phy_Start_Autonegotiate (  uint32_t channel ,   uint8_t pause )
```

8.30.3.1 Brief description

Sets Auto-Negotiation advertisement and starts auto-negotiation.

8.30.3.2 Detailed description

Table 757:Parameters

Name	Direction	Description
channel	in	Ethernet channel number
pause	in	Using state of pause frames

Table 758:Return values

Name	Description
void	

8.30.4 Phy_Get_Autonegotiate

```
Phy_Get_Autonegotiate (  uint32_t channel ,   uint16_t * p_line_speed_duplex ,
 uint16_t * p_local_pause ,   uint16_t * p_partner_pause )
```

8.30.4.1 Brief description

Gets capabilities of an Ethernet PHY device.

8.30.4.2 Detailed description

Table 759:Parameters

Name	Direction	Description
channel	in	Ethernet channel number
p_line_speed_duplex	in	A pointer to the location of both the line speed and the duplex
p_local_pause	in	A pointer to the location to store the local pause bits
p_partner_pause	in	A pointer to the location to store the partner pause bits

Table 760:Return values

Name	Description
R_PHY_OK	Got information successfully
R_PHY_ERROR	PHY device is yet to be initialized

NOTE: The value returned to local_pause and patner_pause is used as it is as an argument of ether_pause_resolution function.

NOTE: Validating parameters is not required by this function as it by design is only called by sf_el_nx framework with valid parameters.

8.30.4.3 Function steps

- Reads the status register. Because reading the first time shows the previous state, the Link status bit should be read twice.
- Checks the link status
- Check the auto-negotiation status
- Gets local pause capability

8.30.5 Phy_GetLinkStatus

```
Phy_GetLinkStatus (  uint32_t channel )
```

8.30.5.1 Brief description

Phy_GetLinkStatus - Returns the status of the physical link.

8.30.5.2 Detailed description

Table 761:Parameters

Name	Direction	Description
channel	in	Ethernet channel number

Table 762:Return values

Name	Description
R_PHY_OK	PHY device is initialized successfully
R_PHY_ERROR	PHY device is not initialized successfully

Chapter 9 API Reference: HAL Driver Interfaces

9.1 API Reference: HAL Interfaces

The HAL Interfaces offer common APIs for functional use cases. They can be implemented by one or more HAL layer drivers. Framework Layer drivers connect to these HAL drivers through the Interface Layer.

- ADC Interface
- CAC Interface
- CAN Interface
- CGC Interface
- CRC Interface
- Crypto Interface
- CTSU Interface
- DAC Interface
- Display Interface
- DOC Interface
- Events and peripheral definitions
- External IRQ Interface
- Flash Interface
- FMI Interface
- I2C Interface
- I2S Interface
- Input Capture Interface
- I/O Port Interface
- JPEG Decode Interface
- Key Matrix Interface
- Low Power Modes Interface
- Low Power Modes V2 Interface
- Low Voltage Detection Interface
- PDC Interface

- Quad SPI Flash Interface
- RTC Interface
- SD/MMC Interface
- SLCDC Interface
- SPI Interface
- Timer Interface
- Transfer Interface
- UART Interface
- WDT Interface

9.2 ADC Interface

Interface for A/D Converters.

9.2.1 Summary

The ADC interface provides standard ADC functionality including one-shot mode (single scan), continuous scan and group scan. It also allows configuration of hardware and software triggers for starting scans. After each conversion an interrupt can be triggered, and if a callback function is provided, the call back is invoked with the appropriate event information.

Implemented by: ADC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

ADC Interface description: ADC Driver

9.2.2 Interface API

adc_api_t

Function name	Description
.open	Initialize ADC Unit; apply power, set the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors.

Function name	Description
.scanCfg	Configure the scan including the channels, groups and scan triggers to be used for the unit that was initialized in the open call.
.scanStart	Start the scan (in case of a software trigger), or enable the hardware trigger.
.scanStop	Stop the ADC scan (in case of a software trigger), or disable the hardware trigger.
.scanStatusGet	Check scan status.
.read	Read ADC conversion result(s).
.sampleStateCountSet	Set the sample state count for the specified channel.
.close	Close the specified ADC unit by ending any scan in progress, disabling interrupts, and removing power to the specified A/D unit.
.infoGet	Return the ADC data register address of the first (lowest number) channel and the total number of bytes to be read in order for the DTC/DMAC to read the conversion results of all configured channels. Return the temperature sensor calibration and slope data.
.versionGet	Retrieve the API version.

9.2.3 Data structures

- adc_sample_state_t
- adc_callback_args_t
- adc_info_t
- adc_channel_cfg_t
- adc_cfg_t
- adc_instance_t

9.2.4 Enumerations

- adc_mode_t
- adc_resolution_t
- adc_alignment_t

- adc_add_t
- adc_clear_t
- adc_trigger_t
- adc_sample_state_reg_t
- adc_cb_event_t
- adc_group_a_t
- adc_register_t

9.2.5 Typedefs

- adc_data_size_t
- adc_ctrl_t

9.2.6 Defines

- #define ADC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Includes board and MCU related header files. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version Number of API.
- #define ADC_API_VERSION_MINOR

  Initial value:`(5U)`

9.2.7 API Data

9.2.7.1 adc_mode_t

`adc_mode_t`

Detailed description

ADC operation mode definitions

Enumerated values

Name	Description
ADC_MODE_SINGLE_SCAN	Single scan - one or more channels.
ADC_MODE_GROUP_SCAN	Two trigger sources to trigger scan for two groups which contain one or more channels.

Name	Description
ADC_MODE_CONTINUOUS_SCAN	Continuous scan - one or more channels.

9.2.7.2 adc_resolution_t

`adc_resolution_t`

Detailed description

ADC data resolution definitions

Enumerated values

Name	Description
ADC_RESOLUTION_12_BIT	12 bit resolution
ADC_RESOLUTION_10_BIT	10 bit resolution
ADC_RESOLUTION_8_BIT	8 bit resolution
ADC_RESOLUTION_14_BIT	14 bit resolution

9.2.7.3 adc_alignment_t

`adc_alignment_t`

Detailed description

ADC data alignment definitions

Enumerated values

Name	Description
ADC_ALIGNMENT_RIGHT	Data alignment right.
ADC_ALIGNMENT_LEFT	Data alignment left.

9.2.7.4 adc_add_t

`adc_add_t`

Detailed description

ADC data sample addition and averaging options

Enumerated values

Name	Description
ADC_ADD_OFF	Addition turned off for channels/sensors.
ADC_ADD_TWO	Add two samples.
ADC_ADD_THREE	Add three samples.
ADC_ADD_FOUR	Add four samples.
ADC_ADD_SIXTEEN	Add sixteen samples.
ADC_ADD_AVERAGE_TWO	Average two samples.
ADC_ADD_AVERAGE_FOUR	Average four samples.
ADC_ADD_AVERAGE_SIXTEEN	Add sixteen samples.

9.2.7.5 adc_clear_t

`adc_clear_t`

Detailed description

ADC clear after read definitions

Enumerated values

Name	Description
ADC_CLEAR_AFTER_READ_OFF	Clear after read off.
ADC_CLEAR_AFTER_READ_ON	Clear after read on.

9.2.7.6 adc_trigger_t

`adc_trigger_t`

Detailed description

ADC trigger mode definitions

Enumerated values

Name	Description
ADC_TRIGGER_ASYNC_EXT_TRG0	External asynchronous trigger; not for group modes.

Name	Description
ADC_TRIGGER_SYNC_ELC	Synchronous trigger via ELC.
ADC_TRIGGER_SOFTWARE	Software trigger; not for group modes.

9.2.7.7 adc_sample_state_reg_t

`adc_sample_state_reg_t`

Detailed description

ADC sample state registers

Enumerated values

Name	Description
ADC_SAMPLE_STATE_CHANNEL_0	Sample state register channel 0.
ADC_SAMPLE_STATE_CHANNEL_1	Sample state register channel 1.
ADC_SAMPLE_STATE_CHANNEL_2	Sample state register channel 2.
ADC_SAMPLE_STATE_CHANNEL_3	Sample state register channel 3.
ADC_SAMPLE_STATE_CHANNEL_4	Sample state register channel 4.
ADC_SAMPLE_STATE_CHANNEL_5	Sample state register channel 5.
ADC_SAMPLE_STATE_CHANNEL_6	Sample state register channel 6.
ADC_SAMPLE_STATE_CHANNEL_7	Sample state register channel 7.
ADC_SAMPLE_STATE_CHANNEL_8	Sample state register channel 8.
ADC_SAMPLE_STATE_CHANNEL_9	Sample state register channel 9.
ADC_SAMPLE_STATE_CHANNEL_10	Sample state register channel 10.
ADC_SAMPLE_STATE_CHANNEL_11	Sample state register channel 11.
ADC_SAMPLE_STATE_CHANNEL_12	Sample state register channel 12.
ADC_SAMPLE_STATE_CHANNEL_13	Sample state register channel 13.
ADC_SAMPLE_STATE_CHANNEL_14	Sample state register channel 14.
ADC_SAMPLE_STATE_CHANNEL_15	Sample state register channel 15.

Name	Description
ADC_SAMPLE_STATE_CHANNEL_16_TO_21	Sample state register channel 16 to 21 for unit 0 on S7G2.
ADC_SAMPLE_STATE_CHANNEL_16_TO_20	Sample state register channel 16 to 20 for unit 1 on S7G2.
ADC_SAMPLE_STATE_CHANNEL_16_TO_27	Sample state register channel 16 to 27 for unit 0 on S3A7.
ADC_SAMPLE_STATE_TEMPERATURE	Sample state register channel temperature.
ADC_SAMPLE_STATE_VOLTAGE	Sample state register channel voltage.

9.2.7.8 adc_cb_event_t

`adc_cb_event_t`

Detailed description

ADC callback event definitions

Enumerated values

Name	Description
ADC_EVENT_SCAN_COMPLETE	Normal/Group A scan complete.
ADC_EVENT_SCAN_COMPLETE_GROUP_B	Group B scan complete.

9.2.7.9 adc_group_a_t

`adc_group_a_t`

Detailed description

ADC action for group A interrupts group B scan. This enumeration is used to specify the priority between Group A and B in group mode.

Enumerated values

Name	Description
ADC_GROUP_A_PRIORITY_OFF	Group A ignored and does not interrupt ongoing group B scan.
ADC_GROUP_A_GROUP_B_WAIT_FOR_TRIGGER	Group A interrupts Group B(single scan) which restarts at next Group B trigger.
ADC_GROUP_A_GROUP_B_RESTART_SCAN	Group A interrupts Group B(single scan) which restarts immediately after Group A scan is complete.

Name	Description
ADC_GROUP_A_GROUP_B_CONTINUOUS_SCAN	Group A interrupts Group B(continuous scan) which continues scanning without a new Group B trigger.

9.2.7.10 adc_register_t

`adc_register_t`

Detailed description

ADC registers used for the Read() argument

Enumerated values

Name	Description
ADC_REG_CHANNEL_0	ADC channel 0.
ADC_REG_CHANNEL_1	ADC channel 1.
ADC_REG_CHANNEL_2	ADC channel 2.
ADC_REG_CHANNEL_3	ADC channel 3.
ADC_REG_CHANNEL_4	ADC channel 4.
ADC_REG_CHANNEL_5	ADC channel 5.
ADC_REG_CHANNEL_6	ADC channel 6.
ADC_REG_CHANNEL_7	ADC channel 7.
ADC_REG_CHANNEL_8	ADC channel 8.
ADC_REG_CHANNEL_9	ADC channel 9.
ADC_REG_CHANNEL_10	ADC channel 10.
ADC_REG_CHANNEL_11	ADC channel 11.
ADC_REG_CHANNEL_12	ADC channel 12.
ADC_REG_CHANNEL_13	ADC channel 13.
ADC_REG_CHANNEL_14	ADC channel 14.
ADC_REG_CHANNEL_15	ADC channel 15.

Name	Description
ADC_REG_CHANNEL_16	ADC channel 16.
ADC_REG_CHANNEL_17	ADC channel 17.
ADC_REG_CHANNEL_18	ADC channel 18.
ADC_REG_CHANNEL_19	ADC channel 19.
ADC_REG_CHANNEL_20	ADC channel 20.
ADC_REG_CHANNEL_21	ADC channel 21.
ADC_REG_CHANNEL_22	ADC channel 22.
ADC_REG_CHANNEL_23	ADC channel 23.
ADC_REG_CHANNEL_24	ADC channel 24.
ADC_REG_CHANNEL_25	ADC channel 25.
ADC_REG_CHANNEL_26	ADC channel 26.
ADC_REG_CHANNEL_27	ADC channel 27.
ADC_REG_TEMPERATURE	ADC channel temperature.
ADC_REG_VOLT	ADC channel volt.

9.2.7.11 adc_data_size_t

```
typedef uint16_t adc_data_size_t
```

Detailed description

Largest supported ADC resolution size.

9.2.7.12 adc_ctrl_t

```
typedef void adc_ctrl_t
```

Detailed description

ADC control block. Allocate using driver instance control structure from driver instance header file.

9.2.8 API Structures

9.2.8.1 adc_sample_state_t

adc_sample_state_t

Detailed description

ADC sample state configuration

Variables

- adc_sample_state_reg_t reg_id

  Sample state register ID.

- uint8_t num_states

  Number of sampling states for conversion. Ch16-20/21 use the same value.

9.2.8.2 adc_callback_args_t

adc_callback_args_t

Detailed description

ADC callback arguments definitions

Variables

- uint16_t unit

  ADC device in use.

- adc_cb_event_t event

  ADC callback event.

- void const * p_context

  Placeholder for user data.

9.2.8.3 adc_info_t

adc_info_t

Detailed description

ADC Information Structure for Transfer Interface

Variables

- __I uint16_t * p_address

  The address to start reading the data from.

- uint32_t length

  The total number of bytes to read.

- elc_peripheral_t elc_peripheral

  Name of the peripheral in the ELC list.

- elc_event_t elc_event

  Name of the ELC event for the peripheral.

- uint32_t calibration_data

  Temperature sensor calibration data (0xFFFFFFFF if unsupported). Refer to hardware manual for steps on using slope with calibration data to determine temperature

- int16_t slope_microvolts

  Temperature sensor slope in microvolts/°C.

9.2.8.4 adc_channel_cfg_t

adc_channel_cfg_t

Detailed description

ADC channel(s) configuration

Variables

- uint32_t scan_mask

  Channels/bits: bit 0 is ch0; bit 15 is ch15. Use #define ADC_MASK_xxx from r_adc.h.

- uint32_t scan_mask_group_b

  Valid for group modes. Use #define ADC_MASK_xxx from r_adc.h.

- adc_group_a_t priority_group_a

  Valid for group modes.

- uint32_t add_mask

  Valid if add enabled in Open(). Use #define ADC_MASK_xxx from r_adc.h.

- uint8_t sample_hold_mask

  Channels/bits 0-2. Use #define ADC_MASK_xxx from r_adc.h.

- uint8_t sample_hold_states

  Number of states to be used for sample and hold. Affects channels 0-2.

9.2.8.5 adc_cfg_t

adc_cfg_t

Detailed description

ADC general configuration

Variables

- uint16_t unit

  ADC Unit to be used.
- adc_mode_t mode

  ADC operation mode.
- adc_resolution_t resolution

  ADC resolution 8, 10, or 12-bit.
- adc_alignment_t alignment

  Specify left or right alignment; ignored if addition used.
- adc_add_t add_average_count

  Add or average samples.
- adc_clear_t clearing

  Clear after read.
- adc_trigger_t trigger

  Default and Group A trigger source.
- adc_trigger_t trigger_group_b

  Group B trigger source; valid only for group mode.
- uint8_t scan_end_ipl

  Scan end interrupt priority.
- uint8_t scan_end_b_ipl

  Scan end group B interrupt priority.
- void(* p_callback)(adc_callback_args_t *p_args)

  Callback function; set to NULL for none.
- void const * p_context

  Placeholder for user data. Passed to the user callback in adc_api_t::adc_callback_args_t.
- void const * p_extend

  Extension parameter for hardware specific settings.

9.2.8.6 adc_api_t

adc_api_t

Detailed description

ADC functions implemented at the HAL layer will follow this API.

9.2.8.7 open

```
ssp_err_t(* adc_api_t::open) (adc_ctrl_t *const p_ctrl, adc_cfg_t const *const
p_cfg)
```

Detailed description

Initialize ADC Unit; apply power, set the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors. Implemented as

- R_ADC_Open

NOTE: Configure peripheral clocks, ADC pins and IRQs prior to calling this function.

Table 763:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

Parameter p_cfg

Definition: `adc_cfg_t const *const p_cfg`

ADC general configuration

- `adc_cfg_t`::`unit`

  ADC Unit to be used.

- `adc_cfg_t`::`adc_mode_t`

  ADC operation mode.

  Enumerated as:

  – ADC_MODE_SINGLE_SCAN

  – ADC_MODE_GROUP_SCAN

  – ADC_MODE_CONTINUOUS_SCAN

- `adc_cfg_t`::`adc_resolution_t`

  ADC resolution 8, 10, or 12-bit.

  Enumerated as:

  – ADC_RESOLUTION_12_BIT

  – ADC_RESOLUTION_10_BIT

  - ADC_RESOLUTION_8_BIT
  - ADC_RESOLUTION_14_BIT
- adc_cfg_t::adc_alignment_t

  Specify left or right alignment; ignored if addition used.

  Enumerated as:
  - ADC_ALIGNMENT_RIGHT
  - ADC_ALIGNMENT_LEFT
- adc_cfg_t::adc_add_t

  Add or average samples.

  Enumerated as:
  - ADC_ADD_OFF
  - ADC_ADD_TWO
  - ADC_ADD_THREE
  - ADC_ADD_FOUR
  - ADC_ADD_SIXTEEN
  - ADC_ADD_AVERAGE_TWO
  - ADC_ADD_AVERAGE_FOUR
  - ADC_ADD_AVERAGE_SIXTEEN
- adc_cfg_t::adc_clear_t

  Clear after read.

  Enumerated as:
  - ADC_CLEAR_AFTER_READ_OFF
  - ADC_CLEAR_AFTER_READ_ON
- adc_cfg_t::adc_trigger_t

  Default and Group A trigger source.

  Enumerated as:
  - ADC_TRIGGER_ASYNC_EXT_TRG0
  - ADC_TRIGGER_SYNC_ELC
  - ADC_TRIGGER_SOFTWARE

- adc_cfg_t::adc_trigger_t

  Group B trigger source; valid only for group mode.

  Enumerated as:

  – ADC_TRIGGER_ASYNC_EXT_TRG0

  – ADC_TRIGGER_SYNC_ELC

  – ADC_TRIGGER_SOFTWARE
- adc_cfg_t::scan_end_ipl

  Scan end interrupt priority.
- adc_cfg_t::scan_end_b_ipl

  Scan end group B interrupt priority.
- adc_cfg_t::p_callback

  Callback function; set to NULL for none.
- adc_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in adc_api_t::adc_callback_args_t.
- adc_cfg_t::p_extend

  Extension parameter for hardware specific settings.

9.2.8.8 scanCfg

```
ssp_err_t(* adc_api_t::scanCfg) (adc_ctrl_t *const p_ctrl, adc_channel_cfg_t
const *const p_channel_cfg)
```

Detailed description

Configure the scan including the channels, groups and scan triggers to be used for the unit that was initialized in the open call. Implemented as

- R_ADC_ScanConfigure

Table 764:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_channel_cfg	in	Pointer to scan configuration structure

Parameter p_ctrl

Definition: adc_ctrl_t*const p_ctrl

ADC control block. Allocate using driver instance control structure from driver instance header file.

Parameter p_channel_cfg

Definition: `adc_channel_cfg_t const *const p_channel_cfg`

ADC channel(s) configuration

- `adc_channel_cfg_t`::`scan_mask`

  Channels/bits: bit 0 is ch0; bit 15 is ch15. Use #define ADC_MASK_xxx from r_adc.h.

- `adc_channel_cfg_t`::`scan_mask_group_b`

  Valid for group modes. Use #define ADC_MASK_xxx from r_adc.h.

- `adc_channel_cfg_t`::`adc_group_a_t`

  Valid for group modes.

  Enumerated as:

  – ADC_GROUP_A_PRIORITY_OFF

  – ADC_GROUP_A_GROUP_B_WAIT_FOR_TRIGGER

  – ADC_GROUP_A_GROUP_B_RESTART_SCAN

  – ADC_GROUP_A_GROUP_B_CONTINUOUS_SCAN

- `adc_channel_cfg_t`::`add_mask`

  Valid if add enabled in Open(). Use #define ADC_MASK_xxx from r_adc.h.

- `adc_channel_cfg_t`::`sample_hold_mask`

  Channels/bits 0-2. Use #define ADC_MASK_xxx from r_adc.h.

- `adc_channel_cfg_t`::`sample_hold_states`

  Number of states to be used for sample and hold. Affects channels 0-2.

9.2.8.9 scanStart

`ssp_err_t(* adc_api_t::scanStart) (adc_ctrl_t *const p_ctrl)`

Detailed description

Start the scan (in case of a software trigger), or enable the hardware trigger. Implemented as

- R_ADC_ScanStart

Table 765:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

9.2.8.10 scanStop

```
ssp_err_t(* adc_api_t::scanStop) (adc_ctrl_t *const p_ctrl)
```

Detailed description

Stop the ADC scan (in case of a software trigger), or disable the hardware trigger. Implemented as

- R_ADC_ScanStop

Table 766:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

9.2.8.11 scanStatusGet

```
ssp_err_t(* adc_api_t::scanStatusGet) (adc_ctrl_t *const p_ctrl)
```

Detailed description

Check scan status. Implemented as

- R_ADC_CheckScanDone

Table 767:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

9.2.8.12 read

```
ssp_err_t(* adc_api_t::read) (adc_ctrl_t *const p_ctrl, adc_register_t const
reg_id, adc_data_size_t *const p_data)
```

Detailed description

Read ADC conversion result(s). Implemented as

- R_ADC_Read

Table 768:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
reg_id	in	Id for the register to read (see enumeration adc_register_t)
p_data	in	Pointer to variable to load value into.

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

Parameter reg_id

Definition: `adc_register_tconst reg_id`

ADC registers used for the Read() argument

Parameter p_data

Definition: `adc_data_size_t*const p_data`

Largest supported ADC resolution size.

9.2.8.13 sampleStateCountSet

```
ssp_err_t(* adc_api_t::sampleStateCountSet) (adc_ctrl_t *const p_ctrl,
adc_sample_state_t *p_sample)
```

Detailed description

Set the sample state count for the specified channel. Implemented as

- R_ADC_SetSampleStateCount

Table 769:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_sample	in	Pointer to the ADC channels and corresponding sample states to be set

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

Parameter p_sample

Definition: `adc_sample_state_t*p_sample`

ADC sample state configuration

- `adc_sample_state_t::reg_id`

  Sample state register ID.

- `adc_sample_state_t::num_states`

  Number of sampling states for conversion. Ch16-20/21 use the same value.

9.2.8.14 close

```
ssp_err_t(* adc_api_t::close) (adc_ctrl_t *const p_ctrl)
```

Detailed description

Close the specified ADC unit by ending any scan in progress, disabling interrupts, and removing power to the specified A/D unit. Implemented as

- R_ADC_Close

Table 770:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

9.2.8.15 infoGet

```
ssp_err_t(* adc_api_t::infoGet) (adc_ctrl_t *const p_ctrl, adc_info_t *const
p_adc_info)
```

Detailed description

Return the ADC data register address of the first (lowest number) channel and the total number of bytes to be read in order for the DTC/DMAC to read the conversion results of all configured channels. Return the temperature sensor calibration and slope data. Implemented as

- R_ADC_InfoGet

Table 771:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Table 771:Parameters (Continued)

Name	Direction	Description
p_adc_info	out	Pointer to ADC information structure

Parameter p_ctrl

Definition: `adc_ctrl_t*const p_ctrl`

ADC control block. Allocate using driver instance control structure from driver instance header file.

Parameter p_adc_info

Definition: `adc_info_t*const p_adc_info`

ADC Information Structure for Transfer Interface

- `adc_info_t::p_address`

  The address to start reading the data from.

- `adc_info_t::length`

  The total number of bytes to read.

- `adc_info_t::elc_peripheral`

  Name of the peripheral in the ELC list.

- `adc_info_t::elc_event`

  Name of the ELC event for the peripheral.

- `adc_info_t::calibration_data`

  Temperature sensor calibration data (0xFFFFFFFF if unsupported). Refer to hardware manual for steps on using slope with calibration data to determine temperature

- `adc_info_t::slope_microvolts`

  Temperature sensor slope in microvolts/°C.

9.2.8.16 versionGet

```
ssp_err_t(* adc_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Retrieve the API version. Implemented as

- R_ADC_VersionGet

NOTE: This function retrieves the API version.

Table 772:Parameters

Name	Direction	Description
p_version	in	Pointer to version structure

Parameter p_version

9.2.8.17 adc_instance_t

adc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- adc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- adc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- adc_channel_cfg_t const * p_channel_cfg

  Pointer to the channel configuration structure for this instance.

- adc_api_t const * p_api

  Pointer to the API structure for this instance.

9.3 CAC Interface

Interface for clock frequency accuracy measurements.

Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

9.3.1 Summary

The interface for the clock frequency accuracy measurement circuit (CAC) peripheral is used to check a system clock frequency with a reference clock signal by counting the number of pulses of the clock (system clock) to be measured.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CAC Interface description: CAC Driver

9.3.2 Interface API

cac_api_t

Function name	Description
.open	Open function for CAC device.
.read	Read function for CAC peripheral.
.close	Close function for CAC device.
.stopMeasurement	End a measurement for the CAC peripheral.
.startMeasurement	Begin a measurement for the CAC peripheral.
.reset	Reset function for CAC device.
.versionGet	Get the CAC API and code version information.

9.3.3 Data structures

- cac_ref_clock_config_t
- cac_meas_clock_config_t
- cac_callback_args_t
- cac_cfg_t
- cac_instance_t

9.3.4 Enumerations

- cac_event_t
- cac_clock_type_t
- cac_clock_source_t
- cac_ref_divider_t
- cac_ref_digfilter_t
- cac_ref_edge_t
- cac_meas_divider_t

9.3.5 Typedefs

- cac_ctrl_t

9.3.6 Defines

- #define CAC_API_VERSION_MAJOR
  Initial value:`(1U)`
- #define CAC_API_VERSION_MINOR
  Initial value:`(2U)`

9.3.7 API Data

9.3.7.1 cac_event_t

`cac_event_t`

Detailed description

Event types returned by the ISR callback when used in CAC interrupt mode

Enumerated values

Name	Description
CAC_EVENT_FREQUENCY_ERROR	Frequency error.
CAC_EVENT_MEASUREMENT_COMPLETE	Measurement complete.
CAC_EVENT_COUNTER_OVERFLOW	Counter overflow.

9.3.7.2 cac_clock_type_t

`cac_clock_type_t`

Detailed description

Enumeration of the two possible clocks.

Enumerated values

Name	Description
CAC_CLOCK_MEASURED	Measurement clock.

Name	Description
CAC_CLOCK_REFERENCE	Reference clock.

9.3.7.3 cac_clock_source_t

`cac_clock_source_t`

Detailed description

Enumeration of the possible clock sources for both the reference and measurement clocks.

Enumerated values

Name	Description
CAC_CLOCK_SOURCE_MAIN_OSC	Main clock oscillator.
CAC_CLOCK_SOURCE_SUBCLOCK	Sub-clock.
CAC_CLOCK_SOURCE_HOCO	HOCO (High speed on chip oscillator)
CAC_CLOCK_SOURCE_MOCO	MOCO (Middle speed on chip oscillator)
CAC_CLOCK_SOURCE_LOCO	LOCO (Middle speed on chip oscillator)
CAC_CLOCK_SOURCE_PCLKB	PCLKB (Peripheral Clock B)
CAC_CLOCK_SOURCE_IWDT	IWDT- Dedicated on-chip oscillator.
CAC_CLOCK_SOURCE_MEAS_MAX	Maximum measured clock.
CAC_CLOCK_SOURCE_EXTERNAL	Externally supplied measurement clock on CACREF pin.
CAC_CLOCK_SOURCE_REF_MAX	Maximum reference clock.

9.3.7.4 cac_ref_divider_t

`cac_ref_divider_t`

Detailed description

Enumeration of available dividers for the reference clock.

Enumerated values

Name	Description
CAC_REF_DIV_32	Reference clock divided by 32.
CAC_REF_DIV_128	Reference clock divided by 128.
CAC_REF_DIV_1024	Reference clock divided by 1024.
CAC_REF_DIV_8192	Reference clock divided by 8192.

9.3.7.5 cac_ref_digfilter_t

`cac_ref_digfilter_t`

Detailed description

Enumeration of available digital filter settings for an external reference clock.

Enumerated values

Name	Description
CAC_REF_DIGITAL_FILTER_OFF	No digital filter on the CACREF pin for reference clock.
CAC_REF_DIGITAL_FILTER_1	Sampling clock for digital filter = measuring frequency.
CAC_REF_DIGITAL_FILTER_4	Sampling clock for digital filter = measuring frequency/4.
CAC_REF_DIGITAL_FILTER_16	Sampling clock for digital filter = measuring frequency/16.

9.3.7.6 cac_ref_edge_t

`cac_ref_edge_t`

Detailed description

Enumeration of available edge detect settings for the reference clock.

Enumerated values

Name	Description
CAC_REF_EDGE_RISE	Rising edge detect for the Reference clock.
CAC_REF_EDGE_FALL	Falling edge detect for the Reference clock.

Name	Description
CAC_REF_EDGE_BOTH	Both Rising and Falling edges detect for the Reference clock.

9.3.7.7 cac_meas_divider_t

```
cac_meas_divider_t
```

Detailed description

Enumeration of available dividers for the measurement clock

Enumerated values

Name	Description
CAC_MEAS_DIV_1	Measurement clock divided by 1.
CAC_MEAS_DIV_4	Measurement clock divided by 4.
CAC_MEAS_DIV_8	Measurement clock divided by 8.
CAC_MEAS_DIV_32	Measurement clock divided by 32.

9.3.7.8 cac_ctrl_t

```
typedef void cac_ctrl_t
```

Detailed description

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented as

- cac_instance_ctrl_t

9.3.8 API Structures

9.3.8.1 cac_ref_clock_config_t

cac_ref_clock_config_t

Detailed description

Structure defining the settings that apply to reference clock configuration.

Variables

- cac_ref_divider_t divider

  Divider specification for the Reference clock.

- cac_clock_source_t clock

  Clock source for the Reference clock.
- cac_ref_digfilter_t digfilter

  Digital filter selection for the CACREF ext clock.
- cac_ref_edge_t edge

  Edge detection for the Reference clock.

9.3.8.2 cac_meas_clock_config_t

cac_meas_clock_config_t

Detailed description

Structure defining the settings that apply to measurement clock configuration.

Variables

- cac_meas_divider_t divider

  Divider specification for the Measurement clock.
- cac_clock_source_t clock

  Clock source for the Measurement clock.

9.3.8.3 cac_callback_args_t

cac_callback_args_t

Detailed description

Callback function parameter data

Variables

- cac_event_t event

  The event can be used to identify what caused the callback (cac ready or error).
- void const * p_context

  Placeholder for user data. Set in cac_api_t::open function in cac_cfg_t.

9.3.8.4 cac_cfg_t

cac_cfg_t

Detailed description

CAC Configuration

Variables

- cac_ref_clock_config_t cac_ref_clock

  reference clock specific settings

- cac_meas_clock_config_t cac_meas_clock
  measurement clock specific settings
- uint16_t cac_upper_limit
  the upper limit counter threshold
- uint16_t cac_lower_limit
  the lower limit counter threshold
- bool mei_interrupt_enabled
  True if Measurement Complete interrupt is enabled.
- bool ovf_interrupt_enabled
  True if Overflow interrupt is enabled.
- bool ferr_interrupt_enabled
  True if Frequency Error interrupt is enabled.
- bool continuous_mode
  True if measurement continuously restarts after completing.
- uint8_t frequency_error_ipl
  Frequency error interrupt priority.
- uint8_t measurement_end_ipl
  Measurement end interrupt priority.
- uint8_t overflow_ipl
  Overflow interrupt priority.
- void(* p_callback)(cac_callback_args_t *p_args)
  Callback provided when a CAC interrupt ISR occurs.
- void const * p_extend
  CAC hardware dependent configuration */.
- void const * p_context
  Placeholder for user data. Passed to user callback in cac_callback_args_t.

9.3.8.5 cac_api_t

cac_api_t

Detailed description

CAC functions implemented at the HAL layer API

9.3.8.6 open

```
ssp_err_t(* cac_api_t::open) (cac_ctrl_t *const p_ctrl, cac_cfg_t const *const p_cfg)
```

Detailed description

Open function for CAC device.

Table 773:Parameters

Name	Direction	Description
p_ctrl	out	Pointer to CAC device control. Must be declared by user. Value set here.
cac_cfg_t	in	Pointer to CAC configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `cac_ctrl_t*const p_ctrl`

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

Parameter cac_cfg_t

Definition: `cac_cfg_t const *const p_cfg`

CAC Configuration

- `cac_cfg_t`::`cac_ref_clock_config_t`

  reference clock specific settings
- `cac_cfg_t`::`cac_meas_clock_config_t`

  measurement clock specific settings
- `cac_cfg_t`::`cac_upper_limit`

  the upper limit counter threshold
- `cac_cfg_t`::`cac_lower_limit`

  the lower limit counter threshold
- `cac_cfg_t`::`mei_interrupt_enabled`

  True if Measurement Complete interrupt is enabled.
- `cac_cfg_t`::`ovf_interrupt_enabled`

  True if Overflow interrupt is enabled.
- `cac_cfg_t`::`ferr_interrupt_enabled`

  True if Frequency Error interrupt is enabled.

- cac_cfg_t::continuous_mode

  True if measurement continuously restarts after completing.
- cac_cfg_t::frequency_error_ipl

  Frequency error interrupt priority.
- cac_cfg_t::measurement_end_ipl

  Measurement end interrupt priority.
- cac_cfg_t::overflow_ipl

  Overflow interrupt priority.
- cac_cfg_t::p_callback

  Callback provided when a CAC interrupt ISR occurs.
- cac_cfg_t::p_extend

  CAC hardware dependent configuration */.
- cac_cfg_t::p_context

  Placeholder for user data. Passed to user callback in cac_callback_args_t.

9.3.8.7 read

```
ssp_err_t(* cac_api_t::read) (cac_ctrl_t *const p_ctrl, uint8_t *const p_status,
uint16_t *const p_counter)
```

Detailed description

Read function for CAC peripheral.

Table 774:Parameters

Name	Direction	Description
p_ctrl	in	Control for the CAC device context.
p_status	in	Pointer to variable in which to store the current CASTR register contents.
p_counter	in	Pointer to variable in which to store the current CACNTBR register contents.

Parameter p_ctrl

Definition: cac_ctrl_t*const p_ctrl

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

Parameter p_status

uint8_t

Parameter p_counter

`uint16_t`

9.3.8.8 close

```
ssp_err_t(* cac_api_t::close) (cac_ctrl_t *const p_ctrl)
```

Detailed description

Close function for CAC device.

Table 775:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAC device control.

Parameter p_ctrl

Definition: `cac_ctrl_t*const p_ctrl`

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

9.3.8.9 stopMeasurement

```
ssp_err_t(* cac_api_t::stopMeasurement) (cac_ctrl_t *const p_ctrl)
```

Detailed description

End a measurement for the CAC peripheral.

Table 776:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAC device control.

Parameter p_ctrl

Definition: `cac_ctrl_t*const p_ctrl`

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

9.3.8.10 startMeasurement

```
ssp_err_t(* cac_api_t::startMeasurement) (cac_ctrl_t *const p_ctrl)
```

Detailed description

Begin a measurement for the CAC peripheral.

Table 777:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAC device control.

Parameter p_ctrl

Definition: `cac_ctrl_t*const p_ctrl`

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

9.3.8.11 reset

```
ssp_err_t(* cac_api_t::reset) (cac_ctrl_t *const p_ctrl)
```

Detailed description

Reset function for CAC device.

Table 778:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAC device control.

Parameter p_ctrl

Definition: `cac_ctrl_t*const p_ctrl`

CAC control block. Allocate an instance specific control block to pass into the CAC API calls. Implemented ascac_instance_ctrl_t

9.3.8.12 versionGet

```
ssp_err_t(* cac_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get the CAC API and code version information.

Table 779:Parameters

Name	Direction	Description
p_version	out	is value returned.

Parameter p_version

9.3.8.13 cac_instance_t

cac_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- cac_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- cac_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- cac_api_t const * p_api

  Pointer to the API structure for this instance.

9.4 CAN Interface

Interface for CAN peripheral.

9.4.1 Summary

The CAN interface provides common APIs for CAN HAL drivers. CAN interface supports following features.

- Full-duplex CAN communication
- Generic CAN parameter setting
- Interrupt driven transmit/receive processing
- Callback function support with returning event code
- Hardware resource locking during a transaction

9.4.2 Interface API

can_api_t

Function name	Description
.open	Open function for CAN device
.read	Read function for CAN device, non-Blocking.
.write	Write function for CAN device

Function name	Description
.close	Close function for CAN device
.control	Control function for CAN device
.infoGet	Get CAN channel info.
.versionGet	Version get function for CAN device

9.4.3 Data structures

- can_status_t
- can_error_t
- can_info_t
- can_callback_args_t
- can_bit_timing_cfg_t
- can_frame_t
- can_mailbox_t
- can_cfg_t
- can_instance_t

9.4.4 Enumerations

- can_event_t
- can_mode_t
- can_id_mode_t
- can_frame_type_t
- can_message_mode_t
- can_clock_source_t
- can_time_segment1_t
- can_time_segment2_t
- can_sync_jump_width_t
- can_mailbox_send_receive_t
- can_command_t

9.4.5 Typedefs

- can_id_t
- can_ctrl_t

9.4.6 Defines

- #define CAN_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CAN_API_VERSION_MINOR

  Initial value:`(2U)`

9.4.7 API Data

9.4.7.1 can_event_t

`can_event_t`

Detailed description

CAN event codes

Enumerated values

Name	Description
CAN_EVENT_RX_COMPLETE	Receive complete event.
CAN_EVENT_TX_COMPLETE	Transmit complete event.
CAN_EVENT_ERR_BUS_OFF	Bus Off event.
CAN_EVENT_BUS_RECOVERY	Bus Off Recovery event.
CAN_EVENT_ERR_PASSIVE	Error Passive event.
CAN_EVENT_ERR_WARNING	Error Warning event.
CAN_EVENT_MAILBOX_OVERWRITE_OVERRUN	DEPRECATED, Mailbox has been overrun. This event is not used when the mailbox is overwritten.
CAN_EVENT_MAILBOX_OVERRUN	Mailbox has been overrun.

9.4.7.2 can_mode_t

`can_mode_t`

Detailed description

CAN Operation modes

Enumerated values

Name	Description
CAN_MODE_NORMAL	CAN Normal Mode.
CAN_MODE_HALT	CAN Halt Mode.
CAN_MODE_SLEEP	CAN SLEEP Mode.
CAN_MODE_EXIT_SLEEP	CAN Exit SLEEP Mode.
CAN_MODE_RESET	CAN Reset Mode.
CAN_MODE_LISTEN	CAN Listen Mode.
CAN_MODE_LOOPBACK_INTERNAL	CAN Internal Loopback Mode.
CAN_MODE_LOOPBACK_EXTERNAL	CAN External Loopback Mode.

9.4.7.3 can_id_mode_t

`can_id_mode_t`

Detailed description

CAN ID modes

Enumerated values

Name	Description
CAN_ID_MODE_STANDARD	Standard IDs of 11 bits used.
CAN_ID_MODE_EXTENDED	Extended IDs of 29 bits used.

9.4.7.4 can_frame_type_t

`can_frame_type_t`

Detailed description

CAN frame types

Enumerated values

Name	Description
CAN_FRAME_TYPE_DATA	Data frame type.
CAN_FRAME_TYPE_REMOTE	Remote frame type.

9.4.7.5 can_message_mode_t

`can_message_mode_t`

Detailed description

CAN Message Modes

Enumerated values

Name	Description
CAN_MESSAGE_MODE_OVERWRITE	Receive data will be overwritten if not read before the next frame.
CAN_MESSAGE_MODE_OVERRUN	Receive data will be retained until it is read.

9.4.7.6 can_clock_source_t

`can_clock_source_t`

Detailed description

CAN Source Clock

Enumerated values

Name	Description
CAN_CLOCK_SOURCE_PCLKB	PCLB is the source of the CAN Clock.
CAN_CLOCK_SOURCE_CANMCLK	CANMCLK is the source of the CAN Clock.

9.4.7.7 can_time_segment1_t

`can_time_segment1_t`

Detailed description

CAN Time Segment 1 Time Quanta

Enumerated values

Name	Description
CAN_TIME_SEGMENT1_TQ4	Time Segment 1 setting for 4 Time Quanta.
CAN_TIME_SEGMENT1_TQ5	Time Segment 1 setting for 5 Time Quanta.
CAN_TIME_SEGMENT1_TQ6	Time Segment 1 setting for 6 Time Quanta.
CAN_TIME_SEGMENT1_TQ7	Time Segment 1 setting for 7 Time Quanta.
CAN_TIME_SEGMENT1_TQ8	Time Segment 1 setting for 8 Time Quanta.
CAN_TIME_SEGMENT1_TQ9	Time Segment 1 setting for 9 Time Quanta.
CAN_TIME_SEGMENT1_TQ10	Time Segment 1 setting for 10 Time Quanta.
CAN_TIME_SEGMENT1_TQ11	Time Segment 1 setting for 11 Time Quanta.
CAN_TIME_SEGMENT1_TQ12	Time Segment 1 setting for 12 Time Quanta.
CAN_TIME_SEGMENT1_TQ13	Time Segment 1 setting for 13 Time Quanta.
CAN_TIME_SEGMENT1_TQ14	Time Segment 1 setting for 14 Time Quanta.
CAN_TIME_SEGMENT1_TQ15	Time Segment 1 setting for 15 Time Quanta.
CAN_TIME_SEGMENT1_TQ16	Time Segment 1 setting for 16 Time Quanta.

9.4.7.8 can_time_segment2_t

`can_time_segment2_t`

Detailed description

CAN Time Segment 2 Time Quanta

Enumerated values

Name	Description
CAN_TIME_SEGMENT2_TQ2	Time Segment 2 setting for 2 Time Quanta.
CAN_TIME_SEGMENT2_TQ3	Time Segment 2 setting for 3 Time Quanta.
CAN_TIME_SEGMENT2_TQ4	Time Segment 2 setting for 4 Time Quanta.

Name	Description
CAN_TIME_SEGMENT2_TQ5	Time Segment 2 setting for 5 Time Quanta.
CAN_TIME_SEGMENT2_TQ6	Time Segment 2 setting for 6 Time Quanta.
CAN_TIME_SEGMENT2_TQ7	Time Segment 2 setting for 7 Time Quanta.
CAN_TIME_SEGMENT2_TQ8	Time Segment 2 setting for 8 Time Quanta.

9.4.7.9 can_sync_jump_width_t

`can_sync_jump_width_t`

Detailed description

CAN Synchronization Jump Width Time Quanta

Enumerated values

Name	Description
CAN_SYNC_JUMP_WIDTH_TQ1	Synchronization Jump Width setting for 1 Time Quanta.
CAN_SYNC_JUMP_WIDTH_TQ2	Synchronization Jump Width setting for 2 Time Quanta.
CAN_SYNC_JUMP_WIDTH_TQ3	Synchronization Jump Width setting for 3 Time Quanta.
CAN_SYNC_JUMP_WIDTH_TQ4	Synchronization Jump Width setting for 4 Time Quanta.

9.4.7.10 can_mailbox_send_receive_t

`can_mailbox_send_receive_t`

Detailed description

CAN Mailbox type

Enumerated values

Name	Description
CAN_MAILBOX_RECEIVE	Mailbox is for receiving.
CAN_MAILBOX_TRANSMIT	Mailbox is for sending.

9.4.7.11 can_command_t

`can_command_t`

Detailed description

CAN control commands.

Enumerated values

Name	Description
CAN_COMMAND_MODE_SWITCH	Switch CAN operating mode..

9.4.7.12 can_id_t

```
typedef uint32_t can_id_t
```

Detailed description

CAN Id

9.4.7.13 can_ctrl_t

```
typedef void can_ctrl_t
```

Detailed description

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented as

- can_instance_ctrl_t

9.4.8 API Structures

9.4.8.1 can_status_t

Detailed description

CAN Status

Variables

status

status_b

9.4.8.2 status

```
uint32_t can_status_t::status
```

9.4.8.3 status_b

`can_status_t::st_status_b::status_b`

9.4.8.4 can_error_t

Detailed description

CAN Error Code

Variables

error

error_b

9.4.8.5 error

`uint32_t can_error_t::error`

9.4.8.6 error_b

`can_error_t::st_error_b::error_b`

9.4.8.7 can_info_t

can_info_t

Detailed description

Variables

- can_mode_t operation_mode

  Can operation mode.
- can_status_t status

  CAN status.
- uint32_t bit_rate

  CAN bit rate.
- uint8_t error_count_transmit

  Transmit error count.
- uint8_t error_count_receive

  Receive error count.
- can_error_t error_code

  Error code, cleared after reading.

9.4.8.8 can_callback_args_t

can_callback_args_t

Detailed description

CAN callback parameter definition

Variables

- uint32_t channel

  Device channel number.

- can_event_t event

  Event code.

- uint32_t mailbox

  Mailbox number of interrupt source.

- void const * p_context

  Context provided to user during callback.

9.4.8.9 can_bit_timing_cfg_t

can_bit_timing_cfg_t

Detailed description

CAN bit rate configuration.

Variables

- uint32_t baud_rate_prescaler

  Baud rate prescaler. Valid values: 1 - 1024.

- can_time_segment1_t time_segment_1

  Time segment 1 control.

- can_time_segment2_t time_segment_2

  Time segment 2 control.

- can_sync_jump_width_t synchronization_jump_width

  Synchronization jump width.

9.4.8.10 can_frame_t

can_frame_t

Detailed description

CAN data Frame

Variables

- can_id_t id

  CAN id.
- uint8_t data_length_code

  CAN Data Length code, number of bytes in the message.
- uint8_t data[8]

  CAN data, up to 8 bytes.
- can_frame_type_t type

  Frame type, data or remote frame.

9.4.8.11 can_mailbox_t

can_mailbox_t

Detailed description

CAN Mailbox

Variables

- can_id_t mailbox_id

  Mailbox ID.
- can_mailbox_send_receive_t mailbox_type

  Receive or Transmit mailbox type.
- can_frame_type_t frame_type

  Frame type for receive mailbox.

9.4.8.12 can_cfg_t

can_cfg_t

Detailed description

CAN Configuration

Variables

- uint32_t channel

  CAN channel.
- can_bit_timing_cfg_t * p_bit_timing

  CAN bit timing.
- can_id_mode_t id_mode

  Standard or Extended ID mode.
- uint32_t mailbox_count

  Number of mailboxes.

- can_mailbox_t * p_mailbox

  Pointer to mailboxes.
- can_message_mode_t message_mode

  Overwrite message or overrun.
- void(* p_callback)(can_callback_args_t *p_args)

  Pointer to callback function.
- void const * p_context

  User defined callback context.
- void const * p_extend

  CAN hardware dependent configuration.
- uint8_t error_ipl

  Error interrupt priority.
- uint8_t mailbox_rx_ipl

  Receive interrupt priority.
- uint8_t mailbox_tx_ipl

  Transmit interrupt priority.

9.4.8.13 can_api_t

can_api_t

Detailed description

Shared Interface definition for CAN

9.4.8.14 open

```
ssp_err_t(* can_api_t::open) (can_ctrl_t *const p_ctrl, can_cfg_t const *const p_cfg)
```

Detailed description

Open function for CAN device Implemented as

- R_CAN_Open

Table 780:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the CAN control block Must be declared by user. Value set here.

Table 780:Parameters (Continued)

Name	Direction	Description
can_cfg_t	in	Pointer to CAN configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

Parameter can_cfg_t

Definition: `can_cfg_t const *const p_cfg`

CAN Configuration

- `can_cfg_t`::`channel`

  CAN channel.

- `can_cfg_t`::`can_bit_timing_cfg_t`

  CAN bit timing.

- `can_cfg_t`::`can_id_mode_t`

  Standard or Extended ID mode.

  Enumerated as:

  – CAN_ID_MODE_STANDARD

  – CAN_ID_MODE_EXTENDED

- `can_cfg_t`::`mailbox_count`

  Number of mailboxes.

- `can_cfg_t`::`can_mailbox_t`

  Pointer to mailboxes.

- `can_cfg_t`::`can_message_mode_t`

  Overwrite message or overrun.

  Enumerated as:

  – CAN_MESSAGE_MODE_OVERWRITE

  – CAN_MESSAGE_MODE_OVERRUN

- `can_cfg_t`::`p_callback`

  Pointer to callback function.

- can_cfg_t::p_context

  User defined callback context.

- can_cfg_t::p_extend

  CAN hardware dependent configuration.

- can_cfg_t::error_ipl

  Error interrupt priority.

- can_cfg_t::mailbox_rx_ipl

  Receive interrupt priority.

- can_cfg_t::mailbox_tx_ipl

  Transmit interrupt priority.

9.4.8.15 read

```
ssp_err_t(* can_api_t::read) (can_ctrl_t *const p_ctrl, uint32_t mailbox,
can_frame_t *const p_frame)
```

Detailed description

Read function for CAN device, non-Blocking. Implemented as

- R_CAN_Read

Table 781:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the CAN control block for the channel.
mailbox	in	Mailbox to read from.
p_frame	out	Pointer for frame of CAN ID, DLC, data and frame type.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

Parameter mailbox

`uint32_t`

Parameter p_frame

Definition: `can_frame_t*const p_frame`

CAN data Frame

- `can_frame_t::id`

  CAN id.
- `can_frame_t::data_length_code`

  CAN Data Length code, number of bytes in the message.
- `can_frame_t::data`

  CAN data, up to 8 bytes.
- `can_frame_t::type`

  Frame type, data or remote frame.

9.4.8.16 write

```
ssp_err_t(* can_api_t::write) (can_ctrl_t *const p_ctrl, uint32_t mailbox,
can_frame_t *const p_frame)
```

Detailed description

Write function for CAN device Implemented as

- R_CAN_Write

Table 782:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the CAN control block.
mailbox	in	Mailbox to write to.
p_frame	in	Pointer for frame of CAN ID, DLC, data and frame type to write.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

Parameter mailbox

`uint32_t`

Parameter p_frame

Definition: `can_frame_t*const p_frame`

CAN data Frame

- `can_frame_t::id`

  CAN id.

- `can_frame_t::data_length_code`

  CAN Data Length code, number of bytes in the message.
- `can_frame_t::data`

  CAN data, up to 8 bytes.
- `can_frame_t::type`

  Frame type, data or remote frame.

9.4.8.17 close

```
ssp_err_t(* can_api_t::close) (can_ctrl_t *const p_ctrl)
```

Detailed description

Close function for CAN device Implemented as

- R_CAN_Close

Table 783:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the CAN control block.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

9.4.8.18 control

```
ssp_err_t(* can_api_t::control) (can_ctrl_t *const p_ctrl, can_command_t const
command, void *p_data)
```

Detailed description

Control function for CAN device Implemented as

- R_CAN_Control

Table 784:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the CAN control block.
command	in	Command type.

Table 784:Parameters (Continued)

Name	Direction	Description
p_data	in	Command data.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

Parameter command

Parameter p_data

`const`

9.4.8.19 infoGet

```
ssp_err_t(* can_api_t::infoGet) (can_ctrl_t *const p_ctrl, can_info_t *const p_info)
```

Detailed description

Get CAN channel info. Implemented as

- R_CAN_InfoGet

Table 785:Parameters

Name	Direction	Description
p_ctrl	in	Handle for channel (pointer to channel control block)
p_info	out	Memory address to return channel specific data to.

Parameter p_ctrl

Definition: `can_ctrl_t*const p_ctrl`

CAN control block. Allocate an instance specific control block to pass into the CAN API calls. Implemented ascan_instance_ctrl_t

Parameter p_info

Definition: `can_info_t*const p_info`

- `can_info_t::operation_mode`

  Can operation mode.

- `can_info_t::status`

  CAN status.

- can_info_t::bit_rate

  CAN bit rate.
- can_info_t::error_count_transmit

  Transmit error count.
- can_info_t::error_count_receive

  Receive error count.
- can_info_t::error_code

  Error code, cleared after reading.

9.4.8.20 versionGet

```
ssp_err_t(* can_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Version get function for CAN device Implemented as

- R_CAN_VersionGet

Table 786:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory to store the version information

Parameter p_version

9.4.8.21 can_instance_t

can_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- can_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- can_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- can_api_t const * p_api

  Pointer to the API structure for this instance.

9.5 CGC Interface

Interface for clock generation.

9.5.1 Summary

The CGC interface provides the ability to configure and use all of the CGC module's capabilities. Among the capabilities is the selection of several clock sources to use as the system clock source. Additionally, the system clocks can be divided down to provide a wide range of frequencies for various system and peripheral needs.

Clock stability can be checked and clocks may also be stopped to save power when not needed. The API has a function to return the frequency of the system and system peripheral clocks at run time. There is also a feature to detect when the main oscillator has stopped, with the option of calling a user provided callback function.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CGC Interface description: CGC Driver

9.5.2 Interface API

cgc_api_t

Function name	Description
.init	Initial configuration
.clocksCfg	Configure all system clocks.
.clockStart	Start a clock.
.clockStop	Stop a clock.
.systemClockSet	Set the system clock.
.systemClockGet	Get the system clock information.
.systemClockFreqGet	Return the frequency of the selected clock.
.clockCheck	Check the stability of the selected clock.
.oscStopDetect	Configure the Main Oscillator stop detection.
.oscStopStatusClear	Clear the oscillator stop detection flag.

Function name	Description
.busClockOutCfg	Configure the bus clock output secondary divider. The primary divider is set using the bsp clock configuration and the systemClockSet function (S7G2 and S3A7 only).
.busClockOutEnable	Enable the bus clock output (S7G2 and S3A7 only).
.busClockOutDisable	Disable the bus clock output (S7G2 and S3A7 only).
.clockOutCfg	Configure clockOut.
.clockOutEnable	Enable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
.clockOutDisable	Disable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
.lcdClockCfg	Configure the segment LCD Clock (S3A7 and S124 only).
.lcdClockEnable	Enable the LCD clock (S3A7 and S124 only).
.lcdClockDisable	Disables the LCD clock (S3A7 and S124 only).
.sdramClockOutEnable	Enables the SDRAM clock output (S7G2 only).
.sdramClockOutDisable	Disables the SDRAM clock (S7G2 only).
.usbClockCfg	Configures the USB clock (S7G2 only).
.systickUpdate	Update the Systick timer.
.versionGet	Gets the CGC driver version.

9.5.3 Data structures

- cgc_callback_args_t
- cgc_clock_cfg_t
- cgc_system_clock_cfg_t
- cgc_clocks_cfg_t
- cgc_instance_t

9.5.4 Enumerations

- cgc_event_t
- cgc_clock_t
- cgc_pll_div_t
- cgc_sys_clock_div_t
- cgc_system_clocks_t
- cgc_clockout_dividers_t
- cgc_bclockout_dividers_t
- cgc_usb_clock_div_t
- cgc_systick_period_units_t
- cgc_clock_change_t

9.5.5 Defines

- #define CGC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CGC_API_VERSION_MINOR

  Initial value:`(2U)`

9.5.6 API Data

9.5.6.1 cgc_event_t

`cgc_event_t`

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
CGC_EVENT_OSC_STOP_DETECT	Oscillator stop detection has caused the event.

9.5.6.2 cgc_clock_t

`cgc_clock_t`

Detailed description

System clock source identifiers - The source of ICLK, BCLK, FCLK, PCLKS A-D and UCLK prior to the system clock divider

Enumerated values

Name	Description
CGC_CLOCK_HOCO	The high speed on chip oscillator.
CGC_CLOCK_MOCO	The middle speed on chip oscillator.
CGC_CLOCK_LOCO	The low speed on chip oscillator.
CGC_CLOCK_MAIN_OSC	The main oscillator.
CGC_CLOCK_SUBCLOCK	The subclock oscillator.
CGC_CLOCK_PLL	The PLL oscillator.

9.5.6.3 cgc_pll_div_t

`cgc_pll_div_t`

Detailed description

PLL divider values

Enumerated values

Name	Description
CGC_PLL_DIV_1	PLL divider of 1.
CGC_PLL_DIV_2	PLL divider of 2.
CGC_PLL_DIV_3	PLL divider of 3 (S7G2 only).
CGC_PLL_DIV_4	PLL divider of 4 (S3A7 only).

9.5.6.4 cgc_sys_clock_div_t

`cgc_sys_clock_div_t`

Detailed description

System clock divider values - The individually selectable divider of each of the system clocks, ICLK, BCLK, FCLK, PCLKS A-D

Enumerated values

Name	Description
CGC_SYS_CLOCK_DIV_1	System clock divided by 1.
CGC_SYS_CLOCK_DIV_2	System clock divided by 2.
CGC_SYS_CLOCK_DIV_4	System clock divided by 4.
CGC_SYS_CLOCK_DIV_8	System clock divided by 8.
CGC_SYS_CLOCK_DIV_16	System clock divided by 16.
CGC_SYS_CLOCK_DIV_32	System clock divided by 32.
CGC_SYS_CLOCK_DIV_64	System clock divided by 64.

9.5.6.5 cgc_system_clocks_t

`cgc_system_clocks_t`

Detailed description

System clock identifiers - Used as an input parameter to the systemClockFreqGet function.

Enumerated values

Name	Description
CGC_SYSTEM_CLOCKS_PCLKA	PCLKA - Peripheral module clock A.
CGC_SYSTEM_CLOCKS_PCLKB	PCLKB - Peripheral module clock B.
CGC_SYSTEM_CLOCKS_PCLKC	PCLKC - Peripheral module clock C.
CGC_SYSTEM_CLOCKS_PCLKD	PCLKD - Peripheral module clock D.
CGC_SYSTEM_CLOCKS_BCLK	BCLK - External bus Clock.
CGC_SYSTEM_CLOCKS_FCLK	FCLK - FlashIF clock.
CGC_SYSTEM_CLOCKS_ICLK	ICLK - System clock.

9.5.6.6 cgc_clockout_dividers_t

`cgc_clockout_dividers_t`

Detailed description

Divider values for the CLKOUT output.

Enumerated values

Name	Description
CGC_CLOCKOUT_DIV_1	Clockout source is divided by 1.
CGC_CLOCKOUT_DIV_2	Clockout source is divided by 2.
CGC_CLOCKOUT_DIV_4	Clockout source is divided by 4.
CGC_CLOCKOUT_DIV_8	Clockout source is divided by 8.
CGC_CLOCKOUT_DIV_16	Clockout source is divided by 16.
CGC_CLOCKOUT_DIV_32	Clockout source is divided by 32.
CGC_CLOCKOUT_DIV_64	Clockout source is divided by 64.
CGC_CLOCKOUT_DIV_128	Clockout source is divided by 128.

9.5.6.7 cgc_bclockout_dividers_t

`cgc_bclockout_dividers_t`

Detailed description

Divider values for the external bus clock output.

Enumerated values

Name	Description
CGC_BCLOCKOUT_DIV_1	External bus clock source is divided by 1.
CGC_BCLOCKOUT_DIV_2	External bus clock source is divided by 2.

9.5.6.8 cgc_usb_clock_div_t

`cgc_usb_clock_div_t`

Detailed description

USB clock divider values

Enumerated values

Name	Description
CGC_USB_CLOCK_DIV_3	Divide USB source clock by 3.
CGC_USB_CLOCK_DIV_4	Divide USB source clock by 4.
CGC_USB_CLOCK_DIV_5	Divide USB source clock by 5.

9.5.6.9 cgc_systick_period_units_t

`cgc_systick_period_units_t`

Detailed description

Available period units for R_CGC_SystickUpdate

Enumerated values

Name	Description
CGC_SYSTICK_PERIOD_UNITS_MILLISECONDS	Requested period in milliseconds.
CGC_SYSTICK_PERIOD_UNITS_MICROSECONDS	Requested period in microseconds.

9.5.6.10 cgc_clock_change_t

`cgc_clock_change_t`

Detailed description

Clock options

Enumerated values

Name	Description
CGC_CLOCK_CHANGE_NONE	No change to the clock.
CGC_CLOCK_CHANGE_STOP	Stop the clock.
CGC_CLOCK_CHANGE_START	Start the clock.

9.5.7 API Structures

9.5.7.1 cgc_callback_args_t

cgc_callback_args_t

Detailed description

Callback function parameter data

Variables

- cgc_event_t event

  The event can be used to identify what caused the callback.

- void const * p_context

  Placeholder for user data.

9.5.7.2 cgc_clock_cfg_t

cgc_clock_cfg_t

Detailed description

Clock configuration structure - Used as an input parameter to the clockStart function for the PLL clock.

Variables

- cgc_clock_t source_clock

  PLL source clock (S7G2 only).

- cgc_pll_div_t divider

  PLL divider.

- float multiplier

  PLL multiplier.

9.5.7.3 cgc_system_clock_cfg_t

cgc_system_clock_cfg_t

Detailed description

Clock configuration structure - Used as an input parameter to the systemClockSet and systemClockGet functions.

Variables

- cgc_sys_clock_div_t pclka_div

  Divider value for PCLKA.

- cgc_sys_clock_div_t pclkb_div

  Divider value for PCLKB.

- cgc_sys_clock_div_t pclkc_div
  Divider value for PCLKC.
- cgc_sys_clock_div_t pclkd_div
  Divider value for PCLKD.
- cgc_sys_clock_div_t bclk_div
  Divider value for BCLK.
- cgc_sys_clock_div_t fclk_div
  Divider value for FCLK.
- cgc_sys_clock_div_t iclk_div
  Divider value for ICLK.

9.5.7.4 cgc_clocks_cfg_t

cgc_clocks_cfg_t

Detailed description

Clock configuration

Variables

- cgc_clock_t system_clock
  System clock source enumeration.
- cgc_clock_cfg_t pll_cfg
  PLL configuration structure.
- cgc_system_clock_cfg_t sys_cfg
  Clock dividers structure.
- cgc_clock_change_t loco_state
  State of LOCO.
- cgc_clock_change_t moco_state
  State of MOCO.
- cgc_clock_change_t hoco_state
  State of HOCO.
- cgc_clock_change_t subosc_state
  State of Sub-oscillator.
- cgc_clock_change_t mainosc_state
  State of Main oscillator.
- cgc_clock_change_t pll_state
  State of PLL.

9.5.7.5 cgc_api_t

cgc_api_t

Detailed description

CGC functions implemented at the HAL layer follow this API.

9.5.7.6 init

```
ssp_err_t(* cgc_api_t::init) (void)
```

Detailed description

Initial configuration Implemented as

- R_CGC_Init

NOTE: *The BSP module calls this function at startup. No further initialization is necessary.*

9.5.7.7 clocksCfg

```
ssp_err_t(* cgc_api_t::clocksCfg) (cgc_clocks_cfg_t const *const p_clock_cfg)
```

Detailed description

Configure all system clocks. Implemented as

- R_CGC_ClocksCfg

NOTE: *The BSP module calls this function at startup, but it can also be called from the application to change clocks at runtime.*

9.5.7.8 clockStart

```
ssp_err_t(* cgc_api_t::clockStart) (cgc_clock_t clock_source, cgc_clock_cfg_t
*p_clock_cfg)
```

Detailed description

Start a clock. Implemented as

- R_CGC_ClockStart

NOTE: Clock to be started must not be running prior to calling this function or an error will be returned.

Table 787:Parameters

Name	Direction	Description
clock_source	in	Clock source to initialize.
p_clock_cfg	in	Pointer to a structure that contains the dividers or multipliers to be used when configuring the PLL.

Parameter clock_source

Definition: `cgc_clock_tclock_source`

System clock source identifiers - The source of ICLK, BCLK, FCLK, PCLKS A-D and UCLK prior to the system clock divider

Parameter p_clock_cfg

Definition: `cgc_clock_cfg_t *p_clock_cfg`

Clock configuration structure - Used as an input parameter to the clockStart function for the PLL clock.

- `cgc_clock_cfg_t`::`cgc_clock_t`

  PLL source clock (S7G2 only).

  Enumerated as:

  – CGC_CLOCK_HOCO

  – CGC_CLOCK_MOCO

  – CGC_CLOCK_LOCO

  – CGC_CLOCK_MAIN_OSC

  – CGC_CLOCK_SUBCLOCK

  – CGC_CLOCK_PLL

- `cgc_clock_cfg_t`::`cgc_pll_div_t`

  PLL divider.

  Enumerated as:

  – CGC_PLL_DIV_1

  – CGC_PLL_DIV_2

  – CGC_PLL_DIV_3

  – CGC_PLL_DIV_4

- `cgc_clock_cfg_t`::`multiplier`

  PLL multiplier.

9.5.7.9 clockStop

```
ssp_err_t(* cgc_api_t::clockStop) (cgc_clock_t clock_source)
```

Detailed description

Stop a clock. Implemented as

- R_CGC_ClockStop

NOTE: Clock to be stopped must not be stopped prior to calling this function or an error will be returned.

Table 788:Parameters

Name	Direction	Description
clock_source	in	The clock source to stop.

Parameter clock_source

Definition: `cgc_clock_tclock_source`

System clock source identifiers - The source of ICLK, BCLK, FCLK, PCLKS A-D and UCLK prior to the system clock divider

9.5.7.10 systemClockSet

```
ssp_err_t(* cgc_api_t::systemClockSet) (cgc_clock_t clock_source,
cgc_system_clock_cfg_t const *const p_clock_cfg)
```

Detailed description

Set the system clock. Implemented as

- R_CGC_SystemClockSet

NOTE: The clock to be set as the system clock must be running prior to calling this function.

Table 789:Parameters

Name	Direction	Description
clock_source	in	Clock source to set as system clock
p_clock_cfg	in	Pointer to the clock dividers configuration passed by the caller.

Parameter clock_source

Definition: `cgc_clock_tclock_source`

System clock source identifiers - The source of ICLK, BCLK, FCLK, PCLKS A-D and UCLK prior to the system clock divider

Parameter p_clock_cfg

Definition: `cgc_system_clock_cfg_t const *const p_clock_cfg`

Clock configuration structure - Used as an input parameter to the systemClockSet and systemClockGet functions.

- `cgc_system_clock_cfg_t`::`cgc_sys_clock_div_t`

  Divider value for PCLKA.

  Enumerated as:

  – CGC_SYS_CLOCK_DIV_1

  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- `cgc_system_clock_cfg_t`::`cgc_sys_clock_div_t`

  Divider value for PCLKB.

  Enumerated as:

  - CGC_SYS_CLOCK_DIV_1
  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- `cgc_system_clock_cfg_t`::`cgc_sys_clock_div_t`

  Divider value for PCLKC.

  Enumerated as:

  - CGC_SYS_CLOCK_DIV_1
  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- `cgc_system_clock_cfg_t`::`cgc_sys_clock_div_t`

  Divider value for PCLKD.

  Enumerated as:

  - CGC_SYS_CLOCK_DIV_1

  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- cgc_system_clock_cfg_t::cgc_sys_clock_div_t

  Divider value for BCLK.

  Enumerated as:
  - CGC_SYS_CLOCK_DIV_1
  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- cgc_system_clock_cfg_t::cgc_sys_clock_div_t

  Divider value for FCLK.

  Enumerated as:
  - CGC_SYS_CLOCK_DIV_1
  - CGC_SYS_CLOCK_DIV_2
  - CGC_SYS_CLOCK_DIV_4
  - CGC_SYS_CLOCK_DIV_8
  - CGC_SYS_CLOCK_DIV_16
  - CGC_SYS_CLOCK_DIV_32
  - CGC_SYS_CLOCK_DIV_64
- cgc_system_clock_cfg_t::cgc_sys_clock_div_t

  Divider value for ICLK.

  Enumerated as:
  - CGC_SYS_CLOCK_DIV_1

- CGC_SYS_CLOCK_DIV_2
- CGC_SYS_CLOCK_DIV_4
- CGC_SYS_CLOCK_DIV_8
- CGC_SYS_CLOCK_DIV_16
- CGC_SYS_CLOCK_DIV_32
- CGC_SYS_CLOCK_DIV_64

9.5.7.11 systemClockGet

```
ssp_err_t(* cgc_api_t::systemClockGet) (cgc_clock_t *p_clock_source,
cgc_system_clock_cfg_t *p_set_clock_cfg)
```

Detailed description

Get the system clock information. Implemented as

- R_CGC_SystemClockGet

Table 790:Parameters

Name	Direction	Description
clock_source	out	Returns the current system clock.
p_clock_cfg	out	Returns the current system clock dividers.

Parameter clock_source

Parameter p_clock_cfg

9.5.7.12 systemClockFreqGet

```
ssp_err_t(* cgc_api_t::systemClockFreqGet) (cgc_system_clocks_t clock, uint32_t
*p_freq_hz)
```

Detailed description

Return the frequency of the selected clock. Implemented as

- R_CGC_SystemClockFreqGet

Table 791:Parameters

Name	Direction	Description
clock	in	Specifies the internal clock whose frequency is returned.
p_freq_hz	out	Returns the frequency in Hz referenced by this pointer.

Parameter clock

Definition: `cgc_system_clocks_t clock`

System clock identifiers - Used as an input parameter to the cgc_api_t::systemClockFreqGet function.

Parameter p_freq_hz

`uint32_t`

9.5.7.13 clockCheck

```
ssp_err_t(* cgc_api_t::clockCheck) (cgc_clock_t clock_source)
```

Detailed description

Check the stability of the selected clock. Implemented as

- R_CGC_ClockCheck

Table 792:Parameters

Name	Direction	Description
clock_source	in	Which clock source to check for stability.

Parameter clock_source

Definition: `cgc_clock_t clock_source`

System clock source identifiers - The source of ICLK, BCLK, FCLK, PCLKS A-D and UCLK prior to the system clock divider

9.5.7.14 oscStopDetect

```
ssp_err_t(* cgc_api_t::oscStopDetect) (void(*p_callback)(cgc_callback_args_t
*p_args), bool enable)
```

Detailed description

Configure the Main Oscillator stop detection. Implemented as

- R_CGC_OscStopDetect

Table 793:Parameters

Name	Direction	Description
p_callback	in	Callback function that will be called by the NMI interrupt when an oscillation stop is detected. If the second argument is "false", then this argument can be NULL.
enable	in	Enable/disable Oscillation Stop Detection.

Parameter p_callback

Parameter enable

`const`

9.5.7.15 oscStopStatusClear

```
ssp_err_t(* cgc_api_t::oscStopStatusClear) (void)
```

Detailed description

Clear the oscillator stop detection flag. Implemented as

- R_CGC_OscStopStatusClear

9.5.7.16 busClockOutCfg

```
ssp_err_t(* cgc_api_t::busClockOutCfg) (cgc_bclockout_dividers_t divider)
```

Detailed description

Configure the bus clock output secondary divider. The primary divider is set using the bsp clock configuration and the systemClockSet function (S7G2 and S3A7 only).

Implemented as

- R_CGC_BusClockOutCfg

Table 794:Parameters

Name	Direction	Description
divider	in	The divider of 1 or 2 of the clock source.

Parameter divider

Definition: `cgc_bclockout_dividers_t`divider

Divider values for the external bus clock output.

9.5.7.17 busClockOutEnable

```
ssp_err_t(* cgc_api_t::busClockOutEnable) (void)
```

Detailed description

Enable the bus clock output (S7G2 and S3A7 only). Implemented as

- R_CGC_BusClockOutEnable

9.5.7.18 busClockOutDisable

```
ssp_err_t(* cgc_api_t::busClockOutDisable) (void)
```

Detailed description

Disable the bus clock output (S7G2 and S3A7 only). Implemented as

- R_CGC_BusClockOutDisable

9.5.7.19 clockOutCfg

```
ssp_err_t(* cgc_api_t::clockOutCfg) (cgc_clock_t clock, cgc_clockout_dividers_t
divider)
```

Detailed description

Configure clockOut. Implemented as

- R_CGC_ClockOutCfg

Table 795:Parameters

Name	Direction	Description
clock	in	Clock source.
divider	in	Divider of between 1 and 128 of the clock source.

Parameter clock

Parameter divider

Definition: `cgc_clockout_dividers_tdivider`

Divider values for the CLKOUT output.

9.5.7.20 clockOutEnable

```
ssp_err_t(* cgc_api_t::clockOutEnable) (void)
```

Detailed description

Enable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg. Implemented as

- R_CGC_ClockOutEnable

9.5.7.21 clockOutDisable

```
ssp_err_t(* cgc_api_t::clockOutDisable) (void)
```

Detailed description

Disable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg. Implemented as

- R_CGC_ClockOutDisable

9.5.7.22 lcdClockCfg

```
ssp_err_t(* cgc_api_t::lcdClockCfg) (cgc_clock_t clock)
```

Detailed description

Configure the segment LCD Clock (S3A7 and S124 only). Implemented as

- R_CGC_LCDClockCfg

Table 796:Parameters

Name	Direction	Description
clock	in	Segment LCD clock source.

Parameter clock

9.5.7.23 lcdClockEnable

```
ssp_err_t(* cgc_api_t::lcdClockEnable) (void)
```

Detailed description

Enable the LCD clock (S3A7 and S124 only). Implemented as

- R_CGC_LCDClockEnable

9.5.7.24 lcdClockDisable

```
ssp_err_t(* cgc_api_t::lcdClockDisable) (void)
```

Detailed description

Disables the LCD clock (S3A7 and S124 only). Implemented as

- R_CGC_LCDClockDisable

9.5.7.25 sdramClockOutEnable

```
ssp_err_t(* cgc_api_t::sdramClockOutEnable) (void)
```

Detailed description

Enables the SDRAM clock output (S7G2 only). Implemented as

- R_CGC_SDRAMClockOutEnable

9.5.7.26 sdramClockOutDisable

```
ssp_err_t(* cgc_api_t::sdramClockOutDisable) (void)
```

Detailed description

Disables the SDRAM clock (S7G2 only). Implemented as

- R_CGC_SDRAMClockOutDisable

9.5.7.27 usbClockCfg

```
ssp_err_t(* cgc_api_t::usbClockCfg) (cgc_usb_clock_div_t divider)
```

Detailed description

Configures the USB clock (S7G2 only). Implemented as

- R_CGC_USBClockCfg

Table 797:Parameters

Name	Direction	Description
divider	in	The divider of 3, 4 or 5, of the clock source.

Parameter divider

Definition: `cgc_usb_clock_div_tdivider`

USB clock divider values

9.5.7.28 systickUpdate

```
ssp_err_t(* cgc_api_t::systickUpdate) (uint32_t period_count,
cgc_systick_period_units_t units)
```

Detailed description

Update the Systick timer. Implemented as

- R_CGC_SystickUpdate

Table 798:Parameters

Name	Direction	Description
period_count	in	The duration for the systick period.
units	in	The units for the provided period.

Parameter period_count

`uint32_t`

Parameter units

9.5.7.29 versionGet

```
ssp_err_t(* cgc_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Gets the CGC driver version. Implemented as

- R_CGC_VersionGet

Table 799:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.5.7.30 cgc_instance_t

cgc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- cgc_clock_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- cgc_api_t const * p_api

  Pointer to the API structure for this instance.

9.6 CRC Interface

Interface for cyclic redundancy checking.

9.6.1 Summary

The CRC (Cyclic Redundancy Check) calculator generates CRC codes using five different polynomials including 8 bit, 16 bit, and 32 bit variations. Calculation can be performed by sending data to the block using the CPU or by snooping on read or write activity on one of 10 SCI channels.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CRC Interface description: CRC Driver

9.6.2 Interface API

crc_api_t

Function name	Description
.open	Open the CRC driver module.
.close	Close the CRC module driver
.crcResultGet	Return the current calculated value.
.snoopEnable	Enable snooping.
.snoopDisable	Disable snooping.
.snoopCfg	Configure the snoop channel and direction.
.calculate	Perform a CRC calculation on a block of data.
.versionGet	Get the driver version based on compile time macros.

9.6.3 Data structures

- crc_cfg_t
- crc_snoop_cfg_t
- crc_instance_t

9.6.4 Enumerations

- crc_polynomial_t
- crc_bit_order_t
- crc_snoop_direction_t

9.6.5 Typedefs

- crc_ctrl_t

9.6.6 Defines

- #define CRC_API_VERSION_MAJOR
  Initial value:`(1U)`
  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CRC_API_VERSION_MINOR
  Initial value:`(2U)`

9.6.7 API Data

9.6.7.1 crc_polynomial_t

`crc_polynomial_t`

Detailed description

CRC Generating Polynomial Switching (GPS).

Enumerated values

Name	Description
CRC_POLYNOMIAL_CRC_8	8-bit CRC-8 (X8 + X2 + X + 1)
CRC_POLYNOMIAL_CRC_16	16-bit CRC-16 (X16 + X15 + X2 + 1)
CRC_POLYNOMIAL_CRC_CCITT	16-bit CRC-CCITT (X16 + X12 + X5 + 1)
CRC_POLYNOMIAL_CRC_32	32-bit CRC-32 (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X + 1)
CRC_POLYNOMIAL_CRC_32C	32-bit CRC-32C (X32 + X28 + X27 + X26 + X25 + X23 + X22 + X20 + X19 + X18 + X14 + X13 + X11 + X10 + X9 + X8 + X6 + 1)

9.6.7.2 crc_bit_order_t

`crc_bit_order_t`

Detailed description

CRC Calculation Switching (LMS)

Enumerated values

Name	Description
CRC_BIT_ORDER_LMS_LSB	Generates CRC for LSB first communication.
CRC_BIT_ORDER_LMS_MSB	Generates CRC for MSB first communication.

9.6.7.3 crc_snoop_direction_t

`crc_snoop_direction_t`

Detailed description

Snoop-On-Write/Read Switch (CRCSWR)

Enumerated values

Name	Description
CRC_SNOOP_DIRECTION_RECEIVE	Snoop-on-read.
CRC_SNOOP_DIRECTION_TRANSMIT	Snoop-on-write.

9.6.7.4 crc_ctrl_t

`typedef void crc_ctrl_t`

Detailed description

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented as

- crc_instance_ctrl_t

9.6.8 API Structures

9.6.8.1 crc_cfg_t

crc_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- crc_polynomial_t polynomial

  CRC Generating Polynomial Switching. (GPS)

- crc_bit_order_t bit_order

  CRC Calculation Switching (LMS)

- void const * p_extend

  CRC Hardware Dependent Configuration.

9.6.8.2 crc_snoop_cfg_t

crc_snoop_cfg_t

Detailed description

Snoop configuration

Variables

- uint32_t snoop_channel

  Register Snoop Address (CRCSA)

- crc_snoop_direction_t snoop_direction

  Snoop-On-Write/Read Switch (CRCSWR)

9.6.8.3 crc_api_t

crc_api_t

Detailed description

CRC driver structure. General CRC functions implemented at the HAL layer will follow this API.

9.6.8.4 open

```
ssp_err_t(* crc_api_t::open) (crc_ctrl_t *const p_ctrl, crc_cfg_t const *const
p_cfg)
```

Detailed description

Open the CRC driver module. Implemented as

- R_CRC_Open

Table 800:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CRC device handle.

Table 800:Parameters (Continued)

Name	Direction	Description
p_cfg	in	Pointer to a configuration structure.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter p_cfg

Definition: `crc_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `crc_cfg_t`::`crc_polynomial_t`

  CRC Generating Polynomial Switching. (GPS)

  Enumerated as:

  – CRC_POLYNOMIAL_CRC_8

  – CRC_POLYNOMIAL_CRC_16

  – CRC_POLYNOMIAL_CRC_CCITT

  – CRC_POLYNOMIAL_CRC_32

  – CRC_POLYNOMIAL_CRC_32C

- `crc_cfg_t`::`crc_bit_order_t`

  CRC Calculation Switching (LMS)

  Enumerated as:

  – CRC_BIT_ORDER_LMS_LSB

  – CRC_BIT_ORDER_LMS_MSB

- `crc_cfg_t`::`p_extend`

  CRC Hardware Dependent Configuration.

9.6.8.5 close

```
ssp_err_t(* crc_api_t::close) (crc_ctrl_t *const p_ctrl)
```

Detailed description

Close the CRC module driver Implemented as

- R_CRC_Close

Table 801:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to crc device handle

Table 802:Return values

Name	Description
SSP_SUCCESS	Configuration was successful.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter SSP_SUCCESS

9.6.8.6 crcResultGet

```
ssp_err_t(* crc_api_t::crcResultGet) (crc_ctrl_t *const p_ctrl, uint32_t
*crc_result)
```

Detailed description

Return the current calculated value. Implemented as

- R_CRC_CalculatedValueGet

Table 803:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CRC device handle.
crc_result	out	The calculated value from the last CRC calculation.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter crc_result

`uint32_t`

9.6.8.7 snoopEnable

```
ssp_err_t(* crc_api_t::snoopEnable) (crc_ctrl_t *const p_ctrl, uint32_t crc_seed)
```

Detailed description

Enable snooping. Implemented as

- R_CRC_SnoopEnable

Table 804:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CRC device handle.
crc_seed	in	CRC seed.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter crc_seed

`uint32_t`

9.6.8.8 snoopDisable

```
ssp_err_t(* crc_api_t::snoopDisable) (crc_ctrl_t *const p_ctrl)
```

Detailed description

Disable snooping. Implemented as

- R_CRC_SnoopDisable

Table 805:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to crc device handle.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

9.6.8.9 snoopCfg

```
ssp_err_t(* crc_api_t::snoopCfg) (crc_ctrl_t *const p_ctrl, crc_snoop_cfg_t
*const p_snoop_cfg)
```

Detailed description

Configure the snoop channel and direction. Implemented as

- R_CRC_SnoopCfg

Table 806:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to crc device handle.
p_snoopCfg	in	Snoop configuration.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter p_snoopCfg

9.6.8.10 calculate

```
ssp_err_t(* crc_api_t::calculate) (crc_ctrl_t *const p_ctrl, void
*p_input_buffer, uint32_t num_bytes, uint32_t crc_seed, uint32_t *p_crc_result)
```

Detailed description

Perform a CRC calculation on a block of data. Implemented as

- R_CRC_Calculate

Table 807:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to crc device handle.
input_buffer	in	A pointer to an array of data values.
num_bytes	in	The number of bytes (not elements) in the array.
crc_seed	in	The seeded value for crc calculations.

Table 807:Parameters (Continued)

Name	Direction	Description
crc_result	out	The calculated value of the CRC calculation.

Parameter p_ctrl

Definition: `crc_ctrl_t*const p_ctrl`

CRC control block. Allocate an instance specific control block to pass into the CRC API calls. Implemented ascrc_instance_ctrl_t

Parameter input_buffer

Parameter num_bytes

`uint32_t`

Parameter crc_seed

`uint32_t`

Parameter crc_result

9.6.8.11 versionGet

```
ssp_err_t(* crc_api_t::versionGet) (ssp_version_t *version)
```

Detailed description

Get the driver version based on compile time macros. Implemented as

- R_CRC_VersionGet

9.6.8.12 crc_instance_t

crc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- crc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- crc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- crc_api_t const * p_api

  Pointer to the API structure for this instance.

9.7 Crypto Interface

Cryptographic algorithm APIs for encryption/decryption, signing/verification, and hashing.

9.7.1 Interface API

crypto_api_t

Function name	Description
.open	Open crypto module using the given configuration
.close	Close the crypto interface module for the given control structure p_ctrl
.interfaceGet	Get API interface structure object based on the interface_info provided.
.statusGet	Get status of SCE initialization
.versionGet	Gets version and stores it in provided pointer p_version.

9.7.2 Data structures

- crypto_ctrl_t
- crypto_cfg_t
- crypto_interface_get_param_t
- crypto_instance_t

9.7.3 Enumerations

- crypto_word_endian_t
- crypto_algorithm_type_t
- crypto_hash_type_t
- crypto_key_type_t
- crypto_key_size_t
- crypto_chaining_mode_t

9.7.4 Modules

- AES Interface
- DSA Interface
- HASH Algorithm Interface
- RSA Interface
- TDES Interface
- Random number generation

9.7.5 Defines

- #define CRYPTO_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CRYPTO_API_VERSION_MINOR

  Initial value:`(00)`

9.7.6 API Data

9.7.6.1 crypto_word_endian_t

`crypto_word_endian_t`

Detailed description

Enumerator for Crypto API uint32_t[] array word endian selection

Enumerated values

Name	Description
CRYPTO_WORD_ENDIAN_BIG	
CRYPTO_WORD_ENDIAN_LITTLE	

9.7.6.2 crypto_algorithm_type_t

`crypto_algorithm_type_t`

Detailed description

MIN and MAX values under enums are for internal use only Enumerated Crypto Algorithm Type

Enumerated values

Name	Description
CRYPTO_ALGORITHM_TYPE_MIN	
CRYPTO_ALGORITHM_TYPE_RSA	
CRYPTO_ALGORITHM_TYPE_HASH	
CRYPTO_ALGORITHM_TYPE_AES	
CRYPTO_ALGORITHM_TYPE_TRNG	
CRYPTO_ALGORITHM_TYPE_MAX	

9.7.6.3 crypto_hash_type_t

`crypto_hash_type_t`

Detailed description

Enumerated HASH Types

Enumerated values

Name	Description
CRYPTO_TYPE_HASH_MIN	
CRYPTO_TYPE_HASH_1	
CRYPTO_TYPE_HASH_224	
CRYPTO_TYPE_HASH_256	
CRYPTO_TYPE_HASH_MAX	

9.7.6.4 crypto_key_type_t

`crypto_key_type_t`

Detailed description

Enumerated Key Types

Enumerated values

Name	Description
CRYPTO_KEY_TYPE_MIN	
CRYPTO_KEY_TYPE_AES_PLAIN_TEXT	
CRYPTO_KEY_TYPE_AES_WRAPPED	
CRYPTO_KEY_TYPE_RSA_PLAIN_TEXT	
CRYPTO_KEY_TYPE_RSA_CRT_PLAIN_TEXT	
CRYPTO_KEY_TYPE_RSA_WRAPPED	
CRYPTO_KEY_TYPE_MAX	

9.7.6.5 crypto_key_size_t

`crypto_key_size_t`

Detailed description

Enumerated Key Sizes

Enumerated values

Name	Description
CRYPTO_KEY_SIZE_MIN	
CRYPTO_KEY_SIZE_AES_128	
CRYPTO_KEY_SIZE_AES_256	
CRYPTO_KEY_SIZE_AES_192	
CRYPTO_KEY_SIZE_RSA_1024	
CRYPTO_KEY_SIZE_RSA_2048	
CRYPTO_KEY_SIZE_MAX	

9.7.6.6 crypto_chaining_mode_t

`crypto_chaining_mode_t`

Detailed description

Enumerated chaining modes

Enumerated values

Name	Description
CRYPTO_CHAINING_MODE_MIN	
CRYPTO_ECB_MODE	
CRYPTO_CBC_MODE	
CRYPTO_CTR_MODE	
CRYPTO_GCM_MODE	
CRYPTO_XTS_MODE	
CRYPTO_CHAINING_MODE_MAX	

9.7.7 API Structures

9.7.7.1 crypto_ctrl_t

crypto_ctrl_t

Detailed description

Crypto_Interface Add API definitions required by user here.

Variables

- uint32_t state

  indicates state of the SCE/SCE-Lite driver e.g whether it is initialized
- uint32_t cb_data
- void(* p_sce_long_plg_start_callback)(void)
- void(* p_sce_long_plg_end_callback)(void)

9.7.7.2 crypto_cfg_t

crypto_cfg_t

Detailed description

Crypto engine configuration parameters

Variables

- void(* p_sce_long_plg_start_callback)(void)

- void(* p_sce_long_plg_end_callback)(void)

  Callback provided when a ISR occurs. Set to NULL for no CPU interrupt.
- crypto_word_endian_t endian_flag

  Endian flag, indicates word endianness for the uint32_t[] array inputs used in Crypto APIs

9.7.7.3 crypto_interface_get_param_t

crypto_interface_get_param_t

Detailed description

Parameters for requesting HAL API interface object

Variables

- crypto_algorithm_type_t algorithm_type
- crypto_key_type_t key_type
- crypto_key_size_t key_size
- crypto_chaining_mode_t chaining_mode
- crypto_hash_type_t hash_type

9.7.7.4 crypto_api_t

crypto_api_t

Detailed description

Crypto_Interface SCE functions implemented at the HAL layer will follow this API.

9.7.7.5 open

```
uint32_t(* crypto_api_t::open) (crypto_ctrl_t *const p_ctrl, crypto_cfg_t const
*const p_cfg)
```

Detailed description

Open crypto module using the given configuration

Table 808:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure. Must be declared by user. Elements set here.
p_cfg	in	pointer to configuration structure. All elements of this structure must be set by user

Parameter p_ctrl

Definition: `crypto_ctrl_t`

Crypto_Interface Add API definitions required by user here.

Parameter p_cfg

Definition: `crypto_cfg_t const *const p_cfg`

Crypto engine configuration parameters

- `crypto_cfg_t`::`p_sce_long_plg_start_callback`
- `crypto_cfg_t`::`p_sce_long_plg_end_callback`

  Callback provided when a ISR occurs. Set to NULL for no CPU interrupt.
- `crypto_cfg_t`::`crypto_word_endian_t`

  Endian flag, indicates word endianness for the uint32_t[] array inputs used in Crypto APIs

  Enumerated as:

  – CRYPTO_WORD_ENDIAN_BIG

  – CRYPTO_WORD_ENDIAN_LITTLE

9.7.7.6 close

```
uint32_t(* crypto_api_t::close) (crypto_ctrl_t *const p_ctrl)
```

Detailed description

Close the crypto interface module for the given control structure p_ctrl

Table 809:Parameters

Name	Direction	Description
p_ctrl	in	pointer to control structure

Parameter p_ctrl

Definition: `crypto_ctrl_t`

Crypto_Interface Add API definitions required by user here.

9.7.7.7 interfaceGet

```
uint32_t(* crypto_api_t::interfaceGet) (crypto_interface_get_param_t *const
interface_info, void *const p_interface)
```

Detailed description

Get API interface structure object based on the interface_info provided.

Table 810:Parameters

Name	Direction	Description
interface_info	in	pointer to structure containing requested interface information.
p_interface	out	pointer which points to interface structure object.

NOTE: Value of p_interface is allowed to be passed as NULL at the time of API call.

Parameter interface_info

Definition: `crypto_interface_get_param_t*const interface_info`

Parameters for requesting HAL API interface object

- `crypto_interface_get_param_t::algorithm_type`
- `crypto_interface_get_param_t::key_type`
- `crypto_interface_get_param_t::key_size`
- `crypto_interface_get_param_t::chaining_mode`
- `crypto_interface_get_param_t::hash_type`

Parameter p_interface

`const`

9.7.7.8 statusGet

```
uint32_t(* crypto_api_t::statusGet) (uint32_t *p_status)
```

Detailed description

Get status of SCE initialization

Table 811:Parameters

Name	Direction	Description
p_status	out	initialization status of SCE module will be written to the memory pointed to by p_status

Parameter p_status

`uint32_t`

9.7.7.9 versionGet

```
uint32_t(* crypto_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 812:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.7.7.10 crypto_instance_t

crypto_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- crypto_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- crypto_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- crypto_api_t const * p_api

  Pointer to the API structure for this instance.

9.7.8 Modules

- AES Interface
- DSA Interface
- HASH Algorithm Interface
- RSA Interface
- TDES Interface
- Random number generation

9.7.8.1 AES Interface

AES encryption and decryption APIs.

Interface API

aes_api_t

Function name	Description
.open	AES module open function. Must be called before performing any encrypt/decrypt operations.
.createKey	Generate an AES key for encrypt / decrypt operations.
.encrypt	AES encryption. Encrypt input data with ECB mode using a 128-bit AES key
.addAdditionalAuthenticationData	Add additional authentication data (called before starting an encryption or decryption operation)
.encryptFinal	AES final encryption using the chaining mode and padding mode specified in the aes.open() function call.
.decrypt	AES Decryption. Decrypt input data with ECB mode using a 128-bit AES key
.setGcmTag	set parameter specific to the mode
.getGcmTag	Get authentication tag data.
.close	Close the AES module.
.zeroPaddingEncrypt	AES zero padding encryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage -.

Function name	Description
.zeroPaddingDecrypt	AES zero padding decryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage -.
.versionGet	Gets version and stores it in provided pointer p_version.

Data structures

- aes_ctrl_t
- aes_cfg_t
- aes_instance_t

Variables

- g_sce_crypto_api
- g_aes128ecb_on_sce
- g_aes128cbc_on_sce
- g_aes128ctr_on_sce
- g_aes256ecb_on_sce
- g_aes256cbc_on_sce
- g_aes256ctr_on_sce
- g_aes128gcm_on_sce
- g_aes128xts_on_sce
- g_aes256gcm_on_sce
- g_aes256xts_on_sce
- g_aes128gcm_on_sceHrk
- g_aes256gcm_on_sceHrk
- g_aes192ecb_on_sce
- g_aes192cbc_on_sce
- g_aes192ctr_on_sce
- g_aes192gcm_on_sce
- g_aes128ecb_on_sceHrk
- g_aes128cbc_on_sceHrk
- g_aes128ctr_on_sceHrk
- g_aes128xts_on_sceHrk
- g_aes256ecb_on_sceHrk
- g_aes256cbc_on_sceHrk

- g_aes256ctr_on_sceHrk
- g_aes256xts_on_sceHrk
- g_aes192ecb_on_sceHrk
- g_aes192cbc_on_sceHrk
- g_aes192ctr_on_sceHrk
- g_aes192gcm_on_sceHrk

Defines

- #define AES_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define AES_API_VERSION_MINOR

  Initial value:`(00)`
- #define DRV_AES_CONTEXT_BUFFER_SIZE

  Initial value:`(64)`
- #define AES_XTS_128_WRAPPPED_SECRET_KEY_SIZE_BYTES

  Initial value:`(52)`

  Return Wrapped AES-XTS secret key size in bytes for a 128-bit AES XTS Mode Key
- #define AES_XTS_256_WRAPPPED_SECRET_KEY_SIZE_BYTES

  Initial value:`(84)`

  Return AES-XTS secret key size in bytes for a 256-bit AES XTS Mode Key
- #define AES128_WRAPPPED_SECRET_KEY_SIZE_BYTES

  Initial value:`(36)`

  Return Wrapped AES secret key size in bytes for a 128-bit AES Key
- #define AES192_WRAPPPED_SECRET_KEY_SIZE_BYTES

  Initial value:`(52)`

  Return Wrapped AES secret key size in bytes for a 192-bit AES Key
- #define AES256_WRAPPPED_SECRET_KEY_SIZE_BYTES

  Initial value:`(52)`

  Return Wrapped AES secret key size in bytes for a 256-bit AES Key
- #define AES128_SECRET_KEY_SIZE_BYTES

  Initial value:`(16)`

  Return AES secret key size in bytes for a 128-bit AES Key

- #define AES192_SECRET_KEY_SIZE_BYTES

  Initial value:`(24)`

  Return AES secret key size in bytes for a 192-bit AES Key
- #define AES256_SECRET_KEY_SIZE_BYTES

  Initial value:`(32)`

  Return AES secret key size in bytes for a 256-bit AES Key

aes_ctrl_t

aes_ctrl_t

Detailed description

AES Interface control structure

Variables

- crypto_ctrl_t * p_crypto_ctrl

  pointer to crypto engine control structure
- crypto_api_t const * p_crypto_api

  pointer to crypto engine API
- uint32_t work_buffer[DRV_AES_CONTEXT_BUFFER_SIZE]

  Examples: AES-GCM mode uses this for storing authentication tag etc.

  used for storing context/state of the Cipher

aes_cfg_t

aes_cfg_t

Detailed description

AES Interface configuration structure. User must fill in these values before invoking the open() function

Variables

- crypto_api_t const * p_crypto_api

  pointer to crypto engine api

aes_api_t

aes_api_t

Detailed description

AES_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* aes_api_t::open) (aes_ctrl_t *const p_ctrl, aes_cfg_t const *const
p_cfg)
```

Detailed description

AES module open function. Must be called before performing any encrypt/decrypt operations.

Table 813:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure for the AES interface. Must be declared by user. Elements are set here.
p_cfg	in	pointer to control structure for the AES configuration. All elemenets of this structure must be set by user.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter p_cfg

Definition: `aes_cfg_t const *const p_cfg`

AES Interface configuration structure. User must fill in these values before invoking the open() function

- `aes_cfg_t`::`crypto_api_t`

  pointer to crypto engine api

createKey

```
uint32_t(* aes_api_t::createKey) (aes_ctrl_t *const p_ctrl, uint32_t num_words,
uint32_t *p_key)
```

Brief description

Generate an AES key for encrypt / decrypt operations.

Detailed description

Table 814:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure for the AES interface.
num_words	in	number of words in buffer p_key
p_key	out	pointer to key buffer. Generated key will be stored at this location.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter num_words

`uint32_t`

Parameter p_key

`uint32_t`

encrypt

```
uint32_t(* aes_api_t::encrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
```

Brief description

AES encryption.

Detailed description

Encrypt input data with ECB mode using a 128-bit AES key

Table 815:Parameters

Name	Direction	Description
*p_key	in	pointer to the AES plain-text key
*p_iv	in	is a pointer to initialization vector. For ECB mode this parameter is unused. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_source	in	input data buffer
*p_dest	out	output data buffer

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

addAdditionalAuthenticationData

```
uint32_t(* aes_api_t::addAdditionalAuthenticationData) (aes_ctrl_t *const p_ctrl,
const uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source)
```

Brief description

Add additional authentication data (called before starting an encryption or decryption operation)

Detailed description

Table 816:Parameters

Name	Direction	Description
*p_key	in	pointer to the AES plain-text key
*p_iv	in	unused for ECB mode
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_source	in	input data buffer

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

encryptFinal

```
uint32_t(* aes_api_t::encryptFinal) (aes_ctrl_t *const p_ctrl, const uint32_t
*p_key, uint32_t *p_iv, uint32_t input_num_words, uint32_t *p_source, uint32_t
output_num_words, uint32_t *p_dest)
```

Brief description

AES final encryption using the chaining mode and padding mode specified in the aes.open() function call.

decrypt

```
uint32_t(* aes_api_t::decrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t imaxcnt, uint32_t *p_source, uint32_t *p_dest)
```

Brief description

AES Decryption.

Detailed description

Decrypt input data with ECB mode using a 128-bit AES key

Table 817:Parameters

Name	Direction	Description
*p_key	in	128-bit plain key
*p_iv	in	is a pointer to initialization vector. For ECB mode this parameter is unused. NULL value is acceptable.
num_words	in	Size in words of p_source and p_dest data buffers. Each word is 4-bytes. Must be multiples of 4 words.
*p_source	in	input data buffer
*p_dest	out	output data buffer

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_words

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

setGcmTag

```
uint32_t(* aes_api_t::setGcmTag) (aes_ctrl_t *const p_ctrl, uint32_t num_words,
uint32_t *p_source)
```

Brief description

set parameter specific to the mode

Detailed description

Table 818:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure
num_words	in	number of words in p_source buffer. This must be atleast 4 words

Table 818:Parameters (Continued)

Name	Direction	Description
p_source	in	pointer to authentication tag data buffer, must be of size 4 words.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter num_words

`uint32_t`

Parameter p_source

`uint32_t`

getGcmTag

```
uint32_t(* aes_api_t::getGcmTag) (aes_ctrl_t *const p_ctrl, uint32_t num_words,
uint32_t *p_dest)
```

Brief description

Get authentication tag data.

Detailed description

Table 819:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure
num_words	in	number of words in p_dest buffer. This must be atleast 4 words
p_dest	in	pointer to data buffer, must be of size 4 words.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter num_words

`uint32_t`

Parameter p_dest

`uint32_t`

close

```
uint32_t(* aes_api_t::close) (aes_ctrl_t *const p_ctrl)
```

Detailed description

Close the AES module.

Table 820:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

zeroPaddingEncrypt

```
uint32_t(* aes_api_t::zeroPaddingEncrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t *p_dest)
```

Brief description

AES zero padding encryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage -.

Detailed description

1) Provide any Add Authentication Data (AAD): set p_dest = NULL
2) Encryption: set p_source to input data and p_dest will return encrypted data
3) Get/Compute Tag: set p_source = NULL

Table 821:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure
*p_key	in	pointer to the AES plain-text key, the buffer size should be equal to the keylength
*p_iv	in	the buffer size must be 16 bytes
num_bytes	in	data buffer size in bytes.
*p_source	in	input data buffer, computes tag when set to NULL.
*p_dest	out	output data buffer, adds authentication data when set to NULL.

NOTE: this function is not thread safe.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_bytes

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

zeroPaddingDecrypt

```
uint32_t(* aes_api_t::zeroPaddingDecrypt) (aes_ctrl_t *const p_ctrl, const
uint32_t *p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t
*p_dest)
```

Brief description

AES zero padding decryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage -.

Detailed description

1) Set expected tag value using the setGcmTag function
2) Provide any Add Authentication Data (AAD), invoke this API using p_dest = NULL
1) Decryption: set p_source to input encrypted data, decrypted data will be returned in p_dest
2) Verify the tag, invoke this API using p_source = NULL and p_dest = NULL, the return value indicates authentication tag verification status.

Table 822:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure
*p_key	in	pointer to the AES plain-text key, the buffer size should be equal to the keylength
*p_iv	in	the buffer size must be 16 bytes

Table 822:Parameters (Continued)

Name	Direction	Description
num_bytes	in	data buffer size in bytes.
*p_source	in	input data buffer, computes tag when set to NULL.
*p_dest	out	output data buffer, adds authentication data when set to NULL.

NOTE: this function is not thread safe.

Parameter p_ctrl

Definition: `aes_ctrl_t`

AES Interface control structure

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_bytes

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

versionGet

```
uint32_t(* aes_api_t::versionGet) ( *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 823:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

aes_instance_t

aes_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- aes_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- aes_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- aes_api_t const * p_api

  Pointer to the API structure for this instance.

g_sce_crypto_api

`crypto_api_t::g_sce_crypto_api`

g_aes128ecb_on_sce

`aes_api_t::g_aes128ecb_on_sce`

Detailed description

AES interface available boards - S7G2, S5D9, S5D5, S3A7, S3A3 and S3A6 - Chaining modes CBC, GCM, CTR, ECB, XTS for 128 & 256-bit S7G2, S5D9, and S5D5 - Chaining modes CBC, GCM, CTR, ECB for 192-bit

AES 128-bit ECB mode implementation

g_aes128cbc_on_sce

`aes_api_t::g_aes128cbc_on_sce`

Detailed description

AES 128-bit CBC mode implementation

g_aes128ctr_on_sce

`aes_api_t::g_aes128ctr_on_sce`

Detailed description

AES 128-bit CTR mode implementation

g_aes256ecb_on_sce

`aes_api_t::g_aes256ecb_on_sce`

Detailed description

AES 256-bit ECB mode implementation

g_aes256cbc_on_sce

`aes_api_t::g_aes256cbc_on_sce`

Detailed description

AES 256-bit CBC mode implementation

g_aes256ctr_on_sce

`aes_api_t::g_aes256ctr_on_sce`

Detailed description

AES 256-bit CTR mode implementation

g_aes128gcm_on_sce

aes_api_t::g_aes128gcm_on_sce

Detailed description

AES 128-bit GCM mode implementation

g_aes128xts_on_sce

aes_api_t::g_aes128xts_on_sce

Detailed description

AES 128-bit CCM mode implementation

g_aes256gcm_on_sce

aes_api_t::g_aes256gcm_on_sce

Detailed description

AES 256-bit GCM mode implementation

g_aes256xts_on_sce

aes_api_t::g_aes256xts_on_sce

Detailed description

AES 256-bit XTS mode implementation

g_aes128gcm_on_sceHrk

aes_api_t::g_aes128gcm_on_sceHrk

g_aes256gcm_on_sceHrk

aes_api_t::g_aes256gcm_on_sceHrk

g_aes192ecb_on_sce

aes_api_t::g_aes192ecb_on_sce

Detailed description

AES 192-bit ECB mode implementation

g_aes192cbc_on_sce

aes_api_t::g_aes192cbc_on_sce

Detailed description

AES 192-bit CBC mode implementation

g_aes192ctr_on_sce

aes_api_t::g_aes192ctr_on_sce

Detailed description

AES 192-bit CTR mode implementation

g_aes192gcm_on_sce

aes_api_t::g_aes192gcm_on_sce

Detailed description

AES 192-bit GCM mode implementation

g_aes128ecb_on_sceHrk

aes_api_t::g_aes128ecb_on_sceHrk

Detailed description

HRK Supported global structure definitions

g_aes128cbc_on_sceHrk

aes_api_t::g_aes128cbc_on_sceHrk

g_aes128ctr_on_sceHrk

aes_api_t::g_aes128ctr_on_sceHrk

g_aes128xts_on_sceHrk

aes_api_t::g_aes128xts_on_sceHrk

g_aes256ecb_on_sceHrk

aes_api_t::g_aes256ecb_on_sceHrk

g_aes256cbc_on_sceHrk

aes_api_t::g_aes256cbc_on_sceHrk

g_aes256ctr_on_sceHrk

aes_api_t::g_aes256ctr_on_sceHrk

g_aes256xts_on_sceHrk

aes_api_t::g_aes256xts_on_sceHrk

g_aes192ecb_on_sceHrk

aes_api_t::g_aes192ecb_on_sceHrk

g_aes192cbc_on_sceHrk

aes_api_t::g_aes192cbc_on_sceHrk

g_aes192ctr_on_sceHrk

aes_api_t::g_aes192ctr_on_sceHrk

g_aes192gcm_on_sceHrk

aes_api_t::g_aes192gcm_on_sceHrk

9.7.8.2 DSA Interface

DSA (Digital Signature Algorithm) signature generation and verification APIs.

Interface API

dsa_api_t

Function name	Description
.open	DSA module open function. Must be called before performing any encrypt/decrypt operations.

Function name	Description
.verify	DSA signature verification using given DSA public key. Verify DSA signature from buffer using the given DSA public key with domain parameters from for the input message hash
.hashVerify	DSA signature verification using given DSA public key. Verify DSA signature from buffer using the given DSA public key with domain parameters from for the input message hash
.sign	DSA Signature generation using DSA private key. Generate signature for the buffer with the given DSA private key for the domain parameters . The result will be written to the buffer
.hashSign	DSA Signature generation using DSA private key. Generate signature for the buffer with the given DSA private key for the domain parameters . The result will be written to the buffer
.close	Close the DSA module.
.versionGet	Gets version and stores it in provided pointer p_version.

Data structures

- dsa_ctrl_t
- dsa_cfg_t
- dsa_instance_t

Variables

- g_sce_crypto_api
- g_dsa1024_160_on_sce
- g_dsa2048_224_on_sce
- g_dsa2048_256_on_sce

Defines

- #define DSA_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define DSA_API_VERSION_MINOR

  Initial value:`(00)`

dsa_ctrl_t

dsa_ctrl_t

Detailed description

DSA Interface control structure

Variables

- crypto_ctrl_t * p_crypto_ctrl

  pointer to crypto engine control structure

- crypto_api_t const * p_crypto_api

  pointer to crypto engine API

dsa_cfg_t

dsa_cfg_t

Detailed description

DSA Interface configuration structure. User must fill in these values before invoking the open() function

Variables

- crypto_api_t const * p_crypto_api

  pointer to crypto engine api

dsa_api_t

dsa_api_t

Detailed description

DSA_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* dsa_api_t::open) (dsa_ctrl_t *const p_ctrl, dsa_cfg_t const *const
p_cfg)
```

Detailed description

DSA module open function. Must be called before performing any encrypt/decrypt operations.

Table 824:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure for the DSA interface. Must be declared by user. Elements are set here.
p_cfg	in	pointer to control structure for the DSA configuration. All elemenets of this structure must be set by user.

Parameter p_ctrl

Definition: dsa_ctrl_t

DSA Interface control structure

Parameter p_cfg

Definition: `dsa_cfg_t const *const p_cfg`

DSA Interface configuration structure. User must fill in these values before invoking the open() function

- `dsa_cfg_t::crypto_api_t`

  pointer to crypto engine api

verify

```
uint32_t(* dsa_api_t::verify) (const uint32_t *p_key, const uint32_t *p_domain,
uint32_t num_words, uint32_t *p_signature, uint32_t *p_paddedHash)
```

Brief description

DSA signature verification using given DSA public key.

Detailed description

Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

Table 825:Parameters

Name	Direction	Description
*p_key	in	pointer to the DSA plain-text key.
*p_domain	in	pointer to DSA domain parameters.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_signature	in	signature data buffer to be verified
*p_paddedHash	in	padded hash of the input message

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_signature

`uint32_t`

Parameter *p_paddedHash

`uint32_t`

hashVerify

```
uint32_t(* dsa_api_t::hashVerify) (dsa_ctrl_t *const p_ctrl, const uint32_t
*p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature,
uint32_t *p_paddedHash)
```

Brief description

DSA signature verification using given DSA public key.

Detailed description

Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

Table 826:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to DSA control structure
*p_key	in	pointer to the DSA plain-text key.
*p_domain	in	pointer to DSA domain parameters.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_signature	in	signature data buffer to be verified
*p_paddedHash	in	padded hash of the input message

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_signature

`uint32_t`

Parameter *p_paddedHash

`uint32_t`

sign

```
uint32_t(* dsa_api_t::sign) (const uint32_t *p_key, const uint32_t *p_domain,
uint32_t num_words, uint32_t *p_paddedHash, uint32_t *p_dest)
```

Brief description

DSA Signature generation using DSA private key.

Detailed description

Generate signature for the buffer p_paddedHash with the given DSA private key p_key for the domain parameters p_domain. The result will be written to the buffer p_dest

Table 827:Parameters

Name	Direction	Description
*p_key	in	DSA plain text private key
*p_domain	in	pointer to DSA domain parameters.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_paddedHash	in	input data buffer
*p_dest	out	output data buffer

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_paddedHash

`uint32_t`

Parameter *p_dest

`uint32_t`

hashSign

```
uint32_t(* dsa_api_t::hashSign) (dsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_paddedHash, uint32_t
*p_dest)
```

Brief description

DSA Signature generation using DSA private key.

Detailed description

Generate signature for the buffer p_paddedHash with the given DSA private key p_key for the domain parameters p_domain. The result will be written to the buffer p_dest

Table 828:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure
*p_key	in	DSA plain text private key

Table 828:Parameters (Continued)

Name	Direction	Description
*p_domain	in	pointer to DSA domain parameters.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_paddedHash	in	input data buffer
*p_dest	out	output data buffer

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_paddedHash

`uint32_t`

Parameter *p_dest

`uint32_t`

close

```
uint32_t(* dsa_api_t::close) (dsa_ctrl_t *const p_ctrl)
```

Detailed description

Close the DSA module.

Table 829:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure

Parameter p_ctrl

Definition: `dsa_ctrl_t`

DSA Interface control structure

versionGet

```
uint32_t(* dsa_api_t::versionGet) ( *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 830:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

dsa_instance_t

dsa_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- dsa_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- dsa_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- dsa_api_t const * p_api

  Pointer to the API structure for this instance.

g_sce_crypto_api

`crypto_api_t::g_sce_crypto_api`

g_dsa1024_160_on_sce

`dsa_api_t::g_dsa1024_160_on_sce`

Detailed description

DSA interface is only available on S7G2, S5D9 and S5D5.

SCE/DSA implementation of DSA API.

g_dsa2048_224_on_sce

`dsa_api_t::g_dsa2048_224_on_sce`

Detailed description

SCE/DSA implementation of DSA API.

g_dsa2048_256_on_sce

`dsa_api_t::g_dsa2048_256_on_sce`

Detailed description

SCE/DSA implementation of DSA API.

9.7.8.3 HASH Algorithm Interface

HASH_Interface Hash algorithm APIs.

Interface API

hash_api_t

Function name	Description
.open	HASH_Interface: Initial configuration
.updateHash	update hash for the words from source buffer .
.hashUpdate	update hash for the words from source buffer .
.close	
.versionGet	Gets version and stores it in provided pointer p_version.

Data structures

- hash_cfg_t
- hash_ctrl_t
- hash_instance_t

Variables

- g_sha1_hash_on_sce
- g_sha256_hash_on_sce

Defines

- #define HASH_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define HASH_API_VERSION_MINOR

  Initial value:`(00)`
- #define HASH_MESSAGE_BLOCK_SIZE_WORDS

  Initial value:`(16)`
- #define HASH_MESSAGE_BLOCK_SIZE_BYTES

  Initial value:`((HASH_MESSAGE_BLOCK_SIZE_WORDS) * 4)`
- #define HASH_MAX_DIGEST_SIZE_WORDS

  Initial value:`(8)`

- #define HASH_MAX_DIGEST_SIZE_BYTES

  Initial value:`((HASH_DIGEST_SIZE_WORDS) * 4)`

hash_cfg_t

hash_cfg_t

Detailed description

HASH_Interface configuration structure

Variables

- crypto_ctrl_t * p_crypto_ctrl

  pointer to crypto engine control structure

- crypto_api_t const * p_crypto_api

  pointer to crypto engine API structure

hash_ctrl_t

hash_ctrl_t

Detailed description

HASH_Interface control structure

Variables

- uint32_t msgbuf[HASH_MESSAGE_BLOCK_SIZE_WORDS]

  message buffer to be hashed

- uint32_t hash[HASH_MAX_DIGEST_SIZE_WORDS]

  current hash value

- uint64_t length

  64-bit message length (number of bits)

hash_api_t

hash_api_t

Detailed description

HASH_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* hash_api_t::open) (hash_ctrl_t *const p_ctrl, hash_cfg_t const *const p_cfg)
```

Detailed description

HASH_Interface: Initial configuration

NOTE: HASH_Interface: Peripheral clocks and any required output pins should be configured prior to calling this function.

Table 831:Parameters

Name	Direction	Description
HASH_Interface	inout	p_ctrl Pointer to control structure. Must be declared by user. Elements set here.
HASH_Interface	in	p_cfg Pointer to configuration structure. All elements of this structure must be set by user.

Parameter HASH_Interface

Parameter HASH_Interface

updateHash

```
uint32_t(* hash_api_t::updateHash) (const uint32_t *p_source, uint32_t num_words,
uint32_t *p_dest)
```

Detailed description

update hash for the num_words words from source buffer p_source.

Table 832:Parameters

Name	Direction	Description
p_source	in	pointer to input message buffer. size must be a multiple of 64-bytes
num_words	in	number of words to be hashed from the buffer p_source
p_dest	inout	pointer to the message digest. on input contains initialization value.

Parameter p_source

```
uint32_t
```

Parameter num_words

```
uint32_t
```

Parameter p_dest

```
uint32_t
```

hashUpdate

```
uint32_t(* hash_api_t::hashUpdate) (hash_ctrl_t *const p_ctrl, const uint32_t
*p_source, uint32_t num_words, uint32_t *p_dest)
```

Detailed description

update hash for the num_words words from source buffer p_source.

Table 833:Parameters

Name	Direction	Description
p_ctrl	in	pointer to hash_ctrl_t control structure.
p_source	in	pointer to input message buffer. size must be a multiple of 64-bytes
num_words	in	number of words to be hashed from the buffer p_source
p_dest	inout	pointer to the message digest. on input contains initialization value.

Parameter p_ctrl

Definition: `hash_ctrl_t`

HASH_Interface control structure

Parameter p_source

`uint32_t`

Parameter num_words

`uint32_t`

Parameter p_dest

`uint32_t`

close

```
uint32_t(* hash_api_t::close) (hash_ctrl_t *const p_ctrl)
```

versionGet

```
uint32_t(* hash_api_t::versionGet) ( *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 834:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

hash_instance_t

hash_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- hash_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- hash_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- hash_api_t const * p_api

  Pointer to the API structure for this instance.

g_sha1_hash_on_sce

`hash_api_t::g_sha1_hash_on_sce`

Detailed description

HASH interface is only available on S7G2, S5D9 and S5D5.

SHA1 implementation of HASH API.

g_sha256_hash_on_sce

`hash_api_t::g_sha256_hash_on_sce`

Detailed description

SHA256 implementation of HASH API.

9.7.8.4 RSA Interface

RSA cryptographic functions for signature generation, verification, encryption and decryption.

Interface API

rsa_api_t

Function name	Description
.open	RSA module open function. Must be called before performing any encrypt/decrypt or sign/verify operations.
.encrypt	Encrypt source data from p_source using an RSA public key from p_key and write the results to destination buffer p_dest.

Function name	Description
.decrypt	Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest. The RSA private key data p_key is specified in the standard format that consists of private exponent and the RSA modulus. The size of the private exponent and the RSA modulus is 1024-bits for the g_rsa1024_on_sce implementation and 2048-bits for the g_rsa2048_on_sce implementation.
.decryptCrt	Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest. RSA private key data is specified in CRT format. The RSA CRT key consists of the exponent2 || prime2 || exponent1 || prime1 || coefficient, starting with exponent2 at index 0. The size of each of these parameter is 512-bits for the g_rsa1024_on_sce implementation and 1024-bits for the g_rsa2048_on_sce implementation.
.verify	Verify signature given in buffer p_signature using the RSA public key p_key for the given padded message hash from buffer p_padded_hash.
.sign	Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. Write the results to the buffer p_dest.
.signCrt	Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. RSA private key p_key is assumed to be in CRT format. Write the results to the buffer p_dest. The RSA CRT key consists of the exponent2 || prime2 || exponent1 || prime1 || coefficient, starting with exponent2 at index 0. The size of each of these parameter is 512-bits for the g_rsa1024_on_sce implementation and 1024-bits for the g_rsa2048_on_sce implementation.
.close	Close the RSA module.
.versionGet	Gets version and stores it in provided pointer p_version.
.keyCreate	Generates an RSA key. This is a blocking call

Data structures

- rsa_key_t
- rsa_ctrl_t
- rsa_cfg_t
- rsa_instance_t

Enumerations

- rsa_key_format_t

Variables

- g_sce_crypto_api
- g_rsa1024_on_sce
- g_rsa2048_on_sce
- g_rsa1024_on_sce_hrk
- g_rsa2048_on_sce_hrk

Defines

- #define RSA_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define RSA_API_VERSION_MINOR

  Initial value:`(01)`
- #define RSA_MODULUS_SIZE_BYTES

  Initial value:`((RSA_SIZE_BITS)/8)`

  Return RSA modulus size in bytes from the specified RSA modulus size in bits
- #define RSA_PLAIN_TEXT_PUBLIC_KEY_SIZE_BYTES

  Initial value:`(((uint32_t)32+(uint32_t)RSA_SIZE_BITS)/8U)`

  Return RSA public key size in bytes from the specified RSA modulus size in bits
- #define RSA_PLAIN_TEXT_PRIVATE_KEY_SIZE_BYTES

  Initial value:`(((uint32_t)2*(uint32_t)RSA_SIZE_BITS)/8U)`

  Return RSA private key size in bytes from the specified RSA modulus size in bits
- #define RSA_PLAIN_TEXT_CRT_KEY_SIZE_BYTES

  Initial value:`(((uint32_t)5*((uint32_t)RSA_SIZE_BITS))/16U)`

  Return RSA CRT private key size in bytes from the specified RSA modulus size in bits
- #define RSA_WRAPPPED_PRIVATE_KEY_SIZE_BYTES

  Initial value:`((((uint32_t)2 * (uint32_t)RSA_SIZE_BITS)+(uint32_t)160)/8U)`

  Return RSA wrapped private key size in bytes from the specified RSA modulus size in bits

rsa_key_t

rsa_key_t

Detailed description

RSA key data structure

Variables

- rsa_key_format_t key_format

  Indicates if the key is in plain-text format or encrypted using device unique key.

- uint32_t length

  Length in bytes of the p_data buffer.

- uint8_t * p_data

  Buffer where the private key is stored.

rsa_ctrl_t

rsa_ctrl_t

Detailed description

RSA Interface control structure

Variables

- crypto_ctrl_t * p_crypto_ctrl

  pointer to crypto engine control structure

- crypto_api_t const * p_crypto_api

  pointer to crypto engine API

- uint32_t stage_num

  processing stage

rsa_cfg_t

rsa_cfg_t

Detailed description

RSA Interface configuration structure. User must fill in these values before invoking the open() function

Variables

- crypto_api_t const * p_crypto_api

  pointer to crypto engine api

rsa_api_t

rsa_api_t

Detailed description

RSA_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* rsa_api_t::open) (rsa_ctrl_t *const p_ctrl, rsa_cfg_t const *const
p_cfg)
```

Detailed description

RSA module open function. Must be called before performing any encrypt/decrypt or sign/verify operations.

Table 835:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure for the RSA interface. Must be declared by user. Elements are set here.
p_cfg	in	pointer to control structure for the RSA configuration. All elemenets of this structure must be set by user.

Parameter p_ctrl

Definition: `rsa_ctrl_t`

RSA Interface control structure

Parameter p_cfg

Definition: `rsa_cfg_t const *const p_cfg`

RSA Interface configuration structure. User must fill in these values before invoking the open() function

- `rsa_cfg_t`::`crypto_api_t`

  pointer to crypto engine api

encrypt

```
uint32_t(* rsa_api_t::encrypt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t
*p_dest)
```

Brief description

Encrypt source data from p_source using an RSA public key from p_key and write the results to destination buffer p_dest.

Detailed description

Table 836:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure for RSA interface
*p_key	in	pointer to the RSA plain-text public key consisting of 32-bit public exponent and RSA public modulus of size either 1024-bits or 2048-bits.
*p_domain	in	unused parameter for RSA encryption. NULL value is acceptable.

Table 836:Parameters (Continued)

Name	Direction	Description
num_words	in	data buffer size in words. Each word is 4-bytes.
*p_source	in	source data buffer to be encrypted.
*p_dest	out	destination data buffer, encryption result will be stored here.

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

decrypt

```
uint32_t(* rsa_api_t::decrypt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t
*p_dest)
```

Brief description

Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest. The RSA private key data p_key is specified in the standard format that consists of private exponent and the RSA modulus. The size of the private exponent and the RSA modulus is 1024-bits for the g_rsa1024_on_sce implementation and 2048-bits for the g_rsa2048_on_sce implementation.

Detailed description

Table 837:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure for RSA interface
*p_key	in	pointer to RSA plain-text private key consisting of private exponent and the RSA modulus.

Table 837:Parameters (Continued)

Name	Direction	Description
*p_domain	in	unused parameter for RSA decryption. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes.
*p_source	in	input data buffer to be decrypted.
*p_dest	out	output destination data buffer, decryption result will be stored here.

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

decryptCrt

```
uint32_t(* rsa_api_t::decryptCrt) (rsa_ctrl_t *const p_ctrl, const uint32_t
*p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t
*p_dest)
```

Brief description

Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest. RSA private key data is specified in CRT format. The RSA CRT key consists of the exponent2 || prime2 || exponent1 || prime1 || coefficient, starting with exponent2 at index 0. The size of each of these parameter is 512-bits for the g_rsa1024_on_sce implementation and 1024-bits for the g_rsa2048_on_sce implementation.

Detailed description

Table 838:Parameters

Name	Direction	Description
*p_key	in	pointer to RSA private key in CRT format.

Table 838:Parameters (Continued)

Name	Direction	Description
*p_domain	in	unused parameter for RSA decryption. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes.
*p_source	in	input data buffer to be decrypted.
*p_dest	out	output destination data buffer, decryption result will be stored here.

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

verify

```
uint32_t(* rsa_api_t::verify) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t
*p_padded_hash)
```

Brief description

Verify signature given in buffer p_signature using the RSA public key p_key for the given padded message hash from buffer p_padded_hash.

Detailed description

Table 839:Parameters

Name	Direction	Description
*p_key	in	pointer to the RSA plain-text public key consisting of 32-bit public exponent and RSA public modulus of size either 1024-bits or 2048-bits.

Table 839:Parameters (Continued)

Name	Direction	Description
*p_domain	in	unused parameter. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes.
*p_signature	in	signature data that needs to be verified
*p_paddedHash	in	padded hash value of the input message buffer

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_signature

`uint32_t`

Parameter *p_paddedHash

sign

```
uint32_t(* rsa_api_t::sign) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t
*p_dest)
```

Brief description

Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. Write the results to the buffer p_dest.

Detailed description

Table 840:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure for RSA interface
*p_key	in	pointer to RSA private key consisting of private exponent and the RSA modulus.

Table 840:Parameters (Continued)

Name	Direction	Description
*p_domain	in	unused parameter. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_padded_hash	in	padded hash for the input message for which an RSA signature is desired
*p_dest	out	generated signature data will be written here.

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_padded_hash

`uint32_t`

Parameter *p_dest

`uint32_t`

signCrt

```
uint32_t(* rsa_api_t::signCrt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t
*p_dest)
```

Brief description

Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. RSA private key p_key is assumed to be in CRT format. Write the results to the buffer p_dest. The RSA CRT key consists of the exponent2 || prime2 || exponent1 || prime1 || coefficient, starting with exponent2 at index 0. The size of each of these parameter is 512-bits for the g_rsa1024_on_sce implementation and 1024-bits for the g_rsa2048_on_sce implementation.

Detailed description

Table 841:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure for RSA interface

Table 841:Parameters (Continued)

Name	Direction	Description
*p_key	in	pointer to RSA private key in CRT format.
*p_domain	in	unused parameter. NULL value is acceptable.
num_words	in	data buffer size in words. Each word is 4-bytes. multiples of 4
*p_padded_hash	in	padded hash for the input message for which an RSA signature is desired
*p_dest	out	generated signature data will be written here.

Parameter *p_ctrl

Parameter *p_key

`uint32_t`

Parameter *p_domain

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_padded_hash

`uint32_t`

Parameter *p_dest

`uint32_t`

close

```
uint32_t(* rsa_api_t::close) (rsa_ctrl_t *const p_ctrl)
```

Detailed description

Close the RSA module.

Table 842:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure

Parameter p_ctrl

Definition: `rsa_ctrl_t`

RSA Interface control structure

versionGet

```
uint32_t(* rsa_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 843:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

keyCreate

```
uint32_t(* rsa_api_t::keyCreate) (rsa_ctrl_t *const p_ctrl, rsa_key_t
*p_private_key, rsa_key_t *p_public_key)
```

Detailed description

Generates an RSA key. This is a blocking call

Table 844:Parameters

Name	Direction	Description
*p_ctrl	in	pointer to control structure for RSA interface
*p_private_key	inout	pointer to a private key structure
*p_public_key	inout	pointer to a public key structure

```
{
  // The following code snippet gives an example for generating and using a 1024
-bit RSA key.
  // For simplicity, the below code snippet does not check return values.

     rsa_ctrl_t rsa_ctrl;
     rsa_cfg_t  rsa_cfg;
     rsa_key_t  rsa_secret_key;
     rsa_key_t  rsa_public_key;
     uint8_t    rsa_secret_key_data[RSA_PLAIN_TEXT_PRIVATE_KEY_SIZE_BYTES(1024)]
;
     uint8_t    rsa_public_key_data[RSA_PLAIN_TEXT_PUBLIC_KEY_SIZE_BYTES(1024)];

     // This example shows generation of an RSA private key in standard format.
```

```
    // To generate a CRT key, the key_format field should be set to RSA_KEY_FOR
MAT_PLAIN_TEXT_CRT_KEY
    // and define rsa_secret_key_data buffer to be of size RSA_KEY_FORMAT_PLAIN
_TEXT_CRT_KEY
    rsa_secret_key.key_format = RSA_KEY_FORMAT_PLAIN_TEXT_PRIVATE_KEY;
    rsa_secret_key.length     = sizeof(rsa_secret_key_data);
    rsa_secret_key.p_data     = rsa_secret_key_data;

    rsa_public_key.key_format = RSA_KEY_FORMAT_PLAIN_TEXT_PUBLIC_KEY;
    rsa_public_key.length     = sizeof(rsa_public_key_data);
    rsa_publi_key.p_data      = rsa_public_key_data;

    g_rsa1024_on_sce.open(&rsa_ctrl, &rsa_cfg);

    g_rsa1024_on_sce.keyCreate(p_ctrl, &rsa_secret_key, &rsa_public_key);

    // p_source is a pointer to a padded hash data. The computed signature will
 be stored at p_dest
    // The example below uses an RSA private key in standard format.
    // To compute signature using an RSA CRT private key, use the signCrt() int
erface function.
    g_rsa1024_on_sce.sign(p_ctrl, num_words, rsa_secret_key.p_data, p_source, p
_dest)

    g_rsa1024_on_sce.verify(p_ctrl, num_words, rsa_public_key.p_data, p_dest, p
_source);
}
```

Parameter *p_ctrl

Parameter *p_private_key

Definition: `rsa_key_t*p_private_key`

RSA key data structure

- `rsa_key_t::key_format`

  Indicates if the key is in plain-text format or encrypted using device unique key.

- `rsa_key_t::length`

  Length in bytes of the p_data buffer.

- `rsa_key_t::p_data`

  Buffer where the private key is stored.

Parameter *p_public_key

Definition: `rsa_key_t*p_public_key`

RSA key data structure

- `rsa_key_t::key_format`

  Indicates if the key is in plain-text format or encrypted using device unique key.
- `rsa_key_t::length`

  Length in bytes of the p_data buffer.
- `rsa_key_t::p_data`

  Buffer where the private key is stored.

rsa_instance_t

rsa_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- rsa_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- rsa_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- rsa_api_t const * p_api

  Pointer to the API structure for this instance.

rsa_key_format_t

`rsa_key_format_t`

Detailed description

RSA key format definitions

Enumerated values

Name	Description
RSA_KEY_FORMAT_PLAIN_TEXT_PUBLIC_KEY	RSA public key in plain text format.
RSA_KEY_FORMAT_PLAIN_TEXT_PRIVATE_KEY	RSA private key in plain text format.
RSA_KEY_FORMAT_PLAIN_TEXT_CRT_KEY	RSA CRT Key in plain text format.
RSA_KEY_FORMAT_WRAPPED_PRIVATE_KEY	RSA private Key wrapped using device specific key.

g_sce_crypto_api

`crypto_api_t::g_sce_crypto_api`

g_rsa1024_on_sce

`rsa_api_t::g_rsa1024_on_sce`

Detailed description

RSA interface is only available on S7G2, S5D9 and S5D5.

SCE/RSA implementation of RSA API.

g_rsa2048_on_sce

`rsa_api_t::g_rsa2048_on_sce`

g_rsa1024_on_sce_hrk

`rsa_api_t::g_rsa1024_on_sce_hrk`

Detailed description

SCE/RSA implementation of RSA API.

g_rsa2048_on_sce_hrk

`rsa_api_t::g_rsa2048_on_sce_hrk`

Detailed description

SCE/RSA implementation of RSA API.

9.7.8.5 TDES Interface

TDES encryption and decryption APIs.

Interface API

tdes_api_t

Function name	Description
.open	TDES module open function. Must be called before performing any encrypt/decrypt operations.
.encrypt	TDES encryption. Encrypt input data with a 192-bit TDES key and the chaining mode specified.
.decrypt	TDES decryption. Decrypt input data with a 192-bit TDES key and the chaining mode specified.
.close	Close the TDES module.
.versionGet	Gets version and stores it in provided pointer p_version.

Data structures

- tdes_ctrl_t
- tdes_cfg_t
- tdes_instance_t

Variables

- g_sce_crypto_api
- g_tdes192ecb_on_sce
- g_tdes192cbc_on_sce
- g_tdes192ctr_on_sce

Defines

- #define TDES_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define TDES_API_VERSION_MINOR

  Initial value:`(00)`

tdes_ctrl_t

tdes_ctrl_t

Detailed description

TDES Interface control structure

Variables

- crypto_ctrl_t crypto_ctrl

  pointer to crypto control structure

- crypto_api_t const * p_crypto_api

  pointer to crypto engine API

tdes_cfg_t

tdes_cfg_t

Detailed description

TDES Interface configuration structure. User must fill in these values before invoking the open() function

Variables

- crypto_api_t const * p_crypto_api

  pointer to crypto engine api

tdes_api_t

tdes_api_t

Detailed description

TDES_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* tdes_api_t::open) (tdes_ctrl_t *const p_ctrl, tdes_cfg_t const *const p_cfg)
```

Detailed description

TDES module open function. Must be called before performing any encrypt/decrypt operations.

Table 845:Parameters

Name	Direction	Description
p_ctrl	inout	pointer to control structure for the TDES interface. Must be declared by user. Elements are set here.
p_cfg	in	pointer to control structure for the TDES configuration. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `tdes_ctrl_t`

TDES Interface control structure

Parameter p_cfg

Definition: `tdes_cfg_t const *const p_cfg`

TDES Interface configuration structure. User must fill in these values before invoking the open() function

- `tdes_cfg_t`::`crypto_api_t`

  pointer to crypto engine api

encrypt

```
uint32_t(* tdes_api_t::encrypt) (tdes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
```

Brief description

TDES encryption.

Detailed description

Encrypt input data with a 192-bit TDES key and the chaining mode specified.

Table 846:Parameters

Name	Direction	Description
*p_key	in	pointer to the TDES plain-text key.
*p_iv	inout	pointer to initialization vector. Should be 8 bytes long. Unused for ECB mode. Next IV value is returned on successful completion.

Table 846:Parameters (Continued)

Name	Direction	Description
num_words	in	Specifies the size of the input data in words. Should be multiples of 2. Note: 1 word is 4-bytes long.
*p_source	in	input data buffer - should be at least num_words long.
*p_dest	out	output data buffer - should be at least num_words long.

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

decrypt

```
uint32_t(* tdes_api_t::decrypt) (tdes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
```

Brief description

TDES decryption.

Detailed description

Decrypt input data with a 192-bit TDES key and the chaining mode specified.

Table 847:Parameters

Name	Direction	Description
*p_key	in	pointer to the 192-bit plain-text key
*p_iv	inout	pointer to initialization vector. Should be 8 bytes long. Unused for ECB mode. Next IV value is returned on successful completion.

Table 847:Parameters (Continued)

Name	Direction	Description
num_words	in	Specifies the size of the input data in words. Should be multiples of 2. Note: 1 word is 4-bytes long.
*p_source	in	input data buffer - should be at least num_words long.
*p_dest	out	output data buffer - should be at least num_words long.

Parameter *p_key

`uint32_t`

Parameter *p_iv

`uint32_t`

Parameter num_words

`uint32_t`

Parameter *p_source

`uint32_t`

Parameter *p_dest

`uint32_t`

close

```
uint32_t(* tdes_api_t::close) (tdes_ctrl_t *const p_ctrl)
```

Detailed description

Close the TDES module.

Table 848:Parameters

Name	Direction	Description
p_ctrl	in	pointer to the control structure

Parameter p_ctrl

Definition: `tdes_ctrl_t`

TDES Interface control structure

versionGet

```
uint32_t(* tdes_api_t::versionGet) ( *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 849:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

tdes_instance_t

tdes_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- tdes_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- tdes_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- tdes_api_t const * p_api

  Pointer to the API structure for this instance.

g_sce_crypto_api

`crypto_api_t::g_sce_crypto_api`

g_tdes192ecb_on_sce

`tdes_api_t::g_tdes192ecb_on_sce`

Detailed description

TDES interface is only available on S7G2 and S5D9.

SCE/TDES implementation of TDES API.

g_tdes192cbc_on_sce

`tdes_api_t::g_tdes192cbc_on_sce`

g_tdes192ctr_on_sce

`tdes_api_t::g_tdes192ctr_on_sce`

9.7.8.6 Random number generation

RNG_Interface Random number generation.

Interface API

trng_api_t

Function name	Description
.open	Open the TRNG driver for reading random data from the hardware TRNG module
.read	Generate of random number words and store them in buffer
.close	Close the TRNG interface driver
.versionGet	Gets version and stores it in provided pointer p_version.

Data structures

- trng_ctrl_t
- trng_cfg_t
- trng_instance_t

Defines

- #define TRNG_API_VERSION_MAJOR

  Initial value:`(01)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define TRNG_API_VERSION_MINOR

  Initial value:`(00)`
- #define TRNG_REGISTER_SIZE_WORDS

  Initial value:`(4)`
- #define TRNG_REGISTER_SIZE_BYTES

  Initial value:`((TRNG_REGISTER_SIZE_WORDS) * 4)`

trng_ctrl_t

trng_ctrl_t

Detailed description

TRNG_Interface control structure.

Variables

- uint32_t nattempts

  number of retries
- crypto_ctrl_t * p_crypto_ctrl

  pointer to crypto control structure

- crypto_api_t const * p_crypto_api

  pointer to crypto-engine API
- uint32_t prevbuf[TRNG_REGISTER_SIZE_WORDS]

  previous random data
- uint32_t currbuf[TRNG_REGISTER_SIZE_WORDS]

  current random data

trng_cfg_t

trng_cfg_t

Detailed description

TRNG interface configuration parameters

Variables

- crypto_api_t const * p_crypto_api

  pointer to crypto API
- uint32_t nattempts

  number of retries when a continuous test failure occurs

trng_api_t

trng_api_t

Detailed description

TRNG_Interface SCE functions implemented at the HAL layer will follow this API.

open

```
uint32_t(* trng_api_t::open) (trng_ctrl_t *const p_ctrl, trng_cfg_t const *const p_cfg)
```

Detailed description

Open the TRNG driver for reading random data from the hardware TRNG module

Table 850:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to control structure. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `trng_ctrl_t`

TRNG_Interface control structure.

Parameter p_cfg

Definition: `trng_cfg_t const *const p_cfg`

TRNG interface configuration parameters

- `trng_cfg_t`::`crypto_api_t`

  pointer to crypto API

- `trng_cfg_t`::`nattempts`

  number of retries when a continuous test failure occurs

read

```
uint32_t(* trng_api_t::read) (trng_ctrl_t *const p_ctrl, uint32_t *const p_rngbuf,
uint32_t nwords)
```

Detailed description

Generate nwords of random number words and store them in p_rngbuf buffer

Table 851:Parameters

Name	Direction	Description
p_ctrl	in	pointer to trng control structure
p_rngbuf	out	generated random numbers will be stored to the buffer p_rngbuf
nwords	in	number of random words to generate

Parameter p_ctrl

Definition: `trng_ctrl_t`

TRNG_Interface control structure.

Parameter p_rngbuf

`uint32_t`

Parameter nwords

`uint32_t`

close

```
uint32_t(* trng_api_t::close) (trng_ctrl_t *const p_ctrl)
```

Detailed description

Close the TRNG interface driver

Table 852:Parameters

Name	Direction	Description
p_ctrl	in	pointer to trng interface control structure

Parameter p_ctrl

Definition: `trng_ctrl_t`

TRNG_Interface control structure.

versionGet

```
uint32_t(* trng_api_t::versionGet) ( *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version.

Table 853:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

trng_instance_t

trng_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- trng_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- trng_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- trng_api_t const * p_api

  Pointer to the API structure for this instance.

9.8 Capacitive Touch Sensing

9.9 CTSU Interface

Interface for Capacitive Touch Controllers.

9.9.1 Summary

The CTSU interface provides the functionality necessary to open, close, run and control the CTSU depending upon the configuration passed as arguments.

Implemented by: CTSU

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

CTSU Interface description: CTSU Driver

9.9.2 Interface API

ctsu_api_t

Function name	Description
.open	Initialize the CTSU; enable power and clock and set the register configuration.
.close	Close the CTSU by ending any scan in progress, disabling interrupts, and removing power to the peripheral and saving configurations according to options.
.scan	Start off a single CTSU scan.
.update	Update the CTSU internal data including the touch decision and other derived data according to the results of the scan.
.read	Read the results from the CTSU including raw data, binary data and other derived data according to the selected options.
.versionGet	Retrieve the API version.

9.9.3 Data structures

- ctsu_callback_args_t
- ctsu_channel_pair_t
- ctsu_channel_setting_t
- ctsu_channel_data_self_t
- ctsu_channel_data_mutual_t
- ctsu_hw_cfg_t
- ctsu_functions_t
- ctsu_cfg_t
- ctsu_instance_t

9.9.4 Enumerations

- ctsu_event_t
- ctsu_read_t
- ctsu_process_option_t
- ctsu_close_option_t
- ctsu_action_t

9.9.5 Typedefs

- ctsu_ctrl_t

9.9.6 Defines

- #define CTSU_API_VERSION_MAJOR

  Initial value:`(1U)`

  Includes board and MCU related header files. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Version of the API defined in this file
- #define CTSU_API_VERSION_MINOR

  Initial value:`(2U)`

9.9.7 API Data

9.9.7.1 ctsu_event_t

```
ctsu_event_t
```

Detailed description

CTSU callback event definitions

Enumerated values

Name	Description
CTSU_EVENT_SCAN_COMPLETE	Scan cycle is complete. This callback is occurring from an ISR.
CTSU_EVENT_PARAMETERS_UPDATED	All derived parameters from raw CTSU data are updated. Callback is not from ISR.

9.9.7.2 ctsu_read_t

`ctsu_read_t`

Detailed description

Different options with which a user can query the results of a CTSU scan. User cannot logic 'or' these together.

Enumerated values

Name	Description
CTSU_READ_BINARY_DATA_ALL	Read binary string of touch determination.
CTSU_READ_BINARY_DATA_SEL	Read binary string of touch determination for selected channels only.
CTSU_READ_RAW_SENSOR_OUTPUT_ALL	Read raw CTSU output for all active channels.
CTSU_READ_RAW_SENSOR_OUTPUT_SEL	Read raw CTSU output for selected channels.
CTSU_READ_FILTERED_SENSOR_VALUES_ALL	Read scount values output from filter for all active channels.
CTSU_READ_FILTERED_SENSOR_VALUES_SEL	Read scount values output from filter for all active channels.
CTSU_READ_DELTA_COUNT_ALL	Write out the difference between sensor and baseline count to p_dest array for all active sensors.
CTSU_READ_DELTA_COUNT_SEL	Write out the difference between sensor and baseline count to p_dest array for selected channels.
CTSU_READ_BASELINE_COUNT_ALL	Read the current sensor baseline used to determine if a channel is being touched for all active channels.
CTSU_READ_BASELINE_COUNT_SEL	Read the current sensor baseline used to determine if a channel is being touched for selected channels.

Name	Description
CTSU_READ_FILTERED_REF_ICO_VALUES_ALL	Read the rcount values output from filter for all active channels.
CTSU_READ_FILTERED_REF_ICO_VALUES_SEL	Read the rcount values output from filter for selected channels only.

9.9.7.3 ctsu_process_option_t

`ctsu_process_option_t`

Detailed description

Different options that can be provided to the Process function.

Enumerated values

Name	Description
CTSU_PROCESS_OPTION_DEFAULT_SETTING	Recommended option for most use cases.
CTSU_PROCESS_OPTION_AUTO_SCAN	Automatically start a scan after all internal variables have been updated.
CTSU_PROCESS_OPTION_ENABLE_DRIFT_COMP	Perform drift compensation and start a new scan when internal variables are updated.
CTSU_PROCESS_OPTION_ENABLE_AUTO_CALIB	Perform auto-calibration and start a new scan when internal variables are updated.
CTSU_PROCESS_OPTION_NONE	Skip all optional actions.

9.9.7.4 ctsu_close_option_t

`ctsu_close_option_t`

Detailed description

Options which can be passed to the Close function. User can logically 'OR' these together.

Enumerated values

Name	Description
CTSU_CLOSE_OPTION_SUSPEND	Suspend the CTSU by setting the CTSUSNZ bit.
CTSU_CLOSE_OPTION_SAVE_CONFIG	Save the configuration in the second argument provided to the close function.

Name	Description
CTSU_CLOSE_OPTION_RESET_SFRS	Bring the SFRs back to their P-O-RS values.
CTSU_CLOSE_OPTION_POWER_DOWN	Disable clock supply to the CTSU and set it into low power mode.

9.9.7.5 ctsu_action_t

```
ctsu_action_t
```

Detailed description

State of the Process/Update CTSU sensor data function

Enumerated values

Name	Description
CTSU_ACTION_START_NEW_SCAN	Start new scan.
CTSU_ACTION_WAITING_FOR_SCAN_COMPLETE	Wait for scan complete.
CTSU_ACTION_CHECK_FOR_ERRORS	Check for errors.
CTSU_ACTION_FILTER_DATA	Filter data.
CTSU_ACTION_UPDATE_TOUCH_INFO	Update touch info.
CTSU_ACTION_UPDATE_BUTTONS	Update buttons.
CTSU_ACTION_UPDATE_SLIDERS	Update sliders.
CTSU_ACTION_RUN_DRIFT_COMP	Run drift compensation.
CTSU_ACTION_RUN_AUTO_TUNE	Run auto tuning.
CTSU_ACTION_RESTART_STATE_MACHINE	Restart state machine.

9.9.7.6 ctsu_ctrl_t

```
typedef void ctsu_ctrl_t
```

Detailed description

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented as

- ctsu_instance_ctrl_t

9.9.8 API Structures

9.9.8.1 ctsu_callback_args_t

ctsu_callback_args_t

Detailed description

CTSU callback arguments definitions

Variables

- ctsu_event_t event

  CTSU callback event.

- void const * p_context

  Placeholder for user data.

9.9.8.2 ctsu_channel_pair_t

ctsu_channel_pair_t

Detailed description

Structure storing information about a touch sensor element

Variables

- int8_t rx

  Denotes the primary channel.

- int8_t tx

  Denotes the secondary channel (used only for mutual capacitance mode)

9.9.8.3 ctsu_channel_setting_t

ctsu_channel_setting_t

Detailed description

Structure to be holding values to be written to the SFRs in the WR ISR

Variables

- uint16_t ctsussc

  Holds value for the CTSUSSC register.

- uint16_t ctsuso0

  Holds value for the CTSUSO0 register.

- uint16_t ctsuso1

  Holds value for the CTSUSO1 register.

9.9.8.4 ctsu_channel_data_self_t

ctsu_channel_data_self_t

Detailed description

Immediate raw data readings from the CTSU registers CTSUSC(sensor count) and CTSURC(Reference count) in self-capacitance mode.

Variables

- uint16_t sensor_count

  Raw sensor count.

- uint16_t reference_count

  Raw reference count.

9.9.8.5 ctsu_channel_data_mutual_t

ctsu_channel_data_mutual_t

Detailed description

Immediate raw data readings from the CTSU registers CTSUSC(sensor count) and CTSURC(Reference count) in mutual-capacitance mode.

Variables

- uint16_t sen_cnt_1

  Raw Sensor count primary reading.

- uint16_t ref_cnt_1

  Raw reference ICO count primary reading.

- uint16_t sen_cnt_2

  Raw sensor ICO count secondary reading.

- uint16_t ref_cnt_2

  Raw reference ICO count secondary reading.

9.9.8.6 ctsu_hw_cfg_t

ctsu_hw_cfg_t

Detailed description

Structure defining a hardware configuration to be passed to the Open function

Variables

- R_CTSU_Type ctsu_settings

  User defined SFR settings for CR0, CR1, SDPRS, SST, CHACn, CHTRCn, DCLKC.

- ctsu_channel_setting_t * write_settings

  User defined initial settings for thresholds for each active channel . Threshold is difference between runtime baseline and filtered output of sensor count.

  User defined settings for SSC, SO0, SO1 for each active channel.

- uint16_t * threshold
- uint16_t * hysteresis

  User defined settings for tolerance in count values.

- uint16_t * baseline

  A baseline of the expected count for each active channel when not touched.

- void * raw_result

  A pointer to a buffer which will hold raw results of CTSU measurement.

- void * filter_output

  A pointer to a buffer which will hold output after filtering raw results.

- void * binary_result

  A pointer to a location where binary data can be stored.

- ctsu_channel_pair_t * excluded

  A pointer to an array which contains a list of channel pairs which need to be ignored in ascending order of rx and then tx.

- int8_t num_excluded

  Number of elements in the array excluded.

- const uint16_t * series_resistance

  Resistance of the channel (to determine RC constant when tuning).

9.9.8.7 ctsu_functions_t

ctsu_functions_t

Detailed description

User defined functions that can be used to override the default processing functions used internally by the driver. For advanced usage only.

Variables

- int32_t(* preFilter)(void *p_args)

  Used for calculating raw data SNR before data gets filtered.

- int32_t(* filter)(volatile uint16_t *output, volatile uint16_t *input)

  Weighted averaging using CTSU_CFG_FILTER_DEPTH.

- int32_t(* postFilter)(void *p_args)

  Processing the filter results.

- int32_t(* ctsuDecode)(void *p_args)

  Algorithm to decide if channel is touched or not.
- int32_t(* otDriftComp)(void *p_args)

  Algorithm to perform manipulations to baseline, envelope, and thresholds on initialization.
- int32_t(* rtDriftComp)(void *p_args)

  Algorithm to perform run time manipulations to baseline, envelope, and thresholds.
- int32_t(* otAutoTune)(void *p_args)

  Function called once on initialization of system.
- int32_t(* rtAutoTune)(void *p_args)

  Function called to auto tune sensor when system is running.

9.9.8.8 ctsu_cfg_t

ctsu_cfg_t

Detailed description

Structure defining a configuration structure for the CTSU driver

Variables

- transfer_instance_t const *const p_lower_lvl_transfer_read

  Pointer to the Transfer instance to read results.
- transfer_instance_t const *const p_lower_lvl_transfer_write

  Pointer to the Transfer instance to write cfg.
- ctsu_hw_cfg_t * p_ctsu_hw_cfg

  Pointer to a CTSU configuration.
- ctsu_functions_t * p_ctsu_functions

  Pointer to a place holder for custom data functions.
- void(* p_callback)(ctsu_callback_args_t *p_args)

  Callback to the function to use when scan is complete.
- void * p_context

  Pointer to data that should be passed to update_complete notification.
- ctsu_process_option_t ctsu_soft_option

  Software options to use when performing Open and Process.
- ctsu_close_option_t ctsu_close_option

  Software options to use when closing touch operation.
- uint8_t write_ipl

  Write interrupt priority.

- uint8_t read_ipl

  Read interrupt priority.

- uint8_t end_ipl

  End interrupt priority.

9.9.8.9 ctsu_api_t

ctsu_api_t

Detailed description

CTSU HAL driver API structure. Functions implemented at the HAL layer will follow this API.

9.9.8.10 open

```
ssp_err_t(* ctsu_api_t::open) (ctsu_ctrl_t *p_ctrl, ctsu_cfg_t *p_cfg)
```

Detailed description

Initialize the CTSU; enable power and clock and set the register configuration. Implemented as

- R_CTSU_Open

Table 854:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_cfg	in	Pointer to configuration structure

Parameter p_ctrl

Definition: `ctsu_ctrl_t*p_ctrl`

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented asctsu_instance_ctrl_t

Parameter p_cfg

Definition: `ctsu_cfg_t *p_cfg`

Structure defining a configuration structure for the CTSU driver

- `ctsu_cfg_t`::`transfer_instance_t`

  Pointer to the Transfer instance to read results.

- `ctsu_cfg_t`::`transfer_instance_t`

  Pointer to the Transfer instance to write cfg.

- `ctsu_cfg_t`::`ctsu_hw_cfg_t`

  Pointer to a CTSU configuration.

- ctsu_cfg_t::ctsu_functions_t

  Pointer to a place holder for custom data functions.
- ctsu_cfg_t::p_callback

  Callback to the function to use when scan is complete.
- ctsu_cfg_t::p_context

  Pointer to data that should be passed to update_complete notification.
- ctsu_cfg_t::ctsu_process_option_t

  Software options to use when performing Open and Process.

  Enumerated as:

  – CTSU_PROCESS_OPTION_DEFAULT_SETTING

  – CTSU_PROCESS_OPTION_AUTO_SCAN

  – CTSU_PROCESS_OPTION_ENABLE_DRIFT_COMP

  – CTSU_PROCESS_OPTION_ENABLE_AUTO_CALIB

  – CTSU_PROCESS_OPTION_NONE
- ctsu_cfg_t::ctsu_close_option_t

  Software options to use when closing touch operation.

  Enumerated as:

  – CTSU_CLOSE_OPTION_SUSPEND

  – CTSU_CLOSE_OPTION_SAVE_CONFIG

  – CTSU_CLOSE_OPTION_RESET_SFRS

  – CTSU_CLOSE_OPTION_POWER_DOWN
- ctsu_cfg_t::write_ipl

  Write interrupt priority.
- ctsu_cfg_t::read_ipl

  Read interrupt priority.
- ctsu_cfg_t::end_ipl

  End interrupt priority.

9.9.8.11 close

```
ssp_err_t(* ctsu_api_t::close) (ctsu_ctrl_t *p_ctrl, ctsu_close_option_t opts)
```

Detailed description

Close the CTSU by ending any scan in progress, disabling interrupts, and removing power to the peripheral and saving configurations according to options. Implemented as

- R_CTSU_Close

Table 855:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
opts	in	Closing options

Parameter p_ctrl

Definition: `ctsu_ctrl_t*p_ctrl`

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented asctsu_instance_ctrl_t

Parameter opts

Definition: `ctsu_close_option_topts`

Options which can be passed to the Close function. User can logically 'OR' these together.

9.9.8.12 scan

```
ssp_err_t(* ctsu_api_t::scan) (ctsu_ctrl_t *p_ctrl)
```

Detailed description

Start off a single CTSU scan. Implemented as

- R_CTSU_Start_Scan

Table 856:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `ctsu_ctrl_t*p_ctrl`

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented asctsu_instance_ctrl_t

9.9.8.13 update

```
ssp_err_t(* ctsu_api_t::update) (ctsu_ctrl_t *p_ctrl)
```

Detailed description

Update the CTSU internal data including the touch decision and other derived data according to the results of the scan. Implemented as

- R_CTSU_Update_Parameters

Table 857:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure

Parameter p_ctrl

Definition: `ctsu_ctrl_t*p_ctrl`

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented asctsu_instance_ctrl_t

9.9.8.14 read

```
ssp_err_t(* ctsu_api_t::read) (ctsu_ctrl_t *p_ctrl, void *p_dest, ctsu_read_t
opts, const ctsu_channel_pair_t *channels, const uint16_t count)
```

Detailed description

Read the results from the CTSU including raw data, binary data and other derived data according to the selected options. Implemented as

- R_CTSU_Read_Results

Table 858:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_dest	out	Pointer to the destination location for the read data
opts	in	Read options, use only one option per call to read, do not logically OR values
channels	in	Specify the channel/channel pairs to read data for
count	in	Specify the number of channel/channel pairs to read data for

Parameter p_ctrl

Definition: `ctsu_ctrl_t*p_ctrl`

CTSU control block. Allocate an instance specific control block to pass into the CTSU API calls. Implemented asctsu_instance_ctrl_t

Parameter p_dest

const

Parameter opts

Definition: ctsu_read_topts

Different options with which a user can query the results of a CTSU scan. User cannot logic 'or' these together.

Parameter channels

Definition: ctsu_channel_pair_tctsu_channel_pair_t *channels

Structure storing information about a touch sensor element

- ctsu_channel_pair_t::rx

  Denotes the primary channel.

- ctsu_channel_pair_t::tx

  Denotes the secondary channel (used only for mutual capacitance mode)

Parameter count

uint16_t

9.9.8.15 versionGet

ssp_err_t(* ctsu_api_t::versionGet) (ssp_version_t *const p_version)

Detailed description

Retrieve the API version. Implemented as

- R_CTSU_VersionGet

NOTE: This function retrieves the API version.

Table 859:Parameters

Name	Direction	Description
p_version	in	Pointer to version structure

Parameter p_version

9.9.8.16 ctsu_instance_t

ctsu_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- ctsu_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- ctsu_cfg_t * p_cfg

  Pointer to the configuration structure for this instance.

- ctsu_api_t const * p_api

  Pointer to the API structure for this instance.

9.10 Digital-to-Analog Converter

9.10.1 DAC Interface

Interface for D/A converters.

9.10.1.1 Summary

The DAC interface provides standard Digital/Analog Converter functionality. A DAC application writes digital sample data to the device and generates analog output on the DAC output pin.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

DAC Interface description: DAC Driver

9.10.1.2 Interface API

dac_api_t

Function name	Description
.open	Initial configuration.
.close	Close the D/A Converter.
.write	Write sample value to the D/A Converter.
.start	Start the D/A Converter if it has not been started yet.
.stop	Stop the D/A Converter if the converter is running.
.versionGet	Get version and store it in provided pointer p_version.

9.10.1.3 Data structures

- dac_cfg_t
- dac_instance_t

9.10.1.4 Enumerations

- dac_data_format_t

9.10.1.5 Typedefs

- dac_size_t
- dac_ctrl_t

9.10.1.6 Defines

- #define DAC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DAC_API_VERSION_MINOR

  Initial value:`(1U)`

9.10.1.7 API Data

dac_data_format_t

`dac_data_format_t`

Detailed description

DAC Open API AD/DA data format settings.

Enumerated values

Name	Description
DAC_DATA_FORMAT_FLUSH_RIGHT	LSB of data is flush to the right leaving the top 4 bits unused.
DAC_DATA_FORMAT_FLUSH_LEFT	MSB of data is flush to the left leaving the bottom 4 bits unused.

dac_size_t

```
typedef uint16_t dac_size_t
```

Detailed description

Data type to store DAC output value.

dac_ctrl_t

```
typedef void dac_ctrl_t
```

Detailed description

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented as

- dac_instance_ctrl_t

9.10.1.8 API Structures

dac_cfg_t

dac_cfg_t

Detailed description

DAC Open API configuration parameter

Variables

- uint8_t channel

  ID associated with this DAC channel.

- bool ad_da_synchronized

  AD/DA synchronization.

- dac_data_format_t data_format

  Data format.

- bool output_amplifier_enabled

  Output amplifier enable.

- void const * p_extend

dac_api_t

dac_api_t

Detailed description

DAC driver structure. General DAC functions implemented at the HAL layer follow this API.

open

```
ssp_err_t(* dac_api_t::open) (dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_DAC_Open

Table 860:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

Parameter p_cfg

Definition: `dac_cfg_t const *const p_cfg`

DAC Open API configuration parameter

- `dac_cfg_t`::`channel`

  ID associated with this DAC channel.

- `dac_cfg_t`::`ad_da_synchronized`

  AD/DA synchronization.

- `dac_cfg_t`::`dac_data_format_t`

  Data format.

  Enumerated as:

  – DAC_DATA_FORMAT_FLUSH_RIGHT

  – DAC_DATA_FORMAT_FLUSH_LEFT

- `dac_cfg_t`::`output_amplifier_enabled`

  Output amplifier enable.

- `dac_cfg_t`::`p_extend`

close

```
ssp_err_t(* dac_api_t::close) (dac_ctrl_t *p_ctrl)
```

Detailed description

Close the D/A Converter. Implemented as

- R_DAC_Close

Table 861:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

write

```
ssp_err_t(* dac_api_t::write) (dac_ctrl_t *p_ctrl, dac_size_t value)
```

Detailed description

Write sample value to the D/A Converter. Implemented as

- R_DAC_Write

Table 862:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
value	in	Sample value to be written to the D/A Converter.

Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

Parameter value

Definition: `dac_size_tvalue`

Data type to store DAC output value.

start

```
ssp_err_t(* dac_api_t::start) (dac_ctrl_t *p_ctrl)
```

Detailed description

Start the D/A Converter if it has not been started yet. Implemented as

- R_DAC_Start

Table 863:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

stop

```
ssp_err_t(* dac_api_t::stop) (dac_ctrl_t *p_ctrl)
```

Detailed description

Stop the D/A Converter if the converter is running. Implemented as

- R_DAC_Stop

Table 864:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

versionGet

```
ssp_err_t(* dac_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_DAC_VersionGet

Table 865:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

dac_instance_t

dac_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- dac_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- dac_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- dac_api_t const * p_api

  Pointer to the API structure for this instance.

9.10.2 dac_api_t

dac_api_t

9.10.2.1 Detailed description

DAC driver structure. General DAC functions implemented at the HAL layer follow this API.

9.10.3 open

```
ssp_err_t(* dac_api_t::open) (dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg)
```

9.10.3.1 Detailed description

Initial configuration. Implemented as

- R_DAC_Open

Table 866:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

9.10.3.2 Parameter p_ctrl

Definition: dac_ctrl_t*p_ctrl

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

9.10.3.3 Parameter p_cfg

Definition: dac_cfg_t const *const p_cfg

DAC Open API configuration parameter

- dac_cfg_t::channel

  ID associated with this DAC channel.

- dac_cfg_t::ad_da_synchronized

  AD/DA synchronization.

- dac_cfg_t::dac_data_format_t

  Data format.

  Enumerated as:

  – DAC_DATA_FORMAT_FLUSH_RIGHT

  – DAC_DATA_FORMAT_FLUSH_LEFT

- dac_cfg_t::output_amplifier_enabled

  Output amplifier enable.

- dac_cfg_t::p_extend

9.10.4 close

ssp_err_t(* dac_api_t::close) (dac_ctrl_t *p_ctrl)

9.10.4.1 Detailed description

Close the D/A Converter. Implemented as

- R_DAC_Close

Table 867:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

9.10.4.2 Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

9.10.5 write

```
ssp_err_t(* dac_api_t::write) (dac_ctrl_t *p_ctrl, dac_size_t value)
```

9.10.5.1 Detailed description

Write sample value to the D/A Converter. Implemented as

- R_DAC_Write

Table 868:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
value	in	Sample value to be written to the D/A Converter.

9.10.5.2 Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

9.10.5.3 Parameter value

Definition: `dac_size_tvalue`

Data type to store DAC output value.

9.10.6 start

```
ssp_err_t(* dac_api_t::start) (dac_ctrl_t *p_ctrl)
```

9.10.6.1 Detailed description

Start the D/A Converter if it has not been started yet. Implemented as

- R_DAC_Start

Table 869:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

9.10.6.2 Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

9.10.7 stop

```
ssp_err_t(* dac_api_t::stop) (dac_ctrl_t *p_ctrl)
```

9.10.7.1 Detailed description

Stop the D/A Converter if the converter is running. Implemented as

- R_DAC_Stop

Table 870:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

9.10.7.2 Parameter p_ctrl

Definition: `dac_ctrl_t*p_ctrl`

DAC control block. Allocate an instance specific control block to pass into the DAC API calls. Implemented asdac_instance_ctrl_t

9.10.8 versionGet

```
ssp_err_t(* dac_api_t::versionGet) (ssp_version_t *p_version)
```

9.10.8.1 Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_DAC_VersionGet

Table 871:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

9.10.8.2 Parameter p_version

9.10.9 dac_cfg_t

dac_cfg_t

9.10.9.1 Detailed description

DAC Open API configuration parameter

9.10.9.2 Variables

- uint8_t channel
  ID associated with this DAC channel.
- bool ad_da_synchronized
  AD/DA synchronization.
- dac_data_format_t data_format
  Data format.
- bool output_amplifier_enabled
  Output amplifier enable.
- void const * p_extend

9.11 Display Interface

Interface for LCD panel displays.

9.11.1 Summary

The display interface provides standard display functionality:

- Signal timing configuration for LCD panels with RGB interface.
- Dot clock source selection (internal or external) and frequency divider.
- Blending of multiple graphics layers on the background screen.
- Color correction (brightness/configuration/gamma correction).
- Interrupts and callback function.

Implemented by: GLCDC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Display Interface description: GLCDC Display Driver

9.11.2 Interface API

display_api_t

Function name	Description
.open	Open display device.
.close	Close display device.
.start	Display start.
.stop	Display stop.
.layerChange	Change layer parameters at runtime.
.correction	Color correction.
.clut	Set CLUT for display device.
.statusGet	Get status for display device.
.versionGet	Get version.

9.11.3 Data structures

- display_timing_t
- display_color_t
- display_coordinate_t
- display_brightness_t
- display_contrast_t
- display_correction_t
- gamma_correction_t
- display_gamma_correction_t
- display_clut_t
- display_input_cfg_t
- display_output_cfg_t
- display_layer_t
- display_callback_args_t
- display_cfg_t
- display_runtime_cfg_t
- display_clut_cfg_t
- display_status_t
- display_instance_t

9.11.4 Enumerations

- display_frame_layer_t
- display_state_t
- display_event_t
- display_in_format_t
- display_out_format_t
- display_endian_t
- display_color_order_t
- display_signal_polarity_t
- display_sync_edge_t
- display_fade_control_t
- display_fade_status_t

9.11.5 Typedefs

- display_ctrl_t

9.11.6 Defines

- #define DISPLAY_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DISPLAY_API_VERSION_MINOR

  Initial value:`(2U)`
- #define DISPLAY_GAMMA_CURVE_ELEMENT_NUM

  Initial value:`(16)`

9.11.7 API Data

9.11.7.1 display_frame_layer_t

`display_frame_layer_t`

Detailed description

Display frame number

Enumerated values

Name	Description
DISPLAY_FRAME_LAYER_1	Frame layer 1.
DISPLAY_FRAME_LAYER_2	Frame layer 2.

9.11.7.2 display_state_t

`display_state_t`

Detailed description

Display interface operation state

Enumerated values

Name	Description
DISPLAY_STATE_CLOSED	Display closed.
DISPLAY_STATE_OPENED	Display opened.
DISPLAY_STATE_DISPLAYING	Displaying.

9.11.7.3 display_event_t

`display_event_t`

Detailed description

Display event codes

Enumerated values

Name	Description
DISPLAY_EVENT_GR1_UNDERFLOW	Graphics frame1 underflow occurs.
DISPLAY_EVENT_GR2_UNDERFLOW	Graphics frame2 underflow occurs.
DISPLAY_EVENT_LINE_DETECTION	Designated line is processed.

9.11.7.4 display_in_format_t

`display_in_format_t`

Detailed description

Input format setting

Enumerated values

Name	Description
DISPLAY_IN_FORMAT_32BITS_ARGB8888	ARGB8888, 32 bits.
DISPLAY_IN_FORMAT_32BITS_RGB888	RGB888, 32 bits.
DISPLAY_IN_FORMAT_16BITS_RGB565	RGB565, 16 bits.
DISPLAY_IN_FORMAT_16BITS_ARGB1555	ARGB1555, 16 bits.
DISPLAY_IN_FORMAT_16BITS_ARGB4444	ARGB4444, 16 bits.

Name	Description
DISPLAY_IN_FORMAT_CLUT8	CLUT8.
DISPLAY_IN_FORMAT_CLUT4	CLUT4.
DISPLAY_IN_FORMAT_CLUT1	CLUT1.

9.11.7.5 display_out_format_t

`display_out_format_t`

Detailed description

Output format setting

Enumerated values

Name	Description
DISPLAY_OUT_FORMAT_24BITS_RGB888	RGB888, 24 bits.
DISPLAY_OUT_FORMAT_18BITS_RGB666	RGB666, 18 bits.
DISPLAY_OUT_FORMAT_16BITS_RGB565	RGB565, 16 bits.
DISPLAY_OUT_FORMAT_8BITS_SERIAL	SERIAL, 8 bits.

9.11.7.6 display_endian_t

`display_endian_t`

Detailed description

Data endian select

Enumerated values

Name	Description
DISPLAY_ENDIAN_LITTLE	Little-endian.
DISPLAY_ENDIAN_BIG	Big-endian.

9.11.7.7 display_color_order_t

`display_color_order_t`

Detailed description

RGB color order select

Enumerated values

Name	Description
DISPLAY_COLOR_ORDER_RGB	Color order RGB.
DISPLAY_COLOR_ORDER_BGR	Color order BGR.

9.11.7.8 display_signal_polarity_t

```
display_signal_polarity_t
```

Detailed description

Polarity of a signal select

Enumerated values

Name	Description
DISPLAY_SIGNAL_POLARITY_LOACTIVE	Low active signal.
DISPLAY_SIGNAL_POLARITY_HIACTIVE	High active signal.

9.11.7.9 display_sync_edge_t

```
display_sync_edge_t
```

Detailed description

Signal synchronization edge select

Enumerated values

Name	Description
DISPLAY_SIGNAL_SYNC_EDGE_RISING	Signal is synchronized to rising edge.
DISPLAY_SIGNAL_SYNC_EDGE_FALLING	Signal is synchronized to falling edge.

9.11.7.10 display_fade_control_t

```
display_fade_control_t
```

Detailed description

Fading control

Enumerated values

Name	Description
DISPLAY_FADE_CONTROL_NONE	Applying no fading control.
DISPLAY_FADE_CONTROL_FADEIN	Applying fade-in control.
DISPLAY_FADE_CONTROL_FADEOUT	Applying fade-out control.

9.11.7.11 display_fade_status_t

```
display_fade_status_t
```

Detailed description

Fading status

Enumerated values

Name	Description
DISPLAY_FADE_STATUS_NOT_UNDERWAY	Fade-in/fade-out is not in progress.
DISPLAY_FADE_STATUS_FADING_UNDERWAY	Fade-in or fade-out is in progress.
DISPLAY_FADE_STATUS_UNCERTAIN	Fade-in/fade-out status is uncertain just before hardware working.

9.11.7.12 display_ctrl_t

```
typedef void display_ctrl_t
```

Detailed description

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented as

- glcd_instance_ctrl_tDisplay control block

9.11.8 API Structures

9.11.8.1 display_timing_t

display_timing_t

Detailed description

Display signal timing setting

Variables

- uint16_t total_cyc

  Total cycles in one line or total lines in one frame.
- uint16_t display_cyc

  Active video cycles or lines.
- uint16_t back_porch

  Back poach cycles or lines.
- uint16_t sync_width

  Sync signal asserting width.
- display_signal_polarity_t sync_polarity

  Sync signal polarity.

9.11.8.2 display_color_t

display_color_t

Detailed description

RGB Color setting

Variables

- uint32_t argb
- uint8_t b

  blue
- uint8_t g

  green
- uint8_t r

  red
- uint8_t a

  a
- struct{} byte

  See source code for definition of this member.
- union{} union{}

  See source code for definition of this member.

9.11.8.3 display_coordinate_t

display_coordinate_t

Detailed description

Contrast (gain) correction setting

Variables

- int16_t x

  Coordinate X, this allows to set signed value.

- int16_t y

  Coordinate Y, this allows to set signed value.

9.11.8.4 display_brightness_t

display_brightness_t

Detailed description

Brightness (DC) correction setting

Variables

- bool enable

  Brightness Correction On/Off.

- uint16_t r

  Brightness (DC) adjustment for R channel.

- uint16_t g

  Brightness (DC) adjustment for G channel.

- uint16_t b

  Brightness (DC) adjustment for B channel.

9.11.8.5 display_contrast_t

display_contrast_t

Detailed description

Contrast (gain) correction setting

Variables

- bool enable

  Contrast Correction On/Off.

- uint8_t r

  Contrast (gain) adjustment for R channel.

- uint8_t g

  Contrast (gain) adjustment for G channel.

- uint8_t b

  Contrast (gain) adjustment for B channel.

9.11.8.6 display_correction_t

display_correction_t

Detailed description

Color correction setting

Variables

- display_brightness_t brightness

  Brightness.

- display_contrast_t contrast

  Contrast.

9.11.8.7 gamma_correction_t

gamma_correction_t

Detailed description

Gamma correction setting for each color

Variables

- bool enable

  Gamma Correction On/Off.

- uint16_t gain[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]

  Gain adjustment.

- uint16_t threshold[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]

  Start threshold.

9.11.8.8 display_gamma_correction_t

display_gamma_correction_t

Detailed description

Gamma correction setting

Variables

- gamma_correction_t r

  Gamma correction for R channel.

- gamma_correction_t g

  Gamma correction for G channel.

- gamma_correction_t b

  Gamma correction for B channel.

9.11.8.9 display_clut_t

display_clut_t

Detailed description

CLUT setting

Variables

- uint32_t color_num

  The number of colors in CLUT.

- const uint32_t * p_clut

  Address of the area storing the CLUT data (in ARGB8888 format)

9.11.8.10 display_input_cfg_t

display_input_cfg_t

Detailed description

Graphics plane input configuration structure

Variables

- uint32_t * p_base

  Base address to the frame buffer.

- uint16_t hsize

  Horizontal pixel size in a line.

- uint16_t vsize

  Vertical pixel size in a frame.

- uint32_t hstride

  Memory stride (bytes) in a line.

- display_in_format_t format

  Input format setting.

- bool line_descending_enable

  Line descending enable.

- bool lines_repeat_enable

  Line repeat enable.

- uint16_t lines_repeat_times

  Expected number of line repeating.

9.11.8.11 display_output_cfg_t

display_output_cfg_t

Detailed description

Display output configuration structure

Variables

- display_timing_t htiming

  Horizontal display cycle setting.
- display_timing_t vtiming

  Vertical display cycle setting.
- display_out_format_t format

  Output format setting.
- display_endian_t endian

  Bit order of output data.
- display_color_order_t color_order

  Color order in pixel.
- display_signal_polarity_t data_enable_polarity

  Data Enable signal polarity.
- display_sync_edge_t sync_edge

  Signal sync edge selection.
- display_color_t bg_color

  Background color.
- display_brightness_t brightness

  Brightness setting.
- display_contrast_t contrast

  Contrast setting.
- display_gamma_correction_t * p_gamma_correction

  Pointer to gamma correction setting.
- bool dithering_on

  Dithering on/off.

9.11.8.12 display_layer_t

display_layer_t

Detailed description

Graphics layer blend setup parameter structure

Variables

- display_coordinate_t coordinate

  Blending location (starting point of image)
- display_color_t bg_color

  Color outside region.
- display_fade_control_t fade_control

  Layer fade-in/out control on/off.
- uint8_t fade_speed

  Layer fade-in/out frame rate.

9.11.8.13 display_callback_args_t

display_callback_args_t

Detailed description

Display callback parameter definition

Variables

- display_event_t event

  Event code.
- void const * p_context

  Context provided to user during callback.

9.11.8.14 display_cfg_t

display_cfg_t

Detailed description

Display main configuration structure

Variables

- display_input_cfg_t input[DISPLAY_FRAME_LAYER_2+1]

  Graphics input frame setting.

  Generic configuration for display devices
- display_output_cfg_t output

  Graphics output frame setting.
- display_layer_t layer[DISPLAY_FRAME_LAYER_2+1]

  Graphics layer blend setting.
- uint8_t line_detect_ipl

  Line detect interrupt priority.

- uint8_t underflow_1_ipl

  Underflow 1 interrupt priority.
- uint8_t underflow_2_ipl

  Underflow 1 interrupt priority.
- void(* p_callback)(display_callback_args_t *p_args)

  Pointer to callback function.

  Configuration for display event processing
- void const * p_context

  User defined context passed into callback function.
- void const * p_extend

  Display hardware dependent configuration.

  Pointer to display peripheral specific configuration

9.11.8.15 display_runtime_cfg_t

display_runtime_cfg_t

Detailed description

Display main configuration structure

Variables

- display_input_cfg_t input

  Graphics input frame setting.

  Generic configuration for display devices
- display_layer_t layer

  Graphics layer alpha blending setting.

9.11.8.16 display_clut_cfg_t

display_clut_cfg_t

Detailed description

Display CLUT configuration structure

Variables

- uint32_t * p_base

  Pointer to CLUT source data.
- uint16_t start

  Beginning of CLUT entry to be updated.

- uint16_t size

  Size of CLUT entry to be updated.

9.11.8.17 display_status_t

display_status_t

Detailed description

Display Status

Variables

- display_state_t state

  Status of GLCD module.

- display_fade_status_t fade_status[DISPLAY_FRAME_LAYER_2+1]

  Status of fade-in/fade-out status.

9.11.8.18 display_api_t

display_api_t

Detailed description

Shared Interface definition for display peripheral

9.11.8.19 open

```
ssp_err_t(* display_api_t::open) (display_ctrl_t *const p_ctrl, display_cfg_t
const *const p_cfg)
```

Detailed description

Open display device. Implemented as

- R_GLCD_Open

Table 872:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to display interface control block. Must be declared by user. Value set here.
p_cfg	in	Pointer to display configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: display_ctrl_t*const p_ctrl

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

Parameter p_cfg

Definition: display_cfg_t const *const p_cfg

Display main configuration structure

- display_cfg_t::display_input_cfg_t

  Graphics input frame setting.

  Generic configuration for display devices

- display_cfg_t::display_output_cfg_t

  Graphics output frame setting.

- display_cfg_t::display_layer_t

  Graphics layer blend setting.

- display_cfg_t::line_detect_ipl

  Line detect interrupt priority.

- display_cfg_t::underflow_1_ipl

  Underflow 1 interrupt priority.

- display_cfg_t::underflow_2_ipl

  Underflow 1 interrupt priority.

- display_cfg_t::p_callback

  Pointer to callback function.

  Configuration for display event processing

- display_cfg_t::p_context

  User defined context passed into callback function.

- display_cfg_t::p_extend

  Display hardware dependent configuration.

  Pointer to display peripheral specific configuration

9.11.8.20 close

```
ssp_err_t(* display_api_t::close) (display_ctrl_t *const p_ctrl)
```

Detailed description

Close display device. Implemented as

- R_GLCD_Close

Table 873:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `display_ctrl_t*const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

9.11.8.21 start

```
ssp_err_t(* display_api_t::start) (display_ctrl_t *const p_ctrl)
```

Detailed description

Display start. Implemented as

- R_GLCD_Start

Table 874:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `display_ctrl_t*const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

9.11.8.22 stop

```
ssp_err_t(* display_api_t::stop) (display_ctrl_t *const p_ctrl)
```

Detailed description

Display stop. Implemented as

- R_GLCD_Stop

Table 875:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `display_ctrl_t*const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

9.11.8.23 layerChange

```
ssp_err_t(* display_api_t::layerChange) (display_ctrl_t const *const p_ctrl,
display_runtime_cfg_t const *const p_cfg, display_frame_layer_t frame)
```

Detailed description

Change layer parameters at runtime. Implemented as

- R_GLCD_LayerChange

Table 876:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
p_cfg	in	Pointer to run-time layer configuration structure.
frame	in	Number of graphic frames.

Parameter p_ctrl

Definition: `display_ctrl_tconst *const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

Parameter p_cfg

Definition: `display_runtime_cfg_t const *const p_cfg`

Display main configuration structure

- `display_runtime_cfg_t`::`display_input_cfg_t`

  Graphics input frame setting.

  Generic configuration for display devices

- display_runtime_cfg_t::display_layer_t

  Graphics layer alpha blending setting.

Parameter frame

9.11.8.24 correction

```
ssp_err_t(* display_api_t::correction) (display_ctrl_t const *const p_ctrl,
display_correction_t const *const p_param)
```

Detailed description

Color correction. Implemented as

- R_GLCD_ColorCorrection

Table 877:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
param	in	Pointer to color correction configuration structure.

Parameter p_ctrl

Definition: display_ctrl_tconst *const p_ctrl

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

Parameter param

9.11.8.25 clut

```
ssp_err_t(* display_api_t::clut) (display_ctrl_t const *const p_ctrl,
display_clut_cfg_t const *const p_clut_cfg, display_frame_layer_t frame)
```

Detailed description

Set CLUT for display device. Implemented as

- R_GLCD_ClutUpdate

Table 878:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Table 878:Parameters (Continued)

Name	Direction	Description
p_clut_cfg	in	Pointer to CLUT configuration structure.
frame	in	Number of frame buffer corresponding to the CLUT.

Parameter p_ctrl

Definition: `display_ctrl_tconst *const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

Parameter p_clut_cfg

Definition: `display_clut_cfg_t const *const p_clut_cfg`

Display CLUT configuration structure

- `display_clut_cfg_t`::`p_base`

  Pointer to CLUT source data.

- `display_clut_cfg_t`::`start`

  Beginning of CLUT entry to be updated.

- `display_clut_cfg_t`::`size`

  Size of CLUT entry to be updated.

Parameter frame

9.11.8.26 statusGet

```
ssp_err_t(* display_api_t::statusGet) (display_ctrl_t const *const p_ctrl,
display_status_t *const p_status)
```

Detailed description

Get status for display device. Implemented as

- R_GLCD_StatusGet

Table 879:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
status	in	Pointer to display interface status structure.

Parameter p_ctrl

Definition: `display_ctrl_tconst *const p_ctrl`

Display control block. Allocate an instance specific control block to pass into the display API calls. Implemented asglcd_instance_ctrl_tDisplay control block

Parameter status

9.11.8.27 versionGet

```
ssp_err_t(* display_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get version. Implemented as

- R_GLCD_VersionGet

Table 880:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory to store the version information.

Parameter p_version

9.11.8.28 display_instance_t

display_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- display_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- display_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- display_api_t const * p_api

  Pointer to the API structure for this instance.

9.12 DOC Interface

Interface for the Data Operation Circuit.

Defines the API and data structures for the DOC implementation of the Data Operation Circuit (DOC) interface.

9.12.1 Summary

This module implements the DOC_API using the Data Operation Circuit (DOC).

Implemented by: DOC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

DOC Interface description: Data Operation Circuit Driver

9.12.2 Interface API

doc_api_t

Function name	Description
.open	Initial configuration.
.close	Allow the driver to be reconfigured. Will reduce power consumption.
.statusGet	Get the DOC status and stores it in the provided pointer p_status.
.statusClear	Clear DOPCF status flag.
.write	Write to the DODIR and DODSR registers.
.inputRegisterWrite	Write to the DODIR register.
.versionGet	Get version and stores it in provided pointer p_version.

9.12.3 Data structures

- doc_callback_args_t
- doc_data_t
- doc_cfg_t
- doc_instance_t

9.12.4 Enumerations

- doc_event_t
- doc_status_t

9.12.5 Typedefs

- doc_size_t
- doc_ctrl_t

9.12.6 Defines

- #define DOC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DOC_API_VERSION_MINOR

  Initial value:`(2U)`

9.12.7 API Data

9.12.7.1 doc_event_t

`doc_event_t`

Detailed description

Event that can trigger a callback function.

Enumerated values

Name	Description
DOC_EVENT_COMPARISON_MISMATCH	Comparison of data has resulted in a mismatch.
DOC_EVENT_ADDITION	Addition of data has resulted in a value greater than H'FFFF.
DOC_EVENT_SUBTRACTION	Subtraction of data has resulted in a value less than H'0000.
DOC_EVENT_COMPARISON_MATCH	Comparison of data has resulted in a match.

9.12.7.2 doc_status_t

```
doc_status_t
```

Detailed description

Status of the data comparison operation.

Enumerated values

Name	Description
DOC_STATUS_CONDITION_FALSE	Data comparison condition NOT met (match or mismatch), addition result NOT > H'FFFF, subtraction result NOT < H'0000.
DOC_STATUS_CONDITION_TRUE	Data comparison condition met (match or mismatch), addition result > H'FFFF, subtraction result < H'0000.

9.12.7.3 doc_size_t

```
typedef uint16_t doc_size_t
```

Detailed description

Size of the comparison data supported by the Data Operation Circuit (DOC)

9.12.7.4 doc_ctrl_t

```
typedef void doc_ctrl_t
```

Detailed description

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented as

- doc_instance_ctrl_t

9.12.8 API Structures

9.12.8.1 doc_callback_args_t

doc_callback_args_t

Detailed description

Callback function parameter data.

Variables

- doc_event_t event

  The event is used to identify what caused the callback.

- void const * p_context

  Placeholder for user data. Set in doc_api_t::open function in doc_cfg_t.

9.12.8.2 doc_data_t

doc_data_t

Detailed description

Data to be written to DOC register for comparison/addition/subtraction.

Variables

- doc_size_t dodir

  Value to be written to the DOC DODIR.

- doc_size_t dodsr

  Value to be written to the DOC DODSR.

9.12.8.3 doc_cfg_t

doc_cfg_t

Detailed description

User configuration structure, used in the open function.

Variables

- doc_event_t event

  Select enumerated value from doc_event_t.

- uint8_t irq_ipl

  DOC interrupt priority.

- void(* p_callback)(doc_callback_args_t *p_args)

  Callback provided when a DOC ISR occurs.

- void const * p_context

  Placeholder for user data. Passed to the user callback in doc_callback_args_t.

9.12.8.4 doc_api_t

doc_api_t

Detailed description

Data Operation Circuit (DOC) API structure. DOC functions implemented at the HAL layer will follow this API.

9.12.8.5 open

```
ssp_err_t(* doc_api_t::open) (doc_ctrl_t *const p_ctrl, doc_cfg_t const *const
p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_DOC_Open

NOTE: Peripheral clocks should be configured prior to calling this function.

Table 881:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `doc_ctrl_t*const p_ctrl`

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

Parameter p_cfg

Definition: `doc_cfg_t const *const p_cfg`

User configuration structure, used in the open function.

- `doc_cfg_t`::`doc_event_t`

  Select enumerated value from doc_event_t.

  Enumerated as:

  – DOC_EVENT_COMPARISON_MISMATCH

  – DOC_EVENT_ADDITION

  – DOC_EVENT_SUBTRACTION

  – DOC_EVENT_COMPARISON_MATCH

- `doc_cfg_t`::`irq_ipl`

  DOC interrupt priority.

- `doc_cfg_t`::`p_callback`

  Callback provided when a DOC ISR occurs.

- doc_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in doc_callback_args_t.

9.12.8.6 close

```
ssp_err_t(* doc_api_t::close) (doc_ctrl_t *const p_ctrl)
```

Detailed description

Allow the driver to be reconfigured. Will reduce power consumption. Implemented as

- R_DOC_Close

Table 882:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.

Parameter p_ctrl

Definition: doc_ctrl_t*const p_ctrl

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

9.12.8.7 statusGet

```
ssp_err_t(* doc_api_t::statusGet) (doc_ctrl_t *const p_ctrl, doc_status_t
*p_status)
```

Detailed description

Get the DOC status and stores it in the provided pointer p_status. Implemented as

- R_DOC_StatusGet

NOTE: Call open to configure the DOC before using this function.

Table 883:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_status	out	Indicates the status of the comparison/addition/subtraction operation. Result will be one of doc_status_t.

Parameter p_ctrl

Definition: doc_ctrl_t*const p_ctrl

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

Parameter p_status

Definition: doc_status_t*p_status

Status of the data comparison operation.

9.12.8.8 statusClear

```
ssp_err_t(* doc_api_t::statusClear) (doc_ctrl_t *const p_ctrl)
```

Detailed description

Clear DOPCF status flag. Implemented as

- R_DOC_StatusClear

NOTE: Call open to configure the DOC before using this function.

Table 884:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.

Parameter p_ctrl

Definition: doc_ctrl_t*const p_ctrl

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

9.12.8.9 write

```
ssp_err_t(* doc_api_t::write) (doc_ctrl_t *const p_ctrl, doc_data_t *const p_data)
```

Detailed description

Write to the DODIR and DODSR registers. Implemented as

- R_DOC_Write

NOTE: Call open to configure the DOC before using this function.

Table 885:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.

Table 885:Parameters (Continued)

Name	Direction	Description
p_data	in	Pointer to data to be written to DOC DODIR and DODSR registers.

Parameter p_ctrl

Definition: `doc_ctrl_t*const p_ctrl`

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

Parameter p_data

Definition: `doc_data_t*const p_data`

Data to be written to DOC register for comparison/addition/subtraction.

- `doc_data_t::dodir`

  Value to be written to the DOC DODIR.

- `doc_data_t::dodsr`

  Value to be written to the DOC DODSR.

9.12.8.10 inputRegisterWrite

```
ssp_err_t(* doc_api_t::inputRegisterWrite) (doc_ctrl_t *const p_ctrl, doc_size_t data)
```

Detailed description

Write to the DODIR register. Implemented as

- R_DOC_InputRegisterWrite

NOTE: Call open to configure the DOC before using this function.

Table 886:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
data	in	Data to be written to DOC DODIR register.

Parameter p_ctrl

Definition: `doc_ctrl_t*const p_ctrl`

DOC control block. Allocate an instance specific control block to pass into the DOC API calls. Implemented asdoc_instance_ctrl_t

Parameter data

Definition: doc_size_tdata

Size of the comparison data supported by the Data Operation Circuit (DOC)

9.12.8.11 versionGet

```
ssp_err_t(* doc_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and stores it in provided pointer p_version. Implemented as

- R_DOC_VersionGet

Table 887:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.12.8.12 doc_instance_t

doc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- doc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- doc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- doc_api_t const * p_api

  Pointer to the API structure for this instance.

9.13 Events and peripheral definitions

Interface for the Event Link Controller.

Related SSP architecture topics:

- SSP Interfaces

- SSP Predefined Layers
- Using SSP Modules

Event Link Controller Interface description: ELC Driver

Related SSP architecture topics:

- What is an SSP Interface? SSP Interfaces
- What is a SSP Layer? SSP Predefined Layers
- How to use SSP Interfaces and Modules? Using SSP Modules

Event Link Controller Interface description: ELC Driver

9.13.1 Interface API

elc_api_t

Function name	Description
.init	Initialize all links in the Event Link Controller.
.softwareEventGenerate	Generate a software event in the Event Link Controller.
.linkSet	Create a single event link.
.linkBreak	Break an event link.
.enable	Enable the operation of the Event Link Controller.
.disable	Disable the operation of the Event Link Controller.
.versionGet	Get the driver version based on compile time macros.

9.13.2 Data structures

- elc_link_t
- elc_cfg_t
- elc_instance_t

9.13.3 Enumerations

- elc_software_event_t

9.13.4 Defines

- #define ELC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define ELC_API_VERSION_MINOR

  Initial value:`(2U)`

9.13.5 API Data

9.13.5.1 elc_software_event_t

`elc_software_event_t`

Detailed description

Software event number

Enumerated values

Name	Description
ELC_SOFTWARE_EVENT_0	Software event 0.
ELC_SOFTWARE_EVENT_1	Software event 1.

9.13.6 API Structures

9.13.6.1 elc_link_t

elc_link_t

Detailed description

Individual event link. The actual peripheral definitions can be found in the MCU specific (ie. /mcu/S124/bsp_elc.h) bsp_elc.h files.

Variables

- elc_peripheral_t peripheral

  Peripheral to receive the signal.
- elc_event_t event

  Signal that gets sent to the Peripheral.

9.13.6.2 elc_cfg_t

elc_cfg_t

Detailed description

Main configuration structure for the Event Link Controller

Variables

- bool autostart

  Start operation and enable interrupts during open().

- uint32_t link_count

  Number of event links.

- elc_link_t const * link_list

  Event links.

9.13.6.3 elc_api_t

elc_api_t

Detailed description

ELC driver structure. General ELC functions implemented at the HAL layer follow this API.

9.13.6.4 init

```
ssp_err_t(* elc_api_t::init) (elc_cfg_t const *const p_cfg)
```

Detailed description

Initialize all links in the Event Link Controller. Implemented as

- R_ELC_Init

Table 888:Parameters

Name	Direction	Description
p_cfg	in	Pointer to configuration structure.

Parameter p_cfg

Definition: `elc_cfg_t const *const p_cfg`

Main configuration structure for the Event Link Controller

- `elc_cfg_t`::`autostart`

  Start operation and enable interrupts during open().

- `elc_cfg_t`::`link_count`

  Number of event links.

- elc_cfg_t::elc_link_t

  Event links.

9.13.6.5 softwareEventGenerate

```
ssp_err_t(* elc_api_t::softwareEventGenerate) (elc_software_event_t event_num)
```

Detailed description

Generate a software event in the Event Link Controller. Implemented as

- R_ELC_SoftwareEventGenerate

Table 889:Parameters

Name	Direction	Description
eventNum	in	Software event number to be generated.

Parameter eventNum

9.13.6.6 linkSet

```
ssp_err_t(* elc_api_t::linkSet) (elc_peripheral_t peripheral, elc_event_t signal)
```

Detailed description

Create a single event link. Implemented as

- R_ELC_LinkSet

Table 890:Parameters

Name	Direction	Description
peripheral	in	The peripheral block that will receive the event signal.
signal	in	The event signal.

Parameter peripheral

Parameter signal

9.13.6.7 linkBreak

```
ssp_err_t(* elc_api_t::linkBreak) (elc_peripheral_t peripheral)
```

Detailed description

Break an event link. Implemented as

- R_ELC_LinkBreak

Table 891:Parameters

Name	Direction	Description
peripheral	in	The peripheral that should no longer be linked.

Parameter peripheral

9.13.6.8 enable

```
ssp_err_t(* elc_api_t::enable) (void)
```

Detailed description

Enable the operation of the Event Link Controller. Implemented as

- R_ELC_Enable

9.13.6.9 disable

```
ssp_err_t(* elc_api_t::disable) (void)
```

Detailed description

Disable the operation of the Event Link Controller. Implemented as

- R_ELC_Disable

9.13.6.10 versionGet

```
ssp_err_t(* elc_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get the driver version based on compile time macros. Implemented as

- R_ELC_VersionGet

Table 892:Parameters

Name	Direction	Description
p_version	out	is value returned.

Parameter p_version

9.13.6.11 elc_instance_t

elc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- elc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- elc_api_t const * p_api

  Pointer to the API structure for this instance.

9.14 External IRQ Interface

Interface for detecting external interrupts.

9.14.1 Summary

The external IRQ interface supports external inputs, for example input from pins or capacitive touch buttons. When an input trigger is detected, a user provided callback function will be called.

Implemented by: ICU

Related interfaces: Key Matrix Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

External IRQ Interface description: External IRQ Driver

9.14.2 Interface API

external_irq_api_t

Function name	Description
.open	Initial configuration.
.enable	Enable callback when external IRQ occurs.
.disable	Disable callback when external IRQ occurs.

Function name	Description
.triggerSet	Set trigger.
.filterEnable	Enables noise filter.
.filterDisable	Disable noise filter.
.close	Allow driver to be reconfigured. May reduce power consumption.
.versionGet	Get version and store it in provided pointer p_version.

9.14.3 Data structures

- external_irq_callback_args_t
- external_irq_cfg_t
- external_irq_instance_t

9.14.4 Enumerations

- external_irq_trigger_t
- external_irq_pclk_div_t

9.14.5 Typedefs

- external_irq_ctrl_t

9.14.6 Defines

- #define EXTERNAL_IRQ_API_VERSION_MAJOR

  Initial value:`(1U)`

  EXTERNAL IRQ API version number (Major)

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define EXTERNAL_IRQ_API_VERSION_MINOR

  Initial value:`(2U)`

  EXTERNAL IRQ API version number (Minor)

9.14.7 API Data

9.14.7.1 external_irq_trigger_t

`external_irq_trigger_t`

Detailed description

Trigger type: rising edge, falling edge, both edges, low level.

Enumerated values

Name	Description
EXTERNAL_IRQ_TRIG_FALLING	Falling edge trigger.
EXTERNAL_IRQ_TRIG_RISING	Rising edge trigger.
EXTERNAL_IRQ_TRIG_BOTH_EDGE	Both edges trigger.
EXTERNAL_IRQ_TRIG_LEVEL_LOW	Low level trigger.

9.14.7.2 external_irq_pclk_div_t

`external_irq_pclk_div_t`

Detailed description

External IRQ input pin digital filtering sample clock divisor settings.

Enumerated values

Name	Description
EXTERNAL_IRQ_PCLK_DIV_BY_1	Filter using PCLK divided by 1.
EXTERNAL_IRQ_PCLK_DIV_BY_8	Filter using PCLK divided by 8.
EXTERNAL_IRQ_PCLK_DIV_BY_32	Filter using PCLK divided by 32.
EXTERNAL_IRQ_PCLK_DIV_BY_64	Filter using PCLK divided by 64.

9.14.7.3 external_irq_ctrl_t

`typedef void external_irq_ctrl_t`

Detailed description

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented as

- icu_instance_ctrl_t

9.14.8 API Structures

9.14.8.1 external_irq_callback_args_t

external_irq_callback_args_t

Detailed description

Callback function parameter data

Variables

- void const * p_context

  Placeholder for user data. Set in external_irq_api_t::open function in external_irq_cfg_t.

- uint32_t channel

  The physical hardware channel that caused the interrupt.

9.14.8.2 external_irq_cfg_t

external_irq_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- uint8_t channel

  Hardware channel used.

- uint8_t irq_ipl

  Interrupt priority.

- external_irq_trigger_t trigger

  Trigger setting.

- external_irq_pclk_div_t pclk_div

  Digital filter clock divisor setting.

- bool autostart

  Start operation and enable interrupts during open().

- bool filter_enable

  Digital filter enable/disable setting.

- void(* p_callback)(external_irq_callback_args_t *p_args)

  Callback provided external input trigger occurs.

- void const * p_context

  Placeholder for user data. Passed to the user callback in external_irq_callback_args_t.

- void const * p_extend

  External IRQ hardware dependent configuration.

9.14.8.3 external_irq_api_t

external_irq_api_t

Detailed description

External interrupt driver structure. External interrupt functions implemented at the HAL layer will follow this API.

9.14.8.4 open

```
ssp_err_t(* external_irq_api_t::open) (external_irq_ctrl_t *const p_ctrl,
external_irq_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_ICU_ExternalIrqOpen

Table 893:Parameters

Name	Direction	Description
p_ctrl	out	Pointer to control block. Must be declared by user. Value set here.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

Parameter p_cfg

Definition: `external_irq_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `external_irq_cfg_t`::`channel`

  Hardware channel used.

- `external_irq_cfg_t`::`irq_ipl`

  Interrupt priority.

- external_irq_cfg_t::external_irq_trigger_t

  Trigger setting.

  Enumerated as:

  – EXTERNAL_IRQ_TRIG_FALLING
  – EXTERNAL_IRQ_TRIG_RISING
  – EXTERNAL_IRQ_TRIG_BOTH_EDGE
  – EXTERNAL_IRQ_TRIG_LEVEL_LOW

- external_irq_cfg_t::external_irq_pclk_div_t

  Digital filter clock divisor setting.

  Enumerated as:

  – EXTERNAL_IRQ_PCLK_DIV_BY_1
  – EXTERNAL_IRQ_PCLK_DIV_BY_8
  – EXTERNAL_IRQ_PCLK_DIV_BY_32
  – EXTERNAL_IRQ_PCLK_DIV_BY_64

- external_irq_cfg_t::autostart

  Start operation and enable interrupts during open().

- external_irq_cfg_t::filter_enable

  Digital filter enable/disable setting.

- external_irq_cfg_t::p_callback

  Callback provided external input trigger occurs.

- external_irq_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in external_irq_callback_args_t.

- external_irq_cfg_t::p_extend

  External IRQ hardware dependent configuration.

9.14.8.5 enable

```
ssp_err_t(* external_irq_api_t::enable) (external_irq_ctrl_t *const p_ctrl)
```

Detailed description

Enable callback when external IRQ occurs. Implemented as

- R_ICU_ExternalIrqEnable

Table 894:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

9.14.8.6 disable

```
ssp_err_t(* external_irq_api_t::disable) (external_irq_ctrl_t *const p_ctrl)
```

Detailed description

Disable callback when external IRQ occurs. Implemented as

- R_ICU_ExternalIrqDisable

Table 895:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

9.14.8.7 triggerSet

```
ssp_err_t(* external_irq_api_t::triggerSet) (external_irq_ctrl_t *const p_ctrl,
external_irq_trigger_t const trigger)
```

Detailed description

Set trigger. Implemented as

- R_ICU_ExternalIrqTriggerSet

Table 896:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.
trigger	in	Trigger type

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

Parameter trigger

9.14.8.8 filterEnable

```
ssp_err_t(* external_irq_api_t::filterEnable) (external_irq_ctrl_t *const p_ctrl)
```

Detailed description

Enables noise filter. Implemented as

- R_ICU_ExternalIrqFilterEnable

Table 897:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

9.14.8.9 filterDisable

```
ssp_err_t(* external_irq_api_t::filterDisable) (external_irq_ctrl_t *const
p_ctrl)
```

Detailed description

Disable noise filter. Implemented as

- R_ICU_ExternalIrqFilterDisable

Table 898:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

9.14.8.10 close

```
ssp_err_t(* external_irq_api_t::close) (external_irq_ctrl_t *const p_ctrl)
```

Detailed description

Allow driver to be reconfigured. May reduce power consumption. Implemented as

- R_ICU_ExternalIrqClose

Table 899:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in Open call for this external interrupt.

Parameter p_ctrl

Definition: `external_irq_ctrl_t*const p_ctrl`

External IRQ control block. Allocate an instance specific control block to pass into the external IRQ API calls. Implemented asicu_instance_ctrl_t

9.14.8.11 versionGet

```
ssp_err_t(* external_irq_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_ICU_ExternalIrqVersionGet

Table 900:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.14.8.12 external_irq_instance_t

external_irq_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- external_irq_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- external_irq_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- external_irq_api_t const * p_api

  Pointer to the API structure for this instance.

9.15 Flash Interface

Interface for the flash controller.

9.15.1 Summary

The Flash interface provides the functionality necessary to read, write, erase and blank check the Flash memory. Additionally functions are provided to configure the access window and swap areas of the flash memory.

Implemented by:

- High-performance Flash
- Low Power Flash

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Flash Interface description: Flash Driver

9.15.2 Interface API

flash_api_t

Function name	Description
.open	Open FLASH device.
.write	Write FLASH device.
.read	Read FLASH device.
.erase	Erase FLASH device.
.blankCheck	Blank check FLASH device.
.infoGet	Close FLASH device.
.close	Close FLASH device.
.statusGet	Get Status for FLASH device.
.accessWindowSet	Set Access Window for FLASH device.
.accessWindowClear	Clear any existing Code Flash access window for FLASH device.
.reset	Reset function for FLASH device.
.updateFlashClockFreq	Update Flash clock frequency (FCLK) and recalculate timeout values
.startupAreaSelect	Select which block - Default (Block 0) or Alternate (Block 1) is used as the start-up area block. swap_type | is_temporary | Operation FLASH_STARTUP_AREA_BLOCK0: false On next reset Startup area will be Block 0.FLASH_STARTUP_AREA_BLOCK0 | false | On next reset Startup area will be Block 0. Block 0.FLASH_STARTUP_AREA_BLOCK1: false On next reset Startup area will be Block 1.FLASH_STARTUP_AREA_BLOCK1 | true | Startup area is immediately, but temporarily switched to Block 1. Block 1.FLASH_STARTUP_AREA_BTFLG | true | Startup area is immediately, but temporarily switched to... taken.the Block determined by the Configuration BTFLG.

Function name	Description
.versionGet	Get Flash driver version.

9.15.3 Data structures

- flash_fmi_block_info_t
- flash_fmi_regions_t
- flash_info_t
- flash_callback_args_t
- flash_cfg_t
- flash_instance_t

9.15.4 Enumerations

- flash_result_t
- flash_startup_area_swap_t
- flash_event_t

9.15.5 Typedefs

- flash_ctrl_t

9.15.6 Defines

- #define FLASH_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
  FLASH HAL API version number (Major)

- #define FLASH_API_VERSION_MINOR

  Initial value:`(2U)`

  FLASH HAL API version number (Minor)

9.15.7 API Data

9.15.7.1 flash_result_t

`flash_result_t`

Detailed description

Result type for certain operations

Enumerated values

Name	Description
FLASH_RESULT_BLANK	Return status for Blank Check Function.
FLASH_RESULT_NOT_BLANK	Return status for Blank Check Function.
FLASH_RESULT_BGO_ACTIVE	Flash is configured for BGO mode. Result is returned in callback.

9.15.7.2 flash_startup_area_swap_t

`flash_startup_area_swap_t`

Detailed description

Parameter for specifying the startup area swap being requested by startupAreaSelect()

Enumerated values

Name	Description
FLASH_STARTUP_AREA_BLOCK1	Startup area will be set to Block 1.
FLASH_STARTUP_AREA_BLOCK0	Startup area will be set to Block 0.
FLASH_STARTUP_AREA_BTFLG	Startup area will be set based on the value of the BTFLG.

9.15.7.3 flash_event_t

`flash_event_t`

Detailed description

Event types returned by the ISR callback when used in Data Flash BGO mode

Enumerated values

Name	Description
FLASH_EVENT_ERASE_COMPLETE	Erase operation successfully completed.
FLASH_EVENT_WRITE_COMPLETE	Write operation successfully completed.

Name	Description
FLASH_EVENT_BLANK	Blank check operation successfully completed. Specified area is blank.
FLASH_EVENT_NOT_BLANK	Blank check operation successfully completed. Specified area is NOT blank.
FLASH_EVENT_ERR_DF_ACCESS	Data Flash operation failed. Can occur when writing an unerased section.
FLASH_EVENT_ERR_CF_ACCESS	Code Flash operation failed. Can occur when writing an unerased section.
FLASH_EVENT_ERR_CMD_LOCKED	Operation failed, FCU is in Locked state (often result of an illegal command)
FLASH_EVENT_ERR_FAILURE	Erase or Program Operation failed.

9.15.7.4 flash_ctrl_t

```
typedef void flash_ctrl_t
```

Detailed description

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented as

- flash_lp_instance_ctrl_t
- flash_hp_instance_ctrl_t

9.15.8 API Structures

9.15.8.1 flash_fmi_block_info_t

flash_fmi_block_info_t

Detailed description

Flash block details stored in factory flash.

Variables

- uint32_t block_section_st_addr

  starting address for this block section (blocks of this size)
- uint32_t block_section_end_addr

  ending address for this block section (blocks of this size)
- uint32_t block_size

  Flash erase block size.

- uint32_t block_size_write

  Flash write block size.

9.15.8.2 flash_fmi_regions_t

flash_fmi_regions_t

Detailed description

Flash block details

Variables

- uint32_t num_regions

  Length of block info array.

- flash_fmi_block_info_t const * p_block_array

  Block info array base address.

9.15.8.3 flash_info_t

flash_info_t

Detailed description

Information about the flash blocks

Variables

- flash_fmi_regions_t code_flash

  Information about the code flash regions.

- flash_fmi_regions_t data_flash

  Information about the code flash regions.

9.15.8.4 flash_callback_args_t

flash_callback_args_t

Detailed description

Callback function parameter data

Variables

- flash_event_t event

  Event can be used to identify what caused the callback (flash ready or error).

- void const * p_context

  Placeholder for user data. Set in flash_api_t::open function in::flash_cfg_t.

9.15.8.5 flash_cfg_t

flash_cfg_t

Detailed description

FLASH Configuration

Variables

- bool data_flash_bgo

  True if BGO (Background Operation) is enabled for Data Flash.

- void(* p_callback)(flash_callback_args_t *p_args)

  Callback provided when a Flash interrupt ISR occurs.

- void const * p_extend

  FLASH hardware dependent configuration.

- void const * p_context

  Placeholder for user data. Passed to user callback in flash_callback_args_t.

- uint8_t irq_ipl

  Flash ready interrupt priority.

- uint8_t err_irq_ipl

  Flash error interrupt priority (unused in r_flash_lp)

9.15.8.6 flash_api_t

flash_api_t

Detailed description

Shared Interface definition for FLASH

9.15.8.7 open

```
ssp_err_t(* flash_api_t::open) (flash_ctrl_t *const p_ctrl, flash_cfg_t const
*const p_cfg)
```

Detailed description

Open FLASH device. Implemented as

- R_FLASH_LP_Open
- R_FLASH_HP_Open

Table 901:Parameters

Name	Direction	Description
p_ctrl	out	Pointer to FLASH device control. Must be declared by user. Value set here.
flash_cfg_t	in	Pointer to FLASH configuration structure. All elements of this structure must be set by the user.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter flash_cfg_t

Definition: `flash_cfg_t const *const p_cfg`

FLASH Configuration

- `flash_cfg_t`::`data_flash_bgo`

  True if BGO (Background Operation) is enabled for Data Flash.

- `flash_cfg_t`::`p_callback`

  Callback provided when a Flash interrupt ISR occurs.

- `flash_cfg_t`::`p_extend`

  FLASH hardware dependent configuration.

- `flash_cfg_t`::`p_context`

  Placeholder for user data. Passed to user callback in flash_callback_args_t.

- `flash_cfg_t`::`irq_ipl`

  Flash ready interrupt priority.

- `flash_cfg_t`::`err_irq_ipl`

  Flash error interrupt priority (unused in r_flash_lp)

9.15.8.8 write

```
ssp_err_t(* flash_api_t::write) (flash_ctrl_t *const p_ctrl, uint32_t const
src_address, uint32_t const flash_address, uint32_t const num_bytes)
```

Detailed description

Write FLASH device. Implemented as

- R_FLASH_LP_Write

- R_FLASH_HP_Write

Table 902:Parameters

Name	Direction	Description
p_ctrl	in	Control for the FLASH device context.
src_address	in	Address of the buffer containing the data to write to Flash.
flash_address	in	Code Flash or Data Flash address to write. The address must be on a programming line boundary.
num_bytes	in	The number of bytes to write. This number must be a multiple of the programming size. For Code Flash this is FLASH_MIN_PGM_SIZE_CF. For Data Flash this is FLASH_MIN_PGM_SIZE_DF.

ATTENTION: Specifying a number that is not a multiple of the programming size will result in SF_FLASH_ERR_BYTES being returned and no data written.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter src_address

`uint32_t`

Parameter flash_address

`uint32_t`

Parameter num_bytes

`uint32_t`

9.15.8.9 read

```
ssp_err_t(* flash_api_t::read) (flash_ctrl_t *const p_ctrl, uint8_t *const
p_dest_address, uint32_t const flash_address, uint32_t const num_bytes)
```

Detailed description

Read FLASH device. Implemented as

- R_FLASH_LP_Read
- R_FLASH_HP_Read

Table 903:Parameters

Name	Direction	Description
p_ctrl	in	Control for the FLASH device context.
p_dest_address	in	Pointer to caller's destination buffer used to hold the data read from Flash.
flash_address	in	Code Flash or Data Flash starting address to read from.
num_bytes	in	The number of bytes to read.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter p_dest_address

`uint8_t`

Parameter flash_address

`uint32_t`

Parameter num_bytes

`uint32_t`

9.15.8.10 erase

```
ssp_err_t(* flash_api_t::erase) (flash_ctrl_t *const p_ctrl, uint32_t const
address, uint32_t const num_blocks)
```

Detailed description

Erase FLASH device. Implemented asR_FLASH_LP_EraseR_FLASH_HP_Erase

Table 904:Parameters

Name	Direction	Description
p_ctrl	in	Control for the FLASH device.
address	in	The block containing this address is the first block erased.

Table 904:Parameters (Continued)

Name	Direction	Description
num_blocks	in	Specifies the number of blocks to be erased, the starting block determined by the block_erase_address.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter address

`uint32_t`

Parameter num_blocks

`uint32_t`

9.15.8.11 blankCheck

```
ssp_err_t(* flash_api_t::blankCheck) (flash_ctrl_t *const p_ctrl, uint32_t const address, uint32_t const num_bytes, flash_result_t *const p_blank_check_result)
```

Detailed description

Blank check FLASH device. Implemented as

- R_FLASH_LP_BlankCheck
- R_FLASH_HP_BlankCheck

Table 905:Parameters

Name	Direction	Description
p_ctrl	in	Control for the FLASH device context.
address	in	The starting address of the Flash area to blank check.
num_bytes	in	Specifies the number of bytes that need to be checked. See the specific handler for details.

Table 905:Parameters (Continued)

Name	Direction	Description
p_blank_check_result	out	Pointer that will be populated by the API with the results of the blank check operation in non-BGO (blocking) mode. In this case the blank check operation completes here and the result is returned. In Data Flash BGO mode the blank check operation is only started here and the result obtained later when the supplied callback routine is called. In this case FLASH_RESULT_BGO_ACTIVE will be returned in p_blank_check_result.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter address

`uint32_t`

Parameter num_bytes

`uint32_t`

Parameter p_blank_check_result

Definition: `flash_result_t*const p_blank_check_result`

Result type for certain operations

9.15.8.12 infoGet

```
ssp_err_t(* flash_api_t::infoGet) (flash_ctrl_t *const p_ctrl, flash_info_t
*const p_info)
```

Detailed description

Close FLASH device. Implemented as

- R_FLASH_LP_InfoGet
- R_FLASH_HP_InfoGet

Table 906:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.

Table 906:Parameters (Continued)

Name	Direction	Description
p_info	out	Pointer to FLASH info structure.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter p_info

Definition: `flash_info_t*const p_info`

Information about the flash blocks

- `flash_info_t::code_flash`

  Information about the code flash regions.

- `flash_info_t::data_flash`

  Information about the code flash regions.

9.15.8.13 close

```
ssp_err_t(* flash_api_t::close) (flash_ctrl_t *const p_ctrl)
```

Detailed description

Close FLASH device. Implemented as

- R_FLASH_LP_Close
- R_FLASH_HP_Close

Table 907:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

9.15.8.14 statusGet

```
ssp_err_t(* flash_api_t::statusGet) (flash_ctrl_t *const p_ctrl)
```

Detailed description

Get Status for FLASH device. Implemented as

- R_FLASH_LP_StatusGet
- R_FLASH_HP_StatusGet

Table 908:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

9.15.8.15 accessWindowSet

```
ssp_err_t(* flash_api_t::accessWindowSet) (flash_ctrl_t *const p_ctrl, uint32_t
const start_addr, uint32_t const end_addr)
```

Detailed description

Set Access Window for FLASH device. Implemented as

- R_FLASH_LP_AccessWindowSet
- R_FLASH_HP_AccessWindowSet

Table 909:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.
start_addr	in	Determines the Starting block for the Code Flash access window.
end_addr	in	Determines the Ending block for the Code Flash access window.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter start_addr

`uint32_t`

Parameter end_addr

`uint32_t`

9.15.8.16 accessWindowClear

```
ssp_err_t(* flash_api_t::accessWindowClear) (flash_ctrl_t *const p_ctrl)
```

Detailed description

Clear any existing Code Flash access window for FLASH device. Implemented as

- R_FLASH_LP_AccessWindowClear
- R_FLASH_HP_AccessWindowClear

Table 910:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.
start_addr	in	Determines the Starting block for the Code Flash access window.
end_addr	in	Determines the Ending block for the Code Flash access window.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter start_addr

Parameter end_addr

9.15.8.17 reset

```
ssp_err_t(* flash_api_t::reset) (flash_ctrl_t *const p_ctrl)
```

Detailed description

Reset function for FLASH device. Implemented as

- R_FLASH_LP_Reset
- R_FLASH_HP_Reset

Table 911:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

9.15.8.18 updateFlashClockFreq

```
ssp_err_t(* flash_api_t::updateFlashClockFreq) (flash_ctrl_t *const p_ctrl)
```

Detailed description

Update Flash clock frequency (FCLK) and recalculate timeout values Implemented as

- R_FLASH_LP_UpdateFlashClockFreq
- R_FLASH_HP_UpdateFlashClockFreq

Table 912:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

9.15.8.19 startupAreaSelect

```
ssp_err_t(* flash_api_t::startupAreaSelect) (flash_ctrl_t *const p_ctrl,
flash_startup_area_swap_t swap_type, bool is_temporary)
```

Detailed description

Select which block - Default (Block 0) or Alternate (Block 1) is used as the start-up area block. Implemented as

- R_FLASH_LP_StartUpAreaSelect
- R_FLASH_HP_StartUpAreaSelect

Table 913:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to FLASH device control.
swap_type	in	FLASH_STARTUP_AREA_BLOCK0, FLASH_STARTUP_AREA_BLOCK1 or FLASH_STARTUP_AREA_BTFLG.
is_temporary	in	True or false. See table below.

swap_type | is_temporary | Operation FLASH_STARTUP_AREA_BLOCK0: false On next reset Startup area will be Block 0.

FLASH_STARTUP_AREA_BLOCK0 | false | On next reset Startup area will be Block 0. Block 0.

FLASH_STARTUP_AREA_BLOCK1: false On next reset Startup area will be Block 1.

FLASH_STARTUP_AREA_BLOCK1 | true | Startup area is immediately, but temporarily switched to Block 1. Block 1.

FLASH_STARTUP_AREA_BTFLG | true | Startup area is immediately, but temporarily switched to... taken.

the Block determined by the Configuration BTFLG.

Parameter p_ctrl

Definition: `flash_ctrl_t*const p_ctrl`

Flash control block. Allocate an instance specific control block to pass into the flash API calls. Implemented asflash_lp_instance_ctrl_tflash_hp_instance_ctrl_t

Parameter swap_type

Definition: `flash_startup_area_swap_tswap_type`

Parameter for specifying the startup area swap being requested by startupAreaSelect()

Parameter is_temporary

`const`

9.15.8.20 versionGet

```
ssp_err_t(* flash_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get Flash driver version. Implemented as

- R_FLASH_LP_VersionGet
- R_FLASH_HP_VersionGet

Table 914:Parameters

Name	Direction	Description
p_version	out	Returns version.

Parameter p_version

9.15.8.21 flash_instance_t

flash_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- flash_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- flash_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- flash_api_t const * p_api

  Pointer to the API structure for this instance.

9.16 FMI Interface

Interface for reading on-chip factory information.

9.16.1 Summary

The FMI (Factory MCU Information) module provides a function for reading the Product Information record.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

FMI Interface description: FMI Driver

9.16.2 Interface API

fmi_api_t

Function name	Description
.init	Initialize the FMI base pointer.
.productInfoGet	Get product information record address into caller's pointer.
.uniqueIdGet	Copy unique ID into p_unique_id.
.productFeatureGet	Get feature information and store it in p_info.
.eventInfoGet	Get event information and store it in p_info.
.versionGet	Get the driver version based on compile time macros.

9.16.3 Data structures

- fmi_header_t
- fmi_product_info_t
- fmi_unique_id_t
- fmi_feature_info_t
- fmi_event_info_t
- fmi_instance_t

9.16.4 Defines

- #define FMI_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define FMI_API_VERSION_MINOR

  Initial value:`(3U)`

9.16.5 API Structures

9.16.5.1 fmi_header_t

fmi_header_t

Detailed description

Variables

- uint32_t contents
- uint32_t variant
- uint32_t count
- uint32_t minor
- uint32_t major

9.16.5.2 fmi_product_info_t

fmi_product_info_t

Detailed description

Variables

- fmi_header_t header
- uint8_t unique_id[16]

  DEPRECATED, use uniqueIdGet instead.
- uint8_t product_name[16]
- uint8_t product_marking[16]
- uint32_t mask_revision
- uint32_t pin_count
- uint32_t pkg_type
- uint32_t temp_range
- uint32_t quality_code
- uint32_t reserved
- struct{} struct{}

  See source code for definition of this member.
- uint32_t max_freq
- uint32_t reserved1
- struct{} struct{}

  See source code for definition of this member.

9.16.5.3 fmi_unique_id_t

fmi_unique_id_t

Detailed description

Variables

- uint32_t unique_id[4]

9.16.5.4 fmi_feature_info_t

fmi_feature_info_t

Detailed description

Variables

- void * ptr
- uint32_t channel_count
- uint32_t variant_data
- uint32_t extended_data_count
- uint32_t version_major
- uint32_t version_minor
- struct{} struct{}

  See source code for definition of this member.
- void * ptr_extended_data

9.16.5.5 fmi_event_info_t

fmi_event_info_t

Detailed description

Variables

- IRQn_Type irq
- elc_event_t event

9.16.5.6 fmi_api_t

fmi_api_t

Detailed description

fmi driver structure. General fmi functions implemented at the HAL layer will follow this API.

9.16.5.7 init

```
(* fmi_api_t::init) (void)
```

Detailed description

Initialize the FMI base pointer. Implemented as

- R_FMI_Init()

9.16.5.8 productInfoGet

```
(* fmi_api_t::productInfoGet) (fmi_product_info_t **pp_product_info)
```

Detailed description

Get product information record address into caller's pointer.

ATTENTION: unique_id is deprecated and will not contain a unique ID if the factory MCU information is linked in by the application code. Use uniqueIdGet for the unique ID.

Table 915:Parameters

Name	Direction	Description
pp_product_info	inout	Pointer to store pointer to product info.

Implemented as

- R_FMI_ProductInfoGet()

Parameter pp_product_info

Definition: `fmi_product_info_t**pp_product_info`

- `fmi_product_info_t::header`
- `fmi_product_info_t::unique_id`

  DEPRECATED, use uniqueIdGet instead.
- `fmi_product_info_t::product_name`
- `fmi_product_info_t::product_marking`
- `fmi_product_info_t::mask_revision`
- `fmi_product_info_t::pin_count`
- `fmi_product_info_t::pkg_type`

- fmi_product_info_t::temp_range
- fmi_product_info_t::quality_code
- fmi_product_info_t::reserved
- fmi_product_info_t::struct{}
- fmi_product_info_t::max_freq
- fmi_product_info_t::reserved1
- fmi_product_info_t::struct{}

9.16.5.9 uniqueIdGet

```
(* fmi_api_t::uniqueIdGet) (fmi_unique_id_t *p_unique_id)
```

Detailed description

Copy unique ID into p_unique_id.

Table 916:Parameters

Name	Direction	Description
p_unique_id	out	Pointer to unique ID.

Implemented as

- R_FMI_UniqueIdGet()

Parameter p_unique_id

Definition: fmi_unique_id_t*p_unique_id

- fmi_unique_id_t::unique_id

9.16.5.10 productFeatureGet

```
(* fmi_api_t::productFeatureGet) ( const *const p_feature, fmi_feature_info_t
*const p_info)
```

Detailed description

Get feature information and store it in p_info.

Table 917:Parameters

Name	Direction	Description
p_feature	in	Definition of SSP feature.
p_info	out	Feature specific information.

Implemented as

- R_FMI_ProductFeatureGet()

Parameter p_feature

Parameter p_info

Definition: `fmi_feature_info_t*const p_info`

- `fmi_feature_info_t::ptr`
- `fmi_feature_info_t::channel_count`
- `fmi_feature_info_t::variant_data`
- `fmi_feature_info_t::extended_data_count`
- `fmi_feature_info_t::version_major`
- `fmi_feature_info_t::version_minor`
- `fmi_feature_info_t::struct{}`
- `fmi_feature_info_t::ptr_extended_data`

9.16.5.11 eventInfoGet

```
(* fmi_api_t::eventInfoGet) ( const *const p_feature, signal, fmi_event_info_t
*const p_info)
```

Detailed description

Get event information and store it in p_info.

Table 918:Parameters

Name	Direction	Description
p_feature	in	Definition of SSP feature.
signal	in	Feature signal ID.
p_info	out	Event information for feature signal.

Implemented as

- R_FMI_EventInfoGet()

Parameter p_feature

Parameter signal

Parameter p_info

Definition: `fmi_event_info_t*const p_info`

- `fmi_event_info_t::irq`

- `fmi_event_info_t::event`

9.16.5.12 versionGet

```
(* fmi_api_t::versionGet) ( *const p_version)
```

Detailed description

Get the driver version based on compile time macros. Implemented as

- R_FMI_VersionGet()

9.16.5.13 fmi_instance_t

fmi_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- fmi_api_t const * p_api

  Pointer to the API structure for this instance.

9.17 I2C Interface

Interface for I2C communication.

9.17.1 Summary

The I2C master interface provides a common API for I2C HAL drivers. The I2C master interface supports:

- Interrupt driven transmit/receive processing
- Callback function support which can return an event code

Implemented by:

- Simple I2C on SCI
- IIC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

I2C Interface description: I2C SCI Driver

9.17.2 Interface API

i2c_api_master_t

Function name	Description
.open	Opens the I2C driver and initializes the hardware.
.close	Closes the driver and releases the I2C device.
.read	Performs a read operation on an I2C device.
.write	Performs a write operation on an I2C device.
.reset	Performs a reset of the peripheral.
.slaveAddressSet	Sets address of the slave device without reconfiguring the bus.
.versionGet	Gets version information and stores it in the provided version struct.

i2c_api_slave_t

Function name	Description
.open	Opens the I2C driver and initializes the hardware.
.close	Closes the driver and releases the I2C device.
.masterWriteSlaveRead	Performs a read operation on an I2C device.
.masterReadSlaveWrite	Performs a write operation on an I2C device.
.versionGet	Gets version information and stores it in the provided version struct.

9.17.3 Data structures

- i2c_callback_args_t
- i2c_cfg_t
- i2c_api_master_t
- i2c_api_slave_t
- i2c_master_instance_t
- i2c_slave_instance_t

9.17.4 Enumerations

- i2c_rate_t
- i2c_addr_mode_t
- i2c_event_t

9.17.5 Typedefs

- i2c_ctrl_t

9.17.6 Defines

- #define I2C_MASTER_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define I2C_MASTER_API_VERSION_MINOR

  Initial value:`(4U)`

9.17.7 API Data

9.17.7.1 i2c_rate_t

`i2c_rate_t`

Detailed description

Communication speed options

Enumerated values

Name	Description
I2C_RATE_STANDARD	100 kHz
I2C_RATE_FAST	400 kHz
I2C_RATE_FASTPLUS	1 MHz

9.17.7.2 i2c_addr_mode_t

`i2c_addr_mode_t`

Detailed description

Addressing mode options

Enumerated values

Name	Description
I2C_ADDR_MODE_7BIT	Use 7-bit addressing mode.
I2C_ADDR_MODE_10BIT	Use 10-bit addressing mode.

9.17.7.3 i2c_event_t

`i2c_event_t`

Detailed description

Callback events

Enumerated values

Name	Description
I2C_EVENT_ABORTED	A transfer was aborted.
I2C_EVENT_RX_COMPLETE	A receive operation was completed successfully.
I2C_EVENT_TX_COMPLETE	A transmit operation was completed successfully.

9.17.7.4 i2c_ctrl_t

```
typedef void i2c_ctrl_t
```

Detailed description

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented as

- sci_i2c_instance_ctrl_t
- riic_instance_ctrl_t

9.17.8 API Structures

9.17.8.1 i2c_callback_args_t

i2c_callback_args_t

Detailed description

I2C callback parameter definition

Variables

- void const *const p_context

  Pointer to user-provided context.

- uint32_t const bytes

  Number of received/transmitted bytes in buff.

- i2c_event_t const event

  Event code.

9.17.8.2 i2c_cfg_t

i2c_cfg_t

Detailed description

I2C configuration block

Variables

- uint8_t channel

  Identifier recognizable by implementation.

  Generic configuration
- i2c_rate_t rate

  Device's maximum clock rate from enum i2c_rate_t.
- uint16_t slave

  The address of the slave device.
- i2c_addr_mode_t addr_mode

  Indicates how slave fields should be interpreted.
- uint16_t sda_delay

  The SDA output delay.
- uint8_t rxi_ipl

  Receive interrupt priority.
- uint8_t txi_ipl

  Transmit interrupt priority.
- uint8_t tei_ipl

  Transmit end interrupt priority.
- uint8_t eri_ipl

  Error interrupt priority.
- transfer_instance_t const * p_transfer_tx

  DTC instance for I2C transmit.Set to NULL if unused.

  DTC/DMA support
- transfer_instance_t const * p_transfer_rx

  DTC instance for I2C receive. Set to NULL if unused.
- void(* p_callback)(i2c_callback_args_t *p_args)

  Pointer to callback function.

  Parameters to control software behavior
- void const * p_context

  Pointer to the user-provided context.
- void const * p_extend

  Any configuration data needed by the hardware.

  Implementation-specific configuration

9.17.8.3 i2c_api_master_t

i2c_api_master_t

Detailed description

Interface definition for I2C access as master

9.17.8.4 open

```
ssp_err_t(* i2c_api_master_t::open) (i2c_ctrl_t *const p_ctrl, i2c_cfg_t const
*const p_cfg)
```

Detailed description

Opens the I2C driver and initializes the hardware. Implemented as

- R_RIIC_MasterOpen
- R_SCI_SIIC_MasterOpen

Table 919:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements are set here.
p_cfg	in	Pointer to configuration structure.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_cfg

Definition: `i2c_cfg_t const *const p_cfg`

I2C configuration block

- `i2c_cfg_t`::`channel`

  Identifier recognizable by implementation.

  Generic configuration

- `i2c_cfg_t`::`i2c_rate_t`

  Device's maximum clock rate from enum i2c_rate_t.

  Enumerated as:

  – I2C_RATE_STANDARD

  – I2C_RATE_FAST

  - I2C_RATE_FASTPLUS
- i2c_cfg_t::slave

  The address of the slave device.
- i2c_cfg_t::i2c_addr_mode_t

  Indicates how slave fields should be interpreted.

  Enumerated as:

  - I2C_ADDR_MODE_7BIT
  - I2C_ADDR_MODE_10BIT
- i2c_cfg_t::sda_delay

  The SDA output delay.
- i2c_cfg_t::rxi_ipl

  Receive interrupt priority.
- i2c_cfg_t::txi_ipl

  Transmit interrupt priority.
- i2c_cfg_t::tei_ipl

  Transmit end interrupt priority.
- i2c_cfg_t::eri_ipl

  Error interrupt priority.
- i2c_cfg_t::transfer_instance_t

  DTC instance for I2C transmit.Set to NULL if unused.

  DTC/DMA support
- i2c_cfg_t::transfer_instance_t

  DTC instance for I2C receive. Set to NULL if unused.
- i2c_cfg_t::p_callback

  Pointer to callback function.

  Parameters to control software behavior
- i2c_cfg_t::p_context

  Pointer to the user-provided context.
- i2c_cfg_t::p_extend

  Any configuration data needed by the hardware.

  Implementation-specific configuration

9.17.8.5 close

```
ssp_err_t(* i2c_api_master_t::close) (i2c_ctrl_t *const p_ctrl)
```

Detailed description

Closes the driver and releases the I2C device. Implemented as

- R_RIIC_MasterClose
- R_SCI_SIIC_MasterClose

Table 920:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

9.17.8.6 read

```
ssp_err_t(* i2c_api_master_t::read) (i2c_ctrl_t *const p_ctrl, uint8_t *const
p_dest, uint32_t const bytes, bool const restart)
```

Detailed description

Performs a read operation on an I2C device. Implemented as

- R_RIIC_MasterRead
- R_SCI_SIIC_MasterRead

Table 921:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.
p_dest	in	Pointer to the location to store read data.
bytes	in	Number of bytes to read.
restart	in	Specify if the restart condition should be issued after reading.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint32_t`

Parameter restart

`const`

9.17.8.7 write

```
ssp_err_t(* i2c_api_master_t::write) (i2c_ctrl_t *const p_ctrl, uint8_t *const
p_src, uint32_t const bytes, bool const restart)
```

Detailed description

Performs a write operation on an I2C device. Implemented as

- R_RIIC_MasterWrite
- R_SCI_SIIC_MasterWrite

Table 922:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.
p_src	in	Pointer to the location to get write data from.
bytes	in	Number of bytes to write.
restart	in	Specify if the restart condition should be issued after writing.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter bytes

`uint32_t`

Parameter restart

const

9.17.8.8 reset

```
ssp_err_t(* i2c_api_master_t::reset) (i2c_ctrl_t *const p_ctrl)
```

Detailed description

Performs a reset of the peripheral. Implemented as

- R_RIIC_MasterReset
- R_SCI_SIIC_MasterReset

Table 923:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

9.17.8.9 slaveAddressSet

```
ssp_err_t(* i2c_api_master_t::slaveAddressSet) (i2c_ctrl_t *const p_ctrl,
uint16_t const slave, i2c_addr_mode_t const addr_mode)
```

Detailed description

Sets address of the slave device without reconfiguring the bus. Implemented as

- R_RIIC_MasterSlaveAddressSet
- R_SCI_SIIC_MasterSlaveAddressSet

Table 924:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.
slave_address	in	Address of the slave device.
address_mode	in	Addressing mode.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter slave_address

Parameter address_mode

9.17.8.10 versionGet

```
ssp_err_t(* i2c_api_master_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version information and stores it in the provided version struct. Implemented as

- R_RIIC_MasterVersionGet
- R_SCI_SIIC_MasterVersionGet

Table 925:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.17.8.11 i2c_api_slave_t

i2c_api_slave_t

Detailed description

Interface definition for I2C access as slave

9.17.8.12 open

```
ssp_err_t(* i2c_api_slave_t::open) (i2c_ctrl_t *const p_ctrl, i2c_cfg_t const
*const p_cfg)
```

Detailed description

Opens the I2C driver and initializes the hardware. Implemented as

- R_RIIC_SlaveOpen

Table 926:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements are set here.

Table 926:Parameters (Continued)

Name	Direction	Description
p_cfg	in	Pointer to configuration structure.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_cfg

Definition: `i2c_cfg_t const *const p_cfg`

I2C configuration block

- `i2c_cfg_t`::`channel`

  Identifier recognizable by implementation.

  Generic configuration

- `i2c_cfg_t`::`i2c_rate_t`

  Device's maximum clock rate from enum i2c_rate_t.

  Enumerated as:

  – I2C_RATE_STANDARD

  – I2C_RATE_FAST

  – I2C_RATE_FASTPLUS

- `i2c_cfg_t`::`slave`

  The address of the slave device.

- `i2c_cfg_t`::`i2c_addr_mode_t`

  Indicates how slave fields should be interpreted.

  Enumerated as:

  – I2C_ADDR_MODE_7BIT

  – I2C_ADDR_MODE_10BIT

- `i2c_cfg_t`::`sda_delay`

  The SDA output delay.

- `i2c_cfg_t`::`rxi_ipl`

  Receive interrupt priority.

- `i2c_cfg_t`::`txi_ipl`

  Transmit interrupt priority.

- i2c_cfg_t::tei_ipl

  Transmit end interrupt priority.
- i2c_cfg_t::eri_ipl

  Error interrupt priority.
- i2c_cfg_t::transfer_instance_t

  DTC instance for I2C transmit.Set to NULL if unused.

  DTC/DMA support
- i2c_cfg_t::transfer_instance_t

  DTC instance for I2C receive. Set to NULL if unused.
- i2c_cfg_t::p_callback

  Pointer to callback function.

  Parameters to control software behavior
- i2c_cfg_t::p_context

  Pointer to the user-provided context.
- i2c_cfg_t::p_extend

  Any configuration data needed by the hardware.

  Implementation-specific configuration

9.17.8.13 close

```
ssp_err_t(* i2c_api_slave_t::close) (i2c_ctrl_t *const p_ctrl)
```

Detailed description

Closes the driver and releases the I2C device. Implemented as

- R_RIIC_SlaveClose

Table 927:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.

Parameter p_ctrl

Definition: i2c_ctrl_t*const p_ctrl

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

9.17.8.14 masterWriteSlaveRead

```
ssp_err_t(* i2c_api_slave_t::masterWriteSlaveRead) (i2c_ctrl_t *const p_ctrl,
uint8_t *const p_dest, uint32_t const bytes)
```

Detailed description

Performs a read operation on an I2C device. Implemented as

- R_RIIC_MasterWriteSlaveRead

Table 928:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.
p_dest	in	Pointer to the location to store read data.
bytes	in	Number of bytes to read.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint32_t`

9.17.8.15 masterReadSlaveWrite

```
ssp_err_t(* i2c_api_slave_t::masterReadSlaveWrite) (i2c_ctrl_t *const p_ctrl,
uint8_t *const p_src, uint32_t const bytes)
```

Detailed description

Performs a write operation on an I2C device. Implemented as

- R_RIIC_MasterReadSlaveWrite

Table 929:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block set in open call.

Table 929:Parameters (Continued)

Name	Direction	Description
p_src	in	Pointer to the location to get write data from.
bytes	in	Number of bytes to write.

Parameter p_ctrl

Definition: `i2c_ctrl_t*const p_ctrl`

I2C control block. Allocate an instance specific control block to pass into the I2C API calls. Implemented assci_i2c_instance_ctrl_triic_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter bytes

`uint32_t`

9.17.8.16 versionGet

```
ssp_err_t(* i2c_api_slave_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version information and stores it in the provided version struct. Implemented as

- R_RIIC_SlaveVersionGet

Table 930:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.17.8.17 i2c_master_instance_t

i2c_master_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- i2c_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- i2c_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- i2c_api_master_t const * p_api

  Pointer to the API structure for this instance.

9.17.8.18 i2c_slave_instance_t

i2c_slave_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- i2c_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- i2c_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- i2c_api_slave_t const * p_api

  Pointer to the API structure for this instance.

9.18 I2S Interface

The I2S (Inter-IC Sound) interface provides APIs and definitions for I2S audio communication.

9.18.1 Summary

The I2S (Inter-IC Sound) interface provides APIs and definitions for I2S audio communication.

9.18.2 Known Implementations

SSI

9.18.3 Related modules

See also: I2S Audio Playback Framework

9.18.4 Interface API

i2s_api_t

Function name	Description
.open	Initial configuration.
.stop	Stop communication. Transmission is stopped when callback is called with I2S_EVENT_IDLE. Reception is stopped when callback is called with I2S_EVENT_RX_EMPTY.
.mute	Enable or disable mute.
.write	Write I2S data. All transmit data is queued when callback is called with I2S_EVENT_TX_EMPTY. Transmission is complete when callback is called with I2S_EVENT_IDLE.
.read	Read I2S data. Reception is complete when callback is called with I2S_EVENT_RX_EMPTY.
.writeRead	Simultaneously write and read I2S data. Transmission and reception are complete when callback is called with I2S_EVENT_IDLE.
.infoGet	Get instance specific information and store it in provided pointer p_info.
.close	Allows driver to be reconfigured and may reduce power consumption.
.versionGet	Get version and store it in provided pointer p_version.

9.18.5 Data structures

- i2s_callback_args_t
- i2s_info_t
- i2s_cfg_t
- i2s_instance_t

9.18.6 Enumerations

- i2s_pcm_width_t
- i2s_word_length_t
- i2s_event_t
- i2s_dir_t

- i2s_mute_t
- i2s_ws_continue_t
- i2s_status_t

9.18.7 Typedefs

- i2s_ctrl_t

9.18.8 Defines

- #define I2S_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define I2S_API_VERSION_MINOR

  Initial value:`(2U)`

9.18.9 API Data

9.18.9.1 i2s_pcm_width_t

`i2s_pcm_width_t`

Detailed description

Audio PCM width

Enumerated values

Name	Description
I2S_PCM_WIDTH_8_BITS	Using 8-bit PCM.
I2S_PCM_WIDTH_16_BITS	Using 16-bit PCM.
I2S_PCM_WIDTH_18_BITS	Using 18-bit PCM.
I2S_PCM_WIDTH_20_BITS	Using 20-bit PCM.
I2S_PCM_WIDTH_22_BITS	Using 22-bit PCM.
I2S_PCM_WIDTH_24_BITS	Using 24-bit PCM.

9.18.9.2 i2s_word_length_t

`i2s_word_length_t`

Detailed description

Audio system word length.

Enumerated values

Name	Description
I2S_WORD_LENGTH_8_BITS	Using 8-bit system word length.
I2S_WORD_LENGTH_16_BITS	Using 16-bit system word length.
I2S_WORD_LENGTH_24_BITS	Using 24-bit system word length.
I2S_WORD_LENGTH_32_BITS	Using 32-bit system word length.

9.18.9.3 i2s_event_t

`i2s_event_t`

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
I2S_EVENT_IDLE	Communication is idle.
I2S_EVENT_TX_EMPTY	Transmit buffer is below FIFO trigger level.
I2S_EVENT_RX_FULL	Receive buffer is above FIFO trigger level.

9.18.9.4 i2s_dir_t

`i2s_dir_t`

Detailed description

I2S communication direction

Enumerated values

Name	Description
I2S_DIR_TX	Transmit direction only.
I2S_DIR_RX	Receive direction only.
I2S_DIR_TX_RX	Transmit and receive direction.

9.18.9.5 i2s_mute_t

`i2s_mute_t`

Detailed description

Mute audio samples.

Enumerated values

Name	Description
I2S_MUTE_ON	Enable mute.
I2S_MUTE_OFF	Disable mute.

9.18.9.6 i2s_ws_continue_t

`i2s_ws_continue_t`

Detailed description

Whether to continue WS (word select line) transmission during idle state.

Enumerated values

Name	Description
I2S_WS_CONTINUE_ON	Enable WS continue mode.
I2S_WS_CONTINUE_OFF	Disable WS continue mode.

9.18.9.7 i2s_status_t

`i2s_status_t`

Detailed description

Possible status values returned by infoGet.

Enumerated values

Name	Description
I2S_STATUS_IN_USE	I2S is in use.
I2S_STATUS_STOPPED	I2S is stopped.

9.18.9.8 i2s_ctrl_t

```
typedef void i2s_ctrl_t
```

Detailed description

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented as

- ssi_instance_ctrl_t

9.18.10 API Structures

9.18.10.1 i2s_callback_args_t

i2s_callback_args_t

Detailed description

Callback function parameter data

Variables

- void const * p_context

  Placeholder for user data. Set in i2s_api_t::open function in i2s_cfg_t.

- i2s_event_t event

  The event can be used to identify what caused the callback (overflow or error).

9.18.10.2 i2s_info_t

i2s_info_t

Detailed description

Timer information structure to store various information for an I2S instance.

Variables

- i2s_status_t status
- uint32_t sampling_freq_hz

  Sampling frequency in Hertz.

9.18.10.3 i2s_cfg_t

i2s_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- uint8_t channel

  Select a channel corresponding to the channel number of the hardware.

- i2s_pcm_width_t pcm_width

  Audio PCM data width.

- i2s_word_length_t word_length

  Audio word length, bits must be >= i2s_cfg_t::pcm_width bits.

- i2s_ws_continue_t ws_continue

  Whether to continue WS transmission during idle state.

- uint32_t sampling_freq_hz

  Sampling frequency in Hertz.

- uint32_t audio_clk_freq_hz

  Audio clock frequency in Hertz. Must be a multiple between 1 and 128 of (16 * i2s_cfg_t::sampling_freq_hz * (i2s_cfg_t::word_length <enum_value> + 1)

- timer_instance_t const * p_timer

  To generate audio clock with GPT, link a timer instance here. Set to NULL if unused.

- transfer_instance_t const * p_transfer_tx

  To use DTC during write, link a DTC instance here. Set to NULL if unused.

- transfer_instance_t const * p_transfer_rx

  To use DTC during read, link a DTC instance here. Set to NULL if unused.

- void(* p_callback)(i2s_callback_args_t *p_args)

  Callback provided when an I2S ISR occurs. Set to NULL for no CPU interrupt.

- void const * p_context

  Placeholder for user data. Passed to the user callback in i2s_callback_args_t.

- void const * p_extend

  Extension parameter for hardware specific settings.

- uint8_t rxi_ipl

  Receive interrupt priority.

- uint8_t txi_ipl

  Transmit interrupt priority.

- uint8_t idle_err_ipl

  Idle/Error interrupt priority.

9.18.10.4 i2s_api_t

i2s_api_t

Detailed description

I2S functions implemented at the HAL layer will follow this API.

9.18.10.5 open

```
ssp_err_t(* i2s_api_t::open) (i2s_ctrl_t *const p_ctrl, i2s_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_SSI_Open

NOTE: Peripheral clocks and any required output pins should be configured prior to calling this function.

NOTE: To reconfigure after calling this function, call close first.

Table 931:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter p_cfg

Definition: `i2s_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `i2s_cfg_t`::`channel`

  Select a channel corresponding to the channel number of the hardware.

- i2s_cfg_t::i2s_pcm_width_t

  Audio PCM data width.

  Enumerated as:

  – I2S_PCM_WIDTH_8_BITS

  – I2S_PCM_WIDTH_16_BITS

  – I2S_PCM_WIDTH_18_BITS

  – I2S_PCM_WIDTH_20_BITS

  – I2S_PCM_WIDTH_22_BITS

  – I2S_PCM_WIDTH_24_BITS

- i2s_cfg_t::i2s_word_length_t

  Audio word length, bits must be >= i2s_cfg_t::pcm_width bits.

  Enumerated as:

  – I2S_WORD_LENGTH_8_BITS

  – I2S_WORD_LENGTH_16_BITS

  – I2S_WORD_LENGTH_24_BITS

  – I2S_WORD_LENGTH_32_BITS

- i2s_cfg_t::i2s_ws_continue_t

  Whether to continue WS transmission during idle state.

  Enumerated as:

  – I2S_WS_CONTINUE_ON

  – I2S_WS_CONTINUE_OFF

- i2s_cfg_t::sampling_freq_hz

  Sampling frequency in Hertz.

- i2s_cfg_t::audio_clk_freq_hz

  Audio clock frequency in Hertz. Must be a multiple between 1 and 128 of (16 * i2s_cfg_t::sampling_freq_hz * (i2s_cfg_t::word_length <enum_value> + 1)

- i2s_cfg_t::timer_instance_t

  To generate audio clock with GPT, link a timer instance here. Set to NULL if unused.

- i2s_cfg_t::transfer_instance_t

  To use DTC during write, link a DTC instance here. Set to NULL if unused.

- i2s_cfg_t::transfer_instance_t

  To use DTC during read, link a DTC instance here. Set to NULL if unused.

- i2s_cfg_t::p_callback

  Callback provided when an I2S ISR occurs. Set to NULL for no CPU interrupt.
- i2s_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in i2s_callback_args_t.
- i2s_cfg_t::p_extend

  Extension parameter for hardware specific settings.
- i2s_cfg_t::rxi_ipl

  Receive interrupt priority.
- i2s_cfg_t::txi_ipl

  Transmit interrupt priority.
- i2s_cfg_t::idle_err_ipl

  Idle/Error interrupt priority.

9.18.10.6 stop

```
ssp_err_t(* i2s_api_t::stop) (i2s_ctrl_t *const p_ctrl, i2s_dir_t const dir)
```

Detailed description

Stop communication. Transmission is stopped when callback is called with I2S_EVENT_IDLE. Reception is stopped when callback is called with I2S_EVENT_RX_EMPTY. Implemented as

- R_SSI_Stop

Table 932:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
dir	in	Direction of communication to stop.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter dir

9.18.10.7 mute

```
ssp_err_t(* i2s_api_t::mute) (i2s_ctrl_t *const p_ctrl, i2s_mute_t const
mute_enable)
```

Detailed description

Enable or disable mute. Implemented as

- R_SSI_Mute

Table 933:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
mute_enable	in	Whether to enable or disable mute.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter mute_enable

Definition: `i2s_mute_tconst mute_enable`

Mute audio samples.

9.18.10.8 write

```
ssp_err_t(* i2s_api_t::write) (i2s_ctrl_t *const p_ctrl, uint8_t const *const
p_src, uint16_t const bytes)
```

Detailed description

Write I2S data. All transmit data is queued when callback is called with I2S_EVENT_TX_EMPTY. Transmission is complete when callback is called with I2S_EVENT_IDLE. Implemented as

- R_SSI_Write

Table 934:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
p_src	in	Buffer of PCM samples. Must be 4 byte aligned.

Table 934:Parameters (Continued)

Name	Direction	Description
bytes	in	Number of bytes in the buffer. Recommended requesting a multiple of 8 bytes. If not a multiple of 8, padding 0s will be added to transmission to make it a multiple of 8.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter bytes

`uint16_t`

9.18.10.9 read

```
ssp_err_t(* i2s_api_t::read) (i2s_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint16_t const bytes)
```

Detailed description

Read I2S data. Reception is complete when callback is called with I2S_EVENT_RX_EMPTY. Implemented as

- R_SSI_Read

Table 935:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
p_dest	in	Buffer to store PCM samples. Must be 4 byte aligned.
bytes	in	Number of bytes in the buffer. Recommended requesting a multiple of 8 bytes. If not a multiple of 8, receive will stop at the multiple of 8 below requested bytes.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint16_t`

9.18.10.10 writeRead

```
ssp_err_t(* i2s_api_t::writeRead) (i2s_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint8_t *const p_dest, uint16_t const bytes)
```

Detailed description

Simultaneously write and read I2S data. Transmission and reception are complete when callback is called with I2S_EVENT_IDLE. Implemented as

- R_SSI_WriteRead

Table 936:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
p_src	in	Buffer of PCM samples. Must be 4 byte aligned.
p_dest	in	Buffer to store PCM samples. Must be 4 byte aligned.
bytes	in	Number of bytes in the buffers. Recommended requesting a multiple of 8 bytes. If not a multiple of 8, padding 0s will be added to transmission to make it a multiple of 8, and receive will stop at the multiple of 8 below requested bytes.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter p_dest

uint8_t

Parameter bytes

uint16_t

9.18.10.11 infoGet

```
ssp_err_t(* i2s_api_t::infoGet) (i2s_ctrl_t *const p_ctrl, i2s_info_t *const
p_info)
```

Detailed description

Get instance specific information and store it in provided pointer p_info. Implemented as

- R_SSI_InfoGet

Table 937:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.
p_info	out	Collection of information for this instance.

Parameter p_ctrl

Definition: i2s_ctrl_t*const p_ctrl

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

Parameter p_info

Definition: i2s_info_t*const p_info

Timer information structure to store various information for an I2S instance.

- i2s_info_t::status
- i2s_info_t::sampling_freq_hz

  Sampling frequency in Hertz.

9.18.10.12 close

```
ssp_err_t(* i2s_api_t::close) (i2s_ctrl_t *const p_ctrl)
```

Detailed description

Allows driver to be reconfigured and may reduce power consumption. Implemented as

- R_SSI_Close

Table 938:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this instance.

Parameter p_ctrl

Definition: `i2s_ctrl_t*const p_ctrl`

I2S control block. Allocate an instance specific control block to pass into the I2S API calls. Implemented asssi_instance_ctrl_t

9.18.10.13 versionGet

```
ssp_err_t(* i2s_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_SSI_VersionGet

Table 939:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.18.10.14 i2s_instance_t

i2s_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- i2s_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- i2s_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- i2s_api_t const * p_api

  Pointer to the API structure for this instance.

9.19 I/O Port Interface

Interface for accessing I/O ports and configuring I/O functionality.

The IOPort shared interface provides the ability to access the IOPorts of a device at both bit and port level. Port and pin direction can be changed.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

IOPORT Interface description: I/O Port Driver

9.19.1 Interface API

ioport_api_t

Function name	Description
.init	Initialize internal driver data and initial pin configurations. Called during startup. Do not call this API during runtime. Use pinsCfg for runtime reconfiguration of multiple pins.
.pinsCfg	Configure multiple pins.
.pinCfg	Configure settings for an individual pin.
.pinDirectionSet	Set the pin direction of a pin.
.pinEventInputRead	Read the event input data of the specified pin and return the level.
.pinEventOutputWrite	Write pin event data.
.pinEthernetModeCfg	Configure the PHY mode of the Ethernet channels.
.pinRead	Read level of a pin.
.pinWrite	Write specified level to a pin.
.portDirectionSet	Set the direction of one or more pins on a port.
.portEventInputRead	Read captured event data for a port.
.portEventOutputWrite	Write event output data for a port.

Function name	Description
.portRead	Read states of pins on the specified port.
.portWrite	Write to multiple pins on a port.
.versionGet	Return the version of the IOPort driver.

9.19.2 Data structures

- ioport_pin_cfg_t
- ioport_cfg_t
- ioport_instance_t

9.19.3 Enumerations

- ioport_level_t
- ioport_direction_t
- ioport_port_t
- ioport_port_pin_t
- ioport_peripheral_t
- ioport_ethernet_channel_t
- ioport_ethernet_mode_t
- ioport_cfg_options_t

9.19.4 Typedefs

- ioport_size_t

9.19.5 Defines

- #define IOPORT_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define IOPORT_API_VERSION_MINOR

  Initial value:`(2U)`
- #define IOPORT_PRV_PFS_PSEL_OFFSET

  Initial value:`(24)`

9.19.6 API Data

9.19.6.1 ioport_level_t

`ioport_level_t`

Detailed description

Levels that can be set and read for individual pins

Enumerated values

Name	Description
IOPORT_LEVEL_LOW	Low.
IOPORT_LEVEL_HIGH	High.

9.19.6.2 ioport_direction_t

`ioport_direction_t`

Detailed description

Direction of individual pins

Enumerated values

Name	Description
IOPORT_DIRECTION_INPUT	Input.
IOPORT_DIRECTION_OUTPUT	Output.

9.19.6.3 ioport_port_t

`ioport_port_t`

Detailed description

Superset list of all possible IO ports.

Enumerated values

Name	Description
IOPORT_PORT_00	IO port 0.

Name	Description
IOPORT_PORT_01	IO port 1.
IOPORT_PORT_02	IO port 2.
IOPORT_PORT_03	IO port 3.
IOPORT_PORT_04	IO port 4.
IOPORT_PORT_05	IO port 5.
IOPORT_PORT_06	IO port 6.
IOPORT_PORT_07	IO port 7.
IOPORT_PORT_08	IO port 8.
IOPORT_PORT_09	IO port 9.
IOPORT_PORT_10	IO port 10.
IOPORT_PORT_11	IO port 11.

9.19.6.4 ioport_port_pin_t

`ioport_port_pin_t`

Detailed description

Superset list of all possible IO port pins.

Enumerated values

Name	Description
IOPORT_PORT_00_PIN_00	IO port 0 pin 0.
IOPORT_PORT_00_PIN_01	IO port 0 pin 1.
IOPORT_PORT_00_PIN_02	IO port 0 pin 2.
IOPORT_PORT_00_PIN_03	IO port 0 pin 3.
IOPORT_PORT_00_PIN_04	IO port 0 pin 4.
IOPORT_PORT_00_PIN_05	IO port 0 pin 5.
IOPORT_PORT_00_PIN_06	IO port 0 pin 6.

Name	Description
IOPORT_PORT_00_PIN_07	IO port 0 pin 7.
IOPORT_PORT_00_PIN_08	IO port 0 pin 8.
IOPORT_PORT_00_PIN_09	IO port 0 pin 9.
IOPORT_PORT_00_PIN_10	IO port 0 pin 10.
IOPORT_PORT_00_PIN_11	IO port 0 pin 11.
IOPORT_PORT_00_PIN_12	IO port 0 pin 12.
IOPORT_PORT_00_PIN_13	IO port 0 pin 13.
IOPORT_PORT_00_PIN_14	IO port 0 pin 14.
IOPORT_PORT_00_PIN_15	IO port 0 pin 15.
IOPORT_PORT_01_PIN_00	IO port 1 pin 0.
IOPORT_PORT_01_PIN_01	IO port 1 pin 1.
IOPORT_PORT_01_PIN_02	IO port 1 pin 2.
IOPORT_PORT_01_PIN_03	IO port 1 pin 3.
IOPORT_PORT_01_PIN_04	IO port 1 pin 4.
IOPORT_PORT_01_PIN_05	IO port 1 pin 5.
IOPORT_PORT_01_PIN_06	IO port 1 pin 6.
IOPORT_PORT_01_PIN_07	IO port 1 pin 7.
IOPORT_PORT_01_PIN_08	IO port 1 pin 8.
IOPORT_PORT_01_PIN_09	IO port 1 pin 9.
IOPORT_PORT_01_PIN_10	IO port 1 pin 10.
IOPORT_PORT_01_PIN_11	IO port 1 pin 11.
IOPORT_PORT_01_PIN_12	IO port 1 pin 12.
IOPORT_PORT_01_PIN_13	IO port 1 pin 13.
IOPORT_PORT_01_PIN_14	IO port 1 pin 14.

Name	Description
IOPORT_PORT_01_PIN_15	IO port 1 pin 15.
IOPORT_PORT_02_PIN_00	IO port 2 pin 0.
IOPORT_PORT_02_PIN_01	IO port 2 pin 1.
IOPORT_PORT_02_PIN_02	IO port 2 pin 2.
IOPORT_PORT_02_PIN_03	IO port 2 pin 3.
IOPORT_PORT_02_PIN_04	IO port 2 pin 4.
IOPORT_PORT_02_PIN_05	IO port 2 pin 5.
IOPORT_PORT_02_PIN_06	IO port 2 pin 6.
IOPORT_PORT_02_PIN_07	IO port 2 pin 7.
IOPORT_PORT_02_PIN_08	IO port 2 pin 8.
IOPORT_PORT_02_PIN_09	IO port 2 pin 9.
IOPORT_PORT_02_PIN_10	IO port 2 pin 10.
IOPORT_PORT_02_PIN_11	IO port 2 pin 11.
IOPORT_PORT_02_PIN_12	IO port 2 pin 12.
IOPORT_PORT_02_PIN_13	IO port 2 pin 13.
IOPORT_PORT_02_PIN_14	IO port 2 pin 14.
IOPORT_PORT_02_PIN_15	IO port 2 pin 15.
IOPORT_PORT_03_PIN_00	IO port 3 pin 0.
IOPORT_PORT_03_PIN_01	IO port 3 pin 1.
IOPORT_PORT_03_PIN_02	IO port 3 pin 2.
IOPORT_PORT_03_PIN_03	IO port 3 pin 3.
IOPORT_PORT_03_PIN_04	IO port 3 pin 4.
IOPORT_PORT_03_PIN_05	IO port 3 pin 5.
IOPORT_PORT_03_PIN_06	IO port 3 pin 6.

Name	Description
IOPORT_PORT_03_PIN_07	IO port 3 pin 7.
IOPORT_PORT_03_PIN_08	IO port 3 pin 8.
IOPORT_PORT_03_PIN_09	IO port 3 pin 9.
IOPORT_PORT_03_PIN_10	IO port 3 pin 10.
IOPORT_PORT_03_PIN_11	IO port 3 pin 11.
IOPORT_PORT_03_PIN_12	IO port 3 pin 12.
IOPORT_PORT_03_PIN_13	IO port 3 pin 13.
IOPORT_PORT_03_PIN_14	IO port 3 pin 14.
IOPORT_PORT_03_PIN_15	IO port 3 pin 15.
IOPORT_PORT_04_PIN_00	IO port 4 pin 0.
IOPORT_PORT_04_PIN_01	IO port 4 pin 1.
IOPORT_PORT_04_PIN_02	IO port 4 pin 2.
IOPORT_PORT_04_PIN_03	IO port 4 pin 3.
IOPORT_PORT_04_PIN_04	IO port 4 pin 4.
IOPORT_PORT_04_PIN_05	IO port 4 pin 5.
IOPORT_PORT_04_PIN_06	IO port 4 pin 6.
IOPORT_PORT_04_PIN_07	IO port 4 pin 7.
IOPORT_PORT_04_PIN_08	IO port 4 pin 8.
IOPORT_PORT_04_PIN_09	IO port 4 pin 9.
IOPORT_PORT_04_PIN_10	IO port 4 pin 10.
IOPORT_PORT_04_PIN_11	IO port 4 pin 11.
IOPORT_PORT_04_PIN_12	IO port 4 pin 12.
IOPORT_PORT_04_PIN_13	IO port 4 pin 13.
IOPORT_PORT_04_PIN_14	IO port 4 pin 14.

Name	Description
IOPORT_PORT_04_PIN_15	IO port 4 pin 15.
IOPORT_PORT_05_PIN_00	IO port 5 pin 0.
IOPORT_PORT_05_PIN_01	IO port 5 pin 1.
IOPORT_PORT_05_PIN_02	IO port 5 pin 2.
IOPORT_PORT_05_PIN_03	IO port 5 pin 3.
IOPORT_PORT_05_PIN_04	IO port 5 pin 4.
IOPORT_PORT_05_PIN_05	IO port 5 pin 5.
IOPORT_PORT_05_PIN_06	IO port 5 pin 6.
IOPORT_PORT_05_PIN_07	IO port 5 pin 7.
IOPORT_PORT_05_PIN_08	IO port 5 pin 8.
IOPORT_PORT_05_PIN_09	IO port 5 pin 9.
IOPORT_PORT_05_PIN_10	IO port 5 pin 10.
IOPORT_PORT_05_PIN_11	IO port 5 pin 11.
IOPORT_PORT_05_PIN_12	IO port 5 pin 12.
IOPORT_PORT_05_PIN_13	IO port 5 pin 13.
IOPORT_PORT_05_PIN_14	IO port 5 pin 14.
IOPORT_PORT_05_PIN_15	IO port 5 pin 15.
IOPORT_PORT_06_PIN_00	IO port 6 pin 0.
IOPORT_PORT_06_PIN_01	IO port 6 pin 1.
IOPORT_PORT_06_PIN_02	IO port 6 pin 2.
IOPORT_PORT_06_PIN_03	IO port 6 pin 3.
IOPORT_PORT_06_PIN_04	IO port 6 pin 4.
IOPORT_PORT_06_PIN_05	IO port 6 pin 5.
IOPORT_PORT_06_PIN_06	IO port 6 pin 6.

Name	Description
IOPORT_PORT_06_PIN_07	IO port 6 pin 7.
IOPORT_PORT_06_PIN_08	IO port 6 pin 8.
IOPORT_PORT_06_PIN_09	IO port 6 pin 9.
IOPORT_PORT_06_PIN_10	IO port 6 pin 10.
IOPORT_PORT_06_PIN_11	IO port 6 pin 11.
IOPORT_PORT_06_PIN_12	IO port 6 pin 12.
IOPORT_PORT_06_PIN_13	IO port 6 pin 13.
IOPORT_PORT_06_PIN_14	IO port 6 pin 14.
IOPORT_PORT_06_PIN_15	IO port 6 pin 15.
IOPORT_PORT_07_PIN_00	IO port 7 pin 0.
IOPORT_PORT_07_PIN_01	IO port 7 pin 1.
IOPORT_PORT_07_PIN_02	IO port 7 pin 2.
IOPORT_PORT_07_PIN_03	IO port 7 pin 3.
IOPORT_PORT_07_PIN_04	IO port 7 pin 4.
IOPORT_PORT_07_PIN_05	IO port 7 pin 5.
IOPORT_PORT_07_PIN_06	IO port 7 pin 6.
IOPORT_PORT_07_PIN_07	IO port 7 pin 7.
IOPORT_PORT_07_PIN_08	IO port 7 pin 8.
IOPORT_PORT_07_PIN_09	IO port 7 pin 9.
IOPORT_PORT_07_PIN_10	IO port 7 pin 10.
IOPORT_PORT_07_PIN_11	IO port 7 pin 11.
IOPORT_PORT_07_PIN_12	IO port 7 pin 12.
IOPORT_PORT_07_PIN_13	IO port 7 pin 13.
IOPORT_PORT_07_PIN_14	IO port 7 pin 14.

Name	Description
IOPORT_PORT_07_PIN_15	IO port 7 pin 15.
IOPORT_PORT_08_PIN_00	IO port 8 pin 0.
IOPORT_PORT_08_PIN_01	IO port 8 pin 1.
IOPORT_PORT_08_PIN_02	IO port 8 pin 2.
IOPORT_PORT_08_PIN_03	IO port 8 pin 3.
IOPORT_PORT_08_PIN_04	IO port 8 pin 4.
IOPORT_PORT_08_PIN_05	IO port 8 pin 5.
IOPORT_PORT_08_PIN_06	IO port 8 pin 6.
IOPORT_PORT_08_PIN_07	IO port 8 pin 7.
IOPORT_PORT_08_PIN_08	IO port 8 pin 8.
IOPORT_PORT_08_PIN_09	IO port 8 pin 9.
IOPORT_PORT_08_PIN_10	IO port 8 pin 10.
IOPORT_PORT_08_PIN_11	IO port 8 pin 11.
IOPORT_PORT_08_PIN_12	IO port 8 pin 12.
IOPORT_PORT_08_PIN_13	IO port 8 pin 13.
IOPORT_PORT_08_PIN_14	IO port 8 pin 14.
IOPORT_PORT_08_PIN_15	IO port 8 pin 15.
IOPORT_PORT_09_PIN_00	IO port 9 pin 0.
IOPORT_PORT_09_PIN_01	IO port 9 pin 1.
IOPORT_PORT_09_PIN_02	IO port 9 pin 2.
IOPORT_PORT_09_PIN_03	IO port 9 pin 3.
IOPORT_PORT_09_PIN_04	IO port 9 pin 4.
IOPORT_PORT_09_PIN_05	IO port 9 pin 5.
IOPORT_PORT_09_PIN_06	IO port 9 pin 6.

Name	Description
IOPORT_PORT_09_PIN_07	IO port 9 pin 7.
IOPORT_PORT_09_PIN_08	IO port 9 pin 8.
IOPORT_PORT_09_PIN_09	IO port 9 pin 9.
IOPORT_PORT_09_PIN_10	IO port 9 pin 10.
IOPORT_PORT_09_PIN_11	IO port 9 pin 11.
IOPORT_PORT_09_PIN_12	IO port 9 pin 12.
IOPORT_PORT_09_PIN_13	IO port 9 pin 13.
IOPORT_PORT_09_PIN_14	IO port 9 pin 14.
IOPORT_PORT_09_PIN_15	IO port 9 pin 15.
IOPORT_PORT_10_PIN_00	IO port 10 pin 0.
IOPORT_PORT_10_PIN_01	IO port 10 pin 1.
IOPORT_PORT_10_PIN_02	IO port 10 pin 2.
IOPORT_PORT_10_PIN_03	IO port 10 pin 3.
IOPORT_PORT_10_PIN_04	IO port 10 pin 4.
IOPORT_PORT_10_PIN_05	IO port 10 pin 5.
IOPORT_PORT_10_PIN_06	IO port 10 pin 6.
IOPORT_PORT_10_PIN_07	IO port 10 pin 7.
IOPORT_PORT_10_PIN_08	IO port 10 pin 8.
IOPORT_PORT_10_PIN_09	IO port 10 pin 9.
IOPORT_PORT_10_PIN_10	IO port 10 pin 10.
IOPORT_PORT_10_PIN_11	IO port 10 pin 11.
IOPORT_PORT_10_PIN_12	IO port 10 pin 12.
IOPORT_PORT_10_PIN_13	IO port 10 pin 13.
IOPORT_PORT_10_PIN_14	IO port 10 pin 14.

Name	Description
IOPORT_PORT_10_PIN_15	IO port 10 pin 15.
IOPORT_PORT_11_PIN_00	IO port 11 pin 0.
IOPORT_PORT_11_PIN_01	IO port 11 pin 1.
IOPORT_PORT_11_PIN_02	IO port 11 pin 2.
IOPORT_PORT_11_PIN_03	IO port 11 pin 3.
IOPORT_PORT_11_PIN_04	IO port 11 pin 4.
IOPORT_PORT_11_PIN_05	IO port 11 pin 5.
IOPORT_PORT_11_PIN_06	IO port 11 pin 6.
IOPORT_PORT_11_PIN_07	IO port 11 pin 7.
IOPORT_PORT_11_PIN_08	IO port 11 pin 8.
IOPORT_PORT_11_PIN_09	IO port 11 pin 9.
IOPORT_PORT_11_PIN_10	IO port 11 pin 10.
IOPORT_PORT_11_PIN_11	IO port 11 pin 11.
IOPORT_PORT_11_PIN_12	IO port 11 pin 12.
IOPORT_PORT_11_PIN_13	IO port 11 pin 13.
IOPORT_PORT_11_PIN_14	IO port 11 pin 14.
IOPORT_PORT_11_PIN_15	IO port 11 pin 15.

9.19.6.5 ioport_peripheral_t

`ioport_peripheral_t`

Detailed description

Superset of all peripheral functions.

Enumerated values

Name	Description
IOPORT_PERIPHERAL_IO	Pin will functions as an IO pin.

Name	Description
IOPORT_PERIPHERAL_DEBUG	Pin will function as a DEBUG pin.
IOPORT_PERIPHERAL_AGT	Pin will function as an AGT.
IOPORT_PERIPHERAL_GPT0	Pin will function as a GPT.
IOPORT_PERIPHERAL_GPT1	Pin will function as a GPT.
IOPORT_PERIPHERAL_SCI0_2_4_6_8	Pin will function as an SCI.
IOPORT_PERIPHERAL_SCI1_3_5_7_9	Pin will function as an SCI.
IOPORT_PERIPHERAL_RSPI	Pin will function as a RSPI.
IOPORT_PERIPHERAL_RIIC	Pin will function as a RIIC.
IOPORT_PERIPHERAL_KEY	Pin will function as a KEY.
IOPORT_PERIPHERAL_CLKOUT_COMP_RTC	Pin will function as a.
IOPORT_PERIPHERAL_CAC_AD	Pin will function as a CAC/ADC.
IOPORT_PERIPHERAL_BUS	Pin will function as a BUS.
IOPORT_PERIPHERAL_CTSU	Pin will function as a CTSU.
IOPORT_PERIPHERAL_LCDC	Pin will function as a segment LCD.
IOPORT_PERIPHERAL_DALI	Pin will function as a DALI.
IOPORT_PERIPHERAL_CAN	Pin will function as a CAN.
IOPORT_PERIPHERAL_QSPI	Pin will function as a QSPI.
IOPORT_PERIPHERAL_SSI	Pin will function as an SSI.
IOPORT_PERIPHERAL_USB_FS	Pin will function as a USB.
IOPORT_PERIPHERAL_USB_HS	Pin will function as a USB.
IOPORT_PERIPHERAL_SDHI_MMC	Pin will function as an SD/MMC.
IOPORT_PERIPHERAL_ETHER_MII	Pin will function as an Ethernet.
IOPORT_PERIPHERAL_ETHER_RMII	Pin will function as an Ethernet.
IOPORT_PERIPHERAL_PDC	Pin will function as a PDC.

Name	Description
IOPORT_PERIPHERAL_LCD_GRAPHICS	Pin will function as a graphics.
IOPORT_PERIPHERAL_TRACE	Pin will function as a debug trace.
IOPORT_PERIPHERAL_END	Marks end of enum - used by.

9.19.6.6 ioport_ethernet_channel_t

`ioport_ethernet_channel_t`

Detailed description

Superset of Ethernet channels.

Enumerated values

Name	Description
IOPORT_ETHERNET_CHANNEL_0	Used to select Ethernet channel 0.
IOPORT_ETHERNET_CHANNEL_1	Used to select Ethernet channel 1.
IOPORT_ETHERNET_CHANNEL_END	Marks end of enum - used by parameter checking.

9.19.6.7 ioport_ethernet_mode_t

`ioport_ethernet_mode_t`

Detailed description

Superset of Ethernet PHY modes.

Enumerated values

Name	Description
IOPORT_ETHERNET_MODE_MII	Ethernet PHY mode set to MII.
IOPORT_ETHERNET_MODE_RMII	Ethernet PHY mode set to RMII.
IOPORT_ETHERNET_MODE_END	Marks end of enum - used by parameter checking.

9.19.6.8 ioport_cfg_options_t

`ioport_cfg_options_t`

Detailed description

Options to configure pin functions

Enumerated values

Name	Description
IOPORT_CFG_PORT_DIRECTION_INPUT	Sets the pin direction to input (default)
IOPORT_CFG_PORT_DIRECTION_OUTPUT	Sets the pin direction to output.
IOPORT_CFG_PORT_OUTPUT_LOW	Sets the pin level to low.
IOPORT_CFG_PORT_OUTPUT_HIGH	Sets the pin level to high.
IOPORT_CFG_PULLUP_ENABLE	Enables the pin's internal pull-up.
IOPORT_CFG_PIM_TTL	Enables the pin's input mode.
IOPORT_CFG_NMOS_ENABLE	Enables the pin's NMOS open-drain output.
IOPORT_CFG_PMOS_ENABLE	Enables the pin's PMOS open-drain ouput.
IOPORT_CFG_DRIVE_MID	Sets pin drive output to medium.
IOPORT_CFG_DRIVE_MID_IIC	Sets pin to drive output needed for IIC on a 20mA port.
IOPORT_CFG_DRIVE_HIGH	Sets pin drive output to high.
IOPORT_CFG_EVENT_RISING_EDGE	Sets pin event trigger to rising edge.
IOPORT_CFG_EVENT_FALLING_EDGE	Sets pin event trigger to falling edge.
IOPORT_CFG_EVENT_BOTH_EDGES	Sets pin event trigger to both edges.
IOPORT_CFG_IRQ_ENABLE	Sets pin as an IRQ pin.
IOPORT_CFG_ANALOG_ENABLE	Enables pin to operate as an analog pin.
IOPORT_CFG_PERIPHERAL_PIN	Enables pin to operate as a peripheral pin.

9.19.6.9 ioport_size_t

```
typedef uint16_t ioport_size_t
```

Brief description

IO port size on this device.

Detailed description

IO port type used with ports

9.19.7 API Structures

9.19.7.1 ioport_pin_cfg_t

ioport_pin_cfg_t

Detailed description

Pin identifier and pin PFS pin configuration value

Variables

- uint32_t pin_cfg

  Pin PFS configuration - Use ioport_cfg_options_t parameters to configure.

- ioport_port_pin_t pin

  Pin identifier.

9.19.7.2 ioport_cfg_t

ioport_cfg_t

Detailed description

Multiple pin configuration data for loading into PFS registers by R_IOPORT_Init

Variables

- uint16_t number_of_pins

  Number of pins for which there is configuration data.

- ioport_pin_cfg_t const * p_pin_cfg_data

  Pin configuration data.

9.19.7.3 ioport_api_t

ioport_api_t

Detailed description

IOPort driver structure. IOPort functions implemented at the HAL layer will follow this API.

9.19.7.4 init

```
ssp_err_t(* ioport_api_t::init) (const ioport_cfg_t *p_cfg)
```

Detailed description

Initialize internal driver data and initial pin configurations. Called during startup. Do not call this API during runtime. Use pinsCfg for runtime reconfiguration of multiple pins. Implemented as

- R_IOPORT_Init

Table 940:Parameters

Name	Direction	Description
p_cfg	in	Pointer to pin configuration data array.

Parameter p_cfg

Definition: `ioport_cfg_t ioport_cfg_t *p_cfg`

Multiple pin configuration data for loading into PFS registers by R_IOPORT_Init

- `ioport_cfg_t`::`number_of_pins`

  Number of pins for which there is configuration data.

- `ioport_cfg_t`::`ioport_pin_cfg_t`

  Pin configuration data.

9.19.7.5 pinsCfg

```
ssp_err_t(* ioport_api_t::pinsCfg) (const ioport_cfg_t *p_cfg)
```

Detailed description

Configure multiple pins. Implemented as

- R_IOPORT_PinsCfg

Table 941:Parameters

Name	Direction	Description
p_cfg	in	Pointer to pin configuration data array.

Parameter p_cfg

Definition: `ioport_cfg_t ioport_cfg_t *p_cfg`

Multiple pin configuration data for loading into PFS registers by R_IOPORT_Init

- `ioport_cfg_t`::`number_of_pins`

  Number of pins for which there is configuration data.

- `ioport_cfg_t`::`ioport_pin_cfg_t`

  Pin configuration data.

9.19.7.6 pinCfg

```
ssp_err_t(* ioport_api_t::pinCfg) (ioport_port_pin_t pin, uint32_t cfg)
```

Detailed description

Configure settings for an individual pin. Implemented as

- R_IOPORT_PinCfg

Table 942:Parameters

Name	Direction	Description
pin	in	Pin to be read.
cfg	in	Configuration options for the pin.

Parameter pin

Parameter cfg

`uint32_t`

9.19.7.7 pinDirectionSet

```
ssp_err_t(* ioport_api_t::pinDirectionSet) (ioport_port_pin_t pin,
ioport_direction_t direction)
```

Detailed description

Set the pin direction of a pin. Implemented as

- R_IOPORT_PinDirectionSet

Table 943:Parameters

Name	Direction	Description
pin	in	Pin being configured.
direction	in	Direction to set pin to which is a member of ioport_direction_t.

Parameter pin

Parameter direction

9.19.7.8 pinEventInputRead

```
ssp_err_t(* ioport_api_t::pinEventInputRead) (ioport_port_pin_t pin,
ioport_level_t *p_pin_event)
```

Detailed description

Read the event input data of the specified pin and return the level. Implemented as

- R_IOPORT_PinEventInputRead

Table 944:Parameters

Name	Direction	Description
pin	in	Pin to be read.
p_pin_event	in	Pointer to return the event data.

Parameter pin

Parameter p_pin_event

Definition: `ioport_level_t*p_pin_event`

Levels that can be set and read for individual pins

9.19.7.9 pinEventOutputWrite

```
ssp_err_t(* ioport_api_t::pinEventOutputWrite) (ioport_port_pin_t pin,
ioport_level_t pin_value)
```

Detailed description

Write pin event data. Implemented as

- R_IOPORT_PinEventOutputWrite

Table 945:Parameters

Name	Direction	Description
pin	in	Pin event data is to be written to.
pin_value	in	Level to be written to pin output event.

Parameter pin

Parameter pin_value

Definition: `ioport_level_tpin_value`

Levels that can be set and read for individual pins

9.19.7.10 pinEthernetModeCfg

```
ssp_err_t(* ioport_api_t::pinEthernetModeCfg) (ioport_ethernet_channel_t channel,
ioport_ethernet_mode_t mode)
```

Detailed description

Configure the PHY mode of the Ethernet channels. Implemented as

- R_IOPORT_EthernetModeCfg

Table 946:Parameters

Name	Direction	Description
channel	in	Channel configuration will be set for.
mode	in	PHY mode to set the channel to.

Parameter channel

Parameter mode

9.19.7.11 pinRead

```
ssp_err_t(* ioport_api_t::pinRead) (ioport_port_pin_t pin, ioport_level_t
*p_pin_value)
```

Detailed description

Read level of a pin. Implemented as

- R_IOPORT_PinRead

Table 947:Parameters

Name	Direction	Description
pin	in	Pin to be read.
p_pin_value	in	Pointer to return the pin level.

Parameter pin

Parameter p_pin_value

Definition: `ioport_level_t*p_pin_value`

Levels that can be set and read for individual pins

9.19.7.12 pinWrite

```
ssp_err_t(* ioport_api_t::pinWrite) (ioport_port_pin_t pin, ioport_level_t level)
```

Detailed description

Write specified level to a pin. Implemented as

- R_IOPORT_PinWrite

Table 948:Parameters

Name	Direction	Description
pin	in	Pin to be written to.
level	in	State to be written to the pin.

Parameter pin

Parameter level

9.19.7.13 portDirectionSet

```
ssp_err_t(* ioport_api_t::portDirectionSet) (ioport_port_t port, ioport_size_t
direction_values, ioport_size_t mask)
```

Detailed description

Set the direction of one or more pins on a port. Implemented as

- R_IOPORT_PortDirectionSet

Table 949:Parameters

Name	Direction	Description
port	in	Port being configured.
direction_values	in	Value controlling direction of pins on port (1 - output, 0 - input).
mask	in	Mask controlling which pins on the port are to be configured.

Parameter port

Parameter direction_values

Definition: `ioport_size_tdirection_values`

IO port type used with ports

Parameter mask

Definition: `ioport_size_tmask`

IO port type used with ports

9.19.7.14 portEventInputRead

```
ssp_err_t(* ioport_api_t::portEventInputRead) (ioport_port_t port, ioport_size_t
*p_event_data)
```

Detailed description

Read captured event data for a port. Implemented as

- R_IOPORT_PortEventInputRead

Table 950:Parameters

Name	Direction	Description
port	in	Port to be read.
p_event_data	in	Pointer to return the event data.

Parameter port

Parameter p_event_data

Definition: `ioport_size_t*p_event_data`

IO port type used with ports

9.19.7.15 portEventOutputWrite

```
ssp_err_t(* ioport_api_t::portEventOutputWrite) (ioport_port_t port,
ioport_size_t event_data, ioport_size_t mask_value)
```

Detailed description

Write event output data for a port. Implemented as

- R_IOPORT_PortEventOutputWrite

Table 951:Parameters

Name	Direction	Description
port	in	Port event data will be written to.
event_data	in	Data to be written as event data to specified port.
mask_value	in	Each bit set to 1 in the mask corresponds to that bit's value in event data. being written to port.

Parameter port

Parameter event_data

Definition: `ioport_size_t`event_data

IO port type used with ports

Parameter mask_value

Definition: `ioport_size_t`mask_value

IO port type used with ports

9.19.7.16 portRead

```
ssp_err_t(* ioport_api_t::portRead) (ioport_port_t port, ioport_size_t
*p_port_value)
```

Detailed description

Read states of pins on the specified port. Implemented as

- R_IOPORT_PortRead

Table 952:Parameters

Name	Direction	Description
port	in	Port to be read.
p_port_value	in	Pointer to return the port value.

Parameter port

Parameter p_port_value

Definition: `ioport_size_t`*p_port_value

IO port type used with ports

9.19.7.17 portWrite

```
ssp_err_t(* ioport_api_t::portWrite) (ioport_port_t port, ioport_size_t value,
ioport_size_t mask)
```

Detailed description

Write to multiple pins on a port. Implemented as

- R_IOPORT_PortWrite

Table 953:Parameters

Name	Direction	Description
port	in	Port to be written to.

Table 953:Parameters (Continued)

Name	Direction	Description
value	in	Value to be written to the port.
mask	in	Mask controlling which pins on the port are written to.

Parameter port

Parameter value

Definition: `ioport_size_tvalue`

IO port type used with ports

Parameter mask

Definition: `ioport_size_tmask`

IO port type used with ports

9.19.7.18 versionGet

```
ssp_err_t(* ioport_api_t::versionGet) (ssp_version_t *p_data)
```

Detailed description

Return the version of the IOPort driver. Implemented as

- R_IOPORT_VersionGet

Table 954:Parameters

Name	Direction	Description
p_data	out	Memory address to return version information to.

Parameter p_data

Definition: `ssp_version_t *p_data`

- `ssp_version_t::version_id`

  Version id
- `ssp_version_t::code_version_minor`

  Code minor version.
- `ssp_version_t::code_version_major`

  Code major version.

- ssp_version_t::api_version_minor

  API minor version.
- ssp_version_t::api_version_major

  API major version.
- ssp_version_tstruct{}

  Code version parameters

9.19.7.19 ioport_instance_t

ioport_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- ioport_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- ioport_api_t const * p_api

  Pointer to the API structure for this instance.

9.20 Input Capture Interface

Interface for sampling input signals for pulse width.

9.20.1 Summary

The input capture interface provides for sampling of input signals to determine the width of a pulse (from one edge to the opposite edge). An interrupt can be triggered after each measurement is captured.

Implemented by: GPT Input Capture

See also: Timer Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

9.20.2 Interface API

input_capture_api_t

Function name	Description
.open	Initial configuration.
.disable	Disables input capture measurement.
.enable	Enables input capture measurement.
.infoGet	Gets the status (running or not) of the measurement counter.
.lastCaptureGet	Gets the last captured timer/counter value
.close	Close the input capture operation. Allows driver to be reconfigured, and may reduce power consumption.
.versionGet	Gets the version of this API and stores it in structure pointed to by p_version.

9.20.3 Data structures

- input_capture_callback_args_t
- input_capture_capture_t
- input_capture_info_t
- input_capture_cfg_t
- input_capture_instance_t

9.20.4 Enumerations

- input_capture_mode_t
- input_capture_signal_edge_t
- input_capture_signal_level_t
- input_capture_repetition_t
- input_capture_event_t
- input_capture_status_t
- input_capture_variant_t

9.20.5 Typedefs

- input_capture_ctrl_t

9.20.6 Defines

- #define INPUT_CAPTURE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define INPUT_CAPTURE_API_VERSION_MINOR

  Initial value:`(3U)`
- #define info_capture_info_t

  Initial value:`input_capture_info_t`

  Mapping of deprecated info_capture_info_t.

9.20.7 API Data

9.20.7.1 input_capture_mode_t

`input_capture_mode_t`

Detailed description

Input capture operational modes

Enumerated values

Name	Description
INPUT_CAPTURE_MODE_PULSE_WIDTH	Measure a signal pulse width.

9.20.7.2 input_capture_signal_edge_t

`input_capture_signal_edge_t`

Detailed description

Input capture signal edge trigger

Enumerated values

Name	Description
INPUT_CAPTURE_SIGNAL_EDGE_RISING	The capture begins at the rising edge.
INPUT_CAPTURE_SIGNAL_EDGE_FALLING	The capture begins at the falling edge.

9.20.7.3 input_capture_signal_level_t

`input_capture_signal_level_t`

Detailed description

Input capture signal level, primarily used for the enable signal

Enumerated values

Name	Description
INPUT_CAPTURE_SIGNAL_LEVEL_NONE	Use this if signal_level is not applicable to a particular measurement.
INPUT_CAPTURE_SIGNAL_LEVEL_LOW	The capture is enabled at the low level.
INPUT_CAPTURE_SIGNAL_LEVEL_HIGH	The capture is enabled at the high level.

9.20.7.4 input_capture_repetition_t

`input_capture_repetition_t`

Detailed description

Specifies either a one-time or continuous measurements

Enumerated values

Name	Description
INPUT_CAPTURE_REPETITION_PERIODIC	Capture continuous measurements, until explicitly stopped or disabled.
INPUT_CAPTURE_REPETITION_ONE_SHOT	Capture a single measurement, then interrupts are disabled.

9.20.7.5 input_capture_event_t

`input_capture_event_t`

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
INPUT_CAPTURE_EVENT_MEASUREMENT	A capture measurement has been captured.

Name	Description
INPUT_CAPTURE_EVENT_OVERFLOW	A capture measurement overflowed the counter.

9.20.7.6 input_capture_status_t

```
input_capture_status_t
```

Detailed description

Input capture status.

Enumerated values

Name	Description
INPUT_CAPTURE_STATUS_IDLE	The input capture timer is idle.
INPUT_CAPTURE_STATUS_CAPTURING	A capture measurement is in progress.

9.20.7.7 input_capture_variant_t

```
input_capture_variant_t
```

Detailed description

Input capture timer variant types.

Enumerated values

Name	Description
INPUT_CAPTURE_VARIANT_32_BIT	32-bit timer
INPUT_CAPTURE_VARIANT_16_BIT	16-bit timer

9.20.7.8 input_capture_ctrl_t

```
typedef void input_capture_ctrl_t
```

Detailed description

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls.
Implemented as

- gpt_input_capture_instance_ctrl_t

9.20.8 API Structures

9.20.8.1 input_capture_callback_args_t

input_capture_callback_args_t

Detailed description

Callback function parameter data

Variables

- uint8_t channel

  The channel being used.

- input_capture_event_t event

  The event that caused the interrupt and callback.

- uint32_t counter

  The value of the timer captured at the time of interrupt.

- uint32_t overflows

  The number of counter overflows that occurred during this measurement.

- void const * p_context

  Placeholder for user data, set in input_capture_cfg_t::p_context.

9.20.8.2 input_capture_capture_t

input_capture_capture_t

Detailed description

Capture data

Variables

- uint32_t counter

  The value of the timer captured at the time of interrupt.

- uint32_t overflows

  The number of counter overflows that occurred during this measurement.

9.20.8.3 input_capture_info_t

input_capture_info_t

Detailed description

Driver information

Variables

- input_capture_status_t status

  Whether or not a capture is in progress.
- input_capture_variant_t variant

  Whether timer is 16 or 32 bits.

9.20.8.4 input_capture_cfg_t

input_capture_cfg_t

Detailed description

User configuration structure, passed to open function

Variables

- uint8_t channel

  The channel in use.
- uint8_t capture_irq_ipl

  Capture interrupt priority.
- uint8_t overflow_irq_ipl

  Overflow interrupt priority.
- input_capture_mode_t mode

  The mode of measurement to be performed.
- input_capture_signal_edge_t edge

  The triggering edge to start a measurement (rise or fall).
- input_capture_repetition_t repetition

  One-shot or periodic measurement.
- bool autostart

  Specifies whether interrupts are enabled or not after open.
- void const * p_extend

  REQUIRED. Pointer to peripheral-specific extension parameters. See gpt_input_capture_extend_t for GPT.
- void(* p_callback)(input_capture_callback_args_t *p_args)

  Pointer to user's callback function, or NULL if no interrupt desired.
- void const * p_context

  Pointer to user's context data, to be passed to the callback.

9.20.8.5 input_capture_api_t

input_capture_api_t

Detailed description

Input capture API structure. Functions implemented at the HAL layer will implement this API.

9.20.8.6 open

```
ssp_err_t(* input_capture_api_t::open) (input_capture_ctrl_t *const p_ctrl,
input_capture_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_GPT_InputCaptureOpen

NOTE: To reconfigure after calling this function, call close first.

Table 955:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block: memory allocated by caller, contents filled in by open.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `input_capture_ctrl_t*const p_ctrl`

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

Parameter p_cfg

Definition: `input_capture_cfg_t const *const p_cfg`

User configuration structure, passed to open function

- `input_capture_cfg_t`::`channel`

  The channel in use.

- `input_capture_cfg_t`::`capture_irq_ipl`

  Capture interrupt priority.

- `input_capture_cfg_t`::`overflow_irq_ipl`

  Overflow interrupt priority.

- `input_capture_cfg_t`::`input_capture_mode_t`

  The mode of measurement to be performed.

  Enumerated as:

  – INPUT_CAPTURE_MODE_PULSE_WIDTH

- input_capture_cfg_t::input_capture_signal_edge_t

  The triggering edge to start a measurement (rise or fall).

  Enumerated as:

  – INPUT_CAPTURE_SIGNAL_EDGE_RISING

  – INPUT_CAPTURE_SIGNAL_EDGE_FALLING
- input_capture_cfg_t::input_capture_repetition_t

  One-shot or periodic measurement.

  Enumerated as:

  – INPUT_CAPTURE_REPETITION_PERIODIC

  – INPUT_CAPTURE_REPETITION_ONE_SHOT
- input_capture_cfg_t::autostart

  Specifies whether interrupts are enabled or not after open.
- input_capture_cfg_t::p_extend

  REQUIRED. Pointer to peripheral-specific extension parameters. See gpt_input_capture_extend_t for GPT.
- input_capture_cfg_t::p_callback

  Pointer to user's callback function, or NULL if no interrupt desired.
- input_capture_cfg_t::p_context

  Pointer to user's context data, to be passed to the callback.

9.20.8.7 disable

```
ssp_err_t(* input_capture_api_t::disable) (input_capture_ctrl_t const *const
p_ctrl)
```

Detailed description

Disables input capture measurement. Implemented as

- R_GPT_InputCaptureDisable

Table 956:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized by open call.

Parameter p_ctrl

Definition: input_capture_ctrl_tconst *const p_ctrl

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

9.20.8.8 enable

```
ssp_err_t(* input_capture_api_t::enable) (input_capture_ctrl_t const *const
p_ctrl)
```

Detailed description

Enables input capture measurement. Implemented as

- R_GPT_InputCaptureEnable

Table 957:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized by open call.

NOTE: Interrupts may already be enabled if specified by input_capture_cfg_t::irq_enable.

Parameter p_ctrl

Definition: `input_capture_ctrl_tconst *const p_ctrl`

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

9.20.8.9 infoGet

```
ssp_err_t(* input_capture_api_t::infoGet) (input_capture_ctrl_t const *const
p_ctrl, input_capture_info_t *const p_info)
```

Detailed description

Gets the status (running or not) of the measurement counter. Implemented as

- R_GPT_InputCaptureInfoGet

Table 958:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized by open call.
p_info	out	Pointer to returned status. Result will be one of input_capture_status_t.

Parameter p_ctrl

Definition: input_capture_ctrl_tconst *const p_ctrl

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

Parameter p_info

Definition: input_capture_info_t*const p_info

Driver information

- input_capture_info_t::status

  Whether or not a capture is in progress.

- input_capture_info_t::variant

  Whether timer is 16 or 32 bits.

9.20.8.10 lastCaptureGet

```
ssp_err_t(* input_capture_api_t::lastCaptureGet) (input_capture_ctrl_t const
*const p_ctrl, input_capture_capture_t *const p_counter)
```

Detailed description

Gets the last captured timer/counter value Implemented as

- R_GPT_InputCaptureLastCaptureGet

Table 959:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized by open call.
p_counter	out	Pointer to location to store last captured counter value.

Parameter p_ctrl

Definition: input_capture_ctrl_tconst *const p_ctrl

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

Parameter p_counter

Definition: input_capture_capture_t*const p_counter

Capture data

- input_capture_capture_t::counter

  The value of the timer captured at the time of interrupt.

- input_capture_capture_t::overflows

  The number of counter overflows that occurred during this measurement.

9.20.8.11 close

```
ssp_err_t(* input_capture_api_t::close) (input_capture_ctrl_t *const p_ctrl)
```

Detailed description

Close the input capture operation. Allows driver to be reconfigured, and may reduce power consumption. Implemented as

- R_GPT_InputCaptureClose

Table 960:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block initialized by open call.

Parameter p_ctrl

Definition: `input_capture_ctrl_t*const p_ctrl`

Input capture control block. Allocate an instance specific control block to pass into the input capture API calls. Implemented asgpt_input_capture_instance_ctrl_t

9.20.8.12 versionGet

```
ssp_err_t(* input_capture_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets the version of this API and stores it in structure pointed to by p_version. Implemented as

- R_GPT_InputCaptureVersionGet

Table 961:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.20.8.13 input_capture_instance_t

input_capture_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- input_capture_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- input_capture_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- input_capture_api_t const * p_api

  Pointer to the API structure for this instance.

9.21 JPEG Decode Interface

Interface for JPEG decode functions.

9.21.1 Summary

The JPEG DECODE interface provides JPEG decoder functionality. It allows application to convert a JPEG image into bitmap data suitable for display frame buffer.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

JPEG DECODE Interface description: JPEG Decode Framework

9.21.2 Interface API

jpeg_decode_api_t

Function name	Description
.open	Initial configuration
.outputBufferSet	Assign output buffer to JPEG codec for storing output data.
.horizontalStrideSet	Configure the horizontal stride value.
.imageSubsampleSet	Configure the horizontal and vertical subsample settings.
.inputBufferSet	Assign input data buffer to JPEG codec.

Function name	Description
.linesDecodedGet	Return the number of lines decoded into the output buffer.
.imageSizeGet	Retrieve image size during decoding operation.
.statusGet	Retrieve current status of the JPEG codec module.
.close	Cancel an outstanding operation.
.versionGet	Get version and store it in provided pointer p_version.
.pixelFormatGet	Get the input pixel format.

9.21.3 Data structures

- jpeg_decode_callback_args_t
- jpeg_decode_cfg_t
- jpeg_decode_instance_t

9.21.4 Enumerations

- jpeg_decode_color_space_t
- jpeg_decode_data_format_t
- jpeg_decode_pixel_format_t
- jpeg_decode_status_t
- jpeg_decode_subsample_t
- jpeg_decode_count_enable_t
- jpeg_decode_resume_mode_t

9.21.5 Typedefs

- jpeg_decode_ctrl_t

9.21.6 Defines

- #define JPEG_DECODE_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Configuration for this module Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define JPEG_DECODE_API_VERSION_MINOR
  Initial value:(3U)

9.21.7 API Data

9.21.7.1 jpeg_decode_color_space_t

`jpeg_decode_color_space_t`

Detailed description

Image color space definitions

Enumerated values

Name	Description
JPEG_DECODE_COLOR_SPACE_YCBCR444	Color Space YCbCr 444.
JPEG_DECODE_COLOR_SPACE_YCBCR422	Color Space YCbCr 422.
JPEG_DECODE_COLOR_SPACE_YCBCR420	Color Space YCbCr 420.
JPEG_DECODE_COLOR_SPACE_YCBCR411	Color Space YCbCr 411.

9.21.7.2 jpeg_decode_data_format_t

`jpeg_decode_data_format_t`

Detailed description

Multi-byte Data Format

Enumerated values

Name	Description
JPEG_DECODE_DATA_FORMAT_NORMAL	(1)(2)(3)(4)(5)(6)(7)(8) Normal byte order
JPEG_DECODE_DATA_FORMAT_BYTE_SWAP	(2)(1)(4)(3)(6)(5)(8)(7) Byte Swap
JPEG_DECODE_DATA_FORMAT_WORD_SWAP	(3)(4)(1)(2)(7)(8)(5)(6) Word Swap
JPEG_DECODE_DATA_FORMAT_WORD_BYTE_SWAP	(4)(3)(2)(1)(8)(7)(6)(5) Word-Byte Swap
JPEG_DECODE_DATA_FORMAT_LONGWORD_SWAP	(5)(6)(7)(8)(1)(2)(3)(4) Longword Swap

Name	Description
JPEG_DECODE_DATA_FORMAT_LONGWORD_BYTE_SWAP	(6)(5)(8)(7)(2)(1)(4)(3) Longword Byte Swap
JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_SWAP	(7)(8)(5)(6)(3)(4)(1)(2) Longword Word Swap
JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_BYTE_SWAP	(8)(7)(6)(5)(4)(3)(2)(1) Longword Word Byte Swap
JPEG_DECODE_DATA_FORMAT_MAX	

9.21.7.3 jpeg_decode_pixel_format_t

`jpeg_decode_pixel_format_t`

Detailed description

Pixel Data Format

Enumerated values

Name	Description
JPEG_DECODE_PIXEL_FORMAT_ARGB8888	Pixel Data ARGB8888 format.
JPEG_DECODE_PIXEL_FORMAT_RGB565	Pixel Data RGB565 format.

9.21.7.4 jpeg_decode_status_t

`jpeg_decode_status_t`

Detailed description

JPEG HLD driver internal status information. The driver can simultaneously be in more than any one status at the same time. Parse the status bit-fields using the definitions in this enum to determine driver status

Enumerated values

Name	Description
JPEG_DECODE_STATUS_FREE	JPEG codec module is not yet open.
JPEG_DECODE_STATUS_IDLE	JPEG Codec module is open, and is not operational.
JPEG_DECODE_STATUS_RUNNING	JPEG Codec is running.
JPEG_DECODE_STATUS_DONE	JPEG Codec has successfully finished the operation.

Name	Description
JPEG_DECODE_STATUS_INPUT_PAUSE	JPEG Codec paused waiting for more input data.
JPEG_DECODE_STATUS_OUTPUT_PAUSE	JPEG Codec paused after decoded the number of lines specified by user.
JPEG_DECODE_STATUS_IMAGE_SIZE_READY	JPEG decoding operation obtained image size, and paused.
JPEG_DECODE_STATUS_ERROR	JPEG Codec module encountered an error.
JPEG_DECODE_STATUS_HEADER_PROCESSING	JPEG Codec module is reading the JPEG header information.

9.21.7.5 jpeg_decode_subsample_t

`jpeg_decode_subsample_t`

Detailed description

Data type for horizontal and vertical subsample settings. This setting applies only to the decoding operation.

Enumerated values

Name	Description
JPEG_DECODE_OUTPUT_NO_SUBSAMPLE	No subsample. The image is decoded with no reduction in size.
JPEG_DECODE_OUTPUT_SUBSAMPLE_HALF	The output image size is reduced by half.
JPEG_DECODE_OUTPUT_SUBSAMPLE_ONE_QUARTER	The output image size is reduced to one-quarter.
JPEG_DECODE_OUTPUT_SUBSAMPLE_ONE_EIGHTH	The output image size is reduced to one-eighth.

9.21.7.6 jpeg_decode_count_enable_t

`jpeg_decode_count_enable_t`

Detailed description

Data type for decoding count mode enable.

Enumerated values

Name	Description
JPEG_DECODE_COUNT_DISABLE	Count mode disable.
JPEG_DECODE_COUNT_ENABLE	Count mode enable.

9.21.7.7 jpeg_decode_resume_mode_t

`jpeg_decode_resume_mode_t`

Detailed description

Data type for decoding count mode enable.

Enumerated values

Name	Description
JPEG_DECODE_COUNT_MODE_ADDRESS_CONTINUE	The data buffer address will not be initialized when resuming image data lines.
JPEG_DECODE_COUNT_MODE_ADDRESS_REINITIALIZE	The data buffer address will be initialized when resuming image data lines.

9.21.7.8 jpeg_decode_ctrl_t

```
typedef void jpeg_decode_ctrl_t
```

Detailed description

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented as

- jpeg_decode_instance_ctrl_t

9.21.8 API Structures

9.21.8.1 jpeg_decode_callback_args_t

jpeg_decode_callback_args_t

Detailed description

Callback status structure

Variables

- jpeg_decode_status_t status

  JPEG status.

- void const * p_context

  Pointer to user-provided context.

9.21.8.2 jpeg_decode_cfg_t

jpeg_decode_cfg_t

Detailed description

User configuration structure, used in open function.

Variables

- jpeg_decode_color_space_t color_space

  Color space.

- jpeg_decode_data_format_t input_data_format

  Input data stream byte order.

- jpeg_decode_data_format_t output_data_format

  Output data stream byte order.

- jpeg_decode_pixel_format_t pixel_format

  Pixel format.

- uint8_t alpha_value

  Alpha value to be applied to decoded pixel data. Only valid for ARGB888 format.

- uint8_t jdti_ipl

  Data transfer interrupt priority.

- uint8_t jedi_ipl

  Decompression interrupt priority.

- void(* p_callback)(jpeg_decode_callback_args_t *p_args)

  User-supplied callback functions.

- void const * p_context

  Placeholder for user data. Passed to user callback in jpeg_decode_callback_args_t.

9.21.8.3 jpeg_decode_api_t

jpeg_decode_api_t

Detailed description

JPEG functions implemented at the HAL layer will follow this API.

9.21.8.4 open

```
ssp_err_t(* jpeg_decode_api_t::open) (jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration Implemented as

- R_JPEG_Decode_Open

NOTE: none

Table 962:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_cfg

Definition: `jpeg_decode_cfg_t const *const p_cfg`

User configuration structure, used in open function.

- `jpeg_decode_cfg_t`::`jpeg_decode_color_space_t`

  Color space.

  Enumerated as:

  – JPEG_DECODE_COLOR_SPACE_YCBCR444

  – JPEG_DECODE_COLOR_SPACE_YCBCR422

  – JPEG_DECODE_COLOR_SPACE_YCBCR420

  – JPEG_DECODE_COLOR_SPACE_YCBCR411

- `jpeg_decode_cfg_t`::`jpeg_decode_data_format_t`

  Input data stream byte order.

  Enumerated as:

  – JPEG_DECODE_DATA_FORMAT_NORMAL

  - JPEG_DECODE_DATA_FORMAT_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_WORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_WORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_MAX
- jpeg_decode_cfg_t::jpeg_decode_data_format_t

  Output data stream byte order.

  Enumerated as:
  - JPEG_DECODE_DATA_FORMAT_NORMAL
  - JPEG_DECODE_DATA_FORMAT_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_WORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_WORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_SWAP
  - JPEG_DECODE_DATA_FORMAT_LONGWORD_WORD_BYTE_SWAP
  - JPEG_DECODE_DATA_FORMAT_MAX
- jpeg_decode_cfg_t::jpeg_decode_pixel_format_t

  Pixel format.

  Enumerated as:
  - JPEG_DECODE_PIXEL_FORMAT_ARGB8888
  - JPEG_DECODE_PIXEL_FORMAT_RGB565
- jpeg_decode_cfg_t::alpha_value

  Alpha value to be applied to decoded pixel data. Only valid for ARGB888 format.
- jpeg_decode_cfg_t::jdti_ipl

  Data transfer interrupt priority.
- jpeg_decode_cfg_t::jedi_ipl

  Decompression interrupt priority.

- jpeg_decode_cfg_t::p_callback

  User-supplied callback functions.

- jpeg_decode_cfg_t::p_context

  Placeholder for user data. Passed to user callback in jpeg_decode_callback_args_t.

9.21.8.5 outputBufferSet

```
ssp_err_t(* jpeg_decode_api_t::outputBufferSet) (jpeg_decode_ctrl_t *const
p_ctrl, void *p_buffer, uint32_t buffer_size)
```

Detailed description

Assign output buffer to JPEG codec for storing output data. Implemented as

- R_JPEG_Decode_OutputBufferSet

NOTE: The JPEG codec module must have been opened properly.

NOTE: The buffer starting address must be 8-byte aligned. For the decoding process, the HLD driver automatically computes the number of lines of the image to decoded so the output data fits into the given space. If the supplied output buffer is not able to hold the entire frame, the application should call the Output Full Callback function so it can be notified when additional buffer space is needed.

Table 963:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_buffer	in	Pointer to the output buffer space
buffer_size	in	Size of the output buffer

Parameter p_ctrl

Definition: jpeg_decode_ctrl_t*const p_ctrl

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_buffer

const

Parameter buffer_size

uint32_t

9.21.8.6 horizontalStrideSet

```
ssp_err_t(* jpeg_decode_api_t::horizontalStrideSet) (jpeg_decode_ctrl_t *const
p_ctrl, uint32_t horizontal_stride)
```

Detailed description

Configure the horizontal stride value. Implemented as

- R_JPEG_Decode_HorizontalStrideSet

NOTE: The JPEG codec module must have been opened properly.

Table 964:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
horizontal_stride	in	Horizontal stride value to be used for the decoded image data.
buffer_size	in	Size of the output buffer

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter horizontal_stride

`uint32_t`

Parameter buffer_size

9.21.8.7 imageSubsampleSet

```
ssp_err_t(* jpeg_decode_api_t::imageSubsampleSet) (jpeg_decode_ctrl_t *const
p_ctrl, jpeg_decode_subsample_t horizontal_subsample, jpeg_decode_subsample_t
vertical_subsample)
```

Detailed description

Configure the horizontal and vertical subsample settings. Implemented as

- R_JPEG_Decode_ImageSubsampleSet

NOTE: The JPEG codec module must have been opened properly.

Table 965:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
horizontal_subsample	in	Horizontal subsample value

Table 965:Parameters (Continued)

Name	Direction	Description
vertical_subsample	in	Vertical subsample value

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter horizontal_subsample

Definition: `jpeg_decode_subsample_thorizontal_subsample`

Data type for horizontal and vertical subsample settings. This setting applies only to the decoding operation.

Parameter vertical_subsample

Definition: `jpeg_decode_subsample_tvertical_subsample`

Data type for horizontal and vertical subsample settings. This setting applies only to the decoding operation.

9.21.8.8 inputBufferSet

```
ssp_err_t(* jpeg_decode_api_t::inputBufferSet) (jpeg_decode_ctrl_t *const p_ctrl,
void *p_buffer, uint32_t buffer_size)
```

Detailed description

Assign input data buffer to JPEG codec. Implemented as

- R_JPEG_Decode_InputBufferSet

NOTE: the JPEG codec module must have been opened properly.

NOTE: The buffer starting address must be 8-byte aligned.

Table 966:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_buffer	in	Pointer to the input buffer space
buffer_size	in	Size of the input buffer

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_buffer

`const`

Parameter buffer_size

`uint32_t`

9.21.8.9 linesDecodedGet

```
ssp_err_t(* jpeg_decode_api_t::linesDecodedGet) (jpeg_decode_ctrl_t *const
p_ctrl, uint32_t *const p_lines)
```

Detailed description

Return the number of lines decoded into the output buffer. Implemented as

- R_JPEG_Decode_LinesDecodedGet

NOTE: the JPEG codec module must have been opened properly.

Table 967:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_lines	out	Number of lines decoded

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_lines

`uint32_t`

9.21.8.10 imageSizeGet

```
ssp_err_t(* jpeg_decode_api_t::imageSizeGet) (jpeg_decode_ctrl_t *const p_ctrl,
uint16_t *p_horizontal_size, uint16_t *p_vertical_size)
```

Detailed description

Retrieve image size during decoding operation. Implemented as

- R_JPEG_Decode_ImageSizeGet

NOTE: the JPEG codec module must have been opened properly.

NOTE: If the encoding or the decoding operation is finished without errors, the HLD driver automatically closes the device. In this case, application does not need to explicitly close the JPEG device.

Table 968:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_horizontal_size	out	Image horizontal size, in number of pixels.
p_vertical_size	out	Image vertical size, in number of pixels.

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_horizontal_size

`uint16_t`

Parameter p_vertical_size

`uint16_t`

9.21.8.11 statusGet

```
ssp_err_t(* jpeg_decode_api_t::statusGet) (jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_status_t *const p_status)
```

Detailed description

Retrieve current status of the JPEG codec module. Implemented as

- R_JPEG_Decode_StatusGet

NOTE: the JPEG codec module must have been opened properly.

Table 969:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_status	out	JPEG module status

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_status

Definition: `jpeg_decode_status_t*const p_status`

JPEG HLD driver internal status information. The driver can simultaneously be in more than any one status at the same time. Parse the status bit-fields using the definitions in this enum to determine driver status

9.21.8.12 close

`ssp_err_t(* jpeg_decode_api_t::close) (jpeg_decode_ctrl_t *const p_ctrl)`

Detailed description

Cancel an outstanding operation. Implemented as

- R_JPEG_Decode_Close

NOTE: the JPEG codec module must have been opened properly.

NOTE: If the encoding or the decoding operation is finished without errors, the HLD driver automatically closes the device. In this case, application does not need to explicitly close the JPEG device.

Table 970:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in jpeg_decode_api_t::Open call.

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

9.21.8.13 versionGet

`ssp_err_t(* jpeg_decode_api_t::versionGet) (ssp_version_t *p_version)`

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_JPEG_Decode_VersionGet

Table 971:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.21.8.14 pixelFormatGet

```
ssp_err_t(* jpeg_decode_api_t::pixelFormatGet) (jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_color_space_t *const p_color_space)
```

Detailed description

Get the input pixel format. Implemented as

- R_JPEG_Decode_PixelFormatGet

NOTE: the JPEG codec module must have been opened properly.

Table 972:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call.
p_color_space	out	JPEG input format.

Parameter p_ctrl

Definition: `jpeg_decode_ctrl_t*const p_ctrl`

JPEG decode control block. Allocate an instance specific control block to pass into the JPEG decode API calls. Implemented asjpeg_decode_instance_ctrl_t

Parameter p_color_space

Definition: `jpeg_decode_color_space_t*const p_color_space`

Image color space definitions

9.21.8.15 jpeg_decode_instance_t

jpeg_decode_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- jpeg_decode_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- jpeg_decode_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- jpeg_decode_api_t const * p_api

  Pointer to the API structure for this instance.

9.22 Key Matrix Interface

Interface for key matrix functions.

9.22.1 Summary

The KEYMATRIX interface provides standard KeyMatrix functionality including event generation on a rising or falling edge for one or more channels at the same time. The generated event indicates all channels that are active in that instant via a bit mask. This allows the interface to be used with a matrix configuration or a one-to-one hardware implementation that is triggered on either a rising or a falling edge.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Key Matrix Interface description: Key Matrix Driver

9.22.2 Interface API

keymatrix_api_t

Function name	Description
.open	Initial configuration.
.enable	Enable Key interrupt
.disable	Disable Key interrupt.
.triggerSet	Set trigger for Key interrupt.
.close	Allow driver to be reconfigured. May reduce power consumption.
.versionGet	Get version and store it in provided pointer p_version.

9.22.3 Data structures

- keymatrix_callback_args_t
- keymatrix_cfg_t
- keymatrix_instance_t

9.22.4 Enumerations

- keymatrix_trigger_t

9.22.5 Typedefs

- keymatrix_channels_t
- keymatrix_ctrl_t

9.22.6 Defines

- #define KEYMATRIX_API_VERSION_MAJOR

  Initial value:`(1U)`

  KEY MATRIX API version number (Major)

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define KEYMATRIX_API_VERSION_MINOR

  Initial value:`(1U)`

  KEY MATRIX API version number (Minor)

9.22.7 API Data

9.22.7.1 keymatrix_trigger_t

`keymatrix_trigger_t`

Detailed description

Trigger type: rising edge, falling edge

Enumerated values

Name	Description
KEYMATRIX_TRIG_FALLING	Falling edge trigger.
KEYMATRIX_TRIG_RISING	Rising edge trigger.

9.22.7.2 keymatrix_channels_t

`typedef uint32_t keymatrix_channels_t`

Detailed description

Channel definition. This is a bit mask with each bit from 0-7 representing channels 0-7 respectively.

9.22.7.3 keymatrix_ctrl_t

```
typedef void keymatrix_ctrl_t
```

Detailed description

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented as

- kint_instance_ctrl_t

9.22.8 API Structures

9.22.8.1 keymatrix_callback_args_t

keymatrix_callback_args_t

Detailed description

Callback function parameter data

Variables

- void const * p_context

  Holder for user data. Set in keymatrix_api_t::open function in keymatrix_cfg_t.

- keymatrix_channels_t channels

  Bit vector representing the physical hardware channel(s) that caused the interrupt. The bit vector is used for compatibility with matrix designs where more than one input will be active at once. Not all HAL drivers support matrix mode. See r_kint.h for details.

9.22.8.2 keymatrix_cfg_t

keymatrix_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- keymatrix_channels_t channels

  Key Input channel(s). Bit mask of channels to open.

- keymatrix_trigger_t trigger

  Key Input trigger setting.

- bool autostart

  Start operation and enable interrupts during open().

- void(* p_callback)(keymatrix_callback_args_t *p_args)

  Callback for key interrupt ISR.

- void const * p_context

  Holder for user data. Passed to callback in keymatrix_user_cb_data_t.

- void const * p_extend

  Extension parameter for hardware specific settings.

- uint8_t irq_ipl

  Interrupt priority level.

9.22.8.3 keymatrix_api_t

keymatrix_api_t

Detailed description

Key Matrix driver structure. Key Matrix functions implemented at the HAL layer will use this API.

9.22.8.4 open

```
ssp_err_t(* keymatrix_api_t::open) (keymatrix_ctrl_t *const p_ctrl,
keymatrix_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_KINT_KEYMATRIX_Open

Table 973:Parameters

Name	Direction	Description
p_ctrl	out	Pointer to control block. Must be declared by user. Value set in this function.
p_cfg	in	Pointer to configuration structure. All elements of the structure must be set by user.

Parameter p_ctrl

Definition: `keymatrix_ctrl_t*const p_ctrl`

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented askint_instance_ctrl_t

Parameter p_cfg

Definition: `keymatrix_cfg_t const *const p_cfg`

User configuration structure, used in open function

- keymatrix_cfg_t::keymatrix_channels_t

  Key Input channel(s). Bit mask of channels to open.
- keymatrix_cfg_t::keymatrix_trigger_t

  Key Input trigger setting.

  Enumerated as:

  – KEYMATRIX_TRIG_FALLING

  – KEYMATRIX_TRIG_RISING
- keymatrix_cfg_t::autostart

  Start operation and enable interrupts during open().
- keymatrix_cfg_t::p_callback

  Callback for key interrupt ISR.
- keymatrix_cfg_t::p_context

  Holder for user data. Passed to callback in keymatrix_user_cb_data_t.
- keymatrix_cfg_t::p_extend

  Extension parameter for hardware specific settings.
- keymatrix_cfg_t::irq_ipl

  Interrupt priority level.

9.22.8.5 enable

```
ssp_err_t(* keymatrix_api_t::enable) (keymatrix_ctrl_t *const p_ctrl)
```

Detailed description

Enable Key interrupt Implemented as

- R_KINT_KEYMATRIX_Enable

Table 974:Parameters

Name	Direction	Description
p_ctrl	in	Control block pointer set in Open call for this Key interrupt.

Parameter p_ctrl

Definition: keymatrix_ctrl_t*const p_ctrl

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented askint_instance_ctrl_t

9.22.8.6 disable

```
ssp_err_t(* keymatrix_api_t::disable) (keymatrix_ctrl_t *const p_ctrl)
```

Detailed description

Disable Key interrupt. Implemented as

- R_KINT_KEYMATRIX_Disable

Table 975:Parameters

Name	Direction	Description
p_ctrl	in	Control block pointer set in Open call for this Key interrupt.

Parameter p_ctrl

Definition: `keymatrix_ctrl_t*const p_ctrl`

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented askint_instance_ctrl_t

9.22.8.7 triggerSet

```
ssp_err_t(* keymatrix_api_t::triggerSet) (keymatrix_ctrl_t *const p_ctrl,
keymatrix_trigger_t const trigger)
```

Detailed description

Set trigger for Key interrupt. Implemented as

- R_KINT_KEYMATRIX_TriggerSet

Table 976:Parameters

Name	Direction	Description
p_ctrl	in	Control block pointer set in Open call for this Key interrupt.
trigger	in	Trigger source for key interrupt; defined in enumeration of keymatrix_trigger_t.

Parameter p_ctrl

Definition: `keymatrix_ctrl_t*const p_ctrl`

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented askint_instance_ctrl_t

Parameter trigger

9.22.8.8 close

```
ssp_err_t(* keymatrix_api_t::close) (keymatrix_ctrl_t *const p_ctrl)
```

Detailed description

Allow driver to be reconfigured. May reduce power consumption. Implemented as

- R_KINT_KEYMATRIX_Close

Table 977:Parameters

Name	Direction	Description
p_ctrl	in	Control block pointer set in Open call for this Key interrupt.

Parameter p_ctrl

Definition: `keymatrix_ctrl_t*const p_ctrl`

Key matrix control block. Allocate an instance specific control block to pass into the key matrix API calls. Implemented askint_instance_ctrl_t

9.22.8.9 versionGet

```
ssp_err_t(* keymatrix_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_KINT_VersionGet

Table 978:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.22.8.10 keymatrix_instance_t

keymatrix_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- keymatrix_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- keymatrix_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- keymatrix_api_t const * p_api

  Pointer to the API structure for this instance.

9.23 Low Power Modes Interface

Interface for accessing low power modes.

9.23.1 Summary

This section defines the API for the LPM (Low Power Mode) Driver. The LPM Driver provides several functions for controlling power consumption including stopping modules, selecting the operating mode, configuring low power modes, and transitioning to low power modes. The LPM driver supports configuration of MCU operating modes and mcu low power modes using the LPM hardware peripheral. The LPM driver supports operating modes low-voltage, low-speed, middle-speed, high-speed, and suboscillator mode. The LPM driver supports low power modes deep standby, standby, sleep, and snooze. The LPM driver supports reducing power consumption when in deep standby mode via internal power supply control and reseting the states of IO ports. The LPM driver supports disabling and enabling of the MCU's other hardware peripherals. ***NOTE:*** *Not all operating modes are available on all MCUs. Not all low power modes are available on all MCUs.*Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

LPM Interface description: Low Power Modes Driver

9.23.2 Interface API

lpm_api_t

Function name	Description
.init	Open the LPM driver module Initialized the LPM block according to the passed in config structure
.mstpcrSet	Set the value of all the Module Stop Control Registers
.mstpcrGet	Get the values of all the Module Stop Control Registers

Function name	Description
.moduleStop	Stop a module
.moduleStart	Run a module
.operatingPowerModeSet	Set the operating power mode
.snoozeEnable	Configure and enable snooze mode
.snoozeDisable	Disable snooze mode
.lowPowerCfg	Configure a low power mode
.wupenSet	Set the value of the Wake Up Interrupt Enable Register WUPEN
.wupenGet	Get the value of the Wake Up Interrupt Enable Register WUPEN
.deepStandbyCancelRequestEnable	Enable a Deep Standby Cancel Request
.deepStandbyCancelRequestDisable	Disable a Deep Standby Cancel Request
.lowPowerModeEnter	Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode.
.versionGet	Get the driver version based on compile time macros.

9.23.3 Data structures

- lpm_cfg_t
- lpm_instance_t

9.23.4 Enumerations

- lpm_cancel_request_edge_t
- lpm_deep_standby_t
- lpm_low_power_mode_t
- lpm_output_port_enable_t
- lpm_dpsby_t
- lpm_io_port_t
- lpm_power_supply_t

- lpm_power_save_memory_t
- lpm_code_flash_t
- lpm_snooze_request_t
- lpm_snooze_end_t
- lpm_snooze_rxd0_t
- lpm_snooze_dtc_t
- lpm_operating_power_t
- lpm_subosc_t
- lpm_mstp_t

9.23.5 Defines

- #define LPM_API_VERSION_MAJOR

  Initial value:`(2U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define LPM_API_VERSION_MINOR

  Initial value:`(3U)`

9.23.6 API Data

9.23.6.1 lpm_cancel_request_edge_t

`lpm_cancel_request_edge_t`

Detailed description

Deep Standby Interrupt Edge

Enumerated values

Name	Description
LPM_CANCEL_REQUEST_EDGE_FALLING	A cancel request is generated at a falling edge.
LPM_CANCEL_REQUEST_EDGE_RISING	A cancel request is generated at a rising edge.

9.23.6.2 lpm_deep_standby_t

`lpm_deep_standby_t`

Detailed description

Deep Standby pins and signals

Enumerated values

Name	Description
LPM_DEEP_STANDBY_IRQ0_DS	IRQ0-DS Pin.
LPM_DEEP_STANDBY_IRQ1_DS	IRQ1-DS Pin.
LPM_DEEP_STANDBY_IRQ2_DS	IRQ2-DS Pin.
LPM_DEEP_STANDBY_IRQ3_DS	IRQ3-DS Pin.
LPM_DEEP_STANDBY_IRQ4_DS	IRQ4-DS Pin.
LPM_DEEP_STANDBY_IRQ5_DS	IRQ5-DS Pin.
LPM_DEEP_STANDBY_IRQ6_DS	IRQ6-DS Pin.
LPM_DEEP_STANDBY_IRQ7_DS	IRQ7-DS Pin.
LPM_DEEP_STANDBY_IRQ8_DS	IRQ8-DS Pin.
LPM_DEEP_STANDBY_IRQ9_DS	IRQ9-DS Pin.
LPM_DEEP_STANDBY_IRQ10_DS	IRQ10-DS Pin.
LPM_DEEP_STANDBY_IRQ11_DS	IRQ11-DS Pin.
LPM_DEEP_STANDBY_IRQ12_DS	IRQ12-DS Pin.
LPM_DEEP_STANDBY_IRQ13_DS	IRQ13-DS Pin.
LPM_DEEP_STANDBY_IRQ14_DS	IRQ14-DS Pin.
LPM_DEEP_STANDBY_IRQ15_DS	IRQ15-DS Pin.
LPM_DEEP_STANDBY_LVD1	LVD1.
LPM_DEEP_STANDBY_LVD2	LVD2.
LPM_DEEP_STANDBY_RTC_INTERVAL	RTC Interval Interrupt.
LPM_DEEP_STANDBY_RTC_ALARM	RTC Alarm Interrupt.
LPM_DEEP_STANDBY_NMI	NMI Pin.
LPM_DEEP_STANDBY_USBFS	USBFS Suspend/Resume.

Name	Description
LPM_DEEP_STANDBY_USBHS	USBHS Suspend/Resume.
LPM_DEEP_STANDBY_AGT1	AGT1 Underflow.

9.23.6.3 lpm_low_power_mode_t

`lpm_low_power_mode_t`

Detailed description

Low power modes

Enumerated values

Name	Description
LPM_LOW_POWER_MODE_SLEEP	Sleep mode (ARM Cortex sleep mode)
LPM_LOW_POWER_MODE_STANDBY	Software Standby mode.
LPM_LOW_POWER_MODE_DEEP	Deep Software Standby mode.

9.23.6.4 lpm_output_port_enable_t

`lpm_output_port_enable_t`

Detailed description

Output port enable

Enumerated values

Name	Description
LPM_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPM_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

9.23.6.5 lpm_dpsby_t

`lpm_dpsby_t`

Detailed description

Select between Software Standby and Deep Software Standby

Enumerated values

Name	Description
LPM_DPSBY_SOFTWARE_STANDBY	Sleep mode (SBYCR.SSBY = 0) / Software Standby mode (SBYCR.SSBY = 1)
LPM_DPSBY_DEEP_SOFTWARE_STANDBY	Sleep mode (SBYCR.SSBY = 0) / Deep Software Standby mode (SBYCR.SSBY = 1)

9.23.6.6 lpm_io_port_t

`lpm_io_port_t`

Detailed description

I/O port state after Deep Software Standby mode

Enumerated values

Name	Description
LPM_IO_PORT_RESET	When the Deep Software Standby mode is canceled, the I/O ports are in the reset state.
LPM_IO_PORT_NO_CHANGE	When the Deep Software Standby mode is canceled, the I/O ports are in the same state as in the Deep Software Standby mode

9.23.6.7 lpm_power_supply_t

`lpm_power_supply_t`

Detailed description

Power supply control

Enumerated values

Name	Description
LPM_POWER_SUPPLY_DEEPCUT0	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is supplied in deep software standby mode
LPM_POWER_SUPPLY_DEEPCUT1	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode
LPM_POWER_SUPPLY_DEEPCUT3	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode. In addition, LVD is disabled and the low power function in a poweron reset circuit is enabled

9.23.6.8 lpm_power_save_memory_t

`lpm_power_save_memory_t`

Detailed description

Power save memory control

Enumerated values

Name	Description
LPM_POWER_SAVE_MEMORY_ALL_RAM	All RAM on in Software Standby mode.
LPM_POWER_SAVE_MEMORY_48KB	48KB RAM (2000 0000h to 2000 BFFFh) on in Software Standby mode

9.23.6.9 lpm_code_flash_t

`lpm_code_flash_t`

Detailed description

Code flash operation mode

Enumerated values

Name	Description
LPM_CODE_FLASH_OPERATES	Code flash memory operates.

Name	Description
LPM_CODE_FLASH_STOPS	Code flash memory stops.

9.23.6.10 lpm_snooze_request_t

`lpm_snooze_request_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPM_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPM_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPM_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.

Name	Description
LPM_SNOOZE_REQUEST_KR	Enable KR snooze request.
LPM_SNOOZE_REQUEST_COMPARATOR_OC0	Enable Comparator-OC0 snooze request.
LPM_SNOOZE_REQUEST_COMPARATOR_LP	Enable Comparator-LP snooze request.
LPM_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPM_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPM_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPM_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPM_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.

9.23.6.11 lpm_snooze_end_t

`lpm_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPM_SNOOZE_END_VIA_WUPEN	Transition from Snooze to Normal mode directly.
LPM_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPM_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPM_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPM_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPM_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC hannel 0 compare mismatch.
LPM_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPM_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.
LPM_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address unmatch.

9.23.6.12 lpm_snooze_rxd0_t

`lpm_snooze_rxd0_t`

Detailed description

RXD0 Snooze Request Enable

Enumerated values

Name	Description
LPM_SNOOZE_RXD0_FALLING_EDGE_IGNORE	Ignore RXD0 falling edge in Software Standby mode.
LPM_SNOOZE_RXD0_FALLING_EDGE_DETECT	Detect RXD0 falling edge in Software Standby mode as a request to transit to Snooze mode. Do not set to 1 other than in asynchronous mode.

9.23.6.13 lpm_snooze_dtc_t

`lpm_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPM_SNOOZE_DTC_DISABLE	Disable DTC operation.
LPM_SNOOZE_DTC_ENABLE	Enable DTC operation.

9.23.6.14 lpm_operating_power_t

`lpm_operating_power_t`

Detailed description

Operating power mode

Enumerated values

Name	Description
LPM_OPERATING_POWER_HIGH_SPEED_MODE	High speed mode.
LPM_OPERATING_POWER_MIDDLE_SPEED_MODE	Middle speed mode.

Name	Description
LPM_OPERATING_POWER_LOW_VOLTAGE_MODE	Low voltage mode.
LPM_OPERATING_POWER_LOW_SPEED_MODE	Low speed mode.

9.23.6.15 lpm_subosc_t

`lpm_subosc_t`

Detailed description

Subosc speed mode select

Enumerated values

Name	Description
LPM_SUBOSC_OTHER	Other than Subosc-speed mode.
LPM_SUBOSC_SELECT	Subosc-speed mode.

9.23.6.16 lpm_mstp_t

`lpm_mstp_t`

Detailed description

Module stop bits

Enumerated values

Name	Description
LPM_MSTP_RAM0	RAM0.
LPM_MSTP_RAM1	RAM1.
LPM_MSTP_HSRAM	HSRAM.
LPM_MSTP_ECCRAM	ECCRAM.
LPM_MSTP_STANDBYRAM	STANDBYRAM.
LPM_MSTP_DMAC_DTC	DMAC_DTC.
LPM_MSTP_RCAN1	RCAN1.

Name	Description
LPM_MSTP_RCAN0	RCAN0.
LPM_MSTP_IRDA	IRDA.
LPM_MSTP_QSPI	QSPI.
LPM_MSTP_I2C2	I2C2.
LPM_MSTP_I2C1	I2C1.
LPM_MSTP_I2C0	I2C0.
LPM_MSTP_USBFS	USBFS.
LPM_MSTP_USBHS	USBHS.
LPM_MSTP_EPTPC_PTPEDMAC	EPTPC_PTPEDMAC.
LPM_MSTP_ETHERC1_EDMAC1	ETHERC1_EDMAC1.
LPM_MSTP_ETHERC0_EDMAC0	ETHERC0_EDMAC0.
LPM_MSTP_RSPI1	RSPI1.
LPM_MSTP_RSPI0	RSPI0.
LPM_MSTP_SCI9	SCI9.
LPM_MSTP_SCI8	SCI8.
LPM_MSTP_SCI7	SCI7.
LPM_MSTP_SCI6	SCI6.
LPM_MSTP_SCI5	SCI5.
LPM_MSTP_SCI4	SCI4.
LPM_MSTP_SCI3	SCI3.
LPM_MSTP_SCI2	SCI2.
LPM_MSTP_SCI1	SCI1.
LPM_MSTP_SCI0	SCI0.
LPM_MSTP_CAC	CAC.

Name	Description
LPM_MSTP_CRC	CRC.
LPM_MSTP_PDC	PDC.
LPM_MSTP_CTSU	CTSU.
LPM_MSTP_GLCDC	GLCDC.
LPM_MSTP_JPEG	JPEG.
LPM_MSTP_2GD	2GD
LPM_MSTP_SSI1	SSI1.
LPM_MSTP_SSI0	SSI0.
LPM_MSTP_SRC	SRC.
LPM_MSTP_SDH_MMC1	SDH_MMC1.
LPM_MSTP_SDH_MMC0	SDH_MMC0.
LPM_MSTP_DOC	DOC.
LPM_MSTP_ELC	ELC.
LPM_MSTP_TSIP	TSIP.
LPM_MSTP_AGT1	AGT1.
LPM_MSTP_AGT0	AGT0.
LPM_MSTP_GPT_CH7_0	GPT_CH7_0.
LPM_MSTP_GPT_CH13_8	GPT_CH13_8.
LPM_MSTP_PGI	PGI.
LPM_MSTP_S12AD_UNIT1	S12AD_UNIT1.
LPM_MSTP_S12AD_UNIT0	S12AD_UNIT0.
LPM_MSTP_D_A0	D_A0.
LPM_MSTP_TEMP_SENSE	TEMP_SENSE.
LPM_MSTP_COMP_OC5	COMP_OC5.

Name	Description
LPM_MSTP_COMP_OC4	COMP_OC4.
LPM_MSTP_COMP_OC3	COMP_OC3.
LPM_MSTP_COMP_OC2	COMP_OC2.
LPM_MSTP_COMP_OC1	COMP_OC1.
LPM_MSTP_COMP_OC0	COMP_OC0.
LPM_MSTP_COMP_LP	COMP_LP.
LPM_MSTP_COMP_RD	COMP_RD.
LPM_MSTP_COMP_OPAMP	COMP_OPAMP.

9.23.7 API Structures

9.23.7.1 lpm_cfg_t

lpm_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- lpm_operating_power_t operating_power

  Operating power mode.

- lpm_subosc_t sub_oscillator

  Sub oscillator.

- lpm_code_flash_t code_flash

  Enable the code flash.

9.23.7.2 lpm_api_t

lpm_api_t

Detailed description

lpm driver structure. General lpm functions implemented at the HAL layer will follow this API.

9.23.7.3 init

```
ssp_err_t(* lpm_api_t::init) (lpm_cfg_t const *const p_cfg)
```

Detailed description

Open the LPM driver module Initialized the LPM block according to the passed in config structure

Table 979:Parameters

Name	Direction	Description
p_cfg	in	Pointer to a configuration structure

Parameter p_cfg

Definition: `lpm_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `lpm_cfg_t`::`lpm_operating_power_t`

  Operating power mode.

  Enumerated as:

  – LPM_OPERATING_POWER_HIGH_SPEED_MODE

  – LPM_OPERATING_POWER_MIDDLE_SPEED_MODE

  – LPM_OPERATING_POWER_LOW_VOLTAGE_MODE

  – LPM_OPERATING_POWER_LOW_SPEED_MODE

- `lpm_cfg_t`::`lpm_subosc_t`

  Sub oscillator.

  Enumerated as:

  – LPM_SUBOSC_OTHER

  – LPM_SUBOSC_SELECT

- `lpm_cfg_t`::`lpm_code_flash_t`

  Enable the code flash.

  Enumerated as:

  – LPM_CODE_FLASH_OPERATES

  – LPM_CODE_FLASH_STOPS

9.23.7.4 mstpcrSet

```
ssp_err_t(* lpm_api_t::mstpcrSet) (uint32_t mstpcra_value, uint32_t
mstpcrb_value, uint32_t mstpcrc_value, uint32_t mstpcrd_value)
```

Detailed description

Set the value of all the Module Stop Control Registers

Table 980:Parameters

Name	Direction	Description
mstpcra_value	in	The value for MSTPCRA
mstpcrb_value	in	The value for MSTPCRB
mstpcrc_value	in	The value for MSTPCRC
mstpcrd_value	in	The value for MSTPCRD

Parameter mstpcra_value

`uint32_t`

Parameter mstpcrb_value

`uint32_t`

Parameter mstpcrc_value

`uint32_t`

Parameter mstpcrd_value

`uint32_t`

9.23.7.5 mstpcrGet

```
ssp_err_t(* lpm_api_t::mstpcrGet) (uint32_t *mstpcra_value, uint32_t
*mstpcrb_value, uint32_t *mstpcrc_value, uint32_t *mstpcrd_value)
```

Detailed description

Get the values of all the Module Stop Control Registers

Table 981:Parameters

Name	Direction	Description
mstpcra_value	inout	The value from MSTPCRA
mstpcrb_value	inout	The value from MSTPCRB
mstpcrc_value	inout	The value from MSTPCRC
mstpcrd_value	inout	The value from MSTPCRD

Parameter mstpcra_value

`uint32_t`

Parameter mstpcrb_value

uint32_t

Parameter mstpcrc_value

uint32_t

Parameter mstpcrd_value

uint32_t

9.23.7.6 moduleStop

```
ssp_err_t(* lpm_api_t::moduleStop) (lpm_mstp_t module)
```

Detailed description

Stop a module

Table 982:Parameters

Name	Direction	Description
module	in	The module to set the state of

Parameter module

Definition: lpm_mstp_tmodule

Module stop bits

9.23.7.7 moduleStart

```
ssp_err_t(* lpm_api_t::moduleStart) (lpm_mstp_t module)
```

Detailed description

Run a module

Table 983:Parameters

Name	Direction	Description
module	in	The module to set the state of

Parameter module

Definition: lpm_mstp_tmodule

Module stop bits

9.23.7.8 operatingPowerModeSet

```
ssp_err_t(* lpm_api_t::operatingPowerModeSet) (lpm_operating_power_t power_mode,
lpm_subosc_t subosc)
```

Detailed description

Set the operating power mode

Table 984:Parameters

Name	Direction	Description
power_mode	in	The power mode to set the chip to
subosc	in	Select the sub oscillator or other oscillator

Parameter power_mode

Definition: `lpm_operating_power_tpower_mode`

Operating power mode

Parameter subosc

9.23.7.9 snoozeEnable

```
ssp_err_t(* lpm_api_t::snoozeEnable) (lpm_snooze_rxd0_t rdx0_mode,
lpm_snooze_dtc_t dtc_mode, lpm_snooze_request_t requests, lpm_snooze_end_t
triggers)
```

Detailed description

Configure and enable snooze mode

Table 985:Parameters

Name	Direction	Description
rdx0_mode	in	Enter Snooze mode on the falling edge of RXD0 (only in asynchronous mode)
dtc_mode	in	Use DTC and RAM while in Snooze Mode
requests	in	Specify the events that cause an entry into Snooze mode
triggers	in	Specify the events that cause an exit from snooze mode

Parameter rdx0_mode

Definition: `lpm_snooze_rxd0_trdx0_mode`

RXD0 Snooze Request Enable

Parameter dtc_mode

Definition: lpm_snooze_dtc_tdtc_mode

DTC Enable in Snooze Mode

Parameter requests

Definition: lpm_snooze_request_trequests

Snooze end control

Parameter triggers

Definition: lpm_snooze_end_ttriggers

Snooze end control

9.23.7.10 snoozeDisable

```
ssp_err_t(* lpm_api_t::snoozeDisable) (void)
```

Detailed description

Disable snooze mode

Table 986:Return values

Name	Description
SSP_SUCCESS	Snooze mode successfully disabled

Parameter SSP_SUCCESS

9.23.7.11 lowPowerCfg

```
ssp_err_t(* lpm_api_t::lowPowerCfg) (lpm_low_power_mode_t power_mode,
lpm_output_port_enable_t output_port_enable, lpm_power_supply_t power_supply,
lpm_io_port_t io_port_state)
```

Detailed description

Configure a low power mode

Table 987:Parameters

Name	Direction	Description
low_power_mode	in	Which low power mode to enter
output_port_enable	in	Retain port output status in sleep or standby
power_supply	in	What remains powered in Deep Software Standby

Table 987:Parameters (Continued)

Name	Direction	Description
io_port_state	in	I/O port state after Deep Software Standby mode

Parameter low_power_mode

Parameter output_port_enable

Parameter power_supply

Parameter io_port_state

Definition: `lpm_io_port_tio_port_state`

I/O port state after Deep Software Standby mode

9.23.7.12 wupenSet

```
ssp_err_t(* lpm_api_t::wupenSet) (uint32_t wupen_value)
```

Detailed description

Set the value of the Wake Up Interrupt Enable Register WUPEN

Table 988:Parameters

Name	Direction	Description
wupen_value	in	The value for WUPEN

Parameter wupen_value

`uint32_t`

9.23.7.13 wupenGet

```
ssp_err_t(* lpm_api_t::wupenGet) (uint32_t *wupen_value)
```

Detailed description

Get the value of the Wake Up Interrupt Enable Register WUPEN

Table 989:Parameters

Name	Direction	Description
wupen_value	inout	The value from WUPEN

Parameter wupen_value

```
uint32_t
```

9.23.7.14 deepStandbyCancelRequestEnable

```
ssp_err_t(* lpm_api_t::deepStandbyCancelRequestEnable) (lpm_deep_standby_t
pin_signal, lpm_cancel_request_edge_t rising_falling)
```

Detailed description

Enable a Deep Standby Cancel Request

Table 990:Parameters

Name	Direction	Description
pin_signal	in	The pin or signal associated with deep standby mode
rising_falling	in	Which edge causes the cancel request (ignore with RTC interrupts)

Parameter pin_signal

Definition: `lpm_deep_standby_tpin_signal`

Deep Standby pins and signals

Parameter rising_falling

Definition: `lpm_cancel_request_edge_trising_falling`

Deep Standby Interrupt Edge

9.23.7.15 deepStandbyCancelRequestDisable

```
ssp_err_t(* lpm_api_t::deepStandbyCancelRequestDisable) (lpm_deep_standby_t
pin_signal)
```

Detailed description

Disable a Deep Standby Cancel Request

Table 991:Parameters

Name	Direction	Description
pin_signal	in	The pin or signal associated with deep standby mode

Parameter pin_signal

Definition: `lpm_deep_standby_tpin_signal`

Deep Standby pins and signals

9.23.7.16 lowPowerModeEnter

```
ssp_err_t(* lpm_api_t::lowPowerModeEnter) (void)
```

Detailed description

Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode.

9.23.7.17 versionGet

```
ssp_err_t(* lpm_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get the driver version based on compile time macros.

Table 992:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_PTR	p_version is NULL.

Parameter SSP_SUCCESS

Parameter SSP_ERR_INVALID_PTR

9.23.7.18 lpm_instance_t

lpm_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- lpm_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- lpm_api_t const * p_api

  Pointer to the API structure for this instance.

9.24 Low Power Modes V2 Interface

Interface for accessing low power modes.

9.24.1 Summary

This section defines the API for the LPMV2 (Low Power Mode) Driver. The LPMV2 Driver provides functions for controlling power consumption by configuring and transitioning to a low power mode. The LPMV2 driver supports configuration of MCU low power modes using the LPMV2 hardware peripheral. The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCUs.*Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

LPMV2 Interface description: HAL LPMV2 Interface

9.24.2 Interface API

lpmv2_api_t

Function name	Description
.init	Initialization function
.lowPowerCfg	Configure a low power mode.
.lowPowerModeEnter	Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode.
.versionGet	Get the driver version based on compile time macros.

9.24.3 Data structures

- lpmv2_cfg_t
- lpmv2_instance_t

9.24.4 Enumerations

- lpmv2_low_power_mode_t

9.24.5 Defines

- #define LPMV2_API_VERSION_MAJOR

  Initial value:(2U)

  Register definitions, common services and error codes.
- #define LPMV2_API_VERSION_MINOR

  Initial value:(3U)

9.24.6 API Data

9.24.6.1 lpmv2_low_power_mode_t

lpmv2_low_power_mode_t

Detailed description

Low power modes

Enumerated values

Name	Description
LPMV2_LOW_POWER_MODE_SLEEP	Sleep mode.
LPMV2_LOW_POWER_MODE_STANDBY	Software Standby mode.
LPMV2_LOW_POWER_MODE_STANDBY_SNOOZE	Software Standby mode with Snooze mode enabled.
LPMV2_LOW_POWER_MODE_DEEP	Deep Software Standby mode.

9.24.7 API Structures

9.24.7.1 lpmv2_cfg_t

lpmv2_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- lpmv2_low_power_mode_t low_power_mode

  Low Power Mode
- void const * p_extend

  MCU Specific configuration

9.24.7.2 lpmv2_api_t

lpmv2_api_t

Detailed description

lpmv2 driver structure. General lpmv2 functions implemented at the HAL layer will follow this API.

9.24.7.3 init

```
ssp_err_t(* lpmv2_api_t::init) (void)
```

Detailed description

Initialization function Implemented as

- R_LPMV2_Init

9.24.7.4 lowPowerCfg

```
ssp_err_t(* lpmv2_api_t::lowPowerCfg) (lpmv2_cfg_t const *const p_cfg)
```

Detailed description

Configure a low power mode. Implemented as

- R_LPMV2_LowPowerConfigure

Table 993:Parameters

Name	Direction	Description
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_cfg

Definition: `lpmv2_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `lpmv2_cfg_t`::`lpmv2_low_power_mode_t`

  Low Power Mode

  Enumerated as:

  – LPMV2_LOW_POWER_MODE_SLEEP

  – LPMV2_LOW_POWER_MODE_STANDBY

  – LPMV2_LOW_POWER_MODE_STANDBY_SNOOZE

  – LPMV2_LOW_POWER_MODE_DEEP

- lpmv2_cfg_t::p_extend

  MCU Specific configuration

9.24.7.5 lowPowerModeEnter

```
ssp_err_t(* lpmv2_api_t::lowPowerModeEnter) (void)
```

Detailed description

Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode. Implemented as

- R_LPMV2_LowPowerModeEnter

9.24.7.6 versionGet

```
ssp_err_t(* lpmv2_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get the driver version based on compile time macros. Implemented as

- R_LPMV2_VersionGet

Table 994:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.24.7.7 lpmv2_instance_t

lpmv2_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- lpmv2_cfg_t const *const p_cfg

  Pointer to the configuration structure for this instance.

- lpmv2_api_t const *const p_api

  Pointer to the API structure for this instance.

9.25 Low Voltage Detection Interface

This section defines the API for the LVD (Low Voltage Detection) Driver.

The LVD driver provides functions for configuring the LVD hardware peripheral.

The process of configuring and enabling a Low Voltage Detection monitor has very specific timing constraints and register write ordering. Because of these constraints, the entire process of configuring and enabling a voltage monitor is most effectively performed by a single function. The API function configure performs configuration and enables the monitor in order to properly enforce the timing and register write ordering constraints.

The LVD driver configures all of the settings of the available configurable LVD monitors.

The settings include:

- voltage_threshold: Determines the voltage detection threshold (i.e. 2.99 Volts).
- sample_clock_divisor: Determines the sample clock rate of the digital filter, based on division of the LOCO clock. Also disables or enables the digital filter if available on the MCU.
- detection_response: Determines which event will occur, reset, interrupt, non-maskable interrupt, or no response, when the voltage threshold is crossed
- voltage_slope: Choose either rising or falling voltage as the trigger for a voltage detection interrupt.
- negation_delay: Determine whether timing of the negation of the voltage event is based upon the reset event or based on the voltage event itself.
- p_callback: Address of user defined function to be called when the voltage event interrupt occurs.

NOTE: *Low Voltage Monitor 0 (LVD0) is not configurable at runtime but can be configured by changing the OFS1 register value on the BSP Properties tab of the Synergy Project Configurator in the e² studio ISDE.****NOTE:*** *Digital filter is not to be used with standby modes. If software standby or deep standby mode is to be used, the digital filter should be disabled.*For details about the implementation of the driver functions see section LVD.

9.25.1 Interface API

lvd_api_t

Function name	Description
.open	Initializes a low voltage detection driver according to the passed in configuration structure. Enables an LVD peripheral based on configuration structure.
.statusGet	Get the current state of the monitor, (threshold crossing detected, voltage currently within range) Can be used to poll the state of the LVD monitor at any time. Must be used if the peripheral was initialized with lvd_response_t set to LVD_RESPONSE_NONE.
.statusClear	Clears the latched status of the monitor. Must be used if the peripheral was initialized with lvd_response_t set to LVD_RESPONSE_NONE.

Function name	Description
.close	Disables the LVD peripheral. Closes the driver instance.
.versionGet	Returns the LVD driver version based on compile time macros.

9.25.2 Data structures

- lvd_status_t
- lvd_callback_args_t
- lvd_cfg_t
- lvd_instance_t

9.25.3 Enumerations

- lvd_threshold_t
- lvd_response_t
- lvd_voltage_slope_t
- lvd_threshold_crossing_t
- lvd_current_state_t

9.25.4 Typedefs

- lvd_ctrl_t

9.25.5 Defines

- #define LVD_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define LVD_API_VERSION_MINOR

  Initial value:`(1U)`

9.25.6 API Data

9.25.6.1 lvd_threshold_t

`lvd_threshold_t`

Detailed description

Voltage detection level The thresholds supported by each MCU is in the MCU User's Manual as well as in the r_lvd module description on the threads tab of the Synergy project.

Enumerated values

Name	Description
LVD_THRESHOLD_MONITOR_1_LEVEL_0	4.29V (Vdet1_0)
LVD_THRESHOLD_MONITOR_1_LEVEL_1	4.14V (Vdet1_1)
LVD_THRESHOLD_MONITOR_1_LEVEL_2	4.02V (Vdet1_2)
LVD_THRESHOLD_MONITOR_1_LEVEL_3	3.84V (Vdet1_3)
LVD_THRESHOLD_MONITOR_1_LEVEL_4	3.10V (Vdet1_4)
LVD_THRESHOLD_MONITOR_1_LEVEL_5	3.00V (Vdet1_5)
LVD_THRESHOLD_MONITOR_1_LEVEL_6	2.90V (Vdet1_6)
LVD_THRESHOLD_MONITOR_1_LEVEL_7	2.79V (Vdet1_7)
LVD_THRESHOLD_MONITOR_1_LEVEL_8	2.68V (Vdet1_8)
LVD_THRESHOLD_MONITOR_1_LEVEL_9	2.58V (Vdet1_9)
LVD_THRESHOLD_MONITOR_1_LEVEL_A	2.48V (Vdet1_A)
LVD_THRESHOLD_MONITOR_1_LEVEL_B	2.20V (Vdet1_B)
LVD_THRESHOLD_MONITOR_1_LEVEL_C	1.96V (Vdet1_C)
LVD_THRESHOLD_MONITOR_1_LEVEL_D	1.86V (Vdet1_D)
LVD_THRESHOLD_MONITOR_1_LEVEL_E	1.75V (Vdet1_E)
LVD_THRESHOLD_MONITOR_1_LEVEL_F	1.65V (Vdet1_F)
LVD_THRESHOLD_MONITOR_1_LEVEL_11	2.99V (Vdet1_11)
LVD_THRESHOLD_MONITOR_1_LEVEL_12	2.92V (Vdet1_12)

Name	Description
LVD_THRESHOLD_MONITOR_1_LEVEL_13	2.85V (Vdet1_13)
LVD_THRESHOLD_MONITOR_2_LEVEL_0	4.29V (Vdet2_0)
LVD_THRESHOLD_MONITOR_2_LEVEL_1	4.14V (Vdet2_1)
LVD_THRESHOLD_MONITOR_2_LEVEL_2	4.02V (Vdet2_2)
LVD_THRESHOLD_MONITOR_2_LEVEL_3	3.84V (Vdet2_3)
LVD_THRESHOLD_MONITOR_2_LEVEL_5	2.99V (Vdet2_5)
LVD_THRESHOLD_MONITOR_2_LEVEL_6	2.92V (Vdet2_6)
LVD_THRESHOLD_MONITOR_2_LEVEL_7	2.85V (Vdet2_7)

9.25.6.2 lvd_response_t

`lvd_response_t`

Detailed description

Response types to a threshold crossing event, interrupt, reset, NMI...

Enumerated values

Name	Description
LVD_RESPONSE_NMI	Non-maskable interrupt.
LVD_RESPONSE_INTERRUPT	Maskable interrupt.
LVD_RESPONSE_RESET	Reset.
LVD_RESPONSE_NONE	No response, status must be requested via statusGet function.

9.25.6.3 lvd_voltage_slope_t

`lvd_voltage_slope_t`

Detailed description

Voltage slope, rising, falling, or both

Enumerated values

Name	Description
LVD_VOLTAGE_SLOPE_RISING	When VCC >= Vdet2 (rise) is detected.
LVD_VOLTAGE_SLOPE_FALLING	When VCC < Vdet2 (drop) is detected.
LVD_VOLTAGE_SLOPE_BOTH	When drop and rise are detected.

9.25.6.4 lvd_threshold_crossing_t

`lvd_threshold_crossing_t`

Detailed description

Threshold crossing detection (latched)

Enumerated values

Name	Description
LVD_THRESHOLD_CROSSING_NOT_DETECTED	Threshold crossing has not been detected.
LVD_THRESHOLD_CROSSING_DETECTED	Threshold crossing has been detected.

9.25.6.5 lvd_current_state_t

`lvd_current_state_t`

Detailed description

Instantaneous status of VCC (above or below threshold)

Enumerated values

Name	Description
LVD_CURRENT_STATE_BELOW_THRESHOLD	VCC < threshold.
LVD_CURRENT_STATE_ABOVE_THRESHOLD	VCC >= threshold or monitor is disabled.

9.25.6.6 lvd_ctrl_t

`typedef void lvd_ctrl_t`

Detailed description

LVD control block. Allocate an instance specific control block to pass into the LVD API calls. Implemented as

- lvd_instance_ctrl_t

9.25.7 API Structures

9.25.7.1 lvd_status_t

lvd_status_t

Detailed description

Voltage monitor status structure, used with statusGet function and p_callback to provide current state of the monitor, (threshold crossing detected, vcc currently within range).

Variables

- lvd_threshold_crossing_t crossing_detected

  Threshold crossing detection (latched)

- lvd_current_state_t current_state

  Instantaneous status of monitored voltage (above or below threshold)

9.25.7.2 lvd_callback_args_t

lvd_callback_args_t

Detailed description

LVD callback parameter definition

Variables

- uint32_t monitor_number

  Monitor number.

- lvd_status_t status

  Status of monitor.

- void const * p_context

  Placeholder for user data.

9.25.7.3 lvd_cfg_t

lvd_cfg_t

Detailed description

LVD configuration structure

Variables

- const uint32_t monitor_number

  Monitor number, 1, 2, ...

- lvd_threshold_t voltage_threshold

  Threshold for out of range voltage detection
- lvd_response_t detection_response

  Response on detecting a threshold crossing
- lvd_voltage_slope_t voltage_slope

  Rising or falling voltage is to be detected
- uint8_t monitor_ipl

  Interrupt priority level.
- void(* p_callback)(lvd_callback_args_t *p_args)

  User function to be called from interrupt
- void const * p_context

  Placeholder for user data. Passed to the user callback in
- void const * p_extend

  Extension parameter for hardware specific settings

9.25.7.4 lvd_api_t

lvd_api_t

Detailed description

LVD driver API structure. LVD driver functions implemented at the HAL layer will adhere to this API.

9.25.7.5 open

```
ssp_err_t(* lvd_api_t::open) (lvd_ctrl_t *const p_ctrl, lvd_cfg_t const *const p_cfg)
```

Detailed description

Initializes a low voltage detection driver according to the passed in configuration structure. Enables an LVD peripheral based on configuration structure. Implemented as

- R_LVD_Open

Table 995:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to monitor control structure for the driver instance
p_cfg	in	Pointer to the configuration structure for the driver instance

Parameter p_ctrl

Definition: `lvd_ctrl_t*const p_ctrl`

LVD control block. Allocate an instance specific control block to pass into the LVD API calls. Implemented aslvd_instance_ctrl_t

Parameter p_cfg

Definition: `lvd_cfg_t const *const p_cfg`

LVD configuration structure

- `lvd_cfg_t`::`monitor_number`

  Monitor number, 1, 2, ...

- `lvd_cfg_t`::`lvd_threshold_t`

  Threshold for out of range voltage detection

  Enumerated as:

  – LVD_THRESHOLD_MONITOR_1_LEVEL_0
  – LVD_THRESHOLD_MONITOR_1_LEVEL_1
  – LVD_THRESHOLD_MONITOR_1_LEVEL_2
  – LVD_THRESHOLD_MONITOR_1_LEVEL_3
  – LVD_THRESHOLD_MONITOR_1_LEVEL_4
  – LVD_THRESHOLD_MONITOR_1_LEVEL_5
  – LVD_THRESHOLD_MONITOR_1_LEVEL_6
  – LVD_THRESHOLD_MONITOR_1_LEVEL_7
  – LVD_THRESHOLD_MONITOR_1_LEVEL_8
  – LVD_THRESHOLD_MONITOR_1_LEVEL_9
  – LVD_THRESHOLD_MONITOR_1_LEVEL_A
  – LVD_THRESHOLD_MONITOR_1_LEVEL_B
  – LVD_THRESHOLD_MONITOR_1_LEVEL_C
  – LVD_THRESHOLD_MONITOR_1_LEVEL_D
  – LVD_THRESHOLD_MONITOR_1_LEVEL_E
  – LVD_THRESHOLD_MONITOR_1_LEVEL_F
  – LVD_THRESHOLD_MONITOR_1_LEVEL_11
  – LVD_THRESHOLD_MONITOR_1_LEVEL_12
  – LVD_THRESHOLD_MONITOR_1_LEVEL_13

  - LVD_THRESHOLD_MONITOR_2_LEVEL_0
  - LVD_THRESHOLD_MONITOR_2_LEVEL_1
  - LVD_THRESHOLD_MONITOR_2_LEVEL_2
  - LVD_THRESHOLD_MONITOR_2_LEVEL_3
  - LVD_THRESHOLD_MONITOR_2_LEVEL_5
  - LVD_THRESHOLD_MONITOR_2_LEVEL_6
  - LVD_THRESHOLD_MONITOR_2_LEVEL_7
- `lvd_cfg_t::lvd_response_t`

  Response on detecting a threshold crossing

  Enumerated as:
  - LVD_RESPONSE_NMI
  - LVD_RESPONSE_INTERRUPT
  - LVD_RESPONSE_RESET
  - LVD_RESPONSE_NONE
- `lvd_cfg_t::lvd_voltage_slope_t`

  Rising or falling voltage is to be detected

  Enumerated as:
  - LVD_VOLTAGE_SLOPE_RISING
  - LVD_VOLTAGE_SLOPE_FALLING
  - LVD_VOLTAGE_SLOPE_BOTH
- `lvd_cfg_t::monitor_ipl`

  Interrupt priority level.
- `lvd_cfg_t::p_callback`

  User function to be called from interrupt
- `lvd_cfg_t::p_context`

  Placeholder for user data. Passed to the user callback in
- `lvd_cfg_t::p_extend`

  Extension parameter for hardware specific settings

9.25.7.6 statusGet

```
ssp_err_t(* lvd_api_t::statusGet) (lvd_ctrl_t *const p_ctrl, lvd_status_t
*p_lvd_status)
```

Detailed description

Get the current state of the monitor, (threshold crossing detected, voltage currently within range) Can be used to poll the state of the LVD monitor at any time. Must be used if the peripheral was initialized with lvd_response_t set to LVD_RESPONSE_NONE. Implemented as

- R_LVD_StatusGet

Table 996:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control structure for the driver instance
p_lvd_status	inout	Pointer to an lvd_status_t instance

Parameter p_ctrl

Definition: `lvd_ctrl_t*const p_ctrl`

LVD control block. Allocate an instance specific control block to pass into the LVD API calls. Implemented aslvd_instance_ctrl_t

Parameter p_lvd_status

Definition: `lvd_status_t*p_lvd_status`

Voltage monitor status structure, used with statusGet function and p_callback to provide current state of the monitor, (threshold crossing detected, vcc currently within range).

- `lvd_status_t::crossing_detected`

  Threshold crossing detection (latched)

- `lvd_status_t::current_state`

  Instantaneous status of monitored voltage (above or below threshold)

9.25.7.7 statusClear

```
ssp_err_t(* lvd_api_t::statusClear) (lvd_ctrl_t *const p_ctrl)
```

Detailed description

Clears the latched status of the monitor. Must be used if the peripheral was initialized with lvd_response_t set to LVD_RESPONSE_NONE. Implemented as

- R_LVD_StatusClear

Table 997:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control structure for the driver instance

Parameter p_ctrl

Definition: `lvd_ctrl_t*const p_ctrl`

LVD control block. Allocate an instance specific control block to pass into the LVD API calls. Implemented aslvd_instance_ctrl_t

9.25.7.8 close

```
ssp_err_t(* lvd_api_t::close) (lvd_ctrl_t *const p_ctrl)
```

Detailed description

Disables the LVD peripheral. Closes the driver instance. Implemented as

- R_LVD_Close

Table 998:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control structure for the driver instance

Parameter p_ctrl

Definition: `lvd_ctrl_t*const p_ctrl`

LVD control block. Allocate an instance specific control block to pass into the LVD API calls. Implemented aslvd_instance_ctrl_t

9.25.7.9 versionGet

```
ssp_err_t(* lvd_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Returns the LVD driver version based on compile time macros. Implemented as

- R_LVD_VersionGet

Table 999:Parameters

Name	Direction	Description
p_version	inout	Pointer to version structure

Parameter p_version

9.25.7.10 lvd_instance_t

lvd_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- lvd_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- lvd_cfg_t const * p_cfg

  Pointer to the configuration structure for this interface instance.

- lvd_api_t const * p_api

  Pointer to the API structure for this interface instance.

9.26 PDC Interface

Interface for PDC functions.

9.26.1 Summary

The PDC interface provides the functionality for capturing an image from a camera. When a capture is complete a transfer complete interrupt is triggered.

9.26.2 Known Implementations

PDCRelated SSP architecture topics:

- What is an SSP Interface? SSP Interfaces
- What is a SSP Layer? SSP Predefined Layers
- How to use SSP Interfaces and Modules? Using SSP Modules

9.26.3 Interface API

pdc_api_t

Function name	Description
.open	Initial configuration.
.close	Closes the driver and allows reconfiguration. May reduce power consumption.
.captureStart	Start a capture.
.stateGet	Get the state of the VSYNC and HSYNC pins.
.versionGet	Return the version of the driver.

9.26.4 Data structures

- pdc_state_t
- pdc_callback_args_t
- pdc_cfg_t
- pdc_instance_t

9.26.5 Enumerations

- pdc_clock_division_t
- pdc_endian_t
- pdc_hsync_polarity_t
- pdc_vsync_polarity_t
- pdc_event_t
- pdc_vsync_state_t
- pdc_hsync_state_t

9.26.6 Typedefs

- pdc_ctrl_t

9.26.7 Defines

- #define PDC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define PDC_API_VERSION_MINOR

  Initial value:`(2U)`

9.26.8 API Data

9.26.8.1 pdc_clock_division_t

`pdc_clock_division_t`

Detailed description

Clock divider applied to PDC clock to provide PCKO output frequency

Enumerated values

Name	Description
PDC_CLOCK_DIVISION_2	CLK / 2.
PDC_CLOCK_DIVISION_4	CLK / 4.
PDC_CLOCK_DIVISION_6	CLK / 6.
PDC_CLOCK_DIVISION_8	CLK / 8.
PDC_CLOCK_DIVISION_10	CLK / 10.
PDC_CLOCK_DIVISION_12	CLK / 12.
PDC_CLOCK_DIVISION_14	CLK / 14.
PDC_CLOCK_DIVISION_16	CLK / 16.

9.26.8.2 pdc_endian_t

`pdc_endian_t`

Detailed description

Endian of captured data

Enumerated values

Name	Description
PDC_ENDIAN_LITTLE	Data is in little endian format.
PDC_ENDIAN_BIG	Data is in big endian format.

9.26.8.3 pdc_hsync_polarity_t

`pdc_hsync_polarity_t`

Detailed description

Polarity of input HSYNC signal

Enumerated values

Name	Description
PDC_HSYNC_POLARITY_HIGH	HSYNC signal is active high.
PDC_HSYNC_POLARITY_LOW	HSYNC signal is active low.

9.26.8.4 pdc_vsync_polarity_t

`pdc_vsync_polarity_t`

Detailed description

Polarity of input VSYNC signal

Enumerated values

Name	Description
PDC_VSYNC_POLARITY_HIGH	VSYNC signal is active high.
PDC_VSYNC_POLARITY_LOW	VSYNC signal is active low.

9.26.8.5 pdc_event_t

`pdc_event_t`

Detailed description

PDC events

Enumerated values

Name	Description
PDC_EVENT_TRANSFER_COMPLETE	Complete frame transferred by DMAC/DTC.
PDC_EVENT_RX_DATA_READY	Receive data ready interrupt.
PDC_EVENT_FRAME_END	Frame end interrupt.
PDC_EVENT_ERR_OVERRUN	Overrun interrupt.
PDC_EVENT_ERR_UNDERRUN	Underrun interrupt.
PDC_EVENT_ERR_V_SET	Vertical line setting error interrupt.
PDC_EVENT_ERR_H_SET	Horizontal byte number setting error interrupt.

9.26.8.6 pdc_vsync_state_t

`pdc_vsync_state_t`

Detailed description

VSYNC signal state

Enumerated values

Name	Description
PDC_VSYNC_STATE_LOW	VSYNC signal is low.
PDC_VSYNC_STATE_HIGH	VSYNC signal is high.

9.26.8.7 pdc_hsync_state_t

`pdc_hsync_state_t`

Detailed description

HSYNC signal state

Enumerated values

Name	Description
PDC_HSYNC_STATE_LOW	HSYNC signal is low.
PDC_HSYNC_STATE_HIGH	HSYNC signal is high.

9.26.8.8 pdc_ctrl_t

```
typedef void pdc_ctrl_t
```

Detailed description

PDC control block. Allocate an instance specific control block to pass into the PDC API calls. Implemented as

- pdc_instance_ctrl_t

9.26.9 API Structures

9.26.9.1 pdc_state_t

pdc_state_t

Detailed description

PDC VSYNC/HSYNC state

Variables

- pdc_vsync_state_t vsync

  VSYNC signal state.
- pdc_hsync_state_t hsync

  HSYNC signal state.

9.26.9.2 pdc_callback_args_t

pdc_callback_args_t

Detailed description

Callback function parameter data

Variables

- pdc_event_t event

  Event causing the callback.
- uint8_t * p_buffer

  Pointer to buffer containing the captured data.
- void const * p_context

  Placeholder for user data. Set in pdc_api_t::open function in pdc_cfg_t.

9.26.9.3 pdc_cfg_t

pdc_cfg_t

Detailed description

PDC configuration parameters.

Variables

- uint16_t x_capture_start_pixel

  Horizontal position to start capture.
- uint16_t x_capture_pixels

  Number of horizontal pixels to capture.
- uint16_t y_capture_start_pixel

  Vertical position to start capture.
- uint16_t y_capture_pixels

  Number of vertical lines/pixels to capture.
- pdc_clock_division_t clock_division

  Clock divider.
- pdc_endian_t endian

  Endian of capture data.
- pdc_hsync_polarity_t hsync_polarity

  Polarity of HSYNC input.
- pdc_vsync_polarity_t vsync_polarity

  Polarity of VSYNC input.
- uint8_t * p_buffer

  Pointer to buffer to write image into.
- uint8_t bytes_per_pixel

  Number of bytes per pixel.
- uint8_t frame_end_ipl

  Frame end interrupt priority.
- uint8_t irq_ipl

  PDC interrupt priority.
- transfer_instance_t const *const p_lower_lvl_transfer

  Pointer to the transfer instance the PDC should use.
- void(* p_callback)(pdc_callback_args_t *p_args)

  Callback provided when a PDC transfer ISR occurs.
- void const * p_context

  Placeholder for user data. Passed to the user callback in pdc_callback_args_t.
- void const * p_extend

9.26.9.4 pdc_api_t

pdc_api_t

Detailed description

PDC functions implemented at the HAL layer will follow this API.

9.26.9.5 open

```
ssp_err_t(* pdc_api_t::open) (pdc_ctrl_t *const p_ctrl, pdc_cfg_t const *const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_PDC_Open

NOTE: To reconfigure after calling this function, call close first.

Table 1000:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_cfg	in	Pointer to pin configuration structure.

Parameter p_ctrl

Definition: `pdc_ctrl_t*const p_ctrl`

PDC control block. Allocate an instance specific control block to pass into the PDC API calls. Implemented aspdc_instance_ctrl_t

Parameter p_cfg

Definition: `pdc_cfg_t const *const p_cfg`

PDC configuration parameters.

- `pdc_cfg_t`::`x_capture_start_pixel`

  Horizontal position to start capture.

- `pdc_cfg_t`::`x_capture_pixels`

  Number of horizontal pixels to capture.

- `pdc_cfg_t`::`y_capture_start_pixel`

  Vertical position to start capture.

- `pdc_cfg_t`::`y_capture_pixels`

  Number of vertical lines/pixels to capture.

- pdc_cfg_t::pdc_clock_division_t

  Clock divider.

  Enumerated as:

  – PDC_CLOCK_DIVISION_2
  – PDC_CLOCK_DIVISION_4
  – PDC_CLOCK_DIVISION_6
  – PDC_CLOCK_DIVISION_8
  – PDC_CLOCK_DIVISION_10
  – PDC_CLOCK_DIVISION_12
  – PDC_CLOCK_DIVISION_14
  – PDC_CLOCK_DIVISION_16
- pdc_cfg_t::pdc_endian_t

  Endian of capture data.

  Enumerated as:

  – PDC_ENDIAN_LITTLE
  – PDC_ENDIAN_BIG
- pdc_cfg_t::pdc_hsync_polarity_t

  Polarity of HSYNC input.

  Enumerated as:

  – PDC_HSYNC_POLARITY_HIGH
  – PDC_HSYNC_POLARITY_LOW
- pdc_cfg_t::pdc_vsync_polarity_t

  Polarity of VSYNC input.

  Enumerated as:

  – PDC_VSYNC_POLARITY_HIGH
  – PDC_VSYNC_POLARITY_LOW
- pdc_cfg_t::p_buffer

  Pointer to buffer to write image into.
- pdc_cfg_t::bytes_per_pixel

  Number of bytes per pixel.

- pdc_cfg_t::frame_end_ipl

  Frame end interrupt priority.
- pdc_cfg_t::irq_ipl

  PDC interrupt priority.
- pdc_cfg_t::transfer_instance_t

  Pointer to the transfer instance the PDC should use.
- pdc_cfg_t::p_callback

  Callback provided when a PDC transfer ISR occurs.
- pdc_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in pdc_callback_args_t.
- pdc_cfg_t::p_extend

9.26.9.6 close

```
ssp_err_t(* pdc_api_t::close) (pdc_ctrl_t *const p_ctrl)
```

Detailed description

Closes the driver and allows reconfiguration. May reduce power consumption. Implemented as

- R_PDC_Close

Table 1001:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.

Parameter p_ctrl

Definition: pdc_ctrl_t*const p_ctrl

PDC control block. Allocate an instance specific control block to pass into the PDC API calls. Implemented aspdc_instance_ctrl_t

9.26.9.7 captureStart

```
ssp_err_t(* pdc_api_t::captureStart) (pdc_ctrl_t *const p_ctrl, uint8_t *const
p_buffer)
```

Detailed description

Start a capture. Implemented as

- R_PDC_CaptureStart

Table 1002:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_buffer	in	Pointer to store captured image data.

Parameter p_ctrl

Definition: `pdc_ctrl_t*const p_ctrl`

PDC control block. Allocate an instance specific control block to pass into the PDC API calls. Implemented aspdc_instance_ctrl_t

Parameter p_buffer

`uint8_t`

9.26.9.8 stateGet

```
ssp_err_t(* pdc_api_t::stateGet) (pdc_ctrl_t *const p_ctrl, pdc_state_t *p_state)
```

Detailed description

Get the state of the VSYNC and HSYNC pins. Implemented as

- R_PDC_StateGet

Table 1003:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_state	in	Pointer to store state data.

Parameter p_ctrl

Definition: `pdc_ctrl_t*const p_ctrl`

PDC control block. Allocate an instance specific control block to pass into the PDC API calls. Implemented aspdc_instance_ctrl_t

Parameter p_state

Definition: `pdc_state_t*p_state`

PDC VSYNC/HSYNC state

- `pdc_state_t::vsync`

  VSYNC signal state.

- pdc_state_t::hsync

  HSYNC signal state.

9.26.9.9 versionGet

```
ssp_err_t(* pdc_api_t::versionGet) (ssp_version_t *const p_data)
```

Detailed description

Return the version of the driver. Implemented as

- R_PDC_VersionGet

Table 1004:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_data	out	Memory address to return version information to.

Parameter p_ctrl

Parameter p_data

Definition: `ssp_version_t *const p_data`

- ssp_version_t::version_id

  Version id

- ssp_version_t::code_version_minor

  Code minor version.

- ssp_version_t::code_version_major

  Code major version.

- ssp_version_t::api_version_minor

  API minor version.

- ssp_version_t::api_version_major

  API major version.

- ssp_version_tstruct{}

  Code version parameters

9.26.9.10 pdc_instance_t

pdc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- pdc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- pdc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- pdc_api_t const * p_api

  Pointer to the API structure for this instance.

9.27 Quad SPI Flash Interface

Interface for accessing external SPI flash devices.

9.27.1 Summary

The QSPI module provides an interface that writes and erases sectors in quad SPI flash devices connected to the QSPI interface.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

QSPI Interface description: QSPI Driver

9.27.2 Interface API

qspi_api_t

Function name	Description
.open	Open the QSPI driver module.
.close	Close the QSPI driver module.
.read	Read a block of data from the flash.
.pageProgram	Program a page of data to the flash.
.erase	Erase a certain number of bytes of the flash.

Function name	Description
.infoGet	Get information about this specific QSPI flash.
.sectorErase	Erase a sector of the flash.
.statusGet	Get the write or erase status of the flash.
.bankSelect	Select the bank to access.
.versionGet	Get the driver version based on compile time macros.

9.27.3 Data structures

- qspi_cfg_t
- qspi_info_t
- qspi_instance_t

9.27.4 Typedefs

- qspi_ctrl_t

9.27.5 Defines

- #define QSPI_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define QSPI_API_VERSION_MINOR

  Initial value:`(2U)`

9.27.6 API Data

9.27.6.1 qspi_ctrl_t

```
typedef void qspi_ctrl_t
```

Detailed description

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented as

- qspi_instance_ctrl_t

9.27.7 API Structures

9.27.7.1 qspi_cfg_t

qspi_cfg_t

Detailed description

User configuration structure used by the open function

Variables

- void * p_extend

  place holder for future development

9.27.7.2 qspi_info_t

qspi_info_t

Detailed description

QSPI information structure to store information specific to the currently used QSPI.

Variables

- uint32_t total_size_bytes

  Size of this QSPI in bytes.

- uint32_t min_program_size_bytes

  Minimum program size in bytes.

- uint32_t * p_erase_sizes_bytes

  Array of available erase sizes. Each entry is in bytes.

- uint8_t num_erase_sizes

  Number of available erase sizes.

9.27.7.3 qspi_api_t

qspi_api_t

Detailed description

QSPI functions implemented at the HAL layer follow this API.

9.27.7.4 open

```
(* qspi_api_t::open) (qspi_ctrl_t *p_ctrl, qspi_cfg_t const *const p_cfg)
```

Detailed description

Open the QSPI driver module. Implemented as

- R_QSPI_Open

Table 1005:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle
p_cfg	in	Pointer to a configuration structure

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_cfg

Definition: `qspi_cfg_t const *const p_cfg`

User configuration structure used by the open function

- `qspi_cfg_t`::`p_extend`

  place holder for future development

9.27.7.5 close

```
(* qspi_api_t::close) (qspi_ctrl_t *p_ctrl)
```

Detailed description

Close the QSPI driver module. Implemented as

- R_QSPI_Close

Table 1006:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

9.27.7.6 read

```
(* qspi_api_t::read) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint8_t
*p_memory_address, uint32_t byte_count)
```

Detailed description

Read a block of data from the flash. Implemented as

- R_QSPI_Read

Table 1007:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle
p_device_address	in	The location in the flash device address space to read
p_memory_address	in	The memory address of a buffer to place the read data in
byte_count	in	The number of bytes to read

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_device_address

`uint8_t`

Parameter p_memory_address

`uint8_t`

Parameter byte_count

`uint32_t`

9.27.7.7 pageProgram

```
(* qspi_api_t::pageProgram) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address,
uint8_t *p_memory_address, uint32_t byte_count)
```

Detailed description

Program a page of data to the flash. Implemented as

- R_QSPI_PageProgram

Table 1008:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle

Table 1008:Parameters (Continued)

Name	Direction	Description
p_device_address	in	The location in the flash device address space to write the data to
p_memory_address	in	The memory address of the data to write to the flash device
byte_count	in	The number of bytes to write

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_device_address

`uint8_t`

Parameter p_memory_address

`uint8_t`

Parameter byte_count

`uint32_t`

9.27.7.8 erase

```
(* qspi_api_t::erase) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint32_t byte_count)
```

Detailed description

Erase a certain number of bytes of the flash. Implemented as

- R_QSPI_Erase

Table 1009:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle
p_device_address	in	The location in the flash device address space to start the erase from
byte_count	in	The number of bytes to erase

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_device_address

`uint8_t`

Parameter byte_count

`uint32_t`

9.27.7.9 infoGet

```
(* qspi_api_t::infoGet) (qspi_ctrl_t *const p_ctrl, qspi_info_t *const p_info)
```

Detailed description

Get information about this specific QSPI flash. Implemented as

- R_QSPI_InfoGet

Table 1010:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
p_info	out	Collection of information for this QSPI.

Parameter p_ctrl

Definition: `qspi_ctrl_t*const p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_info

Definition: `qspi_info_t*const p_info`

QSPI information structure to store information specific to the currently used QSPI.

- `qspi_info_t::total_size_bytes`

  Size of this QSPI in bytes.

- `qspi_info_t::min_program_size_bytes`

  Minimum program size in bytes.

- `qspi_info_t::p_erase_sizes_bytes`

  Array of available erase sizes. Each entry is in bytes.

- `qspi_info_t::num_erase_sizes`

  Number of available erase sizes.

9.27.7.10 sectorErase

```
(* qspi_api_t::sectorErase) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address)
```

Detailed description

Erase a sector of the flash. Implemented as

- R_QSPI_SectorErase

Table 1011:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle
p_device_address	in	The location in the flash device address space to start the erase from

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_device_address

`uint8_t`

9.27.7.11 statusGet

```
(* qspi_api_t::statusGet) (qspi_ctrl_t *p_ctrl, bool *p_write_in_progress)
```

Detailed description

Get the write or erase status of the flash. Implemented as

- R_QSPI_StatusGet

Table 1012:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to a driver handle
p_write_in_progress	in	The write or erase status - TRUE = write/erase in progress

Parameter p_ctrl

Definition: `qspi_ctrl_t*p_ctrl`

QSPI control block. Allocate an instance specific control block to pass into the QSPI API calls. Implemented asqspi_instance_ctrl_t

Parameter p_write_in_progress

`const`

9.27.7.12 bankSelect

```
(* qspi_api_t::bankSelect) (uint32_t bank)
```

Detailed description

Select the bank to access. Implemented as

- R_QSPI_BankSelect

Table 1013:Parameters

Name	Direction	Description
bank	in	The bank number

Parameter bank

`uint32_t`

9.27.7.13 versionGet

```
(* qspi_api_t::versionGet) ( *const p_version)
```

Detailed description

Get the driver version based on compile time macros. Implemented as

- R_QSPI_VersionGet

Table 1014:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.27.7.14 qspi_instance_t

qspi_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- qspi_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.
- qspi_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.
- qspi_api_t const * p_api

  Pointer to the API structure for this instance.

9.28 RTC Interface

Interface for accessing the Realtime Clock.

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

RTC description: RTC Driver

9.28.1 Interface API

rtc_api_t

Function name	Description
.open	Open the RTC driver.
.close	Close the RTC driver.
.calendarTimeSet	Set the calendar time.
.calendarTimeGet	Get the calendar time.
.calendarAlarmSet	Set the calendar alarm time.
.calendarAlarmGet	Get the calendar alarm time.
.calendarCounterStart	Start the calendar counter.
.calendarCounterStop	Stop the calendar counter.
.irqEnable	Enable the alarm irq.

Function name	Description
.irqDisable	Disable the alarm irq.
.periodicIrqRateSet	Set the periodic irq rate
.infoGet	Return the currently configure clock source for the RTC
.versionGet	Gets version and stores it in provided pointer p_version.

9.28.2 Data structures

- rtc_callback_args_t
- rtc_alarm_time_t
- rtc_info_t
- rtc_cfg_t
- rtc_instance_t

9.28.3 Enumerations

- rtc_event_t
- rtc_clock_source_t
- rtc_error_adjustment_t
- rtc_periodic_irq_select_t

9.28.4 Typedefs

- rtc_time_t
- rtc_ctrl_t

9.28.5 Defines

- #define RTC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Use of time structure, tm Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define RTC_API_VERSION_MINOR

  Initial value:`(3U)`

9.28.6 API Data

9.28.6.1 rtc_event_t

`rtc_event_t`

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
RTC_EVENT_ALARM_IRQ	Real Time Clock ALARM IRQ.
RTC_EVENT_PERIODIC_IRQ	Real Time Clock PERIODIC IRQ.
RTC_EVENT_CARRY_IRQ	Real Time Clock CARRY IRQ.

9.28.6.2 rtc_clock_source_t

`rtc_clock_source_t`

Detailed description

Clock source for the RTC block

Enumerated values

Name	Description
RTC_CLOCK_SOURCE_SUBCLK	Sub-clock oscillator.
RTC_CLOCK_SOURCE_LOCO	Low power On Chip Oscillator.

9.28.6.3 rtc_error_adjustment_t

`rtc_error_adjustment_t`

Detailed description

Time error adjustment settings

Enumerated values

Name	Description
RTC_ERROR_ADJUSTMENT_NONE	Adjustment is not performed.
RTC_ERROR_ADJUSTMENT_ADD_PRESCALER	Adjustment is performed by the addition to the prescaler.
RTC_ERROR_ADJUSTMENT_SUBTRACT_PRESCALER	Adjustment is performed by the subtraction from the prescaler.

9.28.6.4 rtc_periodic_irq_select_t

```
rtc_periodic_irq_select_t
```

Detailed description

Periodic Interrupt select

Enumerated values

Name	Description
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_256_SECOND	A periodic irq is generated every 1/256 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_128_SECOND	A periodic irq is generated every 1/128 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_64_SECOND	A periodic irq is generated every 1/64 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_32_SECOND	A periodic irq is generated every 1/32 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_16_SECOND	A periodic irq is generated every 1/16 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_8_SECOND	A periodic irq is generated every 1/8 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_4_SECOND	A periodic irq is generated every 1/4 second.
RTC_PERIODIC_IRQ_SELECT_1_DIV_BY_2_SECOND	A periodic irq is generated every 1/2 second.
RTC_PERIODIC_IRQ_SELECT_1_SECOND	A periodic irq is generated every 1 second.
RTC_PERIODIC_IRQ_SELECT_2_SECONDS	A periodic irq is generated every 2 seconds.

9.28.6.5 rtc_time_t

```
typedef struct tm rtc_time_t
```

Detailed description

Date and time structure defined in C standard library <time.h>

9.28.6.6 rtc_ctrl_t

```
typedef void rtc_ctrl_t
```

Detailed description

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented as

- rtc_instance_ctrl_t

9.28.7 API Structures

9.28.7.1 rtc_callback_args_t

rtc_callback_args_t

Detailed description

Callback function parameter data

Variables

- rtc_event_t event

  The event can be used to identify what caused the callback (compare match or error).
- void const * p_context

  Placeholder for user data. Set in r_timer_t::open function in timer_cfg_t.

9.28.7.2 rtc_alarm_time_t

rtc_alarm_time_t

Detailed description

Alarm time setting structure

Variables

- rtc_time_t time

  Time structure.
- bool sec_match

  Enable the alarm based on a match of the seconds field.
- bool min_match

  Enable the alarm based on a match of the minutes field.
- bool hour_match

  Enable the alarm based on a match of the hours field.
- bool mday_match

  Enable the alarm based on a match of the days field.

- bool mon_match

  Enable the alarm based on a match of the months field.

- bool year_match

  Enable the alarm based on a match of the years field.

- bool dayofweek_match

  Enable the alarm based on a match of the dayofweek field.

9.28.7.3 rtc_info_t

rtc_info_t

Detailed description

RTC Information Structure for information returned by infoGet()

Variables

- rtc_clock_source_t clock_source

  Clock source for the RTC block.

9.28.7.4 rtc_cfg_t

rtc_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- rtc_clock_source_t clock_source

  Clock source for the RTC block.

- uint32_t error_adjustment_value

  Value of the prescaler for error adjustment.

- rtc_error_adjustment_t error_adjustment_type

  How the prescaler value is applied.

- uint8_t alarm_ipl

  Alarm interrupt priority.

- uint8_t periodic_ipl

  Periodic interrupt priority.

- uint8_t carry_ipl

  Carry interrupt priority.

- void(* p_callback)(rtc_callback_args_t *p_args)

  Called from the ISR.

- void const * p_context

  Passed to the callback.

- void const * p_extend

  RTC hardware dependant configuration.

9.28.7.5 rtc_api_t

rtc_api_t

Detailed description

RTC driver structure. General RTC functions implemented at the HAL layer follow this API.

9.28.7.6 open

```
ssp_err_t(* rtc_api_t::open) (rtc_ctrl_t *const p_ctrl, rtc_cfg_t const *const p_cfg)
```

Detailed description

Open the RTC driver. Implemented as

- R_RTC_Open

Table 1015:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
p_cfg	in	Pointer to the configuration structure

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_cfg

Definition: `rtc_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `rtc_cfg_t`::`rtc_clock_source_t`

  Clock source for the RTC block.

  Enumerated as:

  – RTC_CLOCK_SOURCE_SUBCLK

  – RTC_CLOCK_SOURCE_LOCO

- rtc_cfg_t::error_adjustment_value

  Value of the prescaler for error adjustment.
- rtc_cfg_t::rtc_error_adjustment_t

  How the prescaler value is applied.

  Enumerated as:

  – RTC_ERROR_ADJUSTMENT_NONE

  – RTC_ERROR_ADJUSTMENT_ADD_PRESCALER

  – RTC_ERROR_ADJUSTMENT_SUBTRACT_PRESCALER
- rtc_cfg_t::alarm_ipl

  Alarm interrupt priority.
- rtc_cfg_t::periodic_ipl

  Periodic interrupt priority.
- rtc_cfg_t::carry_ipl

  Carry interrupt priority.
- rtc_cfg_t::p_callback

  Called from the ISR.
- rtc_cfg_t::p_context

  Passed to the callback.
- rtc_cfg_t::p_extend

  RTC hardware dependant configuration.

9.28.7.7 close

```
ssp_err_t(* rtc_api_t::close) (rtc_ctrl_t *const p_ctrl)
```

Detailed description

Close the RTC driver. Implemented as

- R_RTC_Close

Table 1016:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle.

Parameter p_ctrl

Definition: rtc_ctrl_t*const p_ctrl

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

9.28.7.8 calendarTimeSet

```
ssp_err_t(* rtc_api_t::calendarTimeSet) (rtc_ctrl_t *const p_ctrl, rtc_time_t
*p_time, bool clock_start)
```

Detailed description

Set the calendar time. Implemented as

- R_RTC_CalendarTimeSet

Table 1017:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
p_time	in	Pointer to a time structure that contains the time to set
clock_start	in	Flag that starts the clock right after it is set

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_time

Definition: `rtc_time_t*p_time`

Date and time structure defined in C standard library <time.h>

Parameter clock_start

`const`

9.28.7.9 calendarTimeGet

```
ssp_err_t(* rtc_api_t::calendarTimeGet) (rtc_ctrl_t *const p_ctrl, rtc_time_t
*p_time)
```

Detailed description

Get the calendar time. Implemented as

- R_RTC_CalendarTimeGet

Table 1018:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
p_time	out	Pointer to a time structure that contains the time to get

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_time

Definition: `rtc_time_t*p_time`

Date and time structure defined in C standard library <time.h>

9.28.7.10 calendarAlarmSet

```
ssp_err_t(* rtc_api_t::calendarAlarmSet) (rtc_ctrl_t *const p_ctrl,
rtc_alarm_time_t *p_alarm, bool irq_enable_flag)
```

Detailed description

Set the calendar alarm time. Implemented as

- R_RTC_CalendarAlarmSet

Table 1019:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
p_alarm	in	Pointer to an alarm structure that contains the alarm time to set
irq_enable_flag	in	Enable the ALARM irq if set

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_alarm

Definition: `rtc_alarm_time_t*p_alarm`

Alarm time setting structure

- `rtc_alarm_time_t::time`

  Time structure.

- `rtc_alarm_time_t::sec_match`

  Enable the alarm based on a match of the seconds field.

- `rtc_alarm_time_t::min_match`

  Enable the alarm based on a match of the minutes field.

- `rtc_alarm_time_t::hour_match`

  Enable the alarm based on a match of the hours field.

- `rtc_alarm_time_t::mday_match`

  Enable the alarm based on a match of the days field.

- `rtc_alarm_time_t::mon_match`

  Enable the alarm based on a match of the months field.

- `rtc_alarm_time_t::year_match`

  Enable the alarm based on a match of the years field.

- `rtc_alarm_time_t::dayofweek_match`

  Enable the alarm based on a match of the dayofweek field.

Parameter irq_enable_flag

`const`

9.28.7.11 calendarAlarmGet

```
ssp_err_t(* rtc_api_t::calendarAlarmGet) (rtc_ctrl_t *const p_ctrl,
rtc_alarm_time_t *p_alarm)
```

Detailed description

Get the calendar alarm time. Implemented as

- R_RTC_CalendarAlarmGet

Table 1020:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
p_alarm	out	Pointer to an alarm structure to fill up with the alarm time

Parameter p_ctrl

Definition: rtc_ctrl_t*const p_ctrl

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_alarm

Definition: rtc_alarm_time_t*p_alarm

Alarm time setting structure

- rtc_alarm_time_t::time

  Time structure.
- rtc_alarm_time_t::sec_match

  Enable the alarm based on a match of the seconds field.
- rtc_alarm_time_t::min_match

  Enable the alarm based on a match of the minutes field.
- rtc_alarm_time_t::hour_match

  Enable the alarm based on a match of the hours field.
- rtc_alarm_time_t::mday_match

  Enable the alarm based on a match of the days field.
- rtc_alarm_time_t::mon_match

  Enable the alarm based on a match of the months field.
- rtc_alarm_time_t::year_match

  Enable the alarm based on a match of the years field.
- rtc_alarm_time_t::dayofweek_match

  Enable the alarm based on a match of the dayofweek field.

9.28.7.12 calendarCounterStart

```
ssp_err_t(* rtc_api_t::calendarCounterStart) (rtc_ctrl_t *const p_ctrl)
```

Detailed description

Start the calendar counter. Implemented as

- R_RTC_CalendarCounterStart

Table 1021:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

9.28.7.13 calendarCounterStop

```
ssp_err_t(* rtc_api_t::calendarCounterStop) (rtc_ctrl_t *const p_ctrl)
```

Detailed description

Stop the calendar counter. Implemented as

- R_RTC_CalendarCounterStop

Table 1022:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

9.28.7.14 irqEnable

```
ssp_err_t(* rtc_api_t::irqEnable) (rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
```

Detailed description

Enable the alarm irq. Implemented as

- R_RTC_IrqEnable

Table 1023:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

9.28.7.15 irqDisable

```
ssp_err_t(* rtc_api_t::irqDisable) (rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
```

Detailed description

Disable the alarm irq. Implemented as

- R_RTC_IrqDisable

Table 1024:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

9.28.7.16 periodicIrqRateSet

```
ssp_err_t(* rtc_api_t::periodicIrqRateSet) (rtc_ctrl_t *const p_ctrl,
rtc_periodic_irq_select_t rate)
```

Detailed description

Set the periodic irq rate Implemented as

- R_RTC_PeriodicIrqRateSet

Table 1025:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to RTC device handle
rate	in	Rate of periodic interrupts

Parameter p_ctrl

Definition: `rtc_ctrl_t*const p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter rate

Definition: `rtc_periodic_irq_select_trate`

Periodic Interrupt select

9.28.7.17 infoGet

```
ssp_err_t(* rtc_api_t::infoGet) (rtc_ctrl_t *p_ctrl, rtc_info_t *p_rtc_info)
```

Detailed description

Return the currently configure clock source for the RTC

Implemented as

- R_RTC_InfoGet

Table 1026:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control handle structure
p_rtc_info	out	Pointer to RTC information structure

Parameter p_ctrl

Definition: `rtc_ctrl_t*p_ctrl`

RTC control block. Allocate an instance specific control block to pass into the RTC API calls. Implemented asrtc_instance_ctrl_t

Parameter p_rtc_info

Definition: `rtc_info_t*p_rtc_info`

RTC Information Structure for information returned by infoGet()

- `rtc_info_t::clock_source`

  Clock source for the RTC block.

9.28.7.18 versionGet

```
ssp_err_t(* rtc_api_t::versionGet) (ssp_version_t *version)
```

Detailed description

Gets version and stores it in provided pointer p_version. Implemented as

- R_RTC_VersionGet

Table 1027:Parameters

Name	Direction	Description
p_version	out	Code and API version used

Parameter p_version

9.28.7.19 rtc_instance_t

rtc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- rtc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- rtc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- rtc_api_t const * p_api

  Pointer to the API structure for this instance.

9.29 SD/MMC Interface

Interface for accessing SD, eMMC, and SDIO devices.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

9.29.1 Summary

The r_sdmmc interface provides standard SD and eMMC media functionality. A complete file system can be implemented with FileX, sf_el_fx, sf_block_media_sdmmc and r_sdmmc modules. This driver also supports SDIO. The SD/MMC interface is implemented by:

- SDMMC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

SD/MMC description: SD/MMC Driver and SDIO Driver

9.29.2 Interface API

sdmmc_api_t

Function name	Description
.open	Open an SD/MMC device. If the device is a card, the card must be plugged in prior to calling this API. This API blocks until the device initialization procedure is complete.
.close	Close open SD/MMC device.
.read	Read data from an SD/MMC channel. This API is not supported for SDIO devices.
.write	Write data to SD/MMC channel. This API is not supported for SDIO devices.
.control	The Control function sends control commands to and receives info from the SD/MMC port.
.readIo	Read one byte of I/O data from an SDIO device. This API is not supported for SD or eMMC memory devices.
.writeIo	Write one byte of I/O data to an SDIO device. This API is not supported for SD or eMMC memory devices.
.readIoExt	Read multiple bytes or blocks of I/O data from an SDIO device. This API is not supported for SD or eMMC memory devices.
.writeIoExt	Write multiple bytes or blocks of I/O data to an SDIO device. This API is not supported for SD or eMMC memory devices.
.ioIntEnable	Enables SDIO interrupt for SD/MMC instance. This API is not supported for SD or eMMC memory devices.
.versionGet	Returns the version of the SD/MMC driver.
.infoGet	Get SD/MMC device info.
.erase	Erase SD/MMC sectors. The sector size for erase is fixed at 512 bytes. This API is not supported for SDIO devices.

9.29.3 Data structures

- sdmmc_hw_t
- sdmmc_info_t
- sdmmc_callback_args_t
- sdmmc_cfg_t
- sdmmc_instance_t

9.29.4 Enumerations

- sdmmc_ready_status_t
- sdmmc_card_type_t
- sdmmc_media_type_t
- sdmmc_bus_width_t
- sdmmc_io_transfer_mode_t
- sdmmc_io_address_mode_t
- sdmmc_io_write_mode_t
- sdmmc_event_t

9.29.5 Typedefs

- sdmmc_ctrl_t

9.29.6 Defines

- #define SDMMC_API_VERSION_MAJOR
  Initial value:`(1U)`
- #define SDMMC_API_VERSION_MINOR
  Initial value:`(2U)`
- #define SDMMC_MAX_BLOCK_SIZE
  Initial value:`(512U)`

9.29.7 API Data

9.29.7.1 sdmmc_ready_status_t

```
sdmmc_ready_status_t
```

Detailed description

SD/MMC status

Enumerated values

Name	Description
SDMMC_STATUS_CARD_NOT_READY	SD card or eMMC device has not been initialized.
SDMMC_STATUS_CARD_READY	SD card or eMMC device has been initialized and is ready to access.

9.29.7.2 sdmmc_card_type_t

`sdmmc_card_type_t`

Detailed description

SD/MMC media uses SD protocol or MMC protocol.

Enumerated values

Name	Description
SDMMC_CARD_TYPE_MMC	The media is an eMMC device.
SDMMC_CARD_TYPE_SD	The media is an SD card.

9.29.7.3 sdmmc_media_type_t

`sdmmc_media_type_t`

Detailed description

SD/MMC media is embedded or it can be inserted and removed.

Enumerated values

Name	Description
SDMMC_MEDIA_TYPE_EMBEDDED	The media is an embedded card, or eMMC device.
SDMMC_MEDIA_TYPE_CARD	The media is an pluggable card.

9.29.7.4 sdmmc_bus_width_t

`sdmmc_bus_width_t`

Detailed description

SD/MMC data bus is 1, 4 or 8 bits wide.

Enumerated values

Name	Description
SDMMC_BUS_WIDTH_1_BIT	Data bus is 1 bit wide.
SDMMC_BUS_WIDTH_4_BITS	Data bus is 4 bits wide.
SDMMC_BUS_WIDTH_8_BITS	Data bus is 8 bits wide.

9.29.7.5 sdmmc_io_transfer_mode_t

`sdmmc_io_transfer_mode_t`

Enumerated values

Name	Description
SDMMC_IO_MODE_TRANSFER_BYTE	
SDMMC_IO_MODE_TRANSFER_BLOCK	

9.29.7.6 sdmmc_io_address_mode_t

`sdmmc_io_address_mode_t`

Enumerated values

Name	Description
SDMMC_IO_ADDRESS_MODE_FIXED	
SDMMC_IO_ADDRESS_MODE_INCREMENT	

9.29.7.7 sdmmc_io_write_mode_t

`sdmmc_io_write_mode_t`

Enumerated values

Name	Description
SDMMC_IO_WRITE_MODE_NO_READ	
SDMMC_IO_WRITE_READ_AFTER_WRITE	

9.29.7.8 sdmmc_event_t

```
sdmmc_event_t
```

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
SDMMC_EVENT_CARD_REMOVED	Card removed event.
SDMMC_EVENT_CARD_INSERTED	Card inserted event.
SDMMC_EVENT_ACCESS	Access event.
SDMMC_EVENT_SDIO	IO event.
SDMMC_EVENT_TRANSFER_COMPLETE	Read or write complete.
SDMMC_EVENT_TRANSFER_ERROR	Read or write failed.
SDMMC_EVENT_NONE	No event.

9.29.7.9 sdmmc_ctrl_t

```
typedef void sdmmc_ctrl_t
```

Detailed description

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented as

- sdmmc_instance_ctrl_t

9.29.8 API Structures

9.29.8.1 sdmmc_hw_t

sdmmc_hw_t

Detailed description

Channel, media type, bus width defined by the hardware.

Variables

- uint8_t channel

  Channel of SD/MMC host interface.

- sdmmc_media_type_t media_type

  Embedded or pluggable card.

- sdmmc_bus_width_t bus_width

  Device bus width is 1, 4 or 8 bits wide.

9.29.8.2 sdmmc_info_t

sdmmc_info_t

Detailed description

Status and other information obtained from the media device.

Variables

- sdmmc_card_type_t card_type

  SD or eMMC.

- bool ready

  False if card was removed (only applies if MCU supports card detection and SDnCD pin is connected).True otherwise.If ready is false, the driver must be closed, then reopened with a card inserted.

- bool hc

  true = Card is High Capacity card

- bool sdio

  true = SDIO present

- bool write_protected

  true = Card is write protected

- bool transfer_in_progress

  true = Card is busy

- uint8_t csd_version

  CSD version.

- uint8_t device_type

  Speed and data rate (eMMC)

- sdmmc_bus_width_t bus_width

  Current media bus width.

- uint8_t hs_timing

  High Speed status.
- uint32_t sdhi_rca

  Relative Card Address.
- uint32_t max_clock_rate

  Maximum clock rate for media card.
- uint32_t clock_rate

  Current clock rate.
- uint32_t sector_size

  Sector size.
- uint32_t sector_count

  Sector count.
- uint32_t erase_sector_count

  Minimum erasable unit (in 512 byte sectors)

9.29.8.3 sdmmc_callback_args_t

sdmmc_callback_args_t

Detailed description

Callback function parameter data

Variables

- sdmmc_event_t event

  The event can be used to identify what caused the callback.
- void const * p_context

  Placeholder for user data.

9.29.8.4 sdmmc_cfg_t

sdmmc_cfg_t

Detailed description

SD/MMC Configuration

Variables

- sdmmc_hw_t hw

  Channel, media type, bus width defined by the hardware.
- transfer_instance_t const * p_lower_lvl_transfer

  Transfer instance used to move data with DMA or DTC.

- void(* p_callback)(sdmmc_callback_args_t *p_args)

  Pointer to callback function.
- void const * p_context

  User defined context passed into callback function.
- void const * p_extend

  SD/MMC hardware dependent configuration.
- uint8_t access_ipl

  Access interrupt priority.
- uint8_t sdio_ipl

  SDIO interrupt priority.
- uint8_t card_ipl

  Card interrupt priority.
- uint8_t dma_req_ipl

  DMA request interrupt priority.

9.29.8.5 sdmmc_api_t

sdmmc_api_t

Detailed description

SD/MMC functions implemented at the HAL layer API.

9.29.8.6 open

```
ssp_err_t(* sdmmc_api_t::open) (sdmmc_ctrl_t *const p_ctrl, sdmmc_cfg_t const
*const p_cfg)
```

Detailed description

Open an SD/MMC device. If the device is a card, the card must be plugged in prior to calling this API. This API blocks until the device initialization procedure is complete.

Implemented asR_SDMMC_Open

Table 1028:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to SD/MMC instance control block.
p_cfg	in	Pointer to SD/MMC instance configuration structure.

Parameter p_ctrl

Definition: sdmmc_ctrl_t*const p_ctrl

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_cfg

Definition: sdmmc_cfg_t const *const p_cfg

SD/MMC Configuration

- sdmmc_cfg_t::sdmmc_hw_t

  Channel, media type, bus width defined by the hardware.
- sdmmc_cfg_t::transfer_instance_t

  Transfer instance used to move data with DMA or DTC.
- sdmmc_cfg_t::p_callback

  Pointer to callback function.
- sdmmc_cfg_t::p_context

  User defined context passed into callback function.
- sdmmc_cfg_t::p_extend

  SD/MMC hardware dependent configuration.
- sdmmc_cfg_t::access_ipl

  Access interrupt priority.
- sdmmc_cfg_t::sdio_ipl

  SDIO interrupt priority.
- sdmmc_cfg_t::card_ipl

  Card interrupt priority.
- sdmmc_cfg_t::dma_req_ipl

  DMA request interrupt priority.

9.29.8.7 close

```
ssp_err_t(* sdmmc_api_t::close) (sdmmc_ctrl_t *const p_ctrl)
```

Detailed description

Close open SD/MMC device.

Implemented asR_SDMMC_Close

Table 1029:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

9.29.8.8 read

```
ssp_err_t(* sdmmc_api_t::read) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const start_sector, uint32_t const sector_count)
```

Detailed description

Read data from an SD/MMC channel. This API is not supported for SDIO devices.

Implemented asR_SDMMC_Read

Table 1030:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_dest	out	Pointer to data buffer to read data to.
start_sector	in	First sector address to read.
sector_count	in	Number of sectors to read. All sectors must be in the range of sector_count.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter start_sector

`uint32_t`

Parameter sector_count

uint32_t

9.29.8.9 write

```
ssp_err_t(* sdmmc_api_t::write) (sdmmc_ctrl_t *const p_ctrl, uint8_t const *const
p_source, uint32_t const start_sector, uint32_t const sector_count)
```

Detailed description

Write data to SD/MMC channel. This API is not supported for SDIO devices.

Implemented asR_SDMMC_Write

Table 1031:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_source	in	Pointer to data buffer to write data from.
start_sector	in	First sector address to write to.
sector_count	in	Number of sectors to write. All sectors must be in the range of sector_count.

Parameter p_ctrl

Definition: sdmmc_ctrl_t*const p_ctrl

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_source

uint8_t

Parameter start_sector

uint32_t

Parameter sector_count

uint32_t

9.29.8.10 control

```
ssp_err_t(* sdmmc_api_t::control) (sdmmc_ctrl_t *const p_ctrl, ssp_command_t
const command, void *p_data)
```

Detailed description

The Control function sends control commands to and receives info from the SD/MMC port.

Implemented asR_SDMMC_Control

Table 1032:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
command	in	Command to execute. The list of supported commands is below.
p_data	inout	Pointer to data in or out. For each command, this data should be cast as follows: • SSP_COMMAND_GET_SECTOR_COUNT : [out] (uint32_t *) p_data • SSP_COMMAND_GET_SECTOR_SIZE : [out] (uint32_t *) p_data • SSP_COMMAND_GET_WRITE_PROTECTED : [out] (bool *) p_data • SSP_COMMAND_SET_BLOCK_SIZE : [in] (uint32_t *) p_data

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter command

Parameter p_data

`const`

9.29.8.11 readIo

```
ssp_err_t(* sdmmc_api_t::readIo) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const
p_data, uint32_t const function, uint32_t const address)
```

Detailed description

Read one byte of I/O data from an SDIO device. This API is not supported for SD or eMMC memory devices.

Implemented asR_SDMMC_ReadIo

Table 1033:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_data	out	Pointer to location to store data byte.
function	in	SDIO Function Number.
address	in	SDIO register address.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_data

`uint8_t`

Parameter function

`uint32_t`

Parameter address

`uint32_t`

9.29.8.12 writeIo

```
ssp_err_t(* sdmmc_api_t::writeIo) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const
p_data, uint32_t const function, uint32_t const address, sdmmc_io_write_mode_t
const read_after_write)
```

Detailed description

Write one byte of I/O data to an SDIO device. This API is not supported for SD or eMMC memory devices.

Implemented asR_SDMMC_WriteIo

Table 1034:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_data	inout	Pointer to data byte to write. Read data is also provided here if read_after_write is true.

Table 1034:Parameters (Continued)

Name	Direction	Description
function	in	SDIO Function Number.
address	in	SDIO register address.
read_after_write	in	Set to true for reading after writing.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_data

`uint8_t`

Parameter function

`uint32_t`

Parameter address

`uint32_t`

Parameter read_after_write

Definition: `sdmmc_io_write_mode_tconst read_after_write`

9.29.8.13 readIoExt

```
ssp_err_t(* sdmmc_api_t::readIoExt) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const
p_dest, uint32_t const function, uint32_t const address, uint32_t *const count,
sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t address_mode)
```

Detailed description

Read multiple bytes or blocks of I/O data from an SDIO device. This API is not supported for SD or eMMC memory devices.

Implemented asR_SDMMC_ReadIoExt

Table 1035:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_dest	out	Pointer to data buffer to read data to.

Table 1035:Parameters (Continued)

Name	Direction	Description
function	in	SDIO Function Number.
address	in	SDIO register address.
count	in	Number of bytes or blocks to read, maximum 512 bytes or 511 blocks.
transfer_mode	in	Byte mode = 0, Block mode = 1.
address_mode	in	0 = fixed address, 1 = Incrementing address.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter function

`uint32_t`

Parameter address

`uint32_t`

Parameter count

`uint32_t`

Parameter transfer_mode

Parameter address_mode

9.29.8.14 writeIoExt

```
ssp_err_t(* sdmmc_api_t::writeIoExt) (sdmmc_ctrl_t *const p_ctrl, uint8_t const
*const p_source, uint32_t const function, uint32_t const address, uint32_t const
count, sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t
address_mode)
```

Detailed description

Write multiple bytes or blocks of I/O data to an SDIO device. This API is not supported for SD or eMMC memory devices.

Implemented asR_SDMMC_WriteIoExt

Table 1036:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_source	in	Pointer to data buffer to write data from.
function_number	in	SDIO Function Number.
address	in	SDIO register address.
count	in	Number of bytes or blocks to write, maximum 512 bytes or 511 blocks.
transfer_mode	in	Byte mode = 0, Block mode = 1.
address_mode	in	0 = fixed address, 1 = Incrementing address.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_source

`uint8_t`

Parameter function_number

Parameter address

`uint32_t`

Parameter count

`uint32_t`

Parameter transfer_mode

Parameter address_mode

9.29.8.15 IoIntEnable

```
ssp_err_t(* sdmmc_api_t::IoIntEnable) (sdmmc_ctrl_t *const p_ctrl, bool enable)
```

Detailed description

Enables SDIO interrupt for SD/MMC instance. This API is not supported for SD or eMMC memory devices.

Implemented asR_SDMMC_IoIntEnable

Table 1037:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
enable	in	Interrupt enable = true, interrupt disable = false.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter enable

`const`

9.29.8.16 versionGet

```
ssp_err_t(* sdmmc_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Returns the version of the SD/MMC driver.

Implemented asR_SDMMC_VersionGet

Table 1038:Parameters

Name	Direction	Description
p_version	out	Pointer to return version information to.

Parameter p_version

9.29.8.17 infoGet

```
ssp_err_t(* sdmmc_api_t::infoGet) (sdmmc_ctrl_t *const p_ctrl, sdmmc_info_t
*const p_info)
```

Detailed description

Get SD/MMC device info.

Implemented asR_SDMMC_InfoGet

Table 1039:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
p_info	out	Pointer to return device information to.

Parameter p_ctrl

Definition: `sdmmc_ctrl_t*const p_ctrl`

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter p_info

Definition: `sdmmc_info_t*const p_info`

Status and other information obtained from the media device.

- `sdmmc_info_t::card_type`

  SD or eMMC.

- `sdmmc_info_t::ready`

  False if card was removed (only applies if MCU supports card detection and SDnCD pin is connected). True otherwise. If ready is false, the driver must be closed, then reopened with a card inserted.

- `sdmmc_info_t::hc`

  true = Card is High Capacity card

- `sdmmc_info_t::sdio`

  true = SDIO present

- `sdmmc_info_t::write_protected`

  true = Card is write protected

- `sdmmc_info_t::transfer_in_progress`

  true = Card is busy

- `sdmmc_info_t::csd_version`

  CSD version.

- `sdmmc_info_t::device_type`

  Speed and data rate (eMMC)

- `sdmmc_info_t::bus_width`

  Current media bus width.

- sdmmc_info_t::hs_timing

  High Speed status.
- sdmmc_info_t::sdhi_rca

  Relative Card Address.
- sdmmc_info_t::max_clock_rate

  Maximum clock rate for media card.
- sdmmc_info_t::clock_rate

  Current clock rate.
- sdmmc_info_t::sector_size

  Sector size.
- sdmmc_info_t::sector_count

  Sector count.
- sdmmc_info_t::erase_sector_count

  Minimum erasable unit (in 512 byte sectors)

9.29.8.18 erase

```
ssp_err_t(* sdmmc_api_t::erase) (sdmmc_ctrl_t *const p_ctrl, uint32_t const
start_sector, uint32_t const sector_count)
```

Detailed description

Erase SD/MMC sectors. The sector size for erase is fixed at 512 bytes. This API is not supported for SDIO devices.

Implemented asR_SDMMC_Erase

Table 1040:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to an open SD/MMC instance control block.
start_sector	in	First sector to erase. Must be a multiple of erase_sector_count.
sector_count	in	Number of sectors to erase. Must be a multiple of erase_sector_count. All sectors must be in the range of sector_count.

Parameter p_ctrl

Definition: sdmmc_ctrl_t*const p_ctrl

SD/MMC control block. Allocate an instance specific control block to pass into the SD/MMC API calls. Implemented assdmmc_instance_ctrl_t

Parameter start_sector

`uint32_t`

Parameter sector_count

`uint32_t`

9.29.8.19 sdmmc_instance_t

sdmmc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- sdmmc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- sdmmc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- sdmmc_api_t const * p_api

  Pointer to the API structure for this instance.

9.30 SLCDC Interface

Interface for Segment LCD controllers.

9.30.1 Summary

This driver uses the Segment LCD controller (SLCDC) to display data on a Segment LCD.

Implemented by: SLCDC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

SLCDC Interface description: Segment LCD Driver

9.30.2 Interface API

slcdc_api_t

Function name	Description
.open	Open SLCD device.
.write	Write data to SLCD segment. Specifies the initial display data. Except for 8-time slice, store values in the lower 4 bits when writing to the A-pattern area, and in the upper 4 bits when writing to the B-pattern area. The display data is stored in the display data register.
.modify	Rewrite data in the SLCD segment. Rewrites the LCD display data in 1-bit units. If a bit is not specified for rewriting, the value stored in the bit is held as it is. Specifies the data to rewrite
.start	Enable display on the SLCD. Displays the specified data on the LCD. Before that data should be written to the segments.
.stop	Disable display on the SLCD. Stops displaying data on the SLCD.
.contrastIncrease	Increase the display contrast. Increase by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator
.contrastDecrease	Decrease the display contrast. Decrease by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator
.setdisplayArea	Set LCD display area. This function sets a specified display area, A-pattern or B-pattern. This function can be used to set blink on where A-pattern and B-pattern area data will be alternately displayed.When using blinking, the RTC is required to operate before this function is executed. To configure the RTC, follow the steps below. 1) Open RTC 2) Set Periodic interrupt request, 1/2 second 3) Start RTC counter 4) Enable IRQ, RTC_EVENT_PERIODIC_IRQ Refer to the User's Manual: Hardware for the detailed procedure.
.close	Close display device.
.versionGet	Get version.

9.30.3 Data structures

- slcdc_cfg_t
- slcdc_instance_t

9.30.4 Enumerations

- slcdc_display_state_t
- slcdc_bias_method_t
- slcdc_time_slice_t
- slcdc_wave_form_t
- slcdc_drive_volt_gen_t
- slcdc_display_area_control_blink_t
- slcdc_display_area_t
- slcdc_display_on_off_t
- slcdc_display_enable_disable_t
- slcdc_display_clock_t
- slcdc_clk_div_t

9.30.5 Typedefs

- slcdc_size_t
- slcdc_ctrl_t

9.30.6 Defines

- #define SLCDC_API_VERSION_MAJOR

  Initial value:`(1U)`

  Register definitions, common services and error codes. Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SLCDC_API_VERSION_MINOR

  Initial value:`(3U)`
- #define SLCDC_VOL_MIN

  Initial value:`(4)`
- #define SLCDC_VOL_MAX

  Initial value:`(19)`

- #define SLCDC_VOL_MAX_4BIAS
  Initial value:`(10)`
- #define SLCDC_CFG_REF_VCC
  Initial value:`(12)`
- #define SLCDC_BOOST_COUNTER
  Initial value:`(5) /* The number of times of performing waiting for 100 ms */`
- #define SLCDC_BOOST_WAIT
  Initial value:`(uint32_t)(100000) /* Waiting for 100ms */`
- #define SLCDC_SETUP_WAIT
  Initial value:`(5) /* Waiting for 5ms */`
- #define MAX_NUM_SEG
  Initial value:`(51U)`

9.30.7 API Data

9.30.7.1 slcdc_display_state_t

`slcdc_display_state_t`

Detailed description

Display interface operation state

Enumerated values

Name	Description
SLCDC_DISPLAY_STATE_CLOSED	Display closed.
SLCDC_DISPLAY_STATE_OPENED	Display opened.

9.30.7.2 slcdc_bias_method_t

`slcdc_bias_method_t`

Detailed description

LCD display bias method.

Enumerated values

Name	Description
SLCDC_BIAS_2	1/2 bias method
SLCDC_BIAS_3	1/3 bias method
SLCDC_BIAS_4	1/4 bias method

9.30.7.3 slcdc_time_slice_t

`slcdc_time_slice_t`

Detailed description

Time slice of LCD display.

Enumerated values

Name	Description
SLCDC_STATIC	Static.
SLCDC_SLICE_2	2-time slice
SLCDC_SLICE_3	3-time slice
SLCDC_SLICE_4	4-time slice
SLCDC_SLICE_8	8-time slice

9.30.7.4 slcdc_wave_form_t

`slcdc_wave_form_t`

Detailed description

LCD display waveform select.

Enumerated values

Name	Description
SLCDC_WAVE_A	Waveform A.
SLCDC_WAVE_B	Waveform B.

9.30.7.5 slcdc_drive_volt_gen_t

`slcdc_drive_volt_gen_t`

Detailed description

LCD Drive Voltage Generator Select.

Enumerated values

Name	Description
SLCDC_VOLT_EXTERNAL	External resistance division method.
SLCDC_VOLT_INTERNAL	Internal voltage boosting method.
SLCDC_VOLT_CAPACITOR	Capacitor split method.

9.30.7.6 slcdc_display_area_control_blink_t

`slcdc_display_area_control_blink_t`

Detailed description

Display Data Area Control

Enumerated values

Name	Description
SLCDC_NOT_BLINKING	Alternately displaying A-pattern and B-pattern area data.
SLCDC_BLINKING	Displaying an A-pattern or B-pattern area data.

9.30.7.7 slcdc_display_area_t

`slcdc_display_area_t`

Detailed description

Display Area data

Enumerated values

Name	Description
SLCDC_DISP_A	Displaying an A-pattern area data.
SLCDC_DISP_B	Displaying an B-pattern area data.

Name	Description
SLCDC_DISP_BLINK	

9.30.7.8 slcdc_display_on_off_t

`slcdc_display_on_off_t`

Detailed description

LCD Display Enable/Disable

Enumerated values

Name	Description
SLCDC_DISP_OFF	Display off.
SLCDC_DISP_ON	Display on.

9.30.7.9 slcdc_display_enable_disable_t

`slcdc_display_enable_disable_t`

Detailed description

LCD Display output enable

Enumerated values

Name	Description
SLCDC_DISP_DISABLE	Output ground level to segment/common pin.
SLCDC_DISP_ENABLE	Output enable.

9.30.7.10 slcdc_display_clock_t

`slcdc_display_clock_t`

Detailed description

LCD Display clock selection

Enumerated values

Name	Description
SLCDC_CLOCK_LOCO	Display clock source LOCO.
SLCDC_CLOCK_SOSC	Display clock source SOSC.
SLCDC_CLOCK_MOSC	Display clock source MOSC.
SLCDC_CLOCK_HOCO	Display clock source HOCO.

9.30.7.11 slcdc_clk_div_t

`slcdc_clk_div_t`

Detailed description

LCD clock settings

Enumerated values

Name	Description
SLCDC_CLK_DIVISOR_LOCO_4	LOCO Clock/4.
SLCDC_CLK_DIVISOR_LOCO_8	LOCO Clock/8.
SLCDC_CLK_DIVISOR_LOCO_16	LOCO Clock/16.
SLCDC_CLK_DIVISOR_LOCO_32	LOCO Clock/32.
SLCDC_CLK_DIVISOR_LOCO_64	LOCO Clock/64.
SLCDC_CLK_DIVISOR_LOCO_128	LOCO Clock/128.
SLCDC_CLK_DIVISOR_LOCO_256	LOCO Clock/256.
SLCDC_CLK_DIVISOR_LOCO_512	LOCO Clock/512.
SLCDC_CLK_DIVISOR_LOCO_1024	LOCO Clock/1024.
SLCDC_CLK_DIVISOR_HOCO_256	HOCO Clock/256.
SLCDC_CLK_DIVISOR_HOCO_512	HOCO Clock/512.
SLCDC_CLK_DIVISOR_HOCO_1024	HOCO Clock/1024.
SLCDC_CLK_DIVISOR_HOCO_2048	HOCO Clock/2048.

Name	Description
SLCDC_CLK_DIVISOR_HOCO_4096	HOCO Clock/4096.
SLCDC_CLK_DIVISOR_HOCO_8192	HOCO Clock/8192.
SLCDC_CLK_DIVISOR_HOCO_16384	HOCO Clock/16384.
SLCDC_CLK_DIVISOR_HOCO_32768	HOCO Clock/32768.
SLCDC_CLK_DIVISOR_HOCO_65536	HOCO Clock/65536.
SLCDC_CLK_DIVISOR_HOCO_131072	HOCO Clock/131072.
SLCDC_CLK_DIVISOR_HOCO_262144	HOCO Clock/262144.
SLCDC_CLK_DIVISOR_HOCO_524288	HOCO Clock/524288.

9.30.7.12 slcdc_size_t

```
typedef uint8_t slcdc_size_t
```

Detailed description

Size definition for slcdc

9.30.7.13 slcdc_ctrl_t

```
typedef void slcdc_ctrl_t
```

Detailed description

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented as

- slcdc_instance_ctrl_tSLCDC control block

9.30.8 API Structures

9.30.8.1 slcdc_cfg_t

slcdc_cfg_t

Detailed description

SLCDC configuration block

Variables

- slcdc_display_clock_t slcdc_clock

  LCD clock source (LCDSCKSEL)

- slcdc_clk_div_t slcdc_clock_setting

  LCD clock setting (LCDC0)
- slcdc_bias_method_t bias_method

  LCD display bias method select (LBAS bit).
- slcdc_time_slice_t time_slice

  Time slice of LCD display select (LDTY bit)
- slcdc_wave_form_t wave_form

  LCD display waveform select (LWAVE bit).
- slcdc_drive_volt_gen_t drive_volt_gen

  LCD Drive Voltage Generator Select (MDSTET bit).

9.30.8.2 slcdc_api_t

slcdc_api_t

Detailed description

SLCDC functions implemented at the HAL layer will follow this API.

9.30.8.3 open

```
ssp_err_t(* slcdc_api_t::open) (slcdc_ctrl_t *const p_ctrl, slcdc_cfg_t const
*const p_cfg)
```

Detailed description

Open SLCD device. Implemented as

- R_SLCDC_Open

Table 1041:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to display interface control block. Must be declared by user. Value set here.
p_cfg	in	Pointer to display configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

Parameter p_cfg

Definition: `slcdc_cfg_t const *const p_cfg`

SLCDC configuration block

- `slcdc_cfg_t`::`slcdc_display_clock_t`

  LCD clock source (LCDSCKSEL)

  Enumerated as:

  – SLCDC_CLOCK_LOCO

  – SLCDC_CLOCK_SOSC

  – SLCDC_CLOCK_MOSC

  – SLCDC_CLOCK_HOCO

- `slcdc_cfg_t`::`slcdc_clk_div_t`

  LCD clock setting (LCDC0)

  Enumerated as:

  – SLCDC_CLK_DIVISOR_LOCO_4

  – SLCDC_CLK_DIVISOR_LOCO_8

  – SLCDC_CLK_DIVISOR_LOCO_16

  – SLCDC_CLK_DIVISOR_LOCO_32

  – SLCDC_CLK_DIVISOR_LOCO_64

  – SLCDC_CLK_DIVISOR_LOCO_128

  – SLCDC_CLK_DIVISOR_LOCO_256

  – SLCDC_CLK_DIVISOR_LOCO_512

  – SLCDC_CLK_DIVISOR_LOCO_1024

  – SLCDC_CLK_DIVISOR_HOCO_256

  – SLCDC_CLK_DIVISOR_HOCO_512

  – SLCDC_CLK_DIVISOR_HOCO_1024

  – SLCDC_CLK_DIVISOR_HOCO_2048

  – SLCDC_CLK_DIVISOR_HOCO_4096

  – SLCDC_CLK_DIVISOR_HOCO_8192

  – SLCDC_CLK_DIVISOR_HOCO_16384

  – SLCDC_CLK_DIVISOR_HOCO_32768

  - SLCDC_CLK_DIVISOR_HOCO_65536
  - SLCDC_CLK_DIVISOR_HOCO_131072
  - SLCDC_CLK_DIVISOR_HOCO_262144
  - SLCDC_CLK_DIVISOR_HOCO_524288
- `slcdc_cfg_t`::`slcdc_bias_method_t`

  LCD display bias method select (LBAS bit).

  Enumerated as:
  - SLCDC_BIAS_2
  - SLCDC_BIAS_3
  - SLCDC_BIAS_4
- `slcdc_cfg_t`::`slcdc_time_slice_t`

  Time slice of LCD display select (LDTY bit)

  Enumerated as:
  - SLCDC_STATIC
  - SLCDC_SLICE_2
  - SLCDC_SLICE_3
  - SLCDC_SLICE_4
  - SLCDC_SLICE_8
- `slcdc_cfg_t`::`slcdc_wave_form_t`

  LCD display waveform select (LWAVE bit).

  Enumerated as:
  - SLCDC_WAVE_A
  - SLCDC_WAVE_B
- `slcdc_cfg_t`::`slcdc_drive_volt_gen_t`

  LCD Drive Voltage Generator Select (MDSTET bit).

  Enumerated as:
  - SLCDC_VOLT_EXTERNAL
  - SLCDC_VOLT_INTERNAL
  - SLCDC_VOLT_CAPACITOR

9.30.8.4 write

```
ssp_err_t(* slcdc_api_t::write) (slcdc_ctrl_t *const p_ctrl, slcdc_size_t const
start_segment, slcdc_size_t const *const p_data, slcdc_size_t const segment_count)
```

Detailed description

Write data to SLCD segment. Specifies the initial display data. Except for 8-time slice, store values in the lower 4 bits when writing to the A-pattern area, and in the upper 4 bits when writing to the B-pattern area. The display data is stored in the display data register. Implemented as

- R_SLCDC_Write

Table 1042:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
start_segment	in	Specify the start segment number to be written.
p_data	in	pointer to the display data to be written to the specified segments
segment_count	in	Number of segments to be written

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

Parameter start_segment

Definition: `slcdc_size_tconst start_segment`

Size definition for slcdc

Parameter p_data

Definition: `slcdc_size_tconst *const p_data`

Size definition for slcdc

Parameter segment_count

Definition: `slcdc_size_tconst segment_count`

Size definition for slcdc

9.30.8.5 modify

```
ssp_err_t(* slcdc_api_t::modify) (slcdc_ctrl_t *const p_ctrl, slcdc_size_t const
segment, slcdc_size_t const data_mask, slcdc_size_t const data)
```

Detailed description

Rewrite data in the SLCD segment. Rewrites the LCD display data in 1-bit units. If a bit is not specified for rewriting, the value stored in the bit is held as it is. Specifies the data to rewrite Implemented as

- R_SLCDC_Modify

Table 1043:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
seg	in	Specify the segment to be written.
data_mask	in	Mask the data being displayed. Set 0 to the bit to be rewritten and set 1 to the other bits. Multiple bits can be rewritten. Setting value of data_mask, Bit 3 - 0xf7 Bit 2 - 0xfb Bit 1 - 0xfd Bit 0 - 0xfe
data	in	Specify display data to rewrite to the specified segment.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

Parameter seg

Definition: `slcdc_size_tconst segment`

Size definition for slcdc

Parameter data_mask

Definition: `slcdc_size_tconst data_mask`

Size definition for slcdc

Parameter data

Definition: `slcdc_size_tconst data`

Size definition for slcdc

9.30.8.6 start

```
ssp_err_t(* slcdc_api_t::start) (slcdc_ctrl_t *const p_ctrl)
```

Detailed description

Enable display on the SLCD. Displays the specified data on the LCD. Before that data should be written to the segments. Implemented as

- R_SLCDC_Start

Table 1044:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

9.30.8.7 stop

```
ssp_err_t(* slcdc_api_t::stop) (slcdc_ctrl_t *const p_ctrl)
```

Detailed description

Disable display on the SLCD. Stops displaying data on the SLCD. Implemented as

- R_SLCDC_Stop

Table 1045:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

9.30.8.8 contrastIncrease

```
ssp_err_t(* slcdc_api_t::contrastIncrease) (slcdc_ctrl_t *const p_ctrl)
```

Detailed description

Increase the display contrast. Increase by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator Implemented as

- R_SLCDC_ContrastIncrease

Table 1046:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

9.30.8.9 contrastDecrease

```
ssp_err_t(* slcdc_api_t::contrastDecrease) (slcdc_ctrl_t *const p_ctrl)
```

Detailed description

Decrease the display contrast. Decrease by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator Implemented as

- R_SLCDC_ContrastDecrease

Table 1047:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

9.30.8.10 setdisplayArea

```
ssp_err_t(* slcdc_api_t::setdisplayArea) (slcdc_ctrl_t *const p_ctrl,
slcdc_display_area_t const display_area)
```

Detailed description

Set LCD display area. This function sets a specified display area, A-pattern or B-pattern. This function can be used to set blink on where A-pattern and B-pattern area data will be alternately displayed.

When using blinking, the RTC is required to operate before this function is executed. To configure the RTC, follow the steps below. 1) Open RTC 2) Set Periodic interrupt request, 1/2 second 3) Start RTC counter 4) Enable IRQ, RTC_EVENT_PERIODIC_IRQ Refer to the User's Manual: Hardware for the detailed procedure.

Implemented as

- R_SLCDC_SetdisplayArea()

Table 1048:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.
display_area	in	Display area to be used, A-pattern or B-pattern area.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

Parameter display_area

9.30.8.11 close

```
ssp_err_t(* slcdc_api_t::close) (slcdc_ctrl_t *const p_ctrl)
```

Detailed description

Close display device. Implemented as

- R_SLCDC_Close

Table 1049:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to display interface control block.

Parameter p_ctrl

Definition: `slcdc_ctrl_t*const p_ctrl`

SLCDC control block. Allocate an instance specific control block to pass into the SLCDC API calls. Implemented asslcdc_instance_ctrl_tSLCDC control block

9.30.8.12 versionGet

```
ssp_err_t(* slcdc_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get version. Implemented as

- R_SLCDC_VersionGet

Table 1050:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory to store the version information.

Parameter p_version

9.30.8.13 slcdc_instance_t

slcdc_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- slcdc_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- slcdc_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- slcdc_api_t const * p_api

  Pointer to the API structure for this instance.

9.31 SPI Interface

Interface for SPI communications.

9.31.1 Summary

The SPI Interface provides access to the SPI bus API. The Interface implements the Simple SPI on SCI HAL layer driver module.

Implemented by:

- SPI
- Simple SPI on SCI

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers

- Using SSP Modules

SPI Interface description: SCI SPI Driver

9.31.2 Interface API

spi_api_t

Function name	Description
.open	Initialize a channel for SPI communication mode.
.read	Receive data from an SPI device.
.write	Transmit data to an SPI device.
.writeRead	Simultaneously transmit data to an SPI device while receiving data from a SPI device (full duplex).
.close	Remove power to the SPI channel designated by the handle and disable the associated interrupts.
.versionGet	Get the version information of the underlying driver.

9.31.3 Data structures

- spi_callback_args_t
- spi_cfg_t
- spi_instance_t

9.31.4 Enumerations

- spi_bit_width_t
- spi_mode_t
- spi_clk_phase_t
- spi_clk_polarity_t
- spi_mode_fault_t
- spi_bit_order_t
- spi_event_t
- spi_operation_t

9.31.5 Typedefs

- spi_ctrl_t

9.31.6 Defines

- #define SPI_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SPI_API_VERSION_MINOR

  Initial value:`(4U)`

9.31.7 API Data

9.31.7.1 spi_bit_width_t

`spi_bit_width_t`

Detailed description

Data bit width

Enumerated values

Name	Description
SPI_BIT_WIDTH_8_BITS	Data bit width is 8 bits byte.
SPI_BIT_WIDTH_16_BITS	Data bit width is 16 bits word.
SPI_BIT_WIDTH_32_BITS	Data bit width is 32 bits long word.

9.31.7.2 spi_mode_t

`spi_mode_t`

Detailed description

Master or slave operating mode

Enumerated values

Name	Description
SPI_MODE_MASTER	Channel operates as SPI master.

Name	Description
SPI_MODE_SLAVE	Channel operates as SPI slave.

9.31.7.3 spi_clk_phase_t

`spi_clk_phase_t`

Detailed description

Clock phase

Enumerated values

Name	Description
SPI_CLK_PHASE_EDGE_ODD	0: Data sampling on odd edge, data variation on even edge
SPI_CLK_PHASE_EDGE_EVEN	1: Data variation on odd edge, data sampling on even edge

9.31.7.4 spi_clk_polarity_t

`spi_clk_polarity_t`

Detailed description

Clock polarity

Enumerated values

Name	Description
SPI_CLK_POLARITY_LOW	0: Clock polarity is low when idle
SPI_CLK_POLARITY_HIGH	1: Clock polarity is high when idle

9.31.7.5 spi_mode_fault_t

`spi_mode_fault_t`

Detailed description

Mode fault error flag. This error occurs when the device is setup as a master, but the SSLA line does not seem to be controlled by the master. This usually happens when the connecting device is also acting as master. A similar situation can also happen when configured as a slave.

Enumerated values

Name	Description
SPI_MODE_FAULT_ERROR_ENABLE	Mode fault error flag on.
SPI_MODE_FAULT_ERROR_DISABLE	Mode fault error flag off.

9.31.7.6 spi_bit_order_t

`spi_bit_order_t`

Detailed description

Bit-order

Enumerated values

Name	Description
SPI_BIT_ORDER_MSB_FIRST	Send MSB first in transmission.
SPI_BIT_ORDER_LSB_FIRST	Send LSB first in transmission.

9.31.7.7 spi_event_t

`spi_event_t`

Detailed description

SPI events

Enumerated values

Name	Description
SPI_EVENT_TRANSFER_COMPLETE	The data transfer was completed.
SPI_EVENT_TRANSFER_ABORTED	The data transfer was aborted.
SPI_EVENT_ERR_MODE_FAULT	Mode fault error.
SPI_EVENT_ERR_READ_OVERFLOW	Read overflow error.
SPI_EVENT_ERR_PARITY	Parity error.
SPI_EVENT_ERR_OVERRUN	Overrun error.
SPI_EVENT_ERR_FRAMING	Framing error.

Name	Description
SPI_EVENT_ERR_MODE_UNDERRUN	Underrun error.

9.31.7.8 spi_operation_t

`spi_operation_t`

Detailed description

Used by control block only.

Enumerated values

Name	Description
SPI_OPERATION_DO_TX	perform SPI transmission operation
SPI_OPERATION_DO_RX	perform SPI reception operation
SPI_OPERATION_DO_TX_RX	perform SPI Transmission and reception operation

9.31.7.9 spi_ctrl_t

`typedef void spi_ctrl_t`

Detailed description

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented as

- sci_spi_instance_ctrl_t
- rspi_instance_ctrl_t

9.31.8 API Structures

9.31.8.1 spi_callback_args_t

spi_callback_args_t

Detailed description

Common callback parameter definition

Variables

- uint32_t channel

  Device channel number.

- spi_event_t event

  Event code.

- void const * p_context

  Context provided to user during callback.

9.31.8.2 spi_cfg_t

spi_cfg_t

Detailed description

SPI interface configuration

Variables

- uint8_t channel

  Channel number to be used.

- uint8_t rxi_ipl

  Receive interrupt priority.

- uint8_t txi_ipl

  Transmit interrupt priority.

- uint8_t tei_ipl

  Transmit end interrupt priority.

- uint8_t eri_ipl

  Error interrupt priority.

- spi_mode_t operating_mode

  Select master or slave operating mode.

- spi_clk_phase_t clk_phase

  Data sampling on odd or even clock edge.

- spi_clk_polarity_t clk_polarity

  Clock level when idle.

- spi_mode_fault_t mode_fault

  Mode fault error (master/slave conflict) flag.

- spi_bit_order_t bit_order

  Select to transmit MSB/LSB first.

- uint32_t bitrate

  Bits Per Second.

- transfer_instance_t const * p_transfer_tx

  To use SPI DTC/DMA write transfer, link a DTC/DMA instance here. Set to NULL if unused.

- transfer_instance_t const * p_transfer_rx

  To use SPI DTC/DMA read transfer, link a DTC/DMA instance here. Set to NULL if unused.

- void(* p_callback)(spi_callback_args_t *p_args)

  Pointer to user callback function.

- void const * p_context

  User defined context passed to callback function.

- void const * p_extend

  Extended SPI hardware dependent configuration.

9.31.8.3 spi_api_t

spi_api_t

Detailed description

Shared Interface definition for SPI

9.31.8.4 open

```
ssp_err_t(* spi_api_t::open) (spi_ctrl_t *p_ctrl, spi_cfg_t const *const p_cfg)
```

Detailed description

Initialize a channel for SPI communication mode. Implemented as

- R_RSPI_Open
- R_SCI_SPI_Open

Table 1051:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to user-provided storage for the control block.
p_cfg	in	Pointer to SPI configuration structure.

Parameter p_ctrl

Definition: `spi_ctrl_t*p_ctrl`

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented assci_spi_instance_ctrl_trspi_instance_ctrl_t

Parameter p_cfg

Definition: `spi_cfg_t const *const p_cfg`

SPI interface configuration

- `spi_cfg_t`::`channel`

  Channel number to be used.

- spi_cfg_t::rxi_ipl

  Receive interrupt priority.
- spi_cfg_t::txi_ipl

  Transmit interrupt priority.
- spi_cfg_t::tei_ipl

  Transmit end interrupt priority.
- spi_cfg_t::eri_ipl

  Error interrupt priority.
- spi_cfg_t::spi_mode_t

  Select master or slave operating mode.

  Enumerated as:

  – SPI_MODE_MASTER

  – SPI_MODE_SLAVE
- spi_cfg_t::spi_clk_phase_t

  Data sampling on odd or even clock edge.

  Enumerated as:

  – SPI_CLK_PHASE_EDGE_ODD

  – SPI_CLK_PHASE_EDGE_EVEN
- spi_cfg_t::spi_clk_polarity_t

  Clock level when idle.

  Enumerated as:

  – SPI_CLK_POLARITY_LOW

  – SPI_CLK_POLARITY_HIGH
- spi_cfg_t::spi_mode_fault_t

  Mode fault error (master/slave conflict) flag.

  Enumerated as:

  – SPI_MODE_FAULT_ERROR_ENABLE

  – SPI_MODE_FAULT_ERROR_DISABLE
- spi_cfg_t::spi_bit_order_t

  Select to transmit MSB/LSB first.

  Enumerated as:

  – SPI_BIT_ORDER_MSB_FIRST

– SPI_BIT_ORDER_LSB_FIRST

- spi_cfg_t::bitrate

  Bits Per Second.

- spi_cfg_t::transfer_instance_t

  To use SPI DTC/DMA write transfer, link a DTC/DMA instance here. Set to NULL if unused.

- spi_cfg_t::transfer_instance_t

  To use SPI DTC/DMA read transfer, link a DTC/DMA instance here. Set to NULL if unused.

- spi_cfg_t::p_callback

  Pointer to user callback function.

- spi_cfg_t::p_context

  User defined context passed to callback function.

- spi_cfg_t::p_extend

  Extended SPI hardware dependent configuration.

9.31.8.5 read

```
ssp_err_t(* spi_api_t::read) (spi_ctrl_t *const p_ctrl, void const *p_dest,
uint32_t const length, spi_bit_width_t const bit_width)
```

Detailed description

Receive data from an SPI device. Implemented as

- R_RSPI_Read
- R_SCI_SPI_Read

Table 1052:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the channel.
length	in	Number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Data bit width to be transferred.

Table 1052:Parameters (Continued)

Name	Direction	Description
p_dest	out	Pointer to destination buffer into which data will be copied that is received from a SPI device. It is the responsibility of the caller to ensure that adequate space is available to hold the requested data count.

Parameter p_ctrl

Definition: `spi_ctrl_t*const p_ctrl`

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented assci_spi_instance_ctrl_trspi_instance_ctrl_t

Parameter length

`uint32_t`

Parameter bit_width

Parameter p_dest

`const`

9.31.8.6 write

```
ssp_err_t(* spi_api_t::write) (spi_ctrl_t *const p_ctrl, void const *p_src,
uint32_t const length, spi_bit_width_t const bit_width)
```

Detailed description

Transmit data to an SPI device. Implemented as

- R_RSPI_Write
- R_SCI_SPI_Write

Table 1053:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the channel.
p_src	in	Pointer to a source data buffer from which data will be transmitted to a SPI device. The argument must not be NULL.

Table 1053:Parameters (Continued)

Name	Direction	Description
length	in	Number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Data bit width to be transferred.

Parameter p_ctrl

Definition: `spi_ctrl_t*const p_ctrl`

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented assci_spi_instance_ctrl_trspi_instance_ctrl_t

Parameter p_src

`const`

Parameter length

`uint32_t`

Parameter bit_width

9.31.8.7 writeRead

```
ssp_err_t(* spi_api_t::writeRead) (spi_ctrl_t *const p_ctrl, void const *p_src,
void const *p_dest, uint32_t const length, spi_bit_width_t const bit_width)
```

Detailed description

Simultaneously transmit data to an SPI device while receiving data from a SPI device (full duplex). Implemented as

- R_RSPI_WriteRead
- R_SCI_SPI_WriteRead

Table 1054:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the channel.
p_src	in	Pointer to a source data buffer from which data will be transmitted to a SPI device. The argument must not be NULL.

Table 1054:Parameters (Continued)

Name	Direction	Description
p_dest	out	Pointer to destination buffer into which data will be copied that is received from a SPI device. It is the responsibility of the caller to ensure that adequate space is available to hold the requested data count. The argument must not be NULL.
length	in	Number of units of data to be transferred (unit size specified by the bit_width).
bit_width	in	Data bit width to be transferred.

Parameter p_ctrl

Definition: `spi_ctrl_t*const p_ctrl`

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented assci_spi_instance_ctrl_trspi_instance_ctrl_t

Parameter p_src

`const`

Parameter p_dest

`const`

Parameter length

`uint32_t`

Parameter bit_width

9.31.8.8 close

```
ssp_err_t(* spi_api_t::close) (spi_ctrl_t *const p_ctrl)
```

Detailed description

Remove power to the SPI channel designated by the handle and disable the associated interrupts. Implemented as

- R_RSPI_Close
- R_SCI_SPI_Close

Table 1055:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the channel.

Parameter p_ctrl

Definition: `spi_ctrl_t*const p_ctrl`

SPI control block. Allocate an instance specific control block to pass into the SPI API calls. Implemented assci_spi_instance_ctrl_trspi_instance_ctrl_t

9.31.8.9 versionGet

```
ssp_err_t(* spi_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get the version information of the underlying driver. Implemented as

- R_RSPI_VersionGet
- R_SCI_SPI_VersionGet

Table 1056:Parameters

Name	Direction	Description
p_version	out	pointer to memory location to return version number

Parameter p_version

9.31.8.10 spi_instance_t

spi_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- spi_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- spi_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- spi_api_t const * p_api

  Pointer to the API structure for this instance.

9.32 Timer Interface

Interface for timer functions.

9.32.1 Summary

The general timer interface provides standard timer functionality including periodic mode, one-shot mode, and free-running timer mode. After each timer cycle (overflow or underflow), an interrupt can be triggered.

If an instance supports output compare mode, it is provided in the extension configuration timer_on_<instance>_cfg_t defined in r_<instance>.h.

Implemented by:

- GPT
- AGT

See also: Input Capture Interface

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

Timer Interface description: GPT Driver

9.32.2 Interface API

timer_api_t

Function name	Description
.open	Initial configuration.
.stop	Stop the counter.
.start	Start the counter.
.reset	Reset the counter to the initial value.
.counterGet	Get current counter value and store it in provided pointer p_value.

Function name	Description
.periodSet	Set the time until the timer expires.
.dutyCycleSet	Sets the time until the duty cycle expires.
.infoGet	Get the time until the timer expires in clock counts and store it in provided pointer p_period_counts.
.close	Allows driver to be reconfigured and may reduce power consumption.
.versionGet	Get version and store it in provided pointer p_version.

9.32.3 Data structures

- timer_callback_args_t
- timer_info_t
- timer_cfg_t
- timer_instance_t

9.32.4 Enumerations

- timer_event_t
- timer_variant_t
- timer_status_t
- timer_mode_t
- timer_unit_t
- timer_pwm_unit_t
- timer_direction_t

9.32.5 Typedefs

- timer_size_t
- timer_ctrl_t
- timer_period_t

9.32.6 Defines

- #define TIMER_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define TIMER_API_VERSION_MINOR

  Initial value:`(2U)`

9.32.7 API Data

9.32.7.1 timer_event_t

`timer_event_t`

Detailed description

Events that can trigger a callback function

Enumerated values

Name	Description
TIMER_EVENT_EXPIRED	Requested timer delay has expired or timer has wrapped around.

9.32.7.2 timer_variant_t

`timer_variant_t`

Detailed description

Timer variant types.

Enumerated values

Name	Description
TIMER_VARIANT_32_BIT	32-bit timer
TIMER_VARIANT_16_BIT	16-bit timer

9.32.7.3 timer_status_t

`timer_status_t`

Detailed description

Possible status values returned by infoGet.

Enumerated values

Name	Description
TIMER_STATUS_COUNTING	Timer is running.
TIMER_STATUS_STOPPED	Timer is stopped.

9.32.7.4 timer_mode_t

`timer_mode_t`

Detailed description

Timer operational modes

Enumerated values

Name	Description
TIMER_MODE_PERIODIC	Timer will restart after delay periods.
TIMER_MODE_ONE_SHOT	Timer will stop after delay periods.
TIMER_MODE_PWM	Timer generate PWM output.

9.32.7.5 timer_unit_t

`timer_unit_t`

Detailed description

Units of timer period value.

Enumerated values

Name	Description
TIMER_UNIT_PERIOD_RAW_COUNTS	Period in clock counts.
TIMER_UNIT_PERIOD_NSEC	Period in nanoseconds.
TIMER_UNIT_PERIOD_USEC	Period in microseconds.
TIMER_UNIT_PERIOD_MSEC	Period in milliseconds.

Name	Description
TIMER_UNIT_PERIOD_SEC	Period in seconds.
TIMER_UNIT_FREQUENCY_HZ	Frequency in Hz.
TIMER_UNIT_FREQUENCY_KHZ	Frequency in kHz.

9.32.7.6 timer_pwm_unit_t

`timer_pwm_unit_t`

Detailed description

Units of timer duty cycle value.

Enumerated values

Name	Description
TIMER_PWM_UNIT_RAW_COUNTS	Period in clock counts.
TIMER_PWM_UNIT_PERCENT	Percent unit used for duty cycle.
TIMER_PWM_UNIT_PERCENT_X_1000	Percent multiplied by 1000 for extra resolution.

9.32.7.7 timer_direction_t

`timer_direction_t`

Detailed description

Direction of timer count

Enumerated values

Name	Description
TIMER_DIRECTION_DOWN	Timer count goes up.
TIMER_DIRECTION_UP	Timer count goes down.

9.32.7.8 timer_size_t

```
typedef uint32_t timer_size_t
```

Detailed description

Largest supported timer size. Currently up to 32-bit timers are supported. If a 16-bit timer is used, only the bottom 16 bits of any timer_size_t parameter can be used. Passing in values larger than 16 bits would result in an error.

9.32.7.9 timer_ctrl_t

```
typedef void timer_ctrl_t
```

Detailed description

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented as

- gpt_instance_ctrl_t
- agt_instance_ctrl_t

9.32.7.10 timer_period_t

```
typedef timer_size_t timer_period_t
```

Detailed description

DEPRECATED: Recommend using timer_size_t for period.

9.32.8 API Structures

9.32.8.1 timer_callback_args_t

timer_callback_args_t

Detailed description

Callback function parameter data

Variables

- void const * p_context

  Placeholder for user data. Set in timer_api_t::open function in timer_cfg_t.

- timer_event_t event

  The event can be used to identify what caused the callback (overflow or error).

9.32.8.2 timer_info_t

timer_info_t

Detailed description

Timer information structure to store various information for a timer resource

Variables

- timer_direction_t count_direction

  Clock counting direction of the timer resource.

- uint32_t clock_frequency

  Clock frequency of the timer resource.
- timer_size_t period_counts

  Time in clock counts until timer will expire.
- timer_status_t status
- elc_event_t elc_event

  ELC event associated with the count operation of the timer resource.

9.32.8.3 timer_cfg_t

timer_cfg_t

Detailed description

User configuration structure, used in open function

Variables

- timer_mode_t mode

  Select enumerated value from timer_mode_t.
- uint32_t period

  Defines when the timer should expire. For a free running counter, set to TIMER_MAX_CLOCK with unit TIMER_UNIT_PERIOD_RAW_COUNTS
- timer_unit_t unit

  Units of timer_cfg_t::period.
- timer_size_t duty_cycle

  Duty cycle in units timer_cfg_t::duty_cycle_unit.
- timer_pwm_unit_t duty_cycle_unit

  Units of timer_cfg_t::duty_cycle.
- uint8_t channel

  Select a channel corresponding to the channel number of the hardware.
- uint8_t irq_ipl

  Timer interrupt priority.
- bool autostart

  Whether to start during Open call or not. True: Start during open. False: Don't start during open.
- void(* p_callback)(timer_callback_args_t *p_args)

  Callback provided when a timer ISR occurs. Set to NULL for no CPU interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user callback in timer_callback_args_t.

- void const * p_extend

  Extension parameter for hardware specific settings.

9.32.8.4 timer_api_t

timer_api_t

Detailed description

Timer API structure. General timer functions implemented at the HAL layer follow this API.

9.32.8.5 open

```
ssp_err_t(* timer_api_t::open) (timer_ctrl_t *const p_ctrl, timer_cfg_t const
*const p_cfg)
```

Detailed description

Initial configuration. Implemented as

- R_GPT_TimerOpen
- R_AGT_TimerOpen

NOTE: To reconfigure after calling this function, call close first.

Table 1057:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

Parameter p_cfg

Definition: `timer_cfg_t const *const p_cfg`

User configuration structure, used in open function

- `timer_cfg_t`::`timer_mode_t`

  Select enumerated value from timer_mode_t.

  Enumerated as:

  - TIMER_MODE_PERIODIC
  - TIMER_MODE_ONE_SHOT
  - TIMER_MODE_PWM
- timer_cfg_t::period

  Defines when the timer should expire. For a free running counter, set to TIMER_MAX_CLOCK with unit TIMER_UNIT_PERIOD_RAW_COUNTS
- timer_cfg_t::timer_unit_t

  Units of timer_cfg_t::period.

  Enumerated as:
  - TIMER_UNIT_PERIOD_RAW_COUNTS
  - TIMER_UNIT_PERIOD_NSEC
  - TIMER_UNIT_PERIOD_USEC
  - TIMER_UNIT_PERIOD_MSEC
  - TIMER_UNIT_PERIOD_SEC
  - TIMER_UNIT_FREQUENCY_HZ
  - TIMER_UNIT_FREQUENCY_KHZ
- timer_cfg_t::timer_size_t

  Duty cycle in units timer_cfg_t::duty_cycle_unit.
- timer_cfg_t::timer_pwm_unit_t

  Units of timer_cfg_t::duty_cycle.

  Enumerated as:
  - TIMER_PWM_UNIT_RAW_COUNTS
  - TIMER_PWM_UNIT_PERCENT
  - TIMER_PWM_UNIT_PERCENT_X_1000
- timer_cfg_t::channel

  Select a channel corresponding to the channel number of the hardware.
- timer_cfg_t::irq_ipl

  Timer interrupt priority.
- timer_cfg_t::autostart

  Whether to start during Open call or not. True: Start during open. False: Don't start during open.
- timer_cfg_t::p_callback

  Callback provided when a timer ISR occurs. Set to NULL for no CPU interrupt.

- `timer_cfg_t`::`p_context`

  Placeholder for user data. Passed to the user callback in timer_callback_args_t.

- `timer_cfg_t`::`p_extend`

  Extension parameter for hardware specific settings.

9.32.8.6 stop

```
ssp_err_t(* timer_api_t::stop) (timer_ctrl_t *const p_ctrl)
```

Detailed description

Stop the counter. Implemented as

- R_GPT_Stop
- R_AGT_Stop

Table 1058:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

9.32.8.7 start

```
ssp_err_t(* timer_api_t::start) (timer_ctrl_t *const p_ctrl)
```

Detailed description

Start the counter. Implemented as

- R_GPT_Start
- R_AGT_Start

NOTE: The counter can also be started in the open function if autostart is true.

Table 1059:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

9.32.8.8 reset

```
ssp_err_t(* timer_api_t::reset) (timer_ctrl_t *const p_ctrl)
```

Detailed description

Reset the counter to the initial value. Implemented as

- R_GPT_Reset
- R_AGT_Reset

Table 1060:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

9.32.8.9 counterGet

```
ssp_err_t(* timer_api_t::counterGet) (timer_ctrl_t *const p_ctrl, timer_size_t
*const p_value)
```

Detailed description

Get current counter value and store it in provided pointer p_value. Implemented as

- R_GPT_CounterGet
- R_AGT_CounterGet

Table 1061:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
p_value	out	Pointer to store current counter value.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

Parameter p_value

Definition: `timer_size_t*const p_value`

Largest supported timer size. Currently up to 32-bit timers are supported. If a 16-bit timer is used, only the bottom 16 bits of any timer_size_t parameter can be used. Passing in values larger than 16 bits would result in an error.

9.32.8.10 periodSet

```
ssp_err_t(* timer_api_t::periodSet) (timer_ctrl_t *const p_ctrl, timer_size_t
const period, timer_unit_t const unit)
```

Detailed description

Set the time until the timer expires. Implemented as

- R_GPT_PeriodSet
- R_AGT_PeriodSet

NOTE: Timer expiration may or may not generate a CPU interrupt based on how the timer is configured in open.

Table 1062:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
period	in	Time until timer should expire.
unit	in	Units of period parameter.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

Parameter period

Definition: `timer_size_tconst period`

Largest supported timer size. Currently up to 32-bit timers are supported. If a 16-bit timer is used, only the bottom 16 bits of any timer_size_t parameter can be used. Passing in values larger than 16 bits would result in an error.

Parameter unit

9.32.8.11 dutyCycleSet

```
ssp_err_t(* timer_api_t::dutyCycleSet) (timer_ctrl_t *const p_ctrl, timer_size_t
const duty_cycle, timer_pwm_unit_t const duty_cycle_unit, uint8_t const pin)
```

Detailed description

Sets the time until the duty cycle expires.

Table 1063:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
duty_cycle	in	Time until duty cycle should expire.
duty_cycle_unit	in	Units of duty_cycle parameter.
pin	in	Which output pin to update. Enter the pin number or if pins are identified by letters, enter 0 for A, 1 for B, 2 for C, etc.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

Parameter duty_cycle

Definition: `timer_size_tconst duty_cycle`

Largest supported timer size. Currently up to 32-bit timers are supported. If a 16-bit timer is used, only the bottom 16 bits of any timer_size_t parameter can be used. Passing in values larger than 16 bits would result in an error.

Parameter duty_cycle_unit

Definition: `timer_pwm_unit_tconst duty_cycle_unit`

Units of timer duty cycle value.

Parameter pin

`uint8_t`

9.32.8.12 infoGet

```
ssp_err_t(* timer_api_t::infoGet) (timer_ctrl_t *const p_ctrl, timer_info_t
*const p_info)
```

Detailed description

Get the time until the timer expires in clock counts and store it in provided pointer p_period_counts. Implemented as

- R_GPT_InfoGet

- R_AGT_InfoGet

Table 1064:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.
p_info	out	Collection of information for this timer.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

Parameter p_info

Definition: `timer_info_t*const p_info`

Timer information structure to store various information for a timer resource

- `timer_info_t::count_direction`

  Clock counting direction of the timer resource.
- `timer_info_t::clock_frequency`

  Clock frequency of the timer resource.
- `timer_info_t::period_counts`

  Time in clock counts until timer will expire.
- `timer_info_t::status`
- `timer_info_t::elc_event`

  ELC event associated with the count operation of the timer resource.

9.32.8.13 close

```
ssp_err_t(* timer_api_t::close) (timer_ctrl_t *const p_ctrl)
```

Detailed description

Allows driver to be reconfigured and may reduce power consumption. Implemented as

- R_GPT_Close
- R_AGT_Close

Table 1065:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this timer.

Parameter p_ctrl

Definition: `timer_ctrl_t*const p_ctrl`

Timer control block. Allocate an instance specific control block to pass into the timer API calls. Implemented asgpt_instance_ctrl_tagt_instance_ctrl_t

9.32.8.14 versionGet

```
ssp_err_t(* timer_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Get version and store it in provided pointer p_version. Implemented as

- R_GPT_VersionGet
- R_AGT_VersionGet

Table 1066:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.32.8.15 timer_instance_t

timer_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- timer_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- timer_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- timer_api_t const * p_api

  Pointer to the API structure for this instance.

9.33 Transfer Interface

Interface for data transfer functions.

9.33.1 Summary

The transfer interface supports background data transfer (no CPU intervention).

The transfer interface can be implemented by:

- DTC
- DMAC

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

9.33.2 Interface API

transfer_api_t

Function name	Description
.open	Initial configuration. Enables the transfer if auto_enable is true and p_src, p_dest, and length are valid. Transfers can also be enabled using enable or reset.
.reset	Reset source address pointer, destination address pointer, and/or length, keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid.
.enable	Enable transfer. Transfers occur after the activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as activation source).

Function name	Description
.disable	Disable transfer. Transfers do not occur after the transfer_info_t::activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as transfer_info_t::activation_source).
.start	Start transfer in software.
.stop	Stop transfer in software. The transfer will stop after completion of the current transfer.
.infoGet	Provides information about this transfer.
.close	Releases hardware lock. This allows a transfer to be reconfigured using open.
.versionGet	Gets version and stores it in provided pointer p_version.
.blockReset	Reset source address pointer, destination address pointer, and/or length, for block transfer keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid.

9.33.3 Data structures

- transfer_properties_t
- transfer_info_t
- transfer_callback_args_t
- transfer_cfg_t
- transfer_instance_t

9.33.4 Enumerations

- transfer_mode_t
- transfer_size_t
- transfer_addr_mode_t
- transfer_repeat_area_t
- transfer_chain_mode_t
- transfer_irq_t

- transfer_start_mode_t

9.33.5 Typedefs

- transfer_ctrl_t

9.33.6 Defines

- #define TRANSFER_API_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define TRANSFER_API_VERSION_MINOR

  Initial value:`(2U)`

9.33.7 API Data

9.33.7.1 transfer_mode_t

`transfer_mode_t`

Detailed description

Transfer mode describes what will happen when a transfer request occurs.

Enumerated values

Name	Description
TRANSFER_MODE_NORMAL	In normal mode, each transfer request causes a transfer of transfer_size_t from the source pointer to the destination pointer. The transfer length is decremented and the source and address pointers are updated according to transfer_addr_mode_t. After the transfer length reaches 0, transfer requests will not cause any further transfers.
TRANSFER_MODE_REPEAT	Repeat mode is like normal mode, except that when the transfer length reaches 0, the pointer to the repeat area and the transfer length will be reset to their initial values. If DMAC is used, the transfer repeats only transfer_info_t::num_blocks times. After the transfer repeats transfer_info_t::num_blocks times, transfer requests will not cause any further transfers. If DTC is used, the transfer repeats continuously (no limit to the number of repeat transfers).

Name	Description
TRANSFER_MODE_BLOCK	In block mode, each transfer request causes transfer_info_t::length transfers of transfer_size_t. After each individual transfer, the source and destination pointers are updated according to transfer_addr_mode_t. After the block transfer is complete, transfer_info_t::num_blocks is decremented. After the transfer_info_t::num_blocks reaches 0, transfer requests will not cause any further transfers.

9.33.7.2 transfer_size_t

`transfer_size_t`

Detailed description

Transfer size specifies the size of each individual transfer. Total transfer length = transfer_size_t * transfer_length_t

Enumerated values

Name	Description
TRANSFER_SIZE_1_BYTE	Each transfer transfers an 8-bit value.
TRANSFER_SIZE_2_BYTE	Address pointer is incremented after each transfer.
TRANSFER_SIZE_4_BYTE	Address pointer is incremented after each transfer.

9.33.7.3 transfer_addr_mode_t

`transfer_addr_mode_t`

Detailed description

Address mode specifies whether to modify (increment or decrement) pointer after each transfer.

Enumerated values

Name	Description
TRANSFER_ADDR_MODE_FIXED	Address pointer remains fixed after each transfer.
TRANSFER_ADDR_MODE_INCREMENTED	Address pointer is incremented by associated transfer_size_t after each transfer.
TRANSFER_ADDR_MODE_DECREMENTED	Address pointer is decremented by associated transfer_size_t after each transfer.

9.33.7.4 transfer_repeat_area_t

`transfer_repeat_area_t`

Detailed description

Repeat area options (source or destination). In TRANSFER_MODE_REPEAT, the selected pointer returns to its original value after length transfers. In TRANSFER_MODE_BLOCK, the selected pointer returns to its original value after each transfer.

Enumerated values

Name	Description
TRANSFER_REPEAT_AREA_DESTINATION	Destination area repeated in TRANSFER_MODE_REPEAT or TRANSFER_MODE_BLOCK.
TRANSFER_REPEAT_AREA_SOURCE	Source area repeated in TRANSFER_MODE_REPEAT or TRANSFER_MODE_BLOCK.

9.33.7.5 transfer_chain_mode_t

`transfer_chain_mode_t`

Detailed description

Chain transfer mode options. ***NOTE:*** *Only applies for DTC.*

Enumerated values

Name	Description
TRANSFER_CHAIN_MODE_DISABLED	Chain mode not used.
TRANSFER_CHAIN_MODE_EACH	Switch to next transfer after a single transfer from this transfer_info_t.
TRANSFER_CHAIN_MODE_END	Complete the entire transfer defined in this transfer_info_t before chaining to next transfer.

9.33.7.6 transfer_irq_t

`transfer_irq_t`

Detailed description

Interrupt options.

Enumerated values

Name	Description
TRANSFER_IRQ_END	Interrupt occurs only after last transfer. If this transfer is chained to a subsequent transfer, the interrupt will occur only after subsequent chained transfer(s) are complete. DTC triggers the interrupt of the activation source. Choosing TRANSFER_IRQ_END with DTC will prevent activation source interrupts until the transfer is complete.
TRANSFER_IRQ_EACH	Interrupt occurs after each transfer. Not available in all HAL drivers. See HAL driver for details. This will prevent chained transfers that would have happened after this one until the next activation source.

9.33.7.7 transfer_start_mode_t

```
transfer_start_mode_t
```

Detailed description

Select whether to start single or repeated transfer with software start.

Enumerated values

Name	Description
TRANSFER_START_MODE_SINGLE	Software start triggers single transfer.
TRANSFER_START_MODE_REPEAT	Software start transfer continues until transfer is complete.

9.33.7.8 transfer_ctrl_t

```
typedef void transfer_ctrl_t
```

Detailed description

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented as

- dtc_instance_ctrl_t
- dmac_instance_ctrl_t

9.33.8 API Structures

9.33.8.1 transfer_properties_t

transfer_properties_t

Detailed description

Driver specific information.

Variables

- uint32_t transfer_length_max

  Maximum number of transfers.

- uint16_t transfer_length_remaining

  Number of transfers remaining.

- bool in_progress

  Whether or not this transfer is in progress.

9.33.8.2 transfer_info_t

transfer_info_t

Detailed description

This structure specifies the properties of the transfer. ***ATTENTION:*** *When using DTC, this structure corresponds to the descriptor block registers required by the DTC. The following components may be modified by the driver: p_src, p_dest, num_blocks, and length.****ATTENTION:*** *When using DTC, do NOT reuse this structure to configure multiple transfers. Each transfer must have a unique .****ATTENTION:*** *When using DTC, this structure must not be allocated in a temporary location. Any instance of this structure must remain in scope until the transfer it is used for is closed.****NOTE:*** *When using DTC, consider placing instances of this structure in a protected section of memory.*

Variables

- uint32_t __pad0__
- uint32_t __pad1__
- transfer_addr_mode_t dest_addr_mode

  Select what happens to destination pointer after each transfer.

- transfer_repeat_area_t repeat_area

  Select to repeat source or destination area, unused in TRANSFER_MODE_NORMAL.

- transfer_irq_t irq

  Select if interrupts should occur after each individual transfer or after the completion of all planned transfers.

- transfer_chain_mode_t chain_mode

  Select when the chain transfer ends.

- uint32_t __pad2__
- transfer_addr_mode_t src_addr_mode

  Select what happens to source pointer after each transfer.

- transfer_size_t size

  Select number of bytes to transfer at once. transfer_info_t::length.

- transfer_mode_t mode

  Select mode from transfer_mode_t.

- void const *volatile p_src

  Source pointer.

- void *volatile p_dest

  Destination pointer.

- uint16_t num_blocks

  Number of blocks to transfer when using TRANSFER_MODE_BLOCK (both DTC an DMAC) and TRANSFER_MODE_REPEAT (DMAC only), unused in other modes.

- uint16_t length

  Length of each transfer. Range limited for TRANSFER_MODE_BLOCK and TRANSFER_MODE_REPEAT, see HAL driver for details.

9.33.8.3 transfer_callback_args_t

transfer_callback_args_t

Detailed description

Callback function parameter data.

Variables

- void const * p_context

  Placeholder for user data. Set in r_transfer_t::open function in transfer_cfg_t.

9.33.8.4 transfer_cfg_t

transfer_cfg_t

Detailed description

Driver configuration set in open. All elements except p_extend are required and must be initialized.

Variables

- transfer_info_t * p_info

  Pointer to transfer configuration options. If using chain transfer (DTC only), this can be a pointer to an array of chained transfers that will be completed in order.

- elc_event_t activation_source

  Select which event will trigger the transfer. Select ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 for software activation. When using DTC, these events may only be used once each. DMAC uses internal software start when either of these events are selected.

- bool auto_enable

  Select whether the transfer should be enabled after open.

- uint8_t irq_ipl

  Interrupt priority level. Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.

- void(* p_callback)(transfer_callback_args_t *p_args)

  Callback for transfer end interrupt. Set to NULL for no CPU interrupt. Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.

- void const * p_context

  Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

- void const * p_extend

  Extension parameter for hardware specific settings.

9.33.8.5 transfer_api_t

transfer_api_t

Detailed description

Transfer functions implemented at the HAL layer will follow this API.

9.33.8.6 open

```
ssp_err_t(* transfer_api_t::open) (transfer_ctrl_t *const p_ctrl, transfer_cfg_t
const *const p_cfg)
```

Detailed description

Initial configuration. Enables the transfer if auto_enable is true and p_src, p_dest, and length are valid. Transfers can also be enabled using enable or reset. Implemented as

- R_DTC_Open
- R_DMAC_Open

Table 1067:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to control block. Must be declared by user. Elements set here.
p_cfg	in	Pointer to configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

Parameter p_cfg

Definition: `transfer_cfg_t const *const p_cfg`

Driver configuration set in open. All elements except p_extend are required and must be initialized.

- `transfer_cfg_t`::`transfer_info_t`

  Pointer to transfer configuration options. If using chain transfer (DTC only), this can be a pointer to an array of chained transfers that will be completed in order.

- `transfer_cfg_t`::`elc_event_t`

  Select which event will trigger the transfer. Select ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 for software activation. When using DTC, these events may only be used once each. DMAC uses internal software start when either of these events are selected.

- `transfer_cfg_t`::`auto_enable`

  Select whether the transfer should be enabled after open.

- `transfer_cfg_t`::`irq_ipl`

  Interrupt priority level. Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.

- `transfer_cfg_t`::`p_callback`

  Callback for transfer end interrupt. Set to NULL for no CPU interrupt. Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.

- `transfer_cfg_t`::`p_context`

  Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

- `transfer_cfg_t`::`p_extend`

  Extension parameter for hardware specific settings.

9.33.8.7 reset

```
ssp_err_t(* transfer_api_t::reset) (transfer_ctrl_t *const p_ctrl, void const
*p_src, void *p_dest, uint16_t const num_transfers)
```

Detailed description

Reset source address pointer, destination address pointer, and/or length, keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid. Implemented as

- R_DTC_Reset
- R_DMAC_Reset

Table 1068:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.
p_src	in	Pointer to source. Set to NULL if source pointer should not change.
p_dest	in	Pointer to destination. Set to NULL if destination pointer should not change.
num_transfers	in	Transfer length in normal mode or number of blocks in block mode. In DMAC only, resets number of repeats (initially stored in num_blocks) in repeat mode. Not used in repeat mode for DTC.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

Parameter p_src

`const`

Parameter p_dest

`const`

Parameter num_transfers

`uint16_t`

9.33.8.8 enable

```
ssp_err_t(* transfer_api_t::enable) (transfer_ctrl_t *const p_ctrl)
```

Detailed description

Enable transfer. Transfers occur after the activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as activation source). Implemented as

- R_DMAC_Enable
- R_DTC_Enable

Table 1069:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

9.33.8.9 disable

```
ssp_err_t(* transfer_api_t::disable) (transfer_ctrl_t *const p_ctrl)
```

Detailed description

Disable transfer. Transfers do not occur after the transfer_info_t::activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as transfer_info_t::activation_source).

NOTE: If a transfer is in progress, it will be completed. Subsequent transfer requests do not cause a transfer.

Implemented as

- R_DMAC_Disable
- R_DTC_Disable

Table 1070:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

9.33.8.10 start

```
ssp_err_t(* transfer_api_t::start) (transfer_ctrl_t *const p_ctrl,
transfer_start_mode_t mode)
```

Detailed description

Start transfer in software.

ATTENTION: Only works if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as transfer_info_t::activation_source.

NOTE: DTC only supports TRANSFER_START_MODE_SINGLE. DTC does not support TRANSFER_START_MODE_REPEAT.

Implemented as

- R_DMAC_Start
- R_DTC_Start

Table 1071:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.
mode	in	Select mode from transfer_start_mode_t.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

Parameter mode

9.33.8.11 stop

```
ssp_err_t(* transfer_api_t::stop) (transfer_ctrl_t *const p_ctrl)
```

Detailed description

Stop transfer in software. The transfer will stop after completion of the current transfer.

NOTE: Not supported for DTC.

NOTE: Only applies for transfers started with TRANSFER_START_MODE_REPEAT.

ATTENTION: Only works if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as transfer_info_t::activation_source.

Implemented as

- R_DMAC_Stop

Table 1072:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

9.33.8.12 infoGet

```
ssp_err_t(* transfer_api_t::infoGet) (transfer_ctrl_t *const p_ctrl,
transfer_properties_t *const p_info)
```

Detailed description

Provides information about this transfer. Implemented as

- R_DTC_InfoGet
- R_DMAC_InfoGet

Table 1073:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.
p_info	out	Driver specific information.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

Parameter p_info

Definition: `transfer_properties_t*const p_info`

Driver specific information.

- `transfer_properties_t::transfer_length_max`

  Maximum number of transfers.

- `transfer_properties_t::transfer_length_remaining`

  Number of transfers remaining.

- `transfer_properties_t::in_progress`

  Whether or not this transfer is in progress.

9.33.8.13 close

```
ssp_err_t(* transfer_api_t::close) (transfer_ctrl_t *const p_ctrl)
```

Detailed description

Releases hardware lock. This allows a transfer to be reconfigured using open. Implemented as

- R_DTC_Close
- R_DMAC_Close

Table 1074:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

9.33.8.14 versionGet

```
ssp_err_t(* transfer_api_t::versionGet) (ssp_version_t *const p_version)
```

Detailed description

Gets version and stores it in provided pointer p_version. Implemented as

- R_DTC_VersionGet
- R_DMAC_VersionGet

Table 1075:Parameters

Name	Direction	Description
p_version	out	Code and API version used.

Parameter p_version

9.33.8.15 blockReset

```
ssp_err_t(* transfer_api_t::blockReset) (transfer_ctrl_t *const p_ctrl, void
const *p_src, void *p_dest, uint16_t const length, transfer_size_t size, uint16_t
const num_transfers)
```

Detailed description

Reset source address pointer, destination address pointer, and/or length, for block transfer keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid. Implemented as

- R_DMAC_BlockReset
- R_DTC_BlockReset

Table 1076:Parameters

Name	Direction	Description
p_ctrl	in	Control block set in open call for this transfer.
p_src	in	Pointer to source. Set to NULL if source pointer should not change.
p_dest	in	Pointer to destination. Set to NULL if destination pointer should not change.
length	in	Transfer length in block mode.In DMAC only.
size	in	Transfer size in block mode. In DMAC only.
num_transfers	in	number of blocks in block mode. In DMAC only.

Parameter p_ctrl

Definition: `transfer_ctrl_t*const p_ctrl`

Transfer control block. Allocate an instance specific control block to pass into the transfer API calls. Implemented asdtc_instance_ctrl_tdmac_instance_ctrl_t

Parameter p_src

`const`

Parameter p_dest

`const`

Parameter length

`uint16_t`

Parameter size

Parameter num_transfers

`uint16_t`

9.33.8.16 transfer_instance_t

transfer_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- transfer_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- transfer_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- transfer_api_t const * p_api

  Pointer to the API structure for this instance.

9.34 UART Interface

Interface for UART communications.

9.34.1 Summary

The UART interface provides common APIs for UART HAL drivers. The UART interface supports the following features:

- Full-duplex UART communication
- Generic UART parameter setting
- Interrupt driven transmit/receive processing
- Callback function with returned event code
- Runtime baud-rate change
- Hardware resource locking during a transaction
- CTS/RTS hardware flow control support (with an associated IOPORT pin)
- Circular buffer support
- Runtime Transmit/Receive circular buffer flushing

Implemented by:

- UART on SCI

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers
- Using SSP Modules

UART Interface description: SCI SPI Driver

9.34.2 Interface API

uart_api_t

Function name	Description
.open	Open UART device.
.read	Read from UART device. If a transfer instance is used for reception, the received bytes are stored directly in the read input buffer. When a transfer is complete, the callback is called with event UART_EVENT_RX_COMPLETE. Bytes received outside an active transfer are received in the callback function with event UART_EVENT_RX_CHAR. The maximum transfer size is reported by infoGet.
.write	Write to UART device. The write buffer is used until write is complete. Do not overwrite write buffer contents until the write is finished. When the write is complete (all bytes are fully transmitted on the wire), the callback called with event UART_EVENT_TX_COMPLETE. The maximum transfer size is reported by infoGet.
.baudSet	Change baud rate.
.infoGet	Get the driver specific information.
.close	Close UART device.
.versionGet	Get version.

9.34.3 Data structures

- uart_info_t
- uart_callback_args_t
- uart_cfg_t
- uart_instance_t

9.34.4 Enumerations

- uart_event_t
- uart_data_bits_t
- uart_parity_t
- uart_stop_bits_t

9.34.5 Typedefs

- uart_ctrl_t

9.34.6 Defines

- #define UART_API_VERSION_MAJOR
  Initial value:`(1U)`
  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define UART_API_VERSION_MINOR
  Initial value:`(4U)`

9.34.7 API Data

9.34.7.1 uart_event_t

`uart_event_t`

Detailed description

UART Event codes

Enumerated values

Name	Description
UART_EVENT_RX_COMPLETE	Receive complete event.
UART_EVENT_TX_COMPLETE	Transmit complete event.
UART_EVENT_ERR_PARITY	Parity error event.
UART_EVENT_ERR_FRAMING	Mode fault error event.
UART_EVENT_BREAK_DETECT	Break detect error event.

Name	Description
UART_EVENT_ERR_OVERFLOW	FIFO Overflow error event.
UART_EVENT_ERR_RXBUF_OVERFLOW	DEPRECATED: Receive buffer overflow error event.
UART_EVENT_RX_CHAR	Character received.
UART_EVENT_TX_DATA_EMPTY	Last byte is transmitting, ready for more data.

9.34.7.2 uart_data_bits_t

`uart_data_bits_t`

Detailed description

UART Data bit length definition

Enumerated values

Name	Description
UART_DATA_BITS_8	Data bits 8-bit.
UART_DATA_BITS_7	Data bits 7-bit.
UART_DATA_BITS_9	Data bits 9-bit.

9.34.7.3 uart_parity_t

`uart_parity_t`

Detailed description

UART Parity definition

Enumerated values

Name	Description
UART_PARITY_OFF	No parity.
UART_PARITY_EVEN	Even parity.
UART_PARITY_ODD	Odd parity.

9.34.7.4 uart_stop_bits_t

`uart_stop_bits_t`

Detailed description

UART Stop bits definition

Enumerated values

Name	Description
UART_STOP_BITS_1	Stop bit 1-bit.
UART_STOP_BITS_2	Stop bits 2-bit.

9.34.7.5 uart_ctrl_t

`typedef void uart_ctrl_t`

Detailed description

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented as

- sci_uart_instance_ctrl_t

9.34.8 API Structures

9.34.8.1 uart_info_t

uart_info_t

Detailed description

UART driver specific information

Variables

- uint32_t write_bytes_max

  Maximum bytes that can be written at this time. Only applies if uart_cfg_t::p_transfer_tx is not NULL.

- uint32_t read_bytes_max

  Maximum bytes that are available to read at one time. Only applies if uart_cfg_t::p_transfer_rx is not NULL.

9.34.8.2 uart_callback_args_t

uart_callback_args_t

Detailed description

UART Callback parameter definition

Variables

- uint32_t channel

  Device channel number.

- uart_event_t event

  Event code.

- uint32_t data

  Contains the next character received for the events UART_EVENT_RX_CHAR, UART_EVENT_ERR_PARITY, UART_EVENT_ERR_FRAMING, or UART_EVENT_ERR_OVERFLOW. Otherwise unused.

- void const * p_context

  Context provided to user during callback.

9.34.8.3 uart_cfg_t

uart_cfg_t

Detailed description

UART Configuration

Variables

- uint8_t channel

  Select a channel corresponding to the channel number of the hardware.

- uint32_t baud_rate

  Baud rate, i.e. 9600, 19200, 115200.

- uart_data_bits_t data_bits

  Data bit length (8 or 7 or 9)

- uart_parity_t parity

  Parity type (none or odd or even)

- uart_stop_bits_t stop_bits

  Stop bit length (1 or 2)

- bool ctsrts_en

  CTS/RTS hardware flow control enable.

- uint8_t rxi_ipl

  Receive interrupt priority.

- uint8_t txi_ipl

  Transmit interrupt priority.

- uint8_t tei_ipl

  Transmit end interrupt priority.

- uint8_t eri_ipl

  Error interrupt priority.

- transfer_instance_t const * p_transfer_rx

  Optional transfer instance used to receive multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the read API is limited to one byte at a time.

- transfer_instance_t const * p_transfer_tx

  Optional transfer instance used to send multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the write APIs is limited to one byte at a time.

- void(* p_callback)(uart_callback_args_t *p_args)

  Pointer to callback function.

- void const * p_context

  User defined context passed into callback function.

- void const * p_extend

  UART hardware dependent configuration.

9.34.8.4 uart_api_t

uart_api_t

Detailed description

Shared Interface definition for UART

9.34.8.5 open

```
ssp_err_t(* uart_api_t::open) (uart_ctrl_t *const p_ctrl, uart_cfg_t const *const p_cfg)
```

Detailed description

Open UART device. Implemented as

- R_SCI_UartOpen

Table 1077:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to the UART control block Must be declared by user. Value set here.
uart_cfg_t	in	Pointer to UART configuration structure. All elements of this structure must be set by user.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

Parameter uart_cfg_t

Definition: `uart_cfg_t const *const p_cfg`

UART Configuration

- `uart_cfg_t`::`channel`

  Select a channel corresponding to the channel number of the hardware.

- `uart_cfg_t`::`baud_rate`

  Baud rate, i.e. 9600, 19200, 115200.

- `uart_cfg_t`::`uart_data_bits_t`

  Data bit length (8 or 7 or 9)

  Enumerated as:

  – UART_DATA_BITS_8

  – UART_DATA_BITS_7

  – UART_DATA_BITS_9

- `uart_cfg_t`::`uart_parity_t`

  Parity type (none or odd or even)

  Enumerated as:

  – UART_PARITY_OFF

  – UART_PARITY_EVEN

  – UART_PARITY_ODD

- `uart_cfg_t`::`uart_stop_bits_t`

  Stop bit length (1 or 2)

  Enumerated as:

  – UART_STOP_BITS_1

  – UART_STOP_BITS_2

- `uart_cfg_t`::`ctsrts_en`

  CTS/RTS hardware flow control enable.

- `uart_cfg_t`::`rxi_ipl`

  Receive interrupt priority.

- uart_cfg_t::txi_ipl

  Transmit interrupt priority.

- uart_cfg_t::tei_ipl

  Transmit end interrupt priority.

- uart_cfg_t::eri_ipl

  Error interrupt priority.

- uart_cfg_t::transfer_instance_t

  Optional transfer instance used to receive multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the read API is limited to one byte at a time.

- uart_cfg_t::transfer_instance_t

  Optional transfer instance used to send multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the write APIs is limited to one byte at a time.

- uart_cfg_t::p_callback

  Pointer to callback function.

- uart_cfg_t::p_context

  User defined context passed into callback function.

- uart_cfg_t::p_extend

  UART hardware dependent configuration.

9.34.8.6 read

```
ssp_err_t(* uart_api_t::read) (uart_ctrl_t *const p_ctrl, uint8_t const *const
p_dest, uint32_t const bytes)
```

Detailed description

Read from UART device. If a transfer instance is used for reception, the received bytes are stored directly in the read input buffer. When a transfer is complete, the callback is called with event UART_EVENT_RX_COMPLETE. Bytes received outside an active transfer are received in the callback function with event UART_EVENT_RX_CHAR. The maximum transfer size is reported by infoGet. Implemented as

- R_SCI_UartRead

Table 1078:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the UART control block for the channel.
p_dest	in	Destination address to read data from.

Table 1078:Parameters (Continued)

Name	Direction	Description
bytes	in	Read data length. Only applicable if p_transfer_rx is not NULL. Otherwise all read bytes will be provided through the callback set in p_callback.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

Parameter p_dest

`uint8_t`

Parameter bytes

`uint32_t`

9.34.8.7 write

```
ssp_err_t(* uart_api_t::write) (uart_ctrl_t *const p_ctrl, uint8_t const *const
p_src, uint32_t const bytes)
```

Detailed description

Write to UART device. The write buffer is used until write is complete. Do not overwrite write buffer contents until the write is finished. When the write is complete (all bytes are fully transmitted on the wire), the callback called with event UART_EVENT_TX_COMPLETE. The maximum transfer size is reported by infoGet. Implemented as

- R_SCI_UartWrite

Table 1079:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the UART control block.
p_src	in	Source address to write data to.
bytes	in	Write data length.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

Parameter p_src

`uint8_t`

Parameter bytes

`uint32_t`

9.34.8.8 baudSet

```
ssp_err_t(* uart_api_t::baudSet) (uart_ctrl_t *const p_ctrl, uint32_t const
baudrate)
```

Detailed description

Change baud rate.

ATTENTION: Calling this API aborts any in-progress transmission and disables reception until the new baud settings have been applied.

Implemented as

- R_SCI_UartBaudSet

Table 1080:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the UART control block.
baudrate	in	Baud rate in bps.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

Parameter baudrate

`uint32_t`

9.34.8.9 infoGet

```
ssp_err_t(* uart_api_t::infoGet) (uart_ctrl_t *const p_ctrl, uart_info_t *const
p_info)
```

Detailed description

Get the driver specific information. Implemented as

- R_SCI_UartInfoGet

Table 1081:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the UART control block.
baudrate	in	Baud rate in bps.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

Parameter baudrate

9.34.8.10 close

```
ssp_err_t(* uart_api_t::close) (uart_ctrl_t *const p_ctrl)
```

Detailed description

Close UART device. Implemented as

- R_SCI_UartClose

Table 1082:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to the UART control block.

Parameter p_ctrl

Definition: `uart_ctrl_t*const p_ctrl`

UART control block. Allocate an instance specific control block to pass into the UART API calls. Implemented assci_uart_instance_ctrl_t

9.34.8.11 versionGet

```
ssp_err_t(* uart_api_t::versionGet) (ssp_version_t *p_version)
```

Detailed description

Get version. Implemented as

- R_SCI_UartVersionGet

Table 1083:Parameters

Name	Direction	Description
p_version	in	Pointer to the memory to store the version information.

Parameter p_version

9.34.8.12 uart_instance_t

uart_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- uart_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- uart_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- uart_api_t const * p_api

  Pointer to the API structure for this instance.

9.35 WDT Interface

Interface for watch dog timer functions.

9.35.1 Summary

The WDT interface for the Watchdog Timer (WDT) peripheral provides watchdog functionality including resetting the device or generating an interrupt.

See Also and Thread Monitor Framework Interface

The watchdog timer interface can be implemented by:

- WDT
- IWDT

Related SSP architecture topics:

- SSP Interfaces
- SSP Predefined Layers

- Using SSP Modules

WDT Interface description: Watchdog Driver

9.35.2 Interface API

wdt_api_t

Function name	Description
.cfgGet	Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.
.open	Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.
.refresh	Refresh the watchdog timer.
.statusGet	Read the status of the WDT.
.statusClear	Clear the status flags of the WDT.
.counterGet	Read the current WDT counter value.
.timeoutGet	Read the watchdog timeout values.
.versionGet	Return the version of the IOPort driver.

9.35.3 Data structures

- wdt_callback_args_t
- wdt_timeout_values_t
- wdt_cfg_t
- wdt_instance_t

9.35.4 Enumerations

- wdt_timeout_t
- wdt_clock_division_t
- wdt_window_start_t
- wdt_window_end_t
- wdt_reset_control_t
- wdt_stop_control_t

- wdt_status_t
- wdt_start_mode_t

9.35.5 Typedefs

- wdt_ctrl_t

9.35.6 Defines

- #define WDT_API_VERSION_MAJOR
  Initial value:`(1U)`
  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define WDT_API_VERSION_MINOR
  Initial value:`(2U)`

9.35.7 API Data

9.35.7.1 wdt_timeout_t

`wdt_timeout_t`

Detailed description

WDT time-out periods.

Enumerated values

Name	Description
WDT_TIMEOUT_128	128 clock cycles
WDT_TIMEOUT_512	512 clock cycles
WDT_TIMEOUT_1024	1024 clock cycles
WDT_TIMEOUT_2048	2048 clock cycles
WDT_TIMEOUT_4096	4096 clock cycles
WDT_TIMEOUT_8192	8192 clock cycles
WDT_TIMEOUT_16384	16384 clock cycles

9.35.7.2 wdt_clock_division_t

`wdt_clock_division_t`

Detailed description

WDT clock division ratio.

Enumerated values

Name	Description
WDT_CLOCK_DIVISION_1	CLK/1.
WDT_CLOCK_DIVISION_4	CLK/4.
WDT_CLOCK_DIVISION_16	CLK/16.
WDT_CLOCK_DIVISION_32	CLK/32.
WDT_CLOCK_DIVISION_64	CLK/64.
WDT_CLOCK_DIVISION_128	CLK/128.
WDT_CLOCK_DIVISION_256	CLK/256.
WDT_CLOCK_DIVISION_512	CLK/512.
WDT_CLOCK_DIVISION_2048	CLK/2048.
WDT_CLOCK_DIVISION_8192	CLK/8192.

9.35.7.3 wdt_window_start_t

`wdt_window_start_t`

Detailed description

WDT refresh permitted period window start position.

Enumerated values

Name	Description
WDT_WINDOW_START_25	Start position = 25%.
WDT_WINDOW_START_50	Start position = 50%.
WDT_WINDOW_START_75	Start position = 75%.

Name	Description
WDT_WINDOW_START_100	Start position = 100%.

9.35.7.4 wdt_window_end_t

`wdt_window_end_t`

Detailed description

WDT refresh permitted period window end position.

Enumerated values

Name	Description
WDT_WINDOW_END_75	End position = 75%.
WDT_WINDOW_END_50	End position = 50%.
WDT_WINDOW_END_25	End position = 25%.
WDT_WINDOW_END_0	End position = 0%.

9.35.7.5 wdt_reset_control_t

`wdt_reset_control_t`

Detailed description

WDT Counter underflow and refresh error control.

Enumerated values

Name	Description
WDT_RESET_CONTROL_NMI	NMI request when counter underflows.
WDT_RESET_CONTROL_RESET	Reset request when counter underflows.

9.35.7.6 wdt_stop_control_t

`wdt_stop_control_t`

Detailed description

WDT Counter operation in sleep mode.

Enumerated values

Name	Description
WDT_STOP_CONTROL_DISABLE	Count will not stop when device enters sleep mode.
WDT_STOP_CONTROL_ENABLE	Count will automatically stop when device enters sleep mode.

9.35.7.7 wdt_status_t

`wdt_status_t`

Detailed description

WDT status

Enumerated values

Name	Description
WDT_STATUS_NO_ERROR	No status flags set.
WDT_STATUS_UNDERFLOW_ERROR	Underflow flag set.
WDT_STATUS_REFRESH_ERROR	Refresh error flag set. Refresh outside of permitted.
WDT_STATUS_UNDERFLOW_AND_REFRESH_ERROR	Underflow and refresh error flags set.

9.35.7.8 wdt_start_mode_t

`wdt_start_mode_t`

Detailed description

WDT start mode. Used to check the WDT is configured correctly.

Enumerated values

Name	Description
WDT_START_MODE_REGISTER	WDT is to be configured using the WDT registers.
WDT_START_MODE_AUTO	WDT is to be configured using OFS0 hardware register.
WDT_START_MODE_DISABLED	WDT is disabled.

9.35.7.9 wdt_ctrl_t

```
typedef void wdt_ctrl_t
```

Detailed description

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented as

- wdt_instance_ctrl_t
- iwdt_instance_ctrl_t

9.35.8 API Structures

9.35.8.1 wdt_callback_args_t

wdt_callback_args_t

Detailed description

Callback function parameter data

Variables

- void const * p_context

  Placeholder for user data. Set in wdt_api_t::open function in wdt_cfg_t.

9.35.8.2 wdt_timeout_values_t

wdt_timeout_values_t

Detailed description

WDT timeout data. Used to return frequency of WDT clock and timeout period

Variables

- uint32_t clock_frequency_hz

  Frequency of watchdog clock after divider.

- uint32_t timeout_clocks

  Timeout period in units of watchdog clock ticks.

9.35.8.3 wdt_cfg_t

wdt_cfg_t

Detailed description

WDT configuration parameters.

Variables

- wdt_start_mode_t start_mode

  The expected start mode for the WDT.

- bool autostart

  When true the WDT is started as part of its configuration (register start mode). If false the WDT needs to be started manually by calling the refresh API.

- wdt_timeout_t timeout

  Timeout period.

- wdt_clock_division_t clock_division

  Clock divider.

- wdt_window_start_t window_start

  Refresh permitted window start position.

- wdt_window_end_t window_end

  Refresh permitted window end position.

- wdt_reset_control_t reset_control

  Select NMI or reset generated on underflow.

- wdt_stop_control_t stop_control

  Select whether counter operates in sleep mode.

- void(* p_callback)(wdt_callback_args_t *p_args)

  Callback provided when a WDT NMI ISR occurs.

- void const * p_context

  Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

- void const * p_extend

  Placeholder for user extension.

9.35.8.4 wdt_api_t

wdt_api_t

Detailed description

WDT functions implemented at the HAL layer will follow this API.

9.35.8.5 cfgGet

```
ssp_err_t(* wdt_api_t::cfgGet) (wdt_ctrl_t *const p_ctrl, wdt_cfg_t *const p_cfg)
```

Detailed description

Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback. Implemented as

- R_WDT_CfgGet
- R_IWDT_CfgGet

Table 1084:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_cfg	out	Pointer to pin configuration structure for reading WDT configuration.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter p_cfg

Definition: `wdt_cfg_t *const p_cfg`

WDT configuration parameters.

- `wdt_cfg_t`::`wdt_start_mode_t`

  The expected start mode for the WDT.

  Enumerated as:

  – WDT_START_MODE_REGISTER

  – WDT_START_MODE_AUTO

  – WDT_START_MODE_DISABLED

- `wdt_cfg_t`::`autostart`

  When true the WDT is started as part of its configuration (register start mode). If false the WDT needs to be started manually by calling the refresh API.

- `wdt_cfg_t`::`wdt_timeout_t`

  Timeout period.

  Enumerated as:

  – WDT_TIMEOUT_128

  – WDT_TIMEOUT_512

  – WDT_TIMEOUT_1024

  – WDT_TIMEOUT_2048

  – WDT_TIMEOUT_4096

  – WDT_TIMEOUT_8192

  – WDT_TIMEOUT_16384

- wdt_cfg_t::wdt_clock_division_t

  Clock divider.

  Enumerated as:

  – WDT_CLOCK_DIVISION_1
  – WDT_CLOCK_DIVISION_4
  – WDT_CLOCK_DIVISION_16
  – WDT_CLOCK_DIVISION_32
  – WDT_CLOCK_DIVISION_64
  – WDT_CLOCK_DIVISION_128
  – WDT_CLOCK_DIVISION_256
  – WDT_CLOCK_DIVISION_512
  – WDT_CLOCK_DIVISION_2048
  – WDT_CLOCK_DIVISION_8192
- wdt_cfg_t::wdt_window_start_t

  Refresh permitted window start position.

  Enumerated as:

  – WDT_WINDOW_START_25
  – WDT_WINDOW_START_50
  – WDT_WINDOW_START_75
  – WDT_WINDOW_START_100
- wdt_cfg_t::wdt_window_end_t

  Refresh permitted window end position.

  Enumerated as:

  – WDT_WINDOW_END_75
  – WDT_WINDOW_END_50
  – WDT_WINDOW_END_25
  – WDT_WINDOW_END_0
- wdt_cfg_t::wdt_reset_control_t

  Select NMI or reset generated on underflow.

  Enumerated as:

  – WDT_RESET_CONTROL_NMI

– WDT_RESET_CONTROL_RESET

- wdt_cfg_t::wdt_stop_control_t

  Select whether counter operates in sleep mode.

  Enumerated as:

  – WDT_STOP_CONTROL_DISABLE

  – WDT_STOP_CONTROL_ENABLE

- wdt_cfg_t::p_callback

  Callback provided when a WDT NMI ISR occurs.

- wdt_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

- wdt_cfg_t::p_extend

  Placeholder for user extension.

9.35.8.6 open

```
ssp_err_t(* wdt_api_t::open) (wdt_ctrl_t *const p_ctrl, wdt_cfg_t const *const p_cfg)
```

Detailed description

Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback. Implemented as

- R_WDT_Open
- R_IWDT_Open

Table 1085:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_cfg	in	Pointer to pin configuration structure.

Parameter p_ctrl

Definition: wdt_ctrl_t*const p_ctrl

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter p_cfg

Definition: wdt_cfg_t const *const p_cfg

WDT configuration parameters.

- wdt_cfg_t::wdt_start_mode_t

  The expected start mode for the WDT.

  Enumerated as:

  – WDT_START_MODE_REGISTER

  – WDT_START_MODE_AUTO

  – WDT_START_MODE_DISABLED

- wdt_cfg_t::autostart

  When true the WDT is started as part of its configuration (register start mode). If false the WDT needs to be started manually by calling the refresh API.

- wdt_cfg_t::wdt_timeout_t

  Timeout period.

  Enumerated as:

  – WDT_TIMEOUT_128

  – WDT_TIMEOUT_512

  – WDT_TIMEOUT_1024

  – WDT_TIMEOUT_2048

  – WDT_TIMEOUT_4096

  – WDT_TIMEOUT_8192

  – WDT_TIMEOUT_16384

- wdt_cfg_t::wdt_clock_division_t

  Clock divider.

  Enumerated as:

  – WDT_CLOCK_DIVISION_1

  – WDT_CLOCK_DIVISION_4

  – WDT_CLOCK_DIVISION_16

  – WDT_CLOCK_DIVISION_32

  – WDT_CLOCK_DIVISION_64

  – WDT_CLOCK_DIVISION_128

  – WDT_CLOCK_DIVISION_256

  – WDT_CLOCK_DIVISION_512

  – WDT_CLOCK_DIVISION_2048

  - WDT_CLOCK_DIVISION_8192
- wdt_cfg_t::wdt_window_start_t

  Refresh permitted window start position.

  Enumerated as:

  - WDT_WINDOW_START_25
  - WDT_WINDOW_START_50
  - WDT_WINDOW_START_75
  - WDT_WINDOW_START_100
- wdt_cfg_t::wdt_window_end_t

  Refresh permitted window end position.

  Enumerated as:

  - WDT_WINDOW_END_75
  - WDT_WINDOW_END_50
  - WDT_WINDOW_END_25
  - WDT_WINDOW_END_0
- wdt_cfg_t::wdt_reset_control_t

  Select NMI or reset generated on underflow.

  Enumerated as:

  - WDT_RESET_CONTROL_NMI
  - WDT_RESET_CONTROL_RESET
- wdt_cfg_t::wdt_stop_control_t

  Select whether counter operates in sleep mode.

  Enumerated as:

  - WDT_STOP_CONTROL_DISABLE
  - WDT_STOP_CONTROL_ENABLE
- wdt_cfg_t::p_callback

  Callback provided when a WDT NMI ISR occurs.
- wdt_cfg_t::p_context

  Placeholder for user data. Passed to the user callback in wdt_callback_args_t.
- wdt_cfg_t::p_extend

  Placeholder for user extension.

9.35.8.7 refresh

```
ssp_err_t(* wdt_api_t::refresh) (wdt_ctrl_t *const p_ctrl)
```

Detailed description

Refresh the watchdog timer. Implemented as

- R_WDT_Refresh
- R_IWDT_Refresh

NOTE: If the WDT is in auto-start mode ensure the OFS0 register is configured before using this function.

ATTENTION: Calling this function in register-start mode before calling R_WDT_Open will start the WDT in it's default state and further changes to the configuration will not be possible.

Table 1086:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

9.35.8.8 statusGet

```
ssp_err_t(* wdt_api_t::statusGet) (wdt_ctrl_t *const p_ctrl, wdt_status_t *const
p_status)
```

Detailed description

Read the status of the WDT. Implemented as

- R_WDT_StatusGet
- R_IWDT_StatusGet

Table 1087:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_status	out	Pointer to variable to return status information through.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter p_status

Definition: `wdt_status_t*const p_status`

WDT status

9.35.8.9 statusClear

```
ssp_err_t(* wdt_api_t::statusClear) (wdt_ctrl_t *const p_ctrl, const wdt_status_t status)
```

Detailed description

Clear the status flags of the WDT. Implemented as

- R_WDT_StatusClear
- R_IWDT_StatusClear

Table 1088:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
status	in	Status condition(s) to clear.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter status

9.35.8.10 counterGet

```
ssp_err_t(* wdt_api_t::counterGet) (wdt_ctrl_t *const p_ctrl, uint32_t *const p_count)
```

Detailed description

Read the current WDT counter value. Implemented as

- R_WDT_CounterGet
- R_IWDT_CounterGet

Table 1089:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_count	out	Pointer to variable to return current WDT counter value.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter p_count

`uint32_t`

9.35.8.11 timeoutGet

```
ssp_err_t(* wdt_api_t::timeoutGet) (wdt_ctrl_t *const p_ctrl,
wdt_timeout_values_t *const p_timeout)
```

Detailed description

Read the watchdog timeout values. Implemented as

- R_WDT_TimeoutGet
- R_IWDT_TimeoutGet

Table 1090:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_timeout	out	Pointer to structure to return timeout values.

Parameter p_ctrl

Definition: `wdt_ctrl_t*const p_ctrl`

WDT control block. Allocate an instance specific control block to pass into the WDT API calls. Implemented aswdt_instance_ctrl_tiwdt_instance_ctrl_t

Parameter p_timeout

Definition: `wdt_timeout_values_t*const p_timeout`

WDT timeout data. Used to return frequency of WDT clock and timeout period

- wdt_timeout_values_t::clock_frequency_hz

  Frequency of watchdog clock after divider.
- wdt_timeout_values_t::timeout_clocks

  Timeout period in units of watchdog clock ticks.

9.35.8.12 versionGet

```
ssp_err_t(* wdt_api_t::versionGet) (ssp_version_t *const p_data)
```

Detailed description

Return the version of the IOPort driver. Implemented as

- R_WDT_VersionGet
- R_IWDT_VersionGet

Table 1091:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to control structure.
p_data	out	Memory address to return version information to.

Parameter p_ctrl

Parameter p_data

Definition: `ssp_version_t *const p_data`

- ssp_version_t::version_id

  Version id
- ssp_version_t::code_version_minor

  Code minor version.
- ssp_version_t::code_version_major

  Code major version.
- ssp_version_t::api_version_minor

  API minor version.
- ssp_version_t::api_version_major

  API major version.
- ssp_version_tstruct{}

  Code version parameters

9.35.8.13 wdt_instance_t

wdt_instance_t

Detailed description

This structure encompasses everything that is needed to use an instance of this interface.

Variables

- wdt_ctrl_t * p_ctrl

  Pointer to the control structure for this instance.

- wdt_cfg_t const * p_cfg

  Pointer to the configuration structure for this instance.

- wdt_api_t const * p_api

  Pointer to the API structure for this instance.

Chapter 10 API Reference: HAL Drivers

10.1 API Reference: HAL Layer

The hardware abstraction layer provides drivers for Renesas peripherals. HAL drivers typically implement Interfaces and provide additional hardware specific APIs.

- ADC
- AGT
- CAC
- CAN
- CGC
- CRC
- CTSU
- DAC
- DAC 8
- DMAC
- DOC
- DTC
- ELC
- High-performance Flash
- Low Power Flash
- FMI
- GLCDC
- GPT
- GPT Input Capture
- ICU
- IOPORT
- IWDT
- JPEG CODEC
- Key Interrupts
- LPM

- LPMV2 S124
- LPMV2 S128
- LPMV2 S3A3
- LPMV2 S3A6
- LPMV2 S3A7
- LPMV2 S5D5
- LPMV2 S5D9
- LPMV2 S7G2
- LVD
- PDC
- QSPI
- IIC
- IIC Slave
- SPI
- RTC
- Simple I2C on SCI
- Simple SPI on SCI
- UART on SCI
- SDMMC
- SLCDC
- SSI
- WDT
- SCE Module

10.2 ADC

Driver for the 14-Bit A/D Converter (ADC14) and 12-bit A/D Converter (ADC12).

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

This module supports the ADC14 and ADC12 peripherals. It implements the following interfaces:

- ADC Interface

10.2.1 Functions

- R_ADC_Open

- R_ADC_SetSampleStateCount
- R_ADC_ScanConfigure
- R_ADC_InfoGet
- R_ADC_ScanStart
- R_ADC_ScanStop
- R_ADC_CheckScanDone
- R_ADC_Read
- R_ADC_Close
- R_ADC_VersionGet

10.2.2 Defines

- #define ADC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Version of code that implements the API defined in this file

- #define ADC_CODE_VERSION_MINOR

  Initial value:`(4U)`

- #define ADC_SAMPLE_STATE_COUNT_MIN

  Initial value:`(7U)`

  Typical values that can be used to modify the sample states. The minimum sample state count value is either 6 or 7 depending on the clock ratios. It is fixed to 7 based on the fact that at the lowest ADC conversion clock supported (1 MHz) this extra state will lead to at worst a "1 microsecond" increase in conversion time. At 60 MHz the extra sample state will add 16.7 ns to the conversion time.

- #define ADC_SAMPLE_STATE_COUNT_MAX

  Initial value:`(255U)`

- #define ADC_SAMPLE_STATE_HOLD_COUNT_MIN

  Initial value:`(4U)`

  Typical values that can be used for the sample and hold counts for the channels 0-2 Minimum sample and hold states

- #define ADC_SAMPLE_STATE_HOLD_COUNT_DEFAULT

  Initial value:`(24U)`

  Default sample and hold states

- #define ADC_MASK_CHANNEL_0

  Initial value:`(1U<<0U)`

  For ADC Scan configuration adc_channel_cfg_t::scan_mask, scan_mask_group_b, add_mask and sample_hold_mask Use bitwise OR to combine these masks for desired channels and sensors.

- #define ADC_MASK_CHANNEL_1
  Initial value:`(1U<<1U)`
- #define ADC_MASK_CHANNEL_2
  Initial value:`(1U<<2U)`
- #define ADC_MASK_CHANNEL_3
  Initial value:`(1U<<3U)`
- #define ADC_MASK_CHANNEL_4
  Initial value:`(1U<<4U)`
- #define ADC_MASK_CHANNEL_5
  Initial value:`(1U<<5U)`
- #define ADC_MASK_CHANNEL_6
  Initial value:`(1U<<6U)`
- #define ADC_MASK_CHANNEL_7
  Initial value:`(1U<<7U)`
- #define ADC_MASK_CHANNEL_8
  Initial value:`(1U<<8U)`
- #define ADC_MASK_CHANNEL_9
  Initial value:`(1U<<9U)`
- #define ADC_MASK_CHANNEL_10
  Initial value:`(1U<<10U)`
- #define ADC_MASK_CHANNEL_11
  Initial value:`(1U<<11U)`
- #define ADC_MASK_CHANNEL_12
  Initial value:`(1U<<12U)`
- #define ADC_MASK_CHANNEL_13
  Initial value:`(1U<<13U)`
- #define ADC_MASK_CHANNEL_14
  Initial value:`(1U<<14U)`
- #define ADC_MASK_CHANNEL_15
  Initial value:`(1U<<15U)`
- #define ADC_MASK_CHANNEL_16
  Initial value:`(1U<<16U)`
- #define ADC_MASK_CHANNEL_17
  Initial value:`(1U<<17U)`

- #define ADC_MASK_CHANNEL_18
  Initial value:`(1U<<18U)`
- #define ADC_MASK_CHANNEL_19
  Initial value:`(1U<<19U)`
- #define ADC_MASK_CHANNEL_20
  Initial value:`(1U<<20U)`
- #define ADC_MASK_CHANNEL_21
  Initial value:`(1U<<21U)`
- #define ADC_MASK_CHANNEL_22
  Initial value:`(1U<<22U)`
- #define ADC_MASK_CHANNEL_23
  Initial value:`(1U<<23U)`
- #define ADC_MASK_CHANNEL_24
  Initial value:`(1U<<24U)`
- #define ADC_MASK_CHANNEL_25
  Initial value:`(1U<<25U)`
- #define ADC_MASK_CHANNEL_26
  Initial value:`(1U<<26U)`
- #define ADC_MASK_CHANNEL_27
  Initial value:`(1U<<27U)`
- #define ADC_MASK_TEMPERATURE
  Initial value:`(1U<<28UL)`
- #define ADC_MASK_VOLT
  Initial value:`(1U<<29UL)`
- #define ADC_MASK_SENSORS
  Initial value:`(ADC_MASK_TEMPERATURE | ADC_MASK_VOLT)`
- #define ADC_MASK_GROUP_B_OFF
  Initial value:`(0UL)`
- #define ADC_MASK_ADD_OFF
  Initial value:`(0UL)`
- #define ADC_MASK_SAMPLE_HOLD_OFF
  Initial value:`(0U)`

- #define ADC_SAMPLE_HOLD_CHANNELS

  Initial value:`(0x07U)`

  Sample and hold Channel mask. Sample and hold is only available for channel 0,1,2

10.2.3 R_ADC_Open

```
ssp_err_t R_ADC_Open ( adc_ctrl_t  * p_api_ctrl ,  adc_cfg_t  const
*const p_cfg )
```

10.2.3.1 Brief description

The Open function applies power to the A/D peripheral, sets the operational mode, trigger sources, interrupt priority, and configurations for the peripheral as a whole. If interrupt priority is non-zero in BSP_IRQ_Cfg.h, the function takes a callback function pointer for notifying the user at interrupt level whenever a scan has completed.

10.2.3.2 Detailed description

Table 1092:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl or p_cfg is NULL.
SSP_ERR_INVALID_ARGUMENT	Mode or element of p_cfg structure has invalid value or is illegal based on mode.
SSP_ERR_IN_USE	Unit has already been initialized.

10.2.3.3 Function steps

- Perform parameter checking
- Verify this unit has not already been initialized
- Set all p_ctrl fields prior to using it in any functions
- Save callback function pointer
- Store the Unit number into the control structure
- Store the user context into the control structure
- Store the mode into the control structure
- Save the regular mode/Group A trigger in the internal control block
- Save the context

- Confirm the requested unit exists on this MCU and record available channels.
- Lock specified ADC channel
- Retrieve temperature sensor information into control block
- Set ADC and Temperature sensors to a stop state
- Initialize the hardware based on the configuration
- Set unused registers to known state (Disable ADC PGA on MCUs that have it)
- Invalid scan mask (initialized for later).
- Mark driver as opened by initializing it to "RADC" in its ASCII equivalent for this unit.
- Return the error code

10.2.4 R_ADC_SetSampleStateCount

```
ssp_err_t R_ADC_SetSampleStateCount ( adc_ctrl_t  * p_api_ctrl ,
 adc_sample_state_t  * p_sample )
```

10.2.4.1 Brief description

Set the sample state count for individual channels. This only needs to be set for special use cases. Normally, use the default values out of Reset.

10.2.4.2 Detailed description

Table 1093:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl or p_sample is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

10.2.4.3 Function steps

- Perform parameter checking
- Set the sample state count for the specified register
- Return the error code

10.2.5 R_ADC_ScanConfigure

```
ssp_err_t R_ADC_ScanConfigure ( adc_ctrl_t  * p_api_ctrl ,  adc_channel_cfg_t
const *const p_channel_cfg )
```

10.2.5.1 Brief description

Configure the ADC scan parameters. Channel specific settings are set in this function.

10.2.5.2 Detailed description

Table 1094:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl or p_channel_cfg is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

NOTE: If the Group Mode Priority configuration is set to ADC_GROUP_A_GROUP_B_CONTINUOUS_SCAN, then since Group B will be scanning continuously, Group B Interrupts are disabled and the application will not receive a call-back for Group B scan completion even if a callback is provided. The application will still receive a callback for Group A scan completion if a callback is provided.

NOTE: If the ADC conversion clock is faster than 50 MHz, the Temperature and Voltage sensor will not be accurate across the operating temperature range, so an error is returned.

10.2.5.3 Function steps

- Perform parameter checking
- Configure the hardware based on the configuration
- Save the scan mask locally; this is required for the infoGet function
- Return the error code

10.2.6 R_ADC_InfoGet

```
ssp_err_t R_ADC_InfoGet ( adc_ctrl_t  * p_api_ctrl ,  adc_info_t  * p_adc_info )
```

10.2.6.1 Brief description

This function returns the address of the lowest number configured channel and the total number of bytes to be read in order to read the results of the configured channels and return the ELC Event name. If no channels are configured, then a length of 0 is returned. This function also retrieves the temperature sensor slope. It also returns the calibration data for the sensor if available on this MCU otherwise an invalid calibration data of 0xFFFFFFFF will be returned.

10.2.6.2 Detailed description

Table 1095:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- eventInfoGet

NOTE: : Currently this function call does not support Group Mode operation.

10.2.6.3 Function steps

- Perform parameter checking
- Get a pointer to the base register for the current unit
- Retrieve the scan mask of active channels from the control structure
- If at least one channel is configured, determine the highest and lowest configured channels
- Determine the lowest channel that is configured
- Determine the highest channel that is configured
- Set the mask count so that we start with the highest bit of the 32 bit mask
- Initialize the mask result
- Determine the size of data that must be read to read all the channels between and including the highest and lowest channels.
- If no channels are configured, set the return length 0
- Specify the peripheral name in the ELC list
- Set calibration data to invalid value
- If calibration register is available, retrieve it from the MCU

- Provide the previously retrieved slope information

10.2.7 R_ADC_ScanStart

```
ssp_err_t R_ADC_ScanStart ( adc_ctrl_t  * p_api_ctrl )
```

10.2.7.1 Brief description

This function starts a software scan or enables the hardware trigger for a scan depending on how the triggers were configured in the Open() call. If the Unit was configured for hardware triggering, then this function simply allows the trigger signal (hardware or software) to get to the ADC Unit. The function is not able to control the generation of the trigger itself. If the Unit was configured for software triggering, then this function starts the software triggered scan.

10.2.7.2 Detailed description

Table 1096:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_IN_USE	Running scan is still in progress

10.2.7.3 Function steps

- Perform parameter checking
- If the the normal/GroupA trigger is not set to software, then that the Unit is configured for hardware triggering
- Otherwise, enable software triggering
- Check to see if there is an ongoing scan else start the scan
- Return the error code

10.2.8 R_ADC_ScanStop

```
ssp_err_t R_ADC_ScanStop ( adc_ctrl_t  * p_api_ctrl )
```

10.2.8.1 Brief description

This function stops the software scan or disables the Unit from being triggered by the hardware trigger (internal or external) based on what type of trigger the unit was configured for in the Open() function. Stopping a hardware triggered

scan via this function does not abort an ongoing scan, but prevents the next scan from occurring. Stopping a software triggered scan aborts an ongoing scan.

10.2.8.2 Detailed description

Table 1097:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.

NOTE: Stopping a software scan results in immediate stoppage of the scan irrespective of current state of of the scan. Stopping the hardware scan results in disabling the trigger to prevent future scans from starting but does not affect the current scan.

10.2.8.3 Function steps

- Perform parameter checking
- If the trigger is not software scan, then disallow hardware triggering
- Otherwise, disable software triggering
- Return the error code

10.2.9 R_ADC_CheckScanDone

```
ssp_err_t R_ADC_CheckScanDone ( adc_ctrl_t  * p_api_ctrl )
```

10.2.9.1 Brief description

This function returns the status of any scan process that was started.

10.2.9.2 Detailed description

Table 1098:Return values

Name	Description
SSP_SUCCESS	Successful; the scan is complete.

Table 1098:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_IN_USE	Running scan is still in progress.

NOTE: If the peripheral was configured in single scan mode, then the return value of this function is an indication of the scan status. However, if the peripheral was configured in group mode, then the return value of this function could be an indication of either the group A or group B scan state. This is because the ADST bit is set when a scan is ongoing and cleared when the scan is done. This function should normally only be used when using software trigger in single scan mode.

10.2.9.3 Function steps

- Perform parameter checking
- Read the status of the ADST bit
- Return the error code

10.2.10 R_ADC_Read

```
ssp_err_t R_ADC_Read ( adc_ctrl_t  * p_api_ctrl ,  adc_register_t  const reg_id ,
 adc_data_size_t  *const p_data )
```

10.2.10.1 Brief description

This function reads conversion results from a single channel or sensor register.

10.2.10.2 Detailed description

Table 1099:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_INVALID_POINTER	The parameter p_data is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.

Table 1099:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

10.2.10.3 Function steps

- Perform parameter checking
- Get pointer to appropriate base address. This is repeated here in case parameter checking is disabled.
- Read the data from the requested ADC conversion register and return it
- Return the error code

10.2.11 R_ADC_Close

```
ssp_err_t R_ADC_Close ( adc_ctrl_t  * p_api_ctrl )
```

10.2.11.1 Brief description

This function ends any scan in progress, disables interrupts, and removes power to the A/D peripheral.

10.2.11.2 Detailed description

Table 1100:Return values

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Unit is not open.

10.2.11.3 Function steps

- Perform parameter checking
- Mark driver as closed
- Perform hardware stop for the specific unit
- Release the lock
- Return the error code

10.2.12 R_ADC_VersionGet

```
ssp_err_t R_ADC_VersionGet ( ssp_version_t  *const p_version )
```

10.2.12.1 Brief description

Retrieve the API version number.

10.2.12.2 Detailed description

Table 1101:Return values

Name	Description
SSP_SUCCESS	Successful return.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.2.12.3 Function steps

- Return the version number

10.2.13 Extensions

10.2.13.1 adc_instance_ctrl_t

adc_instance_ctrl_t

Detailed description

ADC instance control block. DO NOT INITIALIZE. Initialized in open.

Variables

- uint16_t unit

  ADC Unit in use.
- int16_t slope_microvolts

  Temperature sensor slope in microvolts/°C.
- adc_mode_t mode

  operational mode
- uint8_t max_resolution

  ADC max resolution: 8, 10, 12, or 14-bit.

- uint8_t pga_available

  PGA available or not on MCU.
- uint8_t tsn_ctrl_available

  Availability of TSN control register.
- uint8_t tsn_calib_available

  Availability of TSn calibration register.
- R_TSN_Control_Type * p_tsn_ctrl_regs

  Pointer to temperature control register.
- R_TSN_Calibration_Type * p_tsn_calib_regs

  Pointer to temperature calibration register.
- void const * p_context

  Placeholder for user data.
- void * p_reg

  Base register for this unit.
- void(* callback)(*p_args)

  User callback pointer.
- adc_trigger_t trigger

  Trigger defined for normal mode.
- uint32_t opened

  Boolean to verify that the Unit has been initialized.
- uint32_t scan_mask

  Scan mask used for Normal scan.
- IRQn_Type scan_end_irq

  Scan end IRQ number.
- IRQn_Type scan_end_b_irq

  Scan end group B IRQ number.

10.3 AGT

Driver for the Asynchronous General Purpose Timer (AGT).

10.3.1 Summary

Extends Timer Interface.

HAL High-Level Driver for accessing and configuring AGT timer modes.

The AGT timer functions are used by the Timer to provide timer services.

10.3.2 Functions

- R_AGT_TimerOpen
- R_AGT_Close
- R_AGT_CounterGet
- R_AGT_PeriodSet
- R_AGT_DutyCycleSet
- R_AGT_Reset
- R_AGT_Start
- R_AGT_InfoGet
- R_AGT_Stop
- R_AGT_VersionGet

10.3.3 R_AGT_TimerOpen

```
ssp_err_t R_AGT_TimerOpen ( timer_ctrl_t  *const p_api_ctrl ,  timer_cfg_t  const
*const p_cfg )
```

10.3.3.1 Brief description

Open the AGT channel as a timer, handles required initialization described in hardware manual. Implements open.

10.3.3.2 Detailed description

The Timer Open function configures a single AGT channel for timer mode with parameters specified in the timer Configuration structure. It also sets up the control block for use with subsequent AGT Timer APIs.

This function must be called once prior to calling any other AGT API functions. After a channel is opened, the Open function should not be called again for the same channel without first calling the associated Close function.

The AGT hardware does not support one-shot functionality natively. The one-shot feature is therefore implemented in the AGT HAL layer. For a timer configured as a one-shot timer, the timer is stopped upon the first timer expiration.

The AGT implementation of the general timer can accept an optional timer_on_agt_cfg_t extension parameter. For AGT, the extension specifies the clock to be used as timer source and the output pin configurations. If the extension parameter is not specified (NULL), the default clock PCLKB is used and the output pins are disabled.

The clock divider is selected based on the source clock frequency and the timer period supplied by the caller.

Table 1102:Return values

Name	Description
SSP_SUCCESS	Initialization was successful and timer has started.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_cfg, p_ctrl, or the configuration channel ID exceeds AGT_MAX_CH, or the configuration mode is invalid.
SSP_ERR_IN_USE	The channel specified has already been opened.
SSP_ERR_IRQ_BSP_DISABLED	A required interrupt has not been enabled in the BSP.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet
- systemClockFreqGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.3.3.3 Function steps

- Verify channel is not already used
- Power on the AGT channel.
- Wait for counter to stop.
- Clear AGTO output
- Clear TEDGSEL bit to normal output.
- Set the AGT mode based on agt_mode value.
- Make sure period is valid, then set period
- Start the timer if requested by user
- All done.

10.3.4 R_AGT_Close

```
ssp_err_t R_AGT_Close ( timer_ctrl_t  *const p_api_ctrl )
```

10.3.4.1 Detailed description

Stops counter, disables interrupts, disables output pins, and clears internal driver data. Implements close.

Table 1103:Return values

Name	Description
SSP_SUCCESS	The AGT Timer channel is successfully closed.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The AGT channel is not opened.

10.3.4.2 Function steps

- Cleanup the device: Stop counter, disable interrupts, and power down if no other channels are in use.
- Clear the TOE (output enable) bit
- Clear the TEDGSEL bit
- Unlock channel

10.3.5 R_AGT_CounterGet

```
ssp_err_t R_AGT_CounterGet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
*const p_value )
```

10.3.5.1 Detailed description

Retrieve and store counter value in provided p_value pointer. Implements counterGet.

Table 1104:Return values

Name	Description
SSP_SUCCESS	Counter value read, p_value is valid.
SSP_ERR_ASSERTION	The p_ctrl or p_value parameter was null
SSP_ERR_NOT_OPEN	The channel is not opened.

10.3.5.2 Function steps

- Read counter value

10.3.6 R_AGT_PeriodSet

```
ssp_err_t R_AGT_PeriodSet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
const period ,  timer_unit_t  const unit )
```

10.3.6.1 Detailed description

Sets period value provided. Implements periodSet.

Table 1105:Return values

Name	Description
SSP_SUCCESS	Period value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_INVALID_ARG	One of the following is invalid: • p_period->unit: must be one of the options from timer_size_t::unit • p_period->value: must result in a period supported by the clock source specified during the Open call.
SSP_ERR_NOT_OPEN	The channel is not opened.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- systemClockFreqGet

10.3.6.2 Function steps

- Make sure period is valid, then set period

10.3.7 R_AGT_DutyCycleSet

```
ssp_err_t R_AGT_DutyCycleSet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
const duty_cycle ,  timer_pwm_unit_t  const unit ,   uint8_t const pin )
```

10.3.7.1 Detailed description

Setting duty cycle is not supported by this driver. Implements dutyCycleSet.

Table 1106:Return values

Name	Description
SSP_SUCCESS	Once duty cycle set successfully.
SSP_ERR_INVALID_ARGUMENT	If any of the argument is invalid
SSP_ERR_NOT_OPEN	The channel is not opened.

10.3.7.2 Function steps

- Set duty cycle.

10.3.8 R_AGT_Reset

```
ssp_err_t R_AGT_Reset ( timer_ctrl_t  *const p_api_ctrl )
```

10.3.8.1 Detailed description

Resets the counter value to the period that was originally set. Implements reset.

Table 1107:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.3.8.2 Function steps

- Save running status for restart, if already running.
- Reset the counter to the period originally sent in.
- Restart the AGT channel if it was running.

10.3.9 R_AGT_Start

```
ssp_err_t R_AGT_Start ( timer_ctrl_t  *const p_api_ctrl )
```

10.3.9.1 Detailed description

Starts timer. Implements start.

Table 1108:Return values

Name	Description
SSP_SUCCESS	Timer successfully started.
SSP_ERR_ASSERTION	The p_ctrl parameter is null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.3.9.2 Function steps

- Start timer

10.3.10 R_AGT_InfoGet

```
ssp_err_t R_AGT_InfoGet ( timer_ctrl_t  *const p_api_ctrl ,  timer_info_t
*const p_info )
```

10.3.10.1 Brief description

Get timer information and store it in provided pointer p_info. Implements infoGet.

10.3.10.2 Detailed description

Table 1109:Return values

Name	Description
SSP_SUCCESS	Period, status, count direction, frequency value written to caller's structure successfully.
SSP_ERR_ASSERTION	The p_ctrl or p_period_counts parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INVALID_HW_CONDITION	Invalid hardware setting is detected.

10.3.10.3 Function steps

- Get and store period
- Get and store clock frequency
- AGT supports only counting down direction

10.3.11 R_AGT_Stop

```
ssp_err_t R_AGT_Stop ( timer_ctrl_t  *const p_api_ctrl )
```

10.3.11.1 Detailed description

Stops the AGT channel specified by the handle (control block). This API implements stop. This API does not reset the channel or power it down.

Table 1110:Return values

Name	Description
SSP_SUCCESS	Timer successfully stopped.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.3.12 R_AGT_VersionGet

```
ssp_err_t R_AGT_VersionGet ( ssp_version_t  *const p_version )
```

10.3.12.1 Detailed description

Sets driver version based on compile time macros. Implements versionGet.

Table 1111:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.3.13 API Data

10.3.13.1 agt_count_source_t

agt_count_source_t

Detailed description

Count source

Enumerated values

Name	Description
AGT_CLOCK_PCLKB	Counter clock source is PCLKB when AGT_CLOCK_PCLKB, AGT_CLOCK_PCLKB_DIV_2, or AGT_CLOCK_PCLKB_DIV_8 is selected. The PCLKB divisor is selected automatically at runtime to the optimal value of PCLKB/1, PCLKB/2, or PCLKB/8. If the timer_cfg_t::unit is TIMER_UNIT_PERIOD_RAW_COUNTS, the timer_cfg_t::period should be the desired value in PCLKB counts, even if the value would exceed 16 bits. For example, if a period of 0x30000 counts is requested, a divisor of PCLKB/8 is be selected and the counter underflows after 0x6000 counts.
AGT_CLOCK_PCLKB_DIV_8	Superseded: See AGT_CLOCK_PCLKB.
AGT_CLOCK_PCLKB_DIV_2	Superseded: See AGT_CLOCK_PCLKB.
AGT_CLOCK_LOCO	Divided clock LOCO specified y bits CKS[2:0] in the AGTMR2 register.
AGT_CLOCK_AGT0_UNDERFLOW	Underflow event signal from AGT0.
AGT_CLOCK_FSUB	Divided clock fSUB specified by bits CKS[2:0] in the AGTMR2 register.

10.3.14 Extensions

10.3.14.1 agt_instance_ctrl_t

agt_instance_ctrl_t

Detailed description

Channel control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- void(* p_callback)(*p_args)

  Callback provided when a timer ISR occurs. NULL indicates no CPU interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user callback in timer_callback_args_t.
- void * p_reg

  Base register for this channel.
- uint32_t open

  Whether or not channel is open.
- uint16_t period

  Current timer period (counts)
- uint8_t channel

  Channel number.
- IRQn_Type irq

  Counter overflow IRQ number.
- timer_mode_t mode

  Timer mode.

10.3.14.2 timer_on_agt_cfg_t

timer_on_agt_cfg_t

Detailed description

Optional AGT extension data structure.

Variables

- agt_count_source_t count_source

  AGT channel clock source. Valid values are: AGT_CLOCK_PCLKB, AGT_CLOCK_LOCO, AGT_-CLOCK_FSUB.
- bool agto_output_enabled

  AGTO pin is enabled for output compare (true, false)
- bool agtio_output_enabled

  AGTIO pin is enabled for output compare (true, false)
- bool output_inverted

  Output inverted (true, false)
- bool agtoa_output_enable

  Enable comparator A output pin (true, false)

- bool agtob_output_enable

  Enable comparator B output pin (true, false)

10.4 CAC

Driver for the Clock Frequency Accuracy Measurement Circuit (CAC).
Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

10.4.1 Summary

This module supports the CAC peripheral.

10.4.2 Functions

- R_CAC_Open
- R_CAC_Close
- R_CAC_StopMeasurement
- R_CAC_StartMeasurement
- R_CAC_Reset
- R_CAC_Read
- R_CAC_VersionGet
- cac_setup
- cac_disable_nvic_interrupts
- cac_fmi_setup
- cac_setup_vectors
- cac_irq_cfg
- cac_enable_interrupts
- cac_frequency_error_isr
- cac_overflow_isr
- cac_measurement_end_isr

10.4.3 Defines

- #define CAC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

- #define CAC_CODE_VERSION_MINOR

  Initial value:(7U)

10.4.4 R_CAC_Open

```
ssp_err_t R_CAC_Open ( cac_ctrl_t  *const p_api_ctrl ,  cac_cfg_t  const
*const p_cfg )
```

10.4.4.1 Brief description

Initialize the CAC peripheral.

10.4.4.2 Detailed description

The Open function applies power to the CAC peripheral, checks/sets the interrupt priority, and configures the CAC based on the provided user configuration settings. If a user defined callback function has been provided in the configuration, then the CAC interrupt(s) will be enabled and the user callback function called accordingly. Implements r_cac_t::open.

Table 1112:Return values

Name	Description
SSP_SUCCESS	CAC is available and available for measurement(s).
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_INVALID_ARGUMENT	One or more configuration options are invalid.
SSP_ERR_HW_LOCKED	Hardware lock for CAC peripheral is already taken.
SSP_ERR_INVALID_CAC_REF_CLOCK	Measured clock rate smaller than reference clock rate.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: There is only a single CAC peripheral. It is not reentrant.

10.4.4.3 Function steps

- g_cac_version is accessed by the ASSERT macro only and so compiler toolchain can issue a warning that they are not accessed. The code below eliminates this warning and also ensures these data structures are not optimised away.
- Eliminate warning if parameter checking is disabled.
- Take the hardware lock for the CAC.

- Setup the interrupt vectors and priorities
- Return the hardware lock for the CAC.
- Apply power to the peripheral
- Configure the CAC per the configuration.
- Store the callback and context information
- Mark driver as open by initializing it to "CAC" - its ASCII equivalent.

10.4.5 R_CAC_Close

```
ssp_err_t R_CAC_Close ( cac_ctrl_t  *const p_api_ctrl )
```

10.4.5.1 Brief description

Release any resources that were allocated by the Open() or any subsequent CAC operations. Implements r_cac_t::close.

10.4.5.2 Detailed description

Table 1113:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.
SSP_ERR_NOT_OPEN	R_CAC_Open has not been successfully called.

10.4.5.3 Function steps

- Eliminate warning if parameter checking is disabled.
- < Disable interrupts in the peripheral and NVIC
- Disable interrupts in NVIC.
- Disable the CAC ints.
- Stop measuring.
- Power down peripheral.
- Return the hardware lock for the CAC.

10.4.6 R_CAC_StopMeasurement

```
ssp_err_t R_CAC_StopMeasurement ( cac_ctrl_t  *const p_api_ctrl )
```

10.4.6.1 Brief description

Stop the CAC measurement process. Implements r_cac_t::stopMeasurement.

10.4.6.2 Detailed description

Table 1114:Return values

Name	Description
SSP_SUCCESS	CAC measuring has been stoped.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.
SSP_ERR_NOT_OPEN	R_CAC_Open has not been successfully called.

10.4.6.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Stop measuring.

10.4.7 R_CAC_StartMeasurement

```
ssp_err_t R_CAC_StartMeasurement ( cac_ctrl_t  *const p_api_ctrl )
```

10.4.7.1 Brief description

Start the CAC measurement process. Implements r_cac_t::startMeasurement.

10.4.7.2 Detailed description

Table 1115:Return values

Name	Description
SSP_SUCCESS	CAC measurement started.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.
SSP_ERR_NOT_OPEN	R_CAC_Open has not been successfully called.
SSP_ERR_CLOCK_INACTIVE	Either the provided Measurement or Reference clock is not running

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- clockCheck

10.4.7.3 Function steps

- Eliminate warnings if parameter checking is disabled.
- Start measuring.

10.4.8 R_CAC_Reset

```
ssp_err_t R_CAC_Reset ( cac_ctrl_t  *const p_api_ctrl )
```

10.4.8.1 Brief description

Resets the Overflow, Measurement End and Frequency Error interrupt flags. This will clear any of the CASTR status bits that have been set, but only if the CFME bit is off (Not measuring). Implements r_cac_t::reset.

10.4.8.2 Detailed description

Table 1116:Return values

Name	Description
SSP_SUCCESS	CAC reset completed.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.
SSP_ERR_NOT_OPEN	R_CAC_Open has not been successfully called.

10.4.8.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Reset the CAC.

10.4.9 R_CAC_Read

```
ssp_err_t R_CAC_Read ( cac_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_status ,   uint16_t *const p_counter )
```

10.4.9.1 Brief description

Read and return the CAC status and counter registers. Implements r_cac_t::read.

10.4.9.2 Detailed description

Table 1117:Return values

Name	Description
SSP_SUCCESS	CAC read successful.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.
SSP_ERR_NOT_OPEN	R_CAC_Open has not been successfully called.

10.4.9.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.4.10 R_CAC_VersionGet

```
ssp_err_t R_CAC_VersionGet ( ssp_version_t  *const p_version )
```

10.4.10.1 Brief description

Get the API and code version information.

10.4.10.2 Detailed description

Table 1118:Return values

Name	Description
SSP_SUCCESS	Version info returned.

NOTE: This function is reentrant.

10.4.11 cac_setup

```
 cac_setup ( cac_instance_ctrl_t  *const p_ctrl ,  cac_cfg_t  const
*const p_cfg )
```

10.4.11.1 Detailed description

Configure the CAC per the user's configuration

Table 1119:Parameters

Name	Direction	Description
p_cfg	in	- pointer to the CAC configuration settings.
p_ctrl	in	- CAC instance control block

10.4.11.2 Function steps

- Disable interrupt in ICU

10.4.12 cac_disable_nvic_interrupts

```
cac_disable_nvic_interrupts ( cac_instance_ctrl_t  *const p_ctrl )
```

10.4.12.1 Detailed description

Disable the CAC interrupts in the NVIC

Table 1120:Parameters

Name	Direction	Description
p_ctrl	in	- CAC instance control block

None

10.4.13 cac_fmi_setup

```
ssp_err_t cac_fmi_setup ( cac_instance_ctrl_t  *const p_ctrl ,  ssp_feature_t
* p_ssp )
```

10.4.13.1 Brief description

This function gets the interrupt vector indexes from the FMI for the CAC interrupts.

10.4.13.2 Detailed description

Table 1121:Parameters

Name	Direction	Description
p_ctrl	in	- CAC instance control block
p_ssp	in	- pointer to ssp feature information

Table 1122:Return values

Name	Description
SSP_SUCCESS	FMI based setup success.
SSP_ERR_ASSERTION	Problem getting FMI information.

10.4.14 cac_setup_vectors

```
ssp_err_t cac_setup_vectors ( cac_instance_ctrl_t  *const p_ctrl ,  cac_cfg_t
const *const p_cfg )
```

10.4.14.1 Brief description

This function verfies that there is a callback function specified if one or more interrupts are enabled and sets the vector table entries and priority for enabled CAC interrupts.

10.4.14.2 Detailed description

Table 1123:Parameters

Name	Direction	Description
p_ctrl	in	- CAC instance control block
p_cfg	in	- pointer to the CAC configuration settings.

Table 1124:Return values

Name	Description
SSP_SUCCESS	Enabled vectors and priorities configured.
SSP_ERR_INVALID_POINTER	Interrupt specified with NULL callback.

10.4.15 cac_irq_cfg

```
 cac_irq_cfg ( cac_instance_ctrl_t  *const p_ctrl ,   bool state ,  cac_cfg_t
const *const p_cfg )
```

10.4.15.1 Brief description

This function enables or disables the CAC interrupts. There are 3 possible CAC interrupts, each of which the user may choose to enable or disable. These are the Frequency Error interrupt, the Measurement Complete interrupt and the Overflow interrupt. If the caller has provided a callback function as part of the provided p_cfg pointer, then that function will be called as a result of the interrupt.

10.4.15.2 Detailed description

Table 1125:Parameters

Name	Direction	Description
p_ctrl	in	CAC instance control block
state	in	true ==> enable interrupts, false ==> disable interrupts.
p_cfg	in	Pointer to the CAC configuration structure.

10.4.15.3 Function steps

- Regardless of whether we will enable or disable, Clear the Interrupt Request bits
- Enable the Interrupts if requested
- Assign the callback
- Disable interrupt in NVIC
- < Disable interrupts in the peripheral

10.4.16 cac_enable_interrupts

```
cac_enable_interrupts ( cac_instance_ctrl_t  *const p_ctrl ,  cac_cfg_t  const
*const p_cfg )
```

10.4.16.1 Brief description

This function enables the CAC interrupts enabled in the supplied configuration.

10.4.16.2 Detailed description

Table 1126:Parameters

Name	Direction	Description
p_ctrl	in	CAC instance control block
p_cfg	in	Pointer to the CAC configuration structure.

10.4.16.3 Function steps

- < Disable interrupts in the peripheral
- Enable/Disable CAC interrupts as requested by the user.

10.4.17 cac_frequency_error_isr

```
cac_frequency_error_isr (  void  )
```

10.4.17.1 Brief description

CAC Frequency error interrupt routine.

10.4.17.2 Detailed description

This function implements the CAC Frequency error isr. The function clears the interrupt request source on entry populates the callback structure with the relevant event, and providing a callback routine has been provided, calls the callback function with the event.

Table 1127:Return values

Name	Description
none	

10.4.17.3 Function steps

- The CAC interrupts are level based which means the bits in the peripheral must be cleared before clearing the IR bit in the IELSRn register.
- Clear the interrupt condition in the CAC peripheral
- Clear the IR bit in the IELSRn register
- Set data to identify callback to user, then call user callback.

10.4.18 cac_overflow_isr

```
cac_overflow_isr (  void  )
```

10.4.18.1 Brief description

CAC Overflow interrupt routine.

10.4.18.2 Detailed description

This function implements the CAC overflow isr. The function clears the interrupt request source on entry populates the callback structure with the relevant event, and providing a callback routine has been provided, calls the callback function with the event.

Table 1128:Return values

Name	Description
none	

10.4.18.3 Function steps

- The CAC interrupts are level based which means the bits in the peripheral must be cleared before clearing the IR bit in the IELSRn register.
- Clear the interrupt condition in the CAC peripheral
- Clear the IR bit in the IELSRn register

- Set data to identify callback to user, then call user callback.

10.4.19 cac_measurement_end_isr

```
cac_measurement_end_isr (  void  )
```

10.4.19.1 Brief description

CAC Measurement Complete interrupt routine.

10.4.19.2 Detailed description

This function implements the CAC measurement complete isr. The function clears the interrupt request source populates the callback structure with the relevant event, and providing a callback routine has been provided, calls the callback function with the event.

Table 1129:Return values

Name	Description
none	

10.4.19.3 Function steps

- The CAC interrupts are level based which means the bits in the peripheral must be cleared before clearing the IR bit in the IELSRn register.
- Clear the interrupt condition in the CAC peripheral
- Clear the IR bit in the IELSRn register
- Set data to identify callback to user, then call user callback.

10.4.20 Extensions

10.4.20.1 cac_instance_ctrl_t

cac_instance_ctrl_t

Detailed description

CAC instance control block. DO NOT INITIALIZE.

Variables

- void * p_reg

  Pointer to register base address.

- void(* p_callback)(*cb_data)

  Called from the ISR.
- void const * p_context

  Passed to the callback.
- IRQn_Type frequency_error_irq

  Frequency error IRQ number.
- IRQn_Type measurement_end_irq

  Measurement end IRQ number.
- IRQn_Type overflow_irq

  Overflow IRQ number.
- uint32_t cac_api_open

  Set to "CAC" once API has been successfully opened.
- bool cac_continous_mode

  Set as a result of the Open() call.
- bsp_lock_t cac_lock

  CAC commands software lock.
- cac_clock_source_t measurement_clock

  Clock specified in Open() as the measurement clock.
- cac_clock_source_t reference_clock

  Clock specified in Open() as the reference clock.
- void const * p_extend

10.5 CAN

Driver for CAN, Controller Area Network.

This module supports the Controller Area Network peripheral. It implements the following interfaces:

- CAN Interface

10.5.1 Functions

- R_CAN_Open
- R_CAN_Close
- R_CAN_Read
- R_CAN_Write

- R_CAN_Control
- R_CAN_InfoGet
- R_CAN_VersionGet
- can_error_interrupt
- can_error_isr
- can_receive_interrupt
- can_mailbox_rx_isr
- can_transmit_interrupt
- can_mailbox_tx_isr
- can_irq_disable
- can_open_parameters_check
- can_open_parameters_check_timing
- can_open_parameters_check_clock
- can_write_parameters_check
- can_module_start
- can_irq_enable
- can_operate_mode
- can_wake_and_init
- can_mailbox_configure

10.5.2 Defines

- #define CAN_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CAN_CODE_VERSION_MINOR

  Initial value:`(3U)`

10.5.3 R_CAN_Open

```
ssp_err_t R_CAN_Open ( can_ctrl_t  *const p_ctrl ,  can_cfg_t  const
*const p_cfg )
```

10.5.3.1 Brief description

Open and configure the CAN channel for operation. Implements open

10.5.3.2 Detailed description

Table 1130:Return values

Name	Description
SSP_SUCCESS	Channel opened successfully
SSP_ERR_INVALID_ARGUMENT	Invalid channel passed as argument.
SSP_ERR_HW_LOCKED	Lock already owned by another user.
SSP_ERR_CAN_MODE_SWITCH_FAILED	Channel failed to switch modes.
SSP_ERR_CAN_INIT_FAILED	Channel failed to initialize.
SSP_ERR_ASSERTION	Null pointer presented.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet
- clockCheck

10.5.3.3 Function steps

- Check for valid parameters.
- Make sure the feature exists on this MCU.
- Return if failed to get feature information.
- Try to get channel lock.
- Return if channel is already open so return error
- Enter module start state.
- Disable interrupts while initializing
- Initialize and configure CAN module to run.
- Set channel, callback function, context, id mode, mailbox count, message mode, op mode and opened status.
- If successful, Lookup and store IRQ numbers. Enable interrupts.
- If successful, Mark the control block as open
- If the device failed to initialize, disable interrupts, stop and unlock the hardware and mark the control block as closed.
- Process errors before returning.
- Log error or assertion.

10.5.4 R_CAN_Close

```
ssp_err_t R_CAN_Close ( can_ctrl_t  *const p_ctrl )
```

10.5.4.1 Brief description

Close the CAN channel. Implements close

10.5.4.2 Detailed description

End of function R_CAN_Open

Table 1131:Return values

Name	Description
SSP_SUCCESS	Channel closed successfully.
SSP_ERR_NOT_OPEN	Port is not open.
SSP_ERR_ASSERTION	Null pointer presented.

10.5.4.3 Function steps

- Mark the channel not open so other APIs cannot use it.
- Disable transmit, receive and error interrupts
- Enable module stop for the CAN channel
- Unlock the CAN channel

10.5.5 R_CAN_Read

```
ssp_err_t R_CAN_Read ( can_ctrl_t  *const p_ctrl ,   uint32_t mailbox ,
 can_frame_t  *const p_frame )
```

10.5.5.1 Brief description

Read data from the CAN channel. Return up to eight bytes read from the channel mailbox. Implements read

10.5.5.2 Detailed description

Table 1132:Return values

Name	Description
SSP_SUCCESS	Data successfully read.
SSP_ERR_CAN_DATA_UNAVAILABLE	No data available.
SSP_ERR_CAN_TRANSMIT_MAILBOX	Mailbox is not setup for receive.
SSP_ERR_ASSERTION	Null pointer presented.

10.5.5.3 Function steps

- Check for receive data
- Get frame data.

10.5.6 R_CAN_Write

```
ssp_err_t R_CAN_Write ( can_ctrl_t  *const p_ctrl ,   uint32_t mailbox ,
 can_frame_t  *const p_frame )
```

10.5.6.1 Brief description

Write data to the CAN channel. Write up to eight bytes to the channel mailbox. Implements write

10.5.6.2 Detailed description

Table 1133:Return values

Name	Description
SSP_SUCCESS	Operation succeeded.
SSP_ERR_CAN_TRANSMIT_NOT_READY	Transmit in progress, cannot write data at this time.
SSP_ERR_CAN_RECEIVE_MAILBOX	Mailbox is setup for receive and cannot send.
SSP_ERR_INVALID_ARGUMENT	Data length or frame type invalid.
SSP_ERR_ASSERTION	Null pointer presented

10.5.6.3 Function steps

- Check transmit ready flag.
- Transmit ready flag is not set, return error/status.
- Transmit ready flag set, so clear it.
- Send transmit frame.

10.5.7 R_CAN_Control

```
ssp_err_t R_CAN_Control ( can_ctrl_t  *const p_ctrl ,  can_command_t
const command ,   void * p_data )
```

10.5.7.1 Brief description

CAN Control is used to control extended features. Implements control

10.5.7.2 Detailed description

Table 1134:Return values

Name	Description
SSP_SUCCESS	Operation succeeded.
SSP_ERR_INVALID_ARGUMENT	Invalid command.
SSP_ERR_ASSERTION	Null pointer presented
SSP_ERR_CAN_MODE_SWITCH_FAILED	Switching modes failed.

10.5.7.3 Function steps

- Verify command is CAN_COMMAND_MODE_SWITCH
- Change operating mode. Returns false if invalid mode or mode switch failed.
- Save mode for diagnostic purposes.

10.5.8 R_CAN_InfoGet

```
ssp_err_t R_CAN_InfoGet ( can_ctrl_t  *const p_ctrl ,  can_info_t
*const p_info )
```

10.5.8.1 Brief description

Get CAN state and status information for the channel. Implements infoGet

10.5.8.2 Detailed description

Table 1135:Return values

Name	Description
SSP_SUCCESS	Operation succeeded.
SSP_ERR_CAN_DATA_UNAVAILABLE	Channel failed to return info.
SSP_ERR_ASSERTION	Null pointer presented

10.5.8.3 Function steps

- Get status for channel.
- Error encountered when retrieving info.
- Save the operation mode

10.5.9 R_CAN_VersionGet

```
ssp_err_t R_CAN_VersionGet ( ssp_version_t  *const p_version )
```

10.5.9.1 Brief description

Get CAN module code and API versions. Implements versionGet

10.5.9.2 Detailed description

Table 1136:Return values

Name	Description
SSP_SUCCESS	Operation succeeded.
SSP_ERR_ASSERTION	Null pointer presented note This function is reentrant.

10.5.9.3 Function steps

- Return module version information.

10.5.10 can_error_interrupt

```
can_error_interrupt ( can_instance_ctrl_t  * p_ctrl )
```

10.5.10.1 Brief description

CAN Common Error ISR.

10.5.10.2 Detailed description

Gets errors, saves events and calls callback function, if used.

Table 1137:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAN instance control block

10.5.10.3 Function steps

- Get source of error interrupt.
- < Check for bus off error.
- < Set event argument to bus error.
- < Check for bus recovery error.
- < Set event argument to bus recovery.
- < Check for bus error passive.
- < Set event argument to error passive.
- < Check for bus error warning.
- < Set event argument to error warning.
- < Check for receive overwrite / overrrun.
- < Set event to overrrun error.
- Follow code rules
- Get user the callback, if set in the open function.
- < Populate callback arguments accordingly.
- < Call the user callback function.

10.5.11 can_error_isr

```
can_error_isr (  void  )
```

10.5.11.1 Brief description

Error ISR.

10.5.11.2 Detailed description

Saves context if RTOS is used, clears interrupts, calls common error function, and restores context if RTOS is used.

10.5.11.3 Function steps

- Save context if RTOS is used
- Clear interrupt
- Check that the pointer is not NULL prior to using it
- Restore context if RTOS is used

10.5.12 can_receive_interrupt

```
can_receive_interrupt ( can_instance_ctrl_t  * p_ctrl )
```

10.5.12.1 Brief description

CAN Common Receive ISR.

10.5.12.2 Detailed description

Identifies mailbox, saves receive event and calls callback function, if used.

Table 1138:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAN instance control block

10.5.12.3 Function steps

- Get user the callback, if set in the open function.
- Set event argument to receive complete.
- Save the receive mailbox number.

- < Populate callback arguments accordingly.
- < Call the user callback function.

10.5.13 can_mailbox_rx_isr

```
can_mailbox_rx_isr (  void  )
```

10.5.13.1 Brief description

Receive ISR.

10.5.13.2 Detailed description

Saves context if RTOS is used, clears interrupts, calls common receive function and restores context if RTOS is used.

10.5.13.3 Function steps

- Save context if RTOS is used
- Clear interrupt
- Check that the pointer is not NULL prior to using it
- Restore context if RTOS is used

10.5.14 can_transmit_interrupt

```
can_transmit_interrupt ( can_instance_ctrl_t  * p_ctrl )
```

10.5.14.1 Brief description

CAN Common Transmit ISR.

10.5.14.2 Detailed description

Identifies mailbox, saves transmit event and calls callback function, if used.

Table 1139:Parameters

Name	Direction	Description
p_ctrl	in	Pointer to CAN instance control block

10.5.14.3 Function steps

- Check CAN transmit.
- Get user the callback, if set in the open function.
- < clear SENTDATA and TRMREQ.
- Save the transmit mailbox number.
- Set event argument to transmit complete.
- < Populate callback arguments accordingly.
- < Call the user callback function.

10.5.15 can_mailbox_tx_isr

```
can_mailbox_tx_isr (  void  )
```

10.5.15.1 Brief description

Transmit ISR.

10.5.15.2 Detailed description

Saves context if RTOS is used, clears interrupts, calls common transmit function and restores context if RTOS is used.

10.5.15.3 Function steps

- Save context if RTOS is used
- Clear interrupt
- Check that the pointer is not NULL prior to using it
- Restore context if RTOS is used

10.5.16 can_irq_disable

```
can_irq_disable ( can_instance_ctrl_t  * p_ctrl )
```

10.5.16.1 Brief description

Disable and clear ISRs.

10.5.16.2 Detailed description

Disables and clears CAN interrupts.

Table 1140:Parameters

Name	Direction	Description
p_ctrl	in	CAN Instance control block

10.5.16.3 Function steps

- Disable and clear error interrupt.
- Disable and clear receive interrupt.
- Disable and clear transmit interrupt.

10.5.17 can_open_parameters_check

```
ssp_err_t can_open_parameters_check ( can_instance_ctrl_t  *const p_ctrl ,
 can_cfg_t  const *const p_cfg )
```

10.5.17.1 Brief description

Check for Open function NULL pointers and valid mailbox settings.

10.5.17.2 Detailed description

Table 1141:Parameters

Name	Direction	Description
p_internal_ctrl	in	CAN instance control block
p_cfg	in	CAN configuration

SSP_SUCCESS Pointers are valid. Mailbox count is valid. Control block is not currently open. SSP_ERR_ASSERTION Invalid pointer SSP_ERR_CAN_INIT_FAILED Invalid mailbox count. SSP_ERR_IN_USE Control block is already open.

10.5.17.3 Function steps

- Check for null pointers
- Check for valid mailbox count
- Check control block isn't already open

10.5.18 can_open_parameters_check_timing

```
ssp_err_t can_open_parameters_check_timing ( can_cfg_t  const *const p_cfg )
```

10.5.18.1 Brief description

Check for Open function valid timing settings.

10.5.18.2 Detailed description

Table 1142:Parameters

Name	Direction	Description
p_cfg	in	CAN Configuration

SSP_SUCCESS Valid timing settings SSP_ERR_CAN_INIT_FAILED Invalid timing settings

10.5.18.3 Function steps

- Check p_bit_timing is not null
- Check baud Rate Prescaler.
- Check Time Segment 1 is greater than Time Segment 2.
- Check Time Segment 2 is greater than or equal to the synchronization jump width

10.5.19 can_open_parameters_check_clock

```
ssp_err_t can_open_parameters_check_clock ( can_cfg_t  const *const p_cfg ,
 bsp_feature_can_t  can_feature )
```

10.5.19.1 Brief description

Check for Open function valid clock settings.

10.5.19.2 Detailed description

Table 1143:Parameters

Name	Direction	Description
p_cfg	in	CAN Configuration

Table 1143:Parameters (Continued)

Name	Direction	Description
can_feature	in	CAN features

SSP_SUCCESS Valid timing settings SSP_ERR_CAN_INIT_FAILED Invalid timing settings See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- systemClockGet
- systemClockFreqGet

10.5.19.3 Function steps

- Get the frequency of pclkb for later validation
- If the hardware manual states pclkb must be a 2:1 ratio of another clock
- Get the other clock defined by the hardware manual from the bsp and verify there is a 2:1 ratio
- If the devices is configured for CANMCLK or can only use CANMCLK
- Verify pclkb is greater than or equal to canmclk
- Otherwise the device is configured for PCLKB. Verify the source clock is the PLL

10.5.20 can_write_parameters_check

```
ssp_err_t can_write_parameters_check ( can_ctrl_t  *const p_ctrl ,  can_frame_t
*const p_frame ,   uint32_t mailbox )
```

10.5.20.1 Brief description

Check for Write function for invalid parameters.

10.5.20.2 Detailed description

Table 1144:Parameters

Name	Direction	Description
p_ctrl	in	CAN instance control block
p_frame	in	CAN frame
mailbox	in	CAN mailbox number

SSP_SUCCESS Vaild pointers, frame size, mailbox type and mailbox number. Control block is open.
SSP_ERR_ASSERTION Invalid pointer SSP_ERR_NOT_OPEN Control block not open
SSP_ERR_CAN_RECEIVE_MAILBOX Mailbox is configured to receive SSP_ERR_INVALID_ARGUMENT Invalid frame size or mailbox number

10.5.20.3 Function steps

- Check for null pointers
- Make sure mailbox is setup for transmit.
- If channel is not open, return an error
- Check mailbox number
- Check Mailbox type
- Check frame length

10.5.21 can_module_start

```
ssp_err_t can_module_start ( can_instance_ctrl_t  * p_internal_ctrl ,
 can_cfg_t  const *const p_cfg ,  bsp_feature_can_t  can_feature )
```

10.5.21.1 Brief description

Wake and initialize the device. Configure the mailboxes. Configure the device for normal mode operation.

10.5.21.2 Detailed description

Table 1145:Parameters

Name	Direction	Description
p_internal_ctrl	in	CAN instance control block
p_cfg	in	CAN configuration
can_feature	in	CAN features

SSP_SUCCESS CAN device has been configured correctly See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- clockCheck

10.5.21.3 Function steps

- Get the extended configuration
- Initialize the control block

- Check if only CANMCLK is supported
- Set clock source to CANMCLK
- Set the clcok source to the user configured source.
- Put the device in reset mode, configure the device, then go to halt mode.
- Configure mailboxes.
- Go to normal operation.

10.5.22 can_irq_enable

```
 can_irq_enable ( can_instance_ctrl_t  * p_ctrl ,  can_cfg_t  const
*const p_cfg ,  ssp_feature_t  * p_feature )
```

10.5.22.1 Brief description

Set vectors.

10.5.22.2 Detailed description

Table 1146:Parameters

Name	Direction	Description
p_ctrl	in	CAN instance control block
p_cfg	in	CAN configuration
p_feature	in	CAN feature definition for this channel This function calls: • eventInfoGet

10.5.22.3 Function steps

- Check for a valid error interrupt vector
- Enable error interrupt.
- Check for a valid receive interrupt vector
- Enable receive interrupt.
- Check for a valid transmit interrupt vector
- Enable transmit interrupt.
- If either the transmit or receive interrupts are enabled, enable TX/RX interrupts on the CAN device.

10.5.23 can_operate_mode

```
ssp_err_t can_operate_mode (  R_CAN0_Type * p_can_regs )
```

10.5.23.1 Brief description

Go to CAN operate mode.

10.5.23.2 Detailed description

Table 1147:Parameters

Name	Direction	Description
p_can_regs	in	CAN registers

SSP_SUCCESS CAN device has entered normal operation mode and reset the timestamp without errors. SSP_ERR_CAN_MODE_SWITCH_FAILED CAN device failed to enter normal mode SSP_ERR_CAN_INIT_FAILED CAN device failed to reset the timestamp or had an error

10.5.23.3 Function steps

- Halt -> Normal OPERATION mode
- Time Stamp Counter reset. Set the TSRC bit to 1 in CAN Operation mode.
- Check for errors so far, report, and clear.

10.5.24 can_wake_and_init

```
ssp_err_t can_wake_and_init ( can_instance_ctrl_t  * p_internal_ctrl ,
 can_bit_timing_cfg_t  *const p_timing )
```

10.5.24.1 Brief description

Checks that CANMCLK is running if that is the target clock. Wakes the device and puts it into reset mode. Initializes registers and clock settings. Puts the device in HALT mode.

10.5.24.2 Detailed description

Table 1148:Parameters

Name	Direction	Description
p_internal_ctrl	in	CAN instance control block
p_timing	in	CAN timing configuration

SSP_SUCCESS Device has entered halt mode and initialized properly See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- clockCheck

10.5.24.3 Function steps

- If clock source is CANMCLK verify it is active before changing modes. If it's not then chanigng modes could result in failing in an unknown mode.
- Check the clock has stabilized
- SSP_ERR_STABILIZED is the expected return value for clockCheck with CGC_CLOCK_MAIN_OSC
- Enter reset mode
- BOM: Bus Off recovery mode acc. to IEC11898-1
- MBM: Select normal mailbox mode.
- Select ID mode. Standard or extended
- Select message overwrite or overrun mode
- Select transmission priority mode
- Set time stamp
- Set timing bits
- Select the clock
- Enter halt mode

10.5.25 can_mailbox_configure

```
 can_mailbox_configure (  R_CAN0_Type * p_can_regs ,  can_cfg_t  const
*const p_cfg )
```

10.5.25.1 Brief description

Configure CAN mailboxes.

10.5.25.2 Detailed description

Table 1149:Parameters

Name	Direction	Description
p_can_regs	in	CAN registers
p_cfg	in	CAN configuration

10.5.25.3 Function steps

- Clear mailboxes in Halt mode.
- Set Error Display mode in Halt mode.
- Set the IDs for each mailbox.
- Set the masks for each mailbox group and initialize the mask invalid register.

10.5.26 Extensions

10.5.26.1 can_instance_ctrl_t

can_instance_ctrl_t

Detailed description

CAN Instance Control Block

Variables

- uint32_t channel

  Channel number.

  Parameters to control CAN peripheral device
- uint32_t open

  Open status of channel.
- can_mode_t operation_mode

  Can operation mode.
- can_id_mode_t id_mode

  Standard or Extended ID mode.
- uint32_t mailbox_count

  Number of mailboxes.

- can_mailbox_t * p_mailbox

  Pointer to mailboxes.
- can_message_mode_t message_mode

  Overwrite message or overrun.
- can_clock_source_t clock_source

  Clock source. CANMCLK or PCLKB.
- void(* p_callback)(*p_args)

  Pointer to callback function.

  Parameters to process CAN Event
- void const * p_context

  Pointer to the higher level device context.
- void * p_reg

  Pointer to register base address.
- IRQn_Type error_irq

  Error IRQ number.
- IRQn_Type mailbox_rx_irq

  Receive mailbox IRQ number.
- IRQn_Type mailbox_tx_irq

  Transmit mailbox IRQ number.

10.5.26.2 can_extended_cfg_t

can_extended_cfg_t

Detailed description

CAN clock configuration and mailbox mask to be pointed to by p_extend.

Variables

- can_clock_source_t clock_source

  Source of the CAN clock.
- uint32_t * p_mailbox_mask

  Mailbox mask, one for every 4 mailboxes.

10.6 CGC

Driver for the Clock Generation Circuit.

Clock Generation Circuit API.

This module supports the Clock Generation Circuit. It implements the following interfaces:

- CGC Interface

10.6.1 Functions

- R_CGC_Init
- R_CGC_ClocksCfg
- R_CGC_ClockStart
- R_CGC_ClockStop
- R_CGC_SystemClockSet
- R_CGC_SystemClockGet
- R_CGC_SystemClockFreqGet
- R_CGC_ClockCheck
- R_CGC_OscStopDetect
- R_CGC_OscStopStatusClear
- R_CGC_BusClockOutCfg
- R_CGC_BusClockOutEnable
- R_CGC_BusClockOutDisable
- R_CGC_ClockOutCfg
- R_CGC_ClockOutEnable
- R_CGC_ClockOutDisable
- R_CGC_LCDClockCfg
- R_CGC_LCDClockEnable
- R_CGC_LCDClockDisable
- R_CGC_SDRAMClockOutEnable
- R_CGC_SDRAMClockOutDisable
- R_CGC_USBClockCfg
- R_CGC_SystickUpdate
- R_CGC_VersionGet
- r_cgc_stabilization_wait
- r_cgc_clock_start_stop

10.6.2 Defines

- #define CGC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CGC_CODE_VERSION_MINOR

  Initial value:`(7U)`

10.6.3 R_CGC_Init

```
ssp_err_t R_CGC_Init (  void  )
```

10.6.3.1 Brief description

Initialize the CGC API.

10.6.3.2 Detailed description

Configures the following for the clock generator module

- SubClock drive capacity (Compile time configurable: CGC_CFG_SUBCLOCK_DRIVE)
- Initial setting for the SubClock

THIS FUNCTION MUST BE EXECUTED ONCE AT STARTUP BEFORE ANY OF THE OTHER CGC FUNCTIONS CAN BE USED OR THE CLOCK SOURCE IS CHANGED FROM THE MOCO.

Table 1150:Return values

Name	Description
SSP_SUCCESS	Clock initialized successfully.
SSP_ERR_HARDWARE_TIMEOUT	Hardware timed out.

10.6.4 R_CGC_ClocksCfg

```
ssp_err_t R_CGC_ClocksCfg ( cgc_clocks_cfg_t  const *const p_clock_cfg )
```

10.6.4.1 Brief description

Reconfigure all main system clocks.

10.6.4.2 Detailed description

Table 1151:Return values

Name	Description
SSP_SUCCESS	Clock initialized successfully.
SSP_ERR_INVALID_ARGUMENT	Invalid argument used.
SSP_ERR_MAIN_OCO_INACTIVE	PLL Initialization attempted with Main OCO turned off/unstable.
SSP_ERR_CLOCK_ACTIVE	Active clock source specified for modification. This applies specifically to the PLL dividers/multipliers which cannot be modified if the PLL is active. It has to be stopped first before modification.
SSP_ERR_NOT_STABILIZED	The Clock source is not stabilized after being turned off.
SSP_ERR_CLKOUT_EXCEEDED	The main oscillator can be only 8 or 16 MHz.
SSP_ERR_ASSERTION	A NULL is passed for configuration data when PLL is the clock_source.
SSP_ERR_INVALID_MODE	Attempt to start a clock in a restricted operating power control mode.

10.6.5 R_CGC_ClockStart

```
ssp_err_t R_CGC_ClockStart ( cgc_clock_t  clock_source ,  cgc_clock_cfg_t
* p_clock_cfg )
```

10.6.5.1 Brief description

Start the specified clock if it is not currently active.

10.6.5.2 Detailed description

Configures the following when starting the Main Clock Oscillator:

- MainClock drive capacity (Configured based on external clock frequency)
- MainClock stabilization wait time (Compile time configurable: CGC_CFG_MAIN_OSC_WAIT)

Table 1152:Return values

Name	Description
SSP_SUCCESS	Clock initialized successfully.
SSP_ERR_INVALID_ARGUMENT	Invalid argument used.
SSP_ERR_MAIN_OCO_INACTIVE	PLL Initialization attempted with Main OCO turned off/unstable.
SSP_ERR_CLOCK_ACTIVE	Active clock source specified for modification. This applies specifically to the PLL dividers/multipliers which cannot be modified if the PLL is active. It has to be stopped first before modification.
SSP_ERR_NOT_STABILIZED	The Clock source is not stabilized after being turned off.
SSP_ERR_CLKOUT_EXCEEDED	The main oscillator can be only 8 or 16 MHz.
SSP_ERR_ASSERTION	A NULL is passed for configuration data when PLL is the clock_source.
SSP_ERR_INVALID_MODE	Attempt to start a clock in a restricted operating power control mode.

10.6.5.3 Function steps

- See if we need to switch to Middle or High Speed mode before starting the PLL.

10.6.6 R_CGC_ClockStop

```
ssp_err_t R_CGC_ClockStop ( cgc_clock_t  clock_source )
```

10.6.6.1 Brief description

Stop the specified clock if it is active and not configured as the system clock.

10.6.6.2 Detailed description

Table 1153:Return values

Name	Description
SSP_SUCCESS	Clock stopped successfully.
SSP_ERR_CLOCK_ACTIVE	Current System clock source specified for stopping. This is not allowed.
SSP_ERR_OSC_DET_ENABLED	MOCO cannot be stopped if Oscillation stop detection is enabled.
SSP_ERR_NOT_STABILIZED	Clock not stabilized after starting. A finite stabilization time after starting the clock has to elapse before it can be stopped.
SSP_ERR_INVALID_ARGUMENT	Invalid argument used.

10.6.7 R_CGC_SystemClockSet

```
ssp_err_t R_CGC_SystemClockSet ( cgc_clock_t  clock_source ,
 cgc_system_clock_cfg_t  const *const p_clock_cfg )
```

10.6.7.1 Brief description

Set the specified clock as the system clock and configure the internal dividers for ICLK, PCLKA, PCLKB, PCLKC, PCLKD and FCLK.

10.6.7.2 Detailed description

THIS FUNCTION DOES NOT CHECK TO SEE IF THE OPERATING MODE SUPPORTS THE SPECIFIED CLOCK SOURCE AND DIVIDER VALUES. SETTING A CLOCK SOURCE AND DVIDER OUTSIDE THE RANGE SUPPORTED BY THE CURRENT OPERATING MODE WILL RESULT IN UNDEFINED OPERATION.

IF THE LOCO MOCO OR SUBCLOCK ARE CHOSEN AS THE SYSTEM CLOCK, THIS FUNCTION WILL SET THOSE AS THE SYSTEM CLOCK WITHOUT CHECKING FOR STABILIZATION. IT IS UP TO THE USER TO ENSURE THAT LOCO, MOCO OR SUBCLOCK ARE STABLE BEFORE USING THEM AS THE SYSTEM CLOCK.

Additionally this function sets the RAM and ROM wait states for the MCU. For the S7 MCU the ROMWT register controls ROM wait states. For the S3 MCU the MEMWAIT register controls ROM wait states.

Table 1154:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

Table 1154:Return values (Continued)

Name	Description
SSP_ERR_CLOCK_INACTIVE	The specified clock source is inactive.
SSP_ERR_ASSERTION	The p_clock_cfg parameter is NULL.
SSP_ERR_STABILIZED	The clock source has not stabilized
SSP_ERR_INVALID_ARGUMENT	Invalid argument used. ICLK is not set as the fastest clock.
SSP_ERR_INVALID_MODE	Peripheral divisions are not valid in sub-osc mode
SSP_ERR_INVALID_MODE	Oscillator stop detect not allowed in sub-osc mode

10.6.7.3 Function steps

- In order to correctly set the ROM and RAM wait state registers we need to know the current (S3A7 only) and requested iclk frequencies.

10.6.8 R_CGC_SystemClockGet

```
ssp_err_t R_CGC_SystemClockGet ( cgc_clock_t  * clock_source ,
 cgc_system_clock_cfg_t  * p_set_clock_cfg )
```

10.6.8.1 Brief description

Return the current system clock source and configuration.

10.6.8.2 Detailed description

Table 1155:Return values

Name	Description
SSP_SUCCESS	Parameters returned successfully.
SSP_ERR_ASSERTION	A NULL is passed for configuration data.
SSP_ERR_ASSERTION	A NULL is passed for clock source.

10.6.9 R_CGC_SystemClockFreqGet

```
ssp_err_t R_CGC_SystemClockFreqGet ( cgc_system_clocks_t  clock ,  uint32_t
* p_freq_hz )
```

10.6.9.1 Brief description

Return the requested internal clock frequency in Hz.

10.6.9.2 Detailed description

Table 1156:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.
SSP_ERR_INVALID_ARGUMENT	Invalid clock specified.
SSP_ERR_ASSERTION	A NULL is passed for frequency data.

10.6.10 R_CGC_ClockCheck

```
ssp_err_t R_CGC_ClockCheck ( cgc_clock_t  clock_source )
```

10.6.10.1 Brief description

Check the specified clock for stability.

10.6.10.2 Detailed description

Table 1157:Return values

Name	Description
SSP_ERR_STABILIZED	Clock stabilized.
SSP_ERR_NOT_STABILIZED	Clock not stabilized.
SSP_ERR_CLOCK_ACTIVE	Clock active but not able to check for stability.
SSP_ERR_CLOCK_INACTIVE	Clock not turned on.
SSP_ERR_INVALID_ARGUMENT	Illegal parameter passed.

10.6.11 R_CGC_OscStopDetect

```
ssp_err_t R_CGC_OscStopDetect ( cgc_callback_args_t void(*)(
*p_args) p_callback ,   bool enable )
```

10.6.11.1 Brief description

Enable or disable the oscillation stop detection for the main clock. The MCU will automatically switch the system clock to MOCO when a stop is detected if Main Clock is the system clock. If the system clock is the PLL, then the clock source will not be changed and the PLL free running frequency will be the system clock frequency.

10.6.11.2 Detailed description

Table 1158:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.
SSP_ERR_OSC_STOP_DETECTED	The Oscillation stop detect status flag is set. Under this condition it is not possible to disable the Oscillation stop detection function.
SSP_ERR_ASSERTION	Null pointer passed for callback function when the second argument is "true".
SSP_ERR_ASSERTION	Cannot enable oscillator stop detect in sub-osc speed mode
SSP_ERR_ASSERTION	Invalid peripheral clock divisions for oscillator stop detect

10.6.12 R_CGC_OscStopStatusClear

```
ssp_err_t R_CGC_OscStopStatusClear (  void  )
```

10.6.12.1 Brief description

Clear the Oscillation Stop Detection Status register.

10.6.12.2 Detailed description

This register is not cleared automatically if the stopped clock is restarted. This function blocks for about 3 ICLK cycles until the status register is cleared.

Table 1159:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.
SSP_ERR_OSC_STOP_CLOCK_ACTIVE	The Oscillation Detect Status flag cannot be cleared if the Main Osc or PLL is set as the system clock. Change the system clock before attempting to clear this bit.

10.6.13 R_CGC_BusClockOutCfg

```
ssp_err_t R_CGC_BusClockOutCfg ( cgc_bclockout_dividers_t  divider )
```

10.6.13.1 Brief description

Configure the secondary dividers for BCLKOUT. The primary divider is set using the bsp clock configuration and the R_CGC_SystemClockSet function.

10.6.13.2 Detailed description

Table 1160:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.14 R_CGC_BusClockOutEnable

```
ssp_err_t R_CGC_BusClockOutEnable (  void  )
```

10.6.14.1 Brief description

Enable the BCLKOUT output.

10.6.14.2 Detailed description

Table 1161:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.15 R_CGC_BusClockOutDisable

```
ssp_err_t R_CGC_BusClockOutDisable (  void  )
```

10.6.15.1 Brief description

Disable the BCLKOUT output.

10.6.15.2 Detailed description

Table 1162:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.16 R_CGC_ClockOutCfg

```
ssp_err_t R_CGC_ClockOutCfg ( cgc_clock_t  clock ,
 cgc_clockout_dividers_t  divider )
```

10.6.16.1 Brief description

Configure the dividers for CLKOUT.

10.6.16.2 Detailed description

Table 1163:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.17 R_CGC_ClockOutEnable

```
ssp_err_t R_CGC_ClockOutEnable (  void  )
```

10.6.17.1 Brief description

Enable the CLKOUT output.

10.6.17.2 Detailed description

Table 1164:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.18 R_CGC_ClockOutDisable

```
ssp_err_t R_CGC_ClockOutDisable (  void  )
```

10.6.18.1 Brief description

Disable the CLKOUT output.

10.6.18.2 Detailed description

Table 1165:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.19 R_CGC_LCDClockCfg

```
ssp_err_t R_CGC_LCDClockCfg ( cgc_clock_t  clock )
```

10.6.19.1 Brief description

Configure the source for the segment LCDCLK.

10.6.19.2 Detailed description

Table 1166:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.20 R_CGC_LCDClockEnable

```
ssp_err_t R_CGC_LCDClockEnable (  void  )
```

10.6.20.1 Brief description

Enable the segment LCDCLK output.

10.6.20.2 Detailed description

Table 1167:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.21 R_CGC_LCDClockDisable

```
ssp_err_t R_CGC_LCDClockDisable (  void  )
```

10.6.21.1 Brief description

Disable the segment LCDCLK output.

10.6.21.2 Detailed description

Table 1168:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.22 R_CGC_SDRAMClockOutEnable

```
ssp_err_t R_CGC_SDRAMClockOutEnable (  void  )
```

10.6.22.1 Brief description

Enable the SDCLK output.

10.6.22.2 Detailed description

Table 1169:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.23 R_CGC_SDRAMClockOutDisable

```
ssp_err_t R_CGC_SDRAMClockOutDisable (  void  )
```

10.6.23.1 Brief description

Disable the SDCLK output.

10.6.23.2 Detailed description

Table 1170:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.24 R_CGC_USBClockCfg

```
ssp_err_t R_CGC_USBClockCfg ( cgc_usb_clock_div_t  divider )
```

10.6.24.1 Brief description

Configure the dividers for UCLK.

10.6.24.2 Detailed description

Table 1171:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.
SSP_ERR_INVALID_ARGUMENT	Invalid divider specified.

10.6.25 R_CGC_SystickUpdate

```
ssp_err_t R_CGC_SystickUpdate (  uint32_t period_count ,
 cgc_systick_period_units_t  units )
```

10.6.25.1 Brief description

Re-Configure the systick based on the provided period and current system clock frequency.

10.6.25.2 Detailed description

Table 1172:Parameters

Name	Direction	Description
period_count	in	The duration for the systick period.
units	in	The units for the provided period.

Table 1173:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.
SSP_ERR_INVALID_ARGUMENT	Invalid period specified.
SSP_ERR_ABORTED	Attempt to update systick timer failed.

10.6.26 R_CGC_VersionGet

```
ssp_err_t R_CGC_VersionGet ( ssp_version_t  *const p_version )
```

10.6.26.1 Brief description

Return the driver version.

10.6.26.2 Detailed description

Table 1174:Return values

Name	Description
SSP_SUCCESS	Operation performed successfully.

10.6.27 r_cgc_stabilization_wait

```
ssp_err_t r_cgc_stabilization_wait ( cgc_clock_t  clock )
```

10.6.28 r_cgc_clock_start_stop

```
ssp_err_t r_cgc_clock_start_stop ( cgc_clock_change_t  clock_state ,
 cgc_clock_t  clock_to_change )
```

10.7 CRC

Driver for the CRC Calculator (CRC).

This module supports the CRC Calculator (CRC). It implements the following interface:

- CRC Interface

10.7.1 Functions

- calculate_polynomial
- R_CRC_Open
- R_CRC_Close
- R_CRC_CalculatedValueGet
- R_CRC_SnoopEnable
- R_CRC_SnoopDisable
- R_CRC_SnoopCfg
- R_CRC_Calculate
- R_CRC_VersionGet

10.7.2 Variables

- s_crc_version
- g_crc_on_crc

10.7.3 Defines

- #define CRC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CRC_CODE_VERSION_MINOR

  Initial value:`(4U)`
- #define CRC_SNOOP_MAX_CHANNEL

  Initial value:`(10UL)`

  Maximum number of channels supported by CRC
- #define CRC_OPEN

  Initial value:`(0x00435243ULL)`

  "CRC" in ASCII, used to determine if channel is open.
- #define CRC_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &s_crc_version)`

  Macro for error logger.

10.7.4 calculate_polynomial

```
ssp_err_t calculate_polynomial ( crc_ctrl_t  *const p_api_ctrl ,   void
* inputBuffer ,   uint32_t length ,   uint32_t crc_seed ,   uint32_t
* calculatedValue )
```

10.7.4.1 Brief description

Perform a CRC calculation on a block of 8-bit data.

10.7.4.2 Detailed description

(end defgroup CRC) Implements calculate

This function performs a CRC calculation on an array of 8-bit/32-bit(for 32-bit polynomial) values and returns an 8-bit32-bit(for 32-bit polynomial) calculated value.

Table 1175:Return values

Name	Description
SSP_SUCCESS	Calculation successful.

10.7.5 R_CRC_Open

```
ssp_err_t R_CRC_Open ( crc_ctrl_t  *const p_api_ctrl ,  crc_cfg_t  const
*const p_cfg )
```

10.7.5.1 Brief description

Open the CRC driver module.

10.7.5.2 Detailed description

Implements open

Open the CRC driver module and initialize the driver control block according to the passed-in configuration structure.

Table 1176:Return values

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	p_ctrl or p_cfg is NULL.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.7.5.3 Function steps

- Mark driver as initialized by setting the open value to the ASCII equivalent of "CRC"

10.7.6 R_CRC_Close

```
ssp_err_t R_CRC_Close ( crc_ctrl_t  *const p_api_ctrl )
```

10.7.6.1 Brief description

Close the CRC module driver.

10.7.6.2 Detailed description

Implements close

Table 1177:Return values

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The driver is not opened.

10.7.7 R_CRC_CalculatedValueGet

```
ssp_err_t R_CRC_CalculatedValueGet ( crc_ctrl_t  *const p_api_ctrl ,   uint32_t
* calculatedValue )
```

10.7.7.1 Brief description

Return the current calculated value.

10.7.7.2 Detailed description

Implements crcResultGet

CRC calculation operates on a running value. This function returns the current calculated value.

Table 1178:Return values

Name	Description
SSP_SUCCESS	Return of calculated value successful.
SSP_ERR_ASSERTION	Either p_ctrl or calculatedValue is NULL.
SSP_ERR_NOT_OPEN	The driver is not opened.

10.7.8 R_CRC_SnoopEnable

```
ssp_err_t R_CRC_SnoopEnable ( crc_ctrl_t  *const p_api_ctrl ,
  uint32_t crc_seed )
```

10.7.8.1 Brief description

Enable snooping.

10.7.8.2 Detailed description

Implements snoopEnable

Table 1179:Return values

Name	Description
SSP_SUCCESS	Snoop enabled.
SSP_ERR_ASSERTION	Either p_ctrl or crc_seed is NULL.
SSP_ERR_NOT_OPEN	The driver is not opened.

10.7.9 R_CRC_SnoopDisable

```
ssp_err_t R_CRC_SnoopDisable ( crc_ctrl_t  *const p_api_ctrl )
```

10.7.9.1 Brief description

Disable snooping.

10.7.9.2 Detailed description

Implements snoopDisable

Table 1180:Return values

Name	Description
SSP_SUCCESS	Snoop disabled.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The driver is not opened.

10.7.10 R_CRC_SnoopCfg

```
ssp_err_t R_CRC_SnoopCfg ( crc_ctrl_t  *const p_api_ctrl ,  crc_snoop_cfg_t
*const p_snoop_cfg )
```

10.7.10.1 Brief description

Configure the snoop channel and direction.

10.7.10.2 Detailed description

Implements snoopCfg

The CRC calculator can operate on reads and writes over any of the first ten SCI channels. For example, if set to channel 0, transmit, every byte written out SCI channel 0 is also sent to the CRC calculator as if the value was explicitly written directly to the CRC calculator.

Table 1181:Return values

Name	Description
SSP_SUCCESS	Snoop configured successfully.
SSP_ERR_ASSERTION	- This is due to below conditions • Either p_ctrl or p_snoop_cfg is NULL • snoop_channel is greater than or equal to CRC_SNOOP_MAX_CHANNEL.
SSP_ERR_NOT_OPEN	The driver is not opened.

10.7.11 R_CRC_Calculate

```
ssp_err_t R_CRC_Calculate ( crc_ctrl_t  *const p_api_ctrl ,   void
* inputBuffer ,   uint32_t length ,   uint32_t crc_seed ,   uint32_t
* calculatedValue )
```

10.7.11.1 Brief description

Perform a CRC calculation on a block of 8-bit/32-bit(for 32-bit polynomial) data.

10.7.11.2 Detailed description

Implements calculate

This function performs a CRC calculation on an array of 8-bit/32-bit(for 32-bit polynomial) values and returns an 8-bit/32-bit(for 32-bit polynomial) calculated value

Table 1182:Return values

Name	Description
SSP_SUCCESS	Calculation successful.
SSP_ERR_ASSERTION	Either p_ctrl, inputBuffer, or calculatedValue is NULL.
SSP_ERR_INVALID_ARGUMENT	length value is NULL.
SSP_ERR_NOT_OPEN	The driver is not opened.
SSP_ERR_IN_USE	CRC peripheral is currently in use by another instance of the driver.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.7.11.3 Function steps

- Lock the peripheral during calculation

10.7.12 R_CRC_VersionGet

```
ssp_err_t R_CRC_VersionGet ( ssp_version_t  *const p_version )
```

10.7.12.1 Brief description

Get the driver version based on compile time macros.

10.7.12.2 Detailed description

Implements versionGet

Table 1183:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_version is NULL.

10.7.13 g_crc_on_crc

crc_api_t::g_crc_on_crc

10.7.13.1 Detailed description

Name of module used by error logger macro Filled in Interface API structure for this Instance.

10.7.13.2 Initialized as

```
g_crc_on_crc=
{
    .open           = R_CRC_Open,
    .close          = R_CRC_Close,
    .crcResultGet   = R_CRC_CalculatedValueGet,
    .snoopEnable    = R_CRC_SnoopEnable,
    .snoopDisable   = R_CRC_SnoopDisable,
    .snoopCfg       = R_CRC_SnoopCfg,
    .calculate      = R_CRC_Calculate,
    .versionGet     = R_CRC_VersionGet
}
```

10.7.14 Extensions

10.7.14.1 crc_instance_ctrl_t

crc_instance_ctrl_t

Detailed description

Driver instance control structure.

Variables

- R_CRC_Type * p_reg

  Pointer to register base address.

- uint32_t open

  Whether or not channel is open.

- crc_polynomial_t polynomial

  CRC Generating Polynomial Switching (GPS).

- crc_bit_order_t bit_order

  CRC Calculation Switching (LMS).

10.8 CTSU

Driver for the Capacitive Touch Sensing Unit (CTSU).

Driver for operating the CTSU. Provides functions to start up the CTSU and maintain necessary parameters to determine if a channel is being touched or not.

It implements the following interface:

- CTSU Interface defined in r_ctsu_api.h

10.8.1 Functions

- R_CTSU_VersionGet
- R_CTSU_Open
- R_CTSU_Start_Scan
- R_CTSU_Update_Parameters
- R_CTSU_Process
- R_CTSU_Read_Results
- R_CTSU_Control
- R_CTSU_Close

10.8.2 Defines

- #define CTSU_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define CTSU_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define CTSU_MAX_CHANNELS

  Initial value:`(36)`

10.8.3 R_CTSU_VersionGet

```
ssp_err_t R_CTSU_VersionGet ( ssp_version_t  *const p_version )
```

10.8.4 R_CTSU_Open

```
ssp_err_t R_CTSU_Open ( ctsu_ctrl_t  * p_api_ctrl ,  ctsu_cfg_t  * p_cfg )
```

10.8.4.1 Brief description

Initializes the CTSU, initializes software driver parameters for touch determination, with initial auto-tuning(optional) and baseline initialization(optional).

10.8.4.2 Detailed description

Table 1184:Return values

Name	Description
SSP_SUCCESS	No errors. CTSU succesfully initialized.
SSP_ERR_HW_LOCKED	CTSU driver already in use
SSP_ERR_ASSERTION	Invalid mode of operation (SELF or MUTUAL capacitance), invalid soft option, or invalid close option
SSP_ERR_INVALID_POINTER	NULL pointer passed as an argument for a required parameter -OR- Memory allocation failed.
SSP_ERR_INVALID_ARGUMENT	Illegal parameter was found.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Initial auto tuning failed because either the filter depth is too large, or the tuning is not valid for either the board or the current conditions.

NOTE: Data processing functions should be used by advanced users only. Beginners should provide a dummy argument (except NULL). Members inside should be kept NULL.

NOTE: If CFG_AUTO_TUNE is enabled, then the user must ensure that global interrupt operation is enabled before Open gets invoked.

10.8.4.3 Function steps

- Perform quick checks
- Set the valid channel mask based on variant data.
- Cannot reconfigure CTSU. You should unload active configuration by calling Close before proceeding.
- Power ON the CTSU
- Blind copy the configuration settings to initialize CTSU
- Discover excluded channels
- The number of active channels must be greater than the number of excluded channels
- Variables to be used in the loop below
- Initialize buffers used

- Populate primary buffers
- Select operation functions
- Enable interrupt operation
- Perform optional actions
- Indicate that R_Touch is ready
- Save which CTSU configuration has been opened
- Save the CTSU options used during open
- Save the CTSU options to use when updating parameters
- Save the notification function argument to be passed

10.8.5 R_CTSU_Start_Scan

```
ssp_err_t R_CTSU_Start_Scan ( ctsu_ctrl_t  * p_api_ctrl )
```

10.8.5.1 Brief description

Starts off a CTSU hardware scan. The rate at which this function is called is essentially the scan rate.

10.8.5.2 Detailed description

Table 1185:Return values

Name	Description
SSP_SUCCESS	No errors. CTSU scan successfully initialized.
SSP_ERR_IN_USE	A scan is ongoing

10.8.5.3 Function steps

- CTSU h/w is ready to start a new scan.
- Start a new scan.
- Hardware is busy

10.8.6 R_CTSU_Update_Parameters

```
ssp_err_t R_CTSU_Update_Parameters ( ctsu_ctrl_t  * p_api_ctrl )
```

10.8.6.1 Brief description

Function that must be called after each scan is complete to process the results of the scanned data.

10.8.6.2 Detailed description

It performs the following tasks:

```
             0. Checks for any errors in CTSU operation.
             1. Runs a filter on the raw values output by the CTSU.
             2. Determines if a channel is being touched or not and updates the
binary and difference between
                  sensor_baseline and the current sensor value.
            3. When the update is complete, the user specified callback is called
from this function with
                the CTSU_EVENT_PARAMETERS_UPDATED event.
            4. (Optional) Performs auto-tuning if (a single/all) channels are not
being touched.
             5. (Optional) Performs drift compensation if (a single/all channels)
are not being touched.
                    Drift compensation := move sensor baseline after N readings.
```

Table 1186:Return values

Name	Description
SSP_SUCCESS	No errors.
SSP_ERR_INVALID_POINTER	NULL pointer passed as an argument for a required parameter. -OR- Memory allocation failed.
SSP_ERR_INVALID_ARGUMENT	Illegal parameter was found.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Auto tuning failed because CTSU encountered a ICOMP error or Check the configuration parameters for each channel by locating the point of failure.

10.8.6.3 Function steps

- Check for CTSU error conditions
- Update each channel measured
- Initialize variables for loop below
- Channel is to be excluded. Bypass calculations.
- Check which channels are being touched.
- Initialize variables for loop below

- Channel is to be excluded. Bypass calculations.
- Do the callback to provide a notification
- Perform drift compensation

10.8.7 R_CTSU_Process

```
ssp_err_t R_CTSU_Process ( ctsu_ctrl_t  * p_api_ctrl ,
 ctsu_process_option_t  opts )
```

10.8.7.1 Brief description

Function that the user must call from the main loop. This function is deprecated by the R_CTSU_Update_Parameters function and is not exposed to the user in the interface API.

10.8.7.2 Detailed description

It performs the following tasks:

```
            0. Checks for any errors in CTSU operation.
            1. Runs a filter on the raw values output by the CTSU.
            2. Determines if a channel is being touched or not and updates the
binary and difference between
                sensor_baseline and the current sensor value.
          3. (Optional) Performs auto-tuning if (a single/all) channels are not
being touched.
           4. (Optional) Performs drift compensation if (a single/all channels)
are not being touched.
                   Drift compensation := move sensor baseline after N readings.
            5. (Optional) Starts off a new scan.
```

Table 1187:Return values

Name	Description
SSP_SUCCESS	No errors.
SSP_ERR_INVALID_POINTER	NULL pointer passed as an argument for a required parameter. -OR- Memory allocation failed.
SSP_ERR_INVALID_ARGUMENT	Illegal parameter was found.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Auto tuning failed because CTSU encountered a ICOMP error or Check the configuration parameters for each channel by locating the point of failure.

10.8.7.3 Function steps

- Check for CTSU error conditions
- Update each channel measured
- Initialize variables for loop below
- Initialize variables for loop below
- Check which channels are being touched.
- Update buttons
- Update Slider
- Perform drift compensation
- CTSU is ready to run
- Start a new scan.
- Save a copy of the state machine

10.8.8 R_CTSU_Read_Results

```
ssp_err_t R_CTSU_Read_Results ( ctsu_ctrl_t  * p_api_ctrl ,   void * p_dest ,
 ctsu_read_t  opts ,  ctsu_channel_pair_t const * channels ,   const
uint16_t count )
```

10.8.8.1 Brief description

Function to allow the user to read the results after executing a scan.

10.8.8.2 Detailed description

Table 1188:Return values

Name	Description
SSP_SUCCESS	No errors.
CTSU_ERR_INVALID_CMD	Command option provided is invalid.
SSP_ERR_INVALID_POINTER	NULL pointer passed as an argument for a required parameter. -OR- Memory allocation failed.
SSP_ERR_INVALID_ARGUMENT	Illegal parameter was found.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Auto tuning failed because CTSU encountered a ICOMP error. Check the configuration parameters for each channel by locating the point of failure.

10.8.8.3 Function steps

- Get results for the requested options
- Assemble arr[3] and arr[2] as the high and low byte
- Assemble arr[3] and arr[2] as the high and low byte
- Assemble arr[3] and arr[2] as the high and low byte
- Assemble arr[3] and arr[2] as the high and low byte
- Assemble arr[3] and arr[2] as the high and low byte
- Assemble arr[3] and arr[2] as the high and low byte

10.8.9 R_CTSU_Control

```
ssp_err_t R_CTSU_Control (  ctsu_cmd_t cmd ,   void * p_data )
```

10.8.9.1 Brief description

Allows the user to query manipulate R_Touch layer, and CTSU operation. This function is not exposed for usage in the interface API.

10.8.9.2 Detailed description

Table 1189:Return values

Name	Description
SSP_SUCCESS	No errors.
CTSU_ERR_INVALID_CMD	Command option provided is invalid.
SSP_ERR_IN_USE	CTSU Scan is ongoing.

10.8.9.3 Function steps

- Execute the control request

10.8.10 R_CTSU_Close

```
ssp_err_t R_CTSU_Close ( ctsu_ctrl_t  * p_api_ctrl ,
 ctsu_close_option_t  opts )
```

10.8.10.1 Brief description

Function used to close the CTSU driver, stop clocking of the CTSU peripheral, reset all internal structures and close the DTC Transfer instances. The function can also optionally save the configuration for a rapid wake-up on re-opening.

10.8.10.2 Detailed description

Table 1190:Return values

Name	Description
SSP_SUCCESS	No errors.

10.8.11 Extensions

10.8.11.1 ctsu_instance_ctrl_t

ctsu_instance_ctrl_t

Detailed description

Instance control structure

Variables

- transfer_api_t const * p_api_transfer

  Pointer to lower level Transfer driver function pointers.
- transfer_ctrl_t * p_lowerl_lvl_transfer_read_ctrl

  Pointer to the Transfer Read control.
- transfer_ctrl_t * p_lowerl_lvl_transfer_write_ctrl

  Pointer to the Transfer Write control.
- bool ctsu_opened

  Store initialization state.
- uint8_t ctsu_unit

  CTSU Unit in use.
- ctsu_hw_cfg_t * p_ctsu_hw_cfg

  Pointer to a CTSU configuration.
- ctsu_process_option_t ctsu_open_option

  Software options to use when performing Open and Process.
- ctsu_process_option_t ctsu_update_option

  Software options to use when performing parameter Update process.

- ctsu_close_option_t ctsu_close_option

  Software options to use when closing touch operation.
- ctsu_action_t ctsu_process_state

  Variable to observe CTSU processing state machine operation.
- void(* p_callback)(*p_args)

  Callback to the function to use when updating dependent parameters is complete.
- void * p_context

  Pointer to data that should be passed to update_complete notification.
- R_CTSU_Type * p_reg

  Pointer to base register address.

10.9 DAC

Driver for the 12-Bit D/C Converter (DAC12).

10.9.1 Summary

This module implements the following interface: DAC Interface.
Name of module used by error logger macro

10.9.2 Functions

- R_DAC_Open
- R_DAC_Close
- R_DAC_Write
- R_DAC_Start
- R_DAC_Stop
- R_DAC_VersionGet

10.9.3 Defines

- #define DAC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DAC_CODE_VERSION_MINOR

  Initial value:`(3U)`

10.9.4 R_DAC_Open

```
ssp_err_t R_DAC_Open ( dac_ctrl_t  * p_api_ctrl ,  dac_cfg_t  const
*const p_cfg )
```

10.9.4.1 Brief description

Perform required initialization described in hardware manual. Implements open. Configures a single DAC channel, starts the channel, and provides a handle for use with the DAC API Write and Close functions. Must be called once prior to calling any other DAC API functions. After a channel is opened, Open should not be called again for the same channel without calling Close first.

10.9.4.2 Detailed description

Table 1191:Return values

Name	Description
SSP_SUCCESS	The channel was successfully opened.
SSP_ERR_ASSERTION	One or both of the following parameters may be NULL: p_api_ctrl or p_cfg Channel ID requested in p_cfg may not available on the device selected in r_bsp_config.h data_format value in p_cfg is out of range. ad_da_synchronized value in p_cfg is out of range.
SSP_ERR_IN_USE	DAC resource is locked.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.9.4.3 Function steps

- Validate the input parameter.
- Make sure the peripheral exists.
- Lock the DAC Hardware Resource.
- Power on the DAC device.
- Stop the channel.
- Configure data format: left or right justified.
- Configure D/A-A/D Synchronous Start Control Register(DAADSCR).
- Set output amplifier configuration for the channel.

- Set the reference voltage.
- Initialize the channel state information.
- All done. Return.

10.9.5 R_DAC_Close

```
ssp_err_t R_DAC_Close ( dac_ctrl_t  * p_api_ctrl )
```

10.9.5.1 Brief description

Stop the D/A conversion, stop output, and close the DAC channel.

10.9.5.2 Detailed description

Table 1192:Return values

Name	Description
SSP_SUCCESS	The channel is successfully closed.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.9.5.3 Function steps

- Validate that the channel is opened.
- Stop the channel
- Update the channel state information.
- Unlock the DAC Hardware Resource
- Power down the DAC device.
- Unlock the DAC Hardware Resource

10.9.6 R_DAC_Write

```
ssp_err_t R_DAC_Write ( dac_ctrl_t  * p_api_ctrl ,  dac_size_t  value )
```

10.9.6.1 Brief description

Write data to the D/A converter and enable the output if it has not been enabled.

10.9.6.2 Detailed description

Table 1193:Return values

Name	Description
SSP_SUCCESS	Data is successfully written to the D/A Converter.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

NOTE: Write function automatically starts the D/A conversion after data is successfully written to the channel.

10.9.6.3 Function steps

- Validate that the channel is opened.
- Write the value to D/A converter.
- Start the converter if it has been idle.
- Start the channel

10.9.7 R_DAC_Start

```
ssp_err_t R_DAC_Start ( dac_ctrl_t  * p_api_ctrl )
```

10.9.7.1 Brief description

Start the D/A conversion output if it has not been started.

10.9.7.2 Detailed description

Table 1194:Return values

Name	Description
SSP_SUCCESS	The channel is started successfully.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.9.7.3 Function steps

- Validate that the channel is opened.
- Enable the output.
- Update the internal state.

10.9.8 R_DAC_Stop

```
ssp_err_t R_DAC_Stop ( dac_ctrl_t  * p_api_ctrl )
```

10.9.8.1 Brief description

Stop the D/A conversion and disable the output signal.

10.9.8.2 Detailed description

Table 1195:Return values

Name	Description
SSP_SUCCESS	The control is successfully stopped.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.9.8.3 Function steps

- Validate that the channel is opened.
- Disable the output.
- Mark the internal state.

10.9.9 R_DAC_VersionGet

```
ssp_err_t R_DAC_VersionGet ( ssp_version_t  * p_version )
```

10.9.9.1 Brief description

Get version and store it in provided pointer p_version.

10.9.9.2 Detailed description

Table 1196:Return values

Name	Description
SSP_SUCCESS	Successfully retrieved version information.
SSP_ERR_ASSERTION	p_version is NULL.

10.9.10 Extensions

10.9.10.1 dac_instance_ctrl_t

dac_instance_ctrl_t

Detailed description

DAC instance control block. DO NOT INITIALIZE.

Variables

- void * p_reg

  Pointer to DAC base register.
- uint8_t channel

  ID associated with this DAC channel.
- uint8_t channel_started

  DAC operation on channel started.
- uint8_t channel_opened

  DAC channel open.

10.10 DAC 8

Driver for the 8-Bit D/C Converter (DAC 8).

10.10.1 Summary

This module implements the following interface: DAC Interface.

Name of module used by error logger macro

10.10.2 Functions

- R_DAC8_Open
- R_DAC8_Close
- R_DAC8_Write
- R_DAC8_Start
- R_DAC8_Stop
- R_DAC8_VersionGet

10.10.3 Defines

- #define DAC8_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DAC8_CODE_VERSION_MINOR

  Initial value:`(1U)`

10.10.4 R_DAC8_Open

```
ssp_err_t R_DAC8_Open ( dac_ctrl_t  * p_api_ctrl ,  dac_cfg_t  const
*const p_cfg )
```

10.10.4.1 Brief description

Perform required initialization described in hardware manual. Implements open. Configures a single DAC channel, starts the channel, and provides a handle for use with the DAC API Write and Close functions. Must be called once prior to calling any other DAC API functions. After a channel is opened, Open should not be called again for the same channel without calling Close first.

10.10.4.2 Detailed description

Table 1197:Return values

Name	Description
SSP_SUCCESS	The channel was successfully opened.
SSP_ERR_ASSERTION	One or both of the following parameters may be NULL: p_api_ctrl or p_cfg
SSP_ERR_IN_USE	DAC8 resource is locked.

Table 1197:Return values (Continued)

Name	Description
SSP_ERR_IP_CHANNEL_NOT_PRESENT	An invalid channel was requested.
SSP_ERR_UNSUPPORTED	Output amplifier is not supported. Real time mode is not supported. Charge pump is not supported. A/D - D/A synchronization is not supported.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.10.4.3 Function steps

- Validate the input parameter.
- Make sure the peripheral exists.
- Lock the DAC Hardware Resource.
- Power on the DAC device.
- Stop the channel.
- Set defaults.
- Configure the charge pump.
- Configure the DAC mode.
- Configure DA/AD mode.
- Initialize the channel state information.
- All done. Return.

10.10.5 R_DAC8_Close

```
ssp_err_t R_DAC8_Close ( dac_ctrl_t  * p_api_ctrl )
```

10.10.5.1 Brief description

Stop the D/A conversion, stop output, and close the DAC channel.

10.10.5.2 Detailed description

Table 1198:Return values

Name	Description
SSP_SUCCESS	The channel is successfully closed.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.10.5.3 Function steps

- Stop the channel
- Update the channel state information.
- Unlock the DAC Hardware Resource
- Unlock the DAC Hardware Resource

10.10.6 R_DAC8_Write

```
ssp_err_t R_DAC8_Write ( dac_ctrl_t  * p_api_ctrl ,  dac_size_t  value )
```

10.10.6.1 Brief description

Write data to the D/A converter and enable the output if it has not been enabled.

10.10.6.2 Detailed description

Table 1199:Return values

Name	Description
SSP_SUCCESS	Data is successfully written to the D/A Converter.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

NOTE: Write function automatically starts the D/A conversion after data is successfully written to the channel.

10.10.6.3 Function steps

- Handle data format.
- Convert to 8 bits.
- Write the value to D/A converter.
- Start the converter if it has been idle.
- Start the channel

10.10.7 R_DAC8_Start

```
ssp_err_t R_DAC8_Start ( dac_ctrl_t  * p_api_ctrl )
```

10.10.7.1 Brief description

Start the D/A conversion output.

10.10.7.2 Detailed description

Table 1200:Return values

Name	Description
SSP_SUCCESS	The channel is started successfully.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.10.7.3 Function steps

- Enable the output.
- Update the internal state.

10.10.8 R_DAC8_Stop

```
ssp_err_t R_DAC8_Stop ( dac_ctrl_t  * p_api_ctrl )
```

10.10.8.1 Brief description

Stop the D/A conversion and disable the output signal.

10.10.8.2 Detailed description

Table 1201:Return values

Name	Description
SSP_SUCCESS	The control is successfully stopped.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	Channel associated with p_ctrl has not been opened.

10.10.8.3 Function steps

- Disable the output.
- Mark the internal state.

10.10.9 R_DAC8_VersionGet

```
ssp_err_t R_DAC8_VersionGet ( ssp_version_t  * p_version )
```

10.10.9.1 Brief description

Get version and store it in provided pointer p_version.

10.10.9.2 Detailed description

Table 1202:Return values

Name	Description
SSP_SUCCESS	Successfully retrieved version information.
SSP_ERR_ASSERTION	p_version is NULL.

10.10.10 API Data

10.10.10.1 dac8_mode_t

`dac8_mode_t`

Enumerated values

Name	Description
DAC8_MODE_NORMAL	DAC Normal mode.
DAC8_MODE_REAL_TIME	DAC Real-time (event link) mode.

10.10.11 Extensions

10.10.11.1 dac8_instance_ctrl_t

dac8_instance_ctrl_t

Detailed description

DAC8 instance control block. DO NOT INITIALIZE.

Variables

- void * p_reg

  Pointer to DAC base register.

- uint8_t channel

  ID associated with this DAC channel.

- uint8_t channel_started

  DAC operation on channel started.

- uint32_t channel_opened

  DAC channel open.

- dac_data_format_t data_format

  DAC data format.

10.10.11.2 dac8_extended_cfg_t

dac8_extended_cfg_t

Detailed description

DAC8 extended configuration

Variables

- bool enable_charge_pump

  Enable DAC charge pump.

- dac8_mode_t dac_mode

  DAC mode.

10.11 DMAC

DMA Controller (DMAC).

10.11.1 Summary

Extends the Transfer Interface.

The Direct Memory Access (DMA) Controller allows data transfers to occur in place of or in addition to any interrupt. It also supports data transfers using software start.

NOTE: *The transfer length is limited to 1024 (10 bits) in TRANSFER_MODE_BLOCK and TRANSFER_MODE_REPEAT.****NOTE:*** *This driver supports only TRANSFER_IRQ_END from transfer_irq_t.*

10.11.2 Functions

- R_DMAC_Open
- R_DMAC_Reset
- R_DMAC_Start
- R_DMAC_Stop
- R_DMAC_Enable
- R_DMAC_Disable
- R_DMAC_InfoGet
- R_DMAC_Close
- R_DMAC_VersionGet
- R_DMAC_BlockReset

10.11.3 Defines

- #define DMAC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DMAC_CODE_VERSION_MINOR

  Initial value:`(4U)`
- #define DMAC_REPEAT_BLOCK_MAX_LENGTH

  Initial value:`(0x400)`

  Length limited to 1024 transfers for repeat and block mode

- #define DMAC_NORMAL_MAX_LENGTH

  Initial value:`(0xFFFF)`

  Length limited to 65535 transfers for normal mode

10.11.4 R_DMAC_Open

```
ssp_err_t R_DMAC_Open ( transfer_ctrl_t  *const p_api_ctrl ,  transfer_cfg_t
const *const p_cfg )
```

10.11.4.1 Brief description

Initialize transfer and enables transfer in ICU. Implements open.

10.11.4.2 Detailed description

Table 1203:Return values

Name	Description
SSP_SUCCESS	Successful open. Transfer is configured and will start when trigger occurs.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_ENABLED	Auto-enable was requested, but enable failed.
SSP_ERR_IRQ_BSP_DISABLED	The IRQ associated with the activation source is not enabled in the BSP.
SSP_ERR_IN_USE	The BSP hardware lock for the DMAC is not available.

See Common Error Codes for other possible return codes.This function calls

- productFeatureGet
- eventInfoGet

10.11.4.3 Function steps

- Acquire BSP hardware lock for channel used.
- Configure the DMAC according to the flowchart "Activating the DMAC" in chapter 16.3.7 of hardware manual NoSecurity_r01uh0488ej0040_sc32.pdf.
- Configure DMA transfer and sources
- **NOTE:** *Transfer escape interrupts not supported.*Update internal variables.

- Mark driver as open by initializing "DMAC" in its ASCII equivalent.
- If auto_enable is true, enable transfer and ELC events if software start is used.

10.11.5 R_DMAC_Reset

```
ssp_err_t R_DMAC_Reset ( transfer_ctrl_t  *const p_api_ctrl ,   void const
*volatile p_src ,   void *volatile p_dest ,   uint16_t const num_transfers )
```

10.11.5.1 Brief description

Reset transfer source, destination, and number of transfers.

10.11.5.2 Detailed description

Table 1204:Return values

Name	Description
SSP_SUCCESS	Transfer reset successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_ENABLED	DMAC is not enabled. A valid source and destination must be provided in either open() or reset().
SSP_ERR_IN_USE	Transfer is in progress. Wait for transfer to complete.

10.11.5.3 Function steps

- Enables transfers on this activation source.

10.11.6 R_DMAC_Start

```
ssp_err_t R_DMAC_Start ( transfer_ctrl_t  *const p_api_ctrl ,
 transfer_start_mode_t  mode )
```

10.11.6.1 Brief description

Start transfer. Implements start.

10.11.6.2 Detailed description

Table 1205:Return values

Name	Description
SSP_SUCCESS	Transfer started written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.
SSP_ERR_UNSUPPORTED	Handle was not configured for software activation.

10.11.6.3 Function steps

- Set autoclear bit and software start bit.

10.11.7 R_DMAC_Stop

```
ssp_err_t R_DMAC_Stop ( transfer_ctrl_t  *const p_api_ctrl )
```

10.11.7.1 Brief description

Stop transfer. Implements stop.

10.11.7.2 Detailed description

Table 1206:Return values

Name	Description
SSP_SUCCESS	Transfer stopped written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.

10.11.7.3 Function steps

- Reset auto clear bit and clear software transfer request.

10.11.8 R_DMAC_Enable

```
ssp_err_t R_DMAC_Enable ( transfer_ctrl_t  *const p_api_ctrl )
```

10.11.8.1 Brief description

Enable transfer. Implements enable.

10.11.8.2 Detailed description

Table 1207:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.

10.11.8.3 Function steps

- Enable transfer.

10.11.9 R_DMAC_Disable

```
ssp_err_t R_DMAC_Disable ( transfer_ctrl_t  *const p_api_ctrl )
```

10.11.9.1 Brief description

Disable transfer. Implements disable.

10.11.9.2 Detailed description

Table 1208:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.

Table 1208:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.

10.11.9.3 Function steps

- Disable transfer.

10.11.10 R_DMAC_InfoGet

```
ssp_err_t R_DMAC_InfoGet ( transfer_ctrl_t  *const p_api_ctrl ,
 transfer_properties_t  *const p_info )
```

10.11.10.1 Brief description

Set driver specific information in provided pointer. Implements infoGet.

10.11.10.2 Detailed description

Table 1209:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.
SSP_ERR_ASSERTION	An input parameter is invalid.

10.11.10.3 Function steps

- If a transfer is active, store it in p_in_progress.
- Store maximum transfer length.
- Store remaining transfer length.

10.11.11 R_DMAC_Close

```
ssp_err_t R_DMAC_Close ( transfer_ctrl_t  *const p_api_ctrl )
```

10.11.11.1 Brief description

Disable transfer and clean up internal data. Implements close.

10.11.11.2 Detailed description

Table 1210:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.
SSP_ERR_IN_USE	Transfer is in progress. Wait for transfer to complete.

10.11.11.3 Function steps

- Disable DMAC transfers, disable DMAC interrupts, and remove DMAC trigger from ICU register.
- Clear ID so control block can be reused.
- Release BSP hardware lock on this channel

10.11.12 R_DMAC_VersionGet

```
ssp_err_t R_DMAC_VersionGet ( ssp_version_t  *const p_version )
```

10.11.12.1 Brief description

Set driver version based on compile time macros.

10.11.12.2 Detailed description

Table 1211:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	An input parameter is invalid.

10.11.13 R_DMAC_BlockReset

```
ssp_err_t R_DMAC_BlockReset ( transfer_ctrl_t  *const p_api_ctrl ,   void const
*volatile p_src ,   void *volatile p_dest ,   uint16_t const length ,
 transfer_size_t  size ,   uint16_t const num_transfers )
```

10.11.13.1 Brief description

Reset transfer source, destination, length and number of transfers for block transfer.

10.11.13.2 Detailed description

Table 1212:Return values

Name	Description
SSP_SUCCESS	Transfer reset successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_IN_USE	Transfer is in progress. Wait for transfer to complete.
SSP_ERR_NOT_ENABLED	DMAC is not enabled. A valid source and destination must be provided in either open() or blockReset().

10.11.13.3 Function steps

- Enables transfers on this activation source.

10.11.14 Extensions

10.11.14.1 dmac_instance_ctrl_t

dmac_instance_ctrl_t

Detailed description

Control block used by driver. DO NOT INITIALIZE - this structure will be initialized in open.

Variables

- uint32_t id

  Driver ID.

- elc_event_t trigger

  Transfer activation event. Matches event returned by transfer_api_t::infoGet.

- IRQn_Type irq

  Transfer activation IRQ.
- uint8_t channel

  Channel number.
- void(* p_callback)(*cb_data)

  Callback for transfer end interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.
- void * p_reg

  Pointer to base register.

10.11.14.2 transfer_on_dmac_cfg_t

transfer_on_dmac_cfg_t

Detailed description

DMAC transfer configuration extension. This extension is required.

Variables

- uint8_t channel

  Channel number, does not apply to all HAL drivers.

10.12 DOC

Driver for the Data Operation Circuit (DOC).

10.12.1 Summary

This module implements the DOC Interface using the Data Operation Circuit (DOC).

10.12.2 Functions

- R_DOC_Open
- R_DOC_Close
- R_DOC_StatusGet
- R_DOC_StatusClear
- R_DOC_Write
- R_DOC_InputRegisterWrite

- R_DOC_VersionGet

10.12.3 Defines

- #define DOC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DOC_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.12.4 R_DOC_Open

```
ssp_err_t R_DOC_Open ( doc_ctrl_t  *const p_api_ctrl ,  doc_cfg_t  const
*const p_cfg )
```

10.12.4.1 Brief description

Configure the Data Operation Circuit (DOC) in comparison, addition or subtraction mode. Implements open.

10.12.4.2 Detailed description

If callback is not NULL in the configuration structure the DOC IRQ will be enabled.

Table 1213:Return values

Name	Description
SSP_SUCCESS	DOC successfully configured.
SSP_ERR_IN_USE	Module already open.
SSP_ERR_ASSERTION	One or more pointers point to NULL.
SSP_ERR_INVALID_ARGUMENT	ISR is not enabled. Enable the ISR in bsp_irq_cfg.h.
SSP_ERR_HW_LOCKED	DOC resource is locked.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: This function is reentrant.

10.12.4.3 Function steps

- Make sure the DOC exists on this part.
- Lock the DOC Hardware Resource
- Configure the DOC via DOCR register.
- If callback parameter is not NULL configure the IRQ
- Mark driver as open by initializing it to "DOCO" in its ASCII equivalent.

10.12.5 R_DOC_Close

```
ssp_err_t R_DOC_Close ( doc_ctrl_t  *const p_api_ctrl )
```

10.12.5.1 Brief description

Close the module driver. Enable module stop mode. Implements close.

10.12.5.2 Detailed description

To close the DOC it must have been opened via the open API. When opened a control structure of type doc_ctrl_t is passed to the open API. The same control structure must be passed to the close API.

Table 1214:Return values

Name	Description
SSP_SUCCESS	Module successfully closed.
SSP_ERR_NOT_OPEN	Driver not open.
SSP_ERR_ASSERTION	Pointer pointing to NULL.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- eventInfoGet

NOTE: This function is reentrant.

NOTE: This function will disable the DOC interrupt in the NVIC.

10.12.5.3 Function steps

- Disable the IRQ in the NVIC in case it has been left enabled.
- Clear DOPCF in DOCR
- Mark driver as closed.
- Unlock the DOC Hardware Resource

10.12.6 R_DOC_StatusGet

```
ssp_err_t R_DOC_StatusGet ( doc_ctrl_t  *const p_api_ctrl ,  doc_status_t
* p_status )
```

10.12.6.1 Brief description

Return the comparison/addition/subtraction status. Implements statusGet.

10.12.6.2 Detailed description

The status is read from the Data Operation Circuit Flag (DOPCF).

Table 1215:Return values

Name	Description
SSP_SUCCESS	Status successfully read.
SSP_ERR_NOT_OPEN	Driver not open.
SSP_ERR_ASSERTION	One or more pointers point to NULL.

NOTE: This function is reentrant.

10.12.7 R_DOC_StatusClear

```
ssp_err_t R_DOC_StatusClear ( doc_ctrl_t  *const p_api_ctrl )
```

10.12.7.1 Brief description

Clear the DOPCF status flag at the hardware layer. This flag indicates the result of a DOC operation. Implements statusClear.

10.12.7.2 Detailed description

Table 1216:Return values

Name	Description
SSP_SUCCESS	Interrupt successfully cleared.
SSP_ERR_NOT_OPEN	Driver not open.

Table 1216:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	Pointer pointing to NULL.

NOTE: This function is reentrant.

10.12.8 R_DOC_Write

```
ssp_err_t R_DOC_Write ( doc_ctrl_t  *const p_api_ctrl ,  doc_data_t
*const p_data )
```

10.12.8.1 Brief description

Write to the DODIR and DODSR registers. Implements write,.

10.12.8.2 Detailed description

In comparison mode the 16-bit reference data is written to the DODSR register and the data for the comparison written to the DODIR. In addition mode the initial data is written to the DODSR and the data to be added to the DODIR. In subtraction mode the initial data is written to the DODSR and the data to be subtracted to the DODIR.

When changing both the DODSR and DODIR the DODSR should be updated first.

Table 1217:Return values

Name	Description
SSP_SUCCESS	Values successfully written to the registers.
SSP_ERR_NOT_OPEN	Driver not open.
SSP_ERR_ASSERTION	One or more pointers point to NULL.

NOTE: This function is reentrant.

10.12.9 R_DOC_InputRegisterWrite

```
ssp_err_t R_DOC_InputRegisterWrite ( doc_ctrl_t  *const p_api_ctrl ,
 doc_size_t  data )
```

10.12.9.1 Brief description

Write to the DODIR register. Implements inputRegisterWrite,.

10.12.9.2 Detailed description

Writes to the DODIR register only.

Table 1218:Return values

Name	Description
SSP_SUCCESS	Value successfully written to the DODIR register.
SSP_ERR_NOT_OPEN	Driver not open.
SSP_ERR_ASSERTION	One or more pointers point to NULL.

NOTE: This function is reentrant.

10.12.10 R_DOC_VersionGet

```
ssp_err_t R_DOC_VersionGet ( ssp_version_t  *const p_version )
```

10.12.10.1 Brief description

Return DOC HAL driver version. Implements versionGet.

10.12.10.2 Detailed description

Table 1219:Return values

Name	Description
SSP_SUCCESS	Version information successfully read.
SSP_ERR_ASSERTION	Pointer pointing to NULL.

NOTE: This function is reentrant.

10.12.11 Extensions

10.12.11.1 doc_instance_ctrl_t

doc_instance_ctrl_t

Detailed description

DOC instance control block. Do not initialize. Initialization occurs when the open function is called.

Variables

- uint32_t open

  Used by driver to check if the control structure is valid.

- void(* p_callback)(*p_args)

  Callback provided when a DOC ISR occurs. NULL indicates no CPU interrupt.

- doc_event_t event

  The event DOC is configured for. Passed in ISR callback.

- void const * p_context

  Placeholder for user data. Passed to the user callback in doc_callback_args_t.

- void * p_reg

  Base register.

10.13 DTC

Driver for the Data Transfer Controller (DTC).

10.13.1 Summary

Extends Transfer Interface.

The Data Transfer Controller allows data transfers to occur in place of or in addition to any interrupt. It does not support data transfers using software start.

NOTE: *The transfer length is limited to 256 (8 bits) in TRANSFER_MODE_BLOCK and TRANSFER_MODE_REPEAT.*

10.13.2 Functions

- R_DTC_Open
- R_DTC_Reset
- R_DTC_Start
- R_DTC_Stop
- R_DTC_Enable
- R_DTC_Disable
- R_DTC_InfoGet
- R_DTC_Close
- R_DTC_VersionGet
- R_DTC_BlockReset

10.13.3 Defines

- #define DTC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define DTC_CODE_VERSION_MINOR

  Initial value:`(6U)`
- #define DTC_REPEAT_BLOCK_MAX_LENGTH

  Initial value:`(0x100)`

  Length limited to 256 transfers for repeat and block mode
- #define DTC_NORMAL_MAX_LENGTH

  Initial value:`(0x10000)`

  Length limited to 65536 transfers for normal mode

10.13.4 R_DTC_Open

```
ssp_err_t R_DTC_Open ( transfer_ctrl_t  *const p_api_ctrl ,  transfer_cfg_t
const *const p_cfg )
```

10.13.4.1 Brief description

Set transfer data in the vector table and enable transfer in ICU. Implements open.

10.13.4.2 Detailed description

Table 1220:Return values

Name	Description
SSP_SUCCESS	Successful open. Transfer is configured and will start when trigger occurs.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_IN_USE	The BSP hardware lock for the DTC is not available, or the index for this IRQ in the DTC vector table is already configured.
SSP_ERR_HW_LOCKED	DTC hardware resource is locked.

Table 1220:Return values (Continued)

Name	Description
SSP_ERR_IRQ_BSP_DISABLED	The IRQ associated with the activation source is not enabled in the BSP.
SSP_ERR_NOT_ENABLED	Auto-enable was requested, but enable failed due to an invalid input parameter.

10.13.4.3 Function steps

- Make sure the activation source is mapped in the ICU.
- Make sure the activation source is not already being used by the DTC.
- Configure the DTC transfer. See the hardware manual for details.
- Update internal variables.
- Mark driver as open by initializing it to "DTC" in its ASCII equivalent.
- If auto_enable is true, enable transfer and ELC events if software start is used.
- Enable read skip after all settings are complete.

10.13.5 R_DTC_Reset

```
ssp_err_t R_DTC_Reset ( transfer_ctrl_t  *const p_api_ctrl ,   void const
*volatile p_src ,   void *volatile p_dest ,   uint16_t const num_transfers )
```

10.13.5.1 Brief description

Reset transfer source, destination, and number of transfers. Implements reset.

10.13.5.2 Detailed description

Table 1221:Return values

Name	Description
SSP_SUCCESS	Transfer reset successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.

Table 1221:Return values (Continued)

Name	Description
SSP_ERR_NOT_ENABLED	Transfer length must not be 0 for repeat and block mode, or enable failed due to an invalid input parameter: • Transfer source must not be NULL. • Transfer destination must not be NULL.

10.13.5.3 Function steps

- Disable transfers on this activation source.
- Disable read skip prior to modifying settings. It will be enabled later.
- Reset transfer based on input parameters.
- Enables transfers on this activation source.
- Enable read skip after all settings are complete.

10.13.6 R_DTC_Start

```
ssp_err_t R_DTC_Start ( transfer_ctrl_t  *const p_api_ctrl ,
 transfer_start_mode_t  mode )
```

10.13.6.1 Brief description

Start transfer. Implements start.

10.13.6.2 Detailed description

Table 1222:Return values

Name	Description
SSP_SUCCESS	Transfer started successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DMAC_Open to initialize the control block.

Table 1222:Return values (Continued)

Name	Description
SSP_ERR_UNSUPPORTED	One of the following is invalid: • Handle was not configured for software activation. • Mode not set to TRANSFER_START_MODE_SINGLE. • DTC_SOFTWARE_START_ENABLE set to 0 (disabled) in ssp_cfg/driver/r_dtc_cfg.h.

10.13.7 R_DTC_Stop

```
ssp_err_t R_DTC_Stop ( transfer_ctrl_t  *const p_ctrl )
```

10.13.7.1 Brief description

Placeholder for unsupported stop function. Implements stop.

10.13.7.2 Detailed description

Table 1223:Return values

Name	Description
SSP_ERR_UNSUPPORTED	DTC software start is not supported.

10.13.8 R_DTC_Enable

```
ssp_err_t R_DTC_Enable ( transfer_ctrl_t  *const p_api_ctrl )
```

10.13.8.1 Brief description

Enable transfer and ELC events if they are used for software start. Implements enable.

10.13.8.2 Detailed description

Table 1224:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.
SSP_ERR_IRQ_BSP_DISABLED	The IRQ associated with the p_ctrl is not enabled in the BSP.

10.13.8.3 Function steps

- Enable transfer.

10.13.9 R_DTC_Disable

```
ssp_err_t R_DTC_Disable ( transfer_ctrl_t  *const p_api_ctrl )
```

10.13.9.1 Brief description

Disable transfer. Implements disable.

10.13.9.2 Detailed description

Table 1225:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.

10.13.9.3 Function steps

- Disable transfer.

10.13.10 R_DTC_InfoGet

```
ssp_err_t R_DTC_InfoGet ( transfer_ctrl_t  *const p_api_ctrl ,
 transfer_properties_t  *const p_info )
```

10.13.10.1 Brief description

Set driver specific information. Implements infoGet.

10.13.10.2 Detailed description

Table 1226:Return values

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.

10.13.10.3 Function steps

- If a transfer is active, store it in p_in_progress.
- Transfer information for the activation source is taken from DTC vector table.
- Mask out the high byte in case of repeat transfer.
- Store maximum transfer length.

10.13.11 R_DTC_Close

```
ssp_err_t R_DTC_Close ( transfer_ctrl_t  *const p_api_ctrl )
```

10.13.11.1 Brief description

Disables transfer in the ICU, then clears transfer data from the DTC vector table. Implements close.

10.13.11.2 Detailed description

Table 1227:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.
SSP_ERR_IRQ_BSP_DISABLED	The IRQ associated with the p_ctrl is not enabled in the BSP.

10.13.11.3 Function steps

- Clear DTC enable bit in ICU.
- Clear pointer in vector table.

10.13.12 R_DTC_VersionGet

```
ssp_err_t R_DTC_VersionGet ( ssp_version_t  *const p_version )
```

10.13.12.1 Brief description

Set driver version based on compile time macros. Implements versionGet.

10.13.12.2 Detailed description

Table 1228:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	An input parameter is invalid.

10.13.13 R_DTC_BlockReset

```
ssp_err_t R_DTC_BlockReset ( transfer_ctrl_t  *const p_api_ctrl ,   void const
*volatile p_src ,   void *volatile p_dest ,   uint16_t const length ,
 transfer_size_t  size ,   uint16_t const num_transfers )
```

10.13.13.1 Brief description

BlockReset transfer source, destination, length and number of transfers. Implements blockReset.

10.13.13.2 Detailed description

Table 1229:Return values

Name	Description
SSP_SUCCESS	Transfer reset successfully.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_NOT_OPEN	Handle is not initialized. Call R_DTC_Open to initialize the control block.
SSP_ERR_UNSUPPORTED	If mode is other than Block Transfer Mode.
SSP_ERR_NOT_ENABLED	Enable failed due to an invalid input parameter: • Transfer source must not be NULL. • Transfer destination must not be NULL.

10.13.13.3 Function steps

- Disable transfers on this activation source.
- Disable read skip prior to modifying settings. It will be enabled later.
- Reset transfer based on input parameters.
- Enables transfers on this activation source.
- Enable read skip after all settings are complete.

10.13.14 Extensions

10.13.14.1 dtc_instance_ctrl_t

dtc_instance_ctrl_t

Detailed description

Control block used by driver. DO NOT INITIALIZE - this structure will be initialized in open.

Variables

- uint32_t id

  Driver ID.

- elc_event_t trigger

  Transfer activation event. Matches event returned by transfer_api_t::infoGet.
- IRQn_Type irq

  Transfer activation IRQ, does not apply to all HAL drivers.
- void(* p_callback)(*cb_data)

  Callback for transfer end interrupt used for ELC software trigger.
- void const * p_context

  Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

10.13.14.2 dtc_reg_t

dtc_reg_t

Detailed description

DTC Registers. Same as transfer_info_t, but uses register names. Provided as service to typecast transfer_info_t.

Variables

- uint32_t __pad0__
- uint8_t MRB

  Mode Register B
- uint8_t __pad0__
- uint8_t DM

  Transfer Destination Address mode.
- uint8_t DTS

  DTC Transfer Mode Select.
- uint8_t DISEL

  DTC Interrupt Select.
- uint8_t CHNS

  DTC Chain Transfer Select.
- uint8_t CHNE

  DTC CHain Transfer Enable.
- struct{} MRB_b

  MRB bits */

  See source code for definition of this member.
- uint8_t MRA

  Mode Register A

- uint8_t SM

  Transfer Source Address mode.
- uint8_t SZ

  DTC Data Transfer Size.
- uint8_t MD

  DTC Transfer Mode Select.
- struct{} MRA_b

  MRA bits */

  See source code for definition of this member.
- struct{} struct{}

  Mode registers */

  See source code for definition of this member.
- void *volatile SAR

  Source address register.
- void *volatile DAR

  Destination address register
- uint16_t CRB

  Transfer count register B
- uint16_t CRA

  Transfer count register A
- uint8_t CRAL

  Transfer counter A lower register.
- uint8_t CRAH

  Transfer counter B upper register.
- struct{} CRA_b

  bits */

  See source code for definition of this member.
- struct{} struct{}

  Transfer count registers */

  See source code for definition of this member.

10.14 ELC

Driver for the Event Link Controller (ELC).

This module supports the Event Link Controller (ELC). It implements the following interface: Events and peripheral definitions

10.14.1 Functions

- R_ELC_Init
- R_ELC_SoftwareEventGenerate
- R_ELC_LinkSet
- R_ELC_LinkBreak
- R_ELC_Enable
- R_ELC_Disable
- R_ELC_VersionGet

10.14.2 Defines

- #define ELC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define ELC_CODE_VERSION_MINOR

  Initial value:`(3U)`

10.14.3 R_ELC_Init

```
ssp_err_t R_ELC_Init ( elc_cfg_t  const *const p_cfg )
```

10.14.3.1 Brief description

Initialize all the links in the Event Link Controller.

10.14.3.2 Detailed description

Implements init

The configuration structure passed in to this function includes links for every event source included in the ELC and sets them all at once. To set an individual link use R_ELC_LinkSet

Table 1230:Return values

Name	Description
SSP_SUCCESS	Initialization was successful
SSP_ERR_ASSERTION	p_config was NULL

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.14.3.3 Function steps

- Power on ELC
- Enable the operation of the Event Link Controller

10.14.4 R_ELC_SoftwareEventGenerate

```
ssp_err_t R_ELC_SoftwareEventGenerate ( elc_software_event_t  event_number )
```

10.14.4.1 Brief description

Generate a software event in the Event Link Controller.

10.14.4.2 Detailed description

Implements softwareEventGenerate

Table 1231:Return values

Name	Description
SSP_SUCCESS	Initialization was successful
SSP_ERR_ASSERTION	Invalid event number

10.14.5 R_ELC_LinkSet

```
ssp_err_t R_ELC_LinkSet ( elc_peripheral_t  peripheral ,  elc_event_t  signal )
```

10.14.5.1 Brief description

Create a single event link.

10.14.5.2 Detailed description

Implements linkSet

Table 1232:Return values

Name	Description
SSP_SUCCESS	Initialization was successful

10.14.6 R_ELC_LinkBreak

```
ssp_err_t R_ELC_LinkBreak ( elc_peripheral_t  peripheral )
```

10.14.6.1 Brief description

Break an event link.

10.14.6.2 Detailed description

Implements linkBreak

Table 1233:Return values

Name	Description
SSP_SUCCESS	Event link broken

10.14.7 R_ELC_Enable

```
ssp_err_t R_ELC_Enable (  void  )
```

10.14.7.1 Brief description

Enable the operation of the Event Link Controller.

10.14.7.2 Detailed description

Implements enable

Table 1234:Return values

Name	Description
SSP_SUCCESS	ELC enabled.

10.14.8 R_ELC_Disable

```
ssp_err_t R_ELC_Disable (  void  )
```

10.14.8.1 Brief description

Disable the operation of the Event Link Controller.

10.14.8.2 Detailed description

Implements disable

Table 1235:Return values

Name	Description
SSP_SUCCESS	ELC disabled.

10.14.9 R_ELC_VersionGet

```
ssp_err_t R_ELC_VersionGet ( ssp_version_t  *const p_version )
```

10.14.9.1 Brief description

Get the driver version based on compile time macros.

10.14.9.2 Detailed description

Implements versionGet

Table 1236:Return values

Name	Description
SSP_SUCCESS	Successful close.

Table 1236:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	p_version is NULL.

10.15 Low Power Flash

Driver for the Low power Flash Memory (S3A7 and S124).

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

This module supports the Flash interface for the Low-power FLASH peripheral.

10.15.1 Functions

- R_FLASH_LP_Open
- R_FLASH_LP_Write
- R_FLASH_LP_Read
- R_FLASH_LP_Erase
- R_FLASH_LP_BlankCheck
- R_FLASH_LP_StatusGet
- R_FLASH_LP_AccessWindowSet
- R_FLASH_LP_AccessWindowClear
- R_FLASH_LP_Reset
- R_FLASH_LP_StartUpAreaSelect
- R_FLASH_LP_UpdateFlashClockFreq
- R_FLASH_LP_InfoGet
- R_FLASH_LP_Close
- R_FLASH_LP_VersionGet

10.15.2 Defines

- #define FLASH_LP_CODE_VERSION_MAJOR
  Initial value:`(1U)`
- #define FLASH_LP_CODE_VERSION_MINOR
  Initial value:`(6U)`

- #define PLACE_IN_RAM_SECTION

  Initial value:

10.15.3 R_FLASH_LP_Open

```
ssp_err_t R_FLASH_LP_Open ( flash_ctrl_t  *const p_api_ctrl ,  flash_cfg_t  const
*const p_cfg )
```

10.15.3.1 Brief description

Initialize the Low Power flash peripheral. Implements open.

10.15.3.2 Detailed description

The Open function initializes the Flash. It first copies the FCU firmware to FCURAM and sets the FCU Clock based on the current FCLK frequency. In addition, if Code Flash programming is enabled, the API code responsible for Code Flash programming will be copied to RAM.

This function must be called once prior to calling any other FLASH API functions. If a user supplied callback function is supplied, then the Flash Ready interrupt will be configured to call the users callback routine with an Event type describing the source of the interrupt for Data Flash operations. Subsequent to successfully completing this call p_ctrl->opened will be true.

NOTE: *Providing a callback function in the supplied p_cfg->callback field, automatically configures the Flash for Data Flash to operate in non-blocking (BGO) mode.*Subsequent to a successful Open(), the Flash is ready to process additional Flash commands.

Table 1237:Return values

Name	Description
SSP_SUCCESS	Initialization was successful and timer has started.
SSP_FLASH_ERR_FAILURE	Failed to successfully enter Programming/Erase mode.
SSP_ERR_TIMEOUT	Timed out waiting for FCU to be ready.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg or problem getting FMI info.
SSP_ERR_IRQ_BSP_DISABLED	Caller is requesting BGO but the Flash interrupts are not enabled.
SSP_ERR_FCLK	FCLK must be a minimum of 4 MHz for Flash operations.
SSP_ERR_HW_LOCKED	FLASH peripheral has already been initialized and is in use.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

- eventInfoGet

10.15.3.3 Function steps

- g_flash_lp_version is accessed by the ASSERT macro only, and so compiler toolchain can issue a warning that it is not accessed. The code below eliminates this warning and also ensures data structures are not optimized away.
- Insure API has not been opened
- While the Flash API is in use we will disable the FLash Cache.
- Allow Initialization if not initialized or if no operation is ongoing and re-initialization is desired
- Check that API is not already Open
- Set the parameters struct based on the user supplied settings
- Setup the Flash interrupt callback based on the caller's info. If the Flash interrupt is not enabled in the BSP then this will return SSP_ERR_IRQ_BSP_DISABLED
- Make sure Flash interrupts are disabled, they are only used in BGO mode
- Save callback function pointer

10.15.4 R_FLASH_LP_Write

```
ssp_err_t R_FLASH_LP_Write ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const src_address ,   uint32_t flash_address ,   uint32_t const num_bytes )
```

10.15.4.1 Brief description

Write to the specified Code or Data Flash memory area. Implements write.

10.15.4.2 Detailed description

The minimum/maximum number of bytes, as well as the invalid address and programming boundaries supported and enforced by this function are dependent on the MCU in use as well as the part package size. Please see the User manual and/or requirements document for additional information.

Table 1238:Return values

Name	Description
SSP_SUCCESS	Operation successful. If BGO is enabled this means the operation was started successfully.
SSP_ERR_IN_USE	The Flash peripheral is busy with a prior on-going transaction.
SSP_ERR_NOT_OPEN	The Flash API is not Open.

Table 1238:Return values (Continued)

Name	Description
SSP_ERR_CMD_LOCKED	FCU is in locked state, typically as a result of attempting to Write an area that is protected by an Access Window.
SSP_ERR_WRITE_FAILED	Status is indicating a Programming error for the requested operation. This may be returned if the requested Flash area is not blank.
SSP_ERR_TIMEOUT	Timed out waiting for FCU operation to complete.
SSP_ERR_INVALID_SIZE	Number of bytes provided was not a multiple of the programming size or exceeded the maximum range.
SSP_ERR_INVALID_ADDRESS	Invalid address was input or address not on programming boundary.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled and a request to write CF was requested.

10.15.4.3 Function steps

- Get the block information for this address

10.15.5 R_FLASH_LP_Read

```
ssp_err_t R_FLASH_LP_Read ( flash_ctrl_t  *const p_api_ctrl ,   uint8_t
* p_dest_address ,   uint32_t const flash_address ,   uint32_t const num_bytes )
```

10.15.5.1 Brief description

Read the requested number of bytes from the supplied Data or Code Flash address. Implements read.

10.15.5.2 Detailed description

The minimum/maximum number of blocks, as well as the invalid address and programming boundaries supported and enforced by this function are dependent on the MCU in use as well as the part package size. Please see the User manual and/or requirements document for additional information.

NOTE: This function is provided simply for the purposes of maintaining a complete interface. It is possible (and recommended), to read Flash memory directly.

Table 1239:Return values

Name	Description
SSP_SUCCESS	Operation successful.
SSP_ERR_INVALID_ADDRESS	Invalid Flash address was supplied.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_dest_address
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.5.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.15.6 R_FLASH_LP_Erase

```
ssp_err_t R_FLASH_LP_Erase ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const address ,   uint32_t const num_blocks )
```

10.15.6.1 Brief description

Erase the specified Code or Data Flash blocks. Implements erase.

10.15.6.2 Detailed description

The minimum/maximum number of blocks, as well as the invalid address and programming boundaries supported and enforced by this function are dependent on the MCU in use as well as the part package size. Please see the User manual and/or requirements document for additional information.

Table 1240:Return values

Name	Description
SSP_SUCCESS	Successful open.
SSP_ERR_INVALID_BLOCKS	Invalid number of blocks specified
SSP_ERR_INVALID_ADDRESS	Invalid address specified
SSP_ERR_IN_USE	Other flash operation in progress, or API not initialized
SSP_ERR_CMD_LOCKED	FCU is in locked state, typically as a result of attempting to Erase an area that is protected by an Access Window.

Table 1240:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	The Flash API is not Open.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled and a request to erase CF was requested.

10.15.6.3 Function steps

- Get the block information for this address
- Update Flash state and enter the respective Code or Data Flash P/E mode, may return SSP_ERR_IN_USE
- Still good to go?
- Is this a request to Erase Code Flash?
- This is a request to erase Data Flash
- Erase the Blocks, give this function the details about this block
- If in non-BGO mode, the current operation is complete. Exit PE mode and return status

10.15.7 R_FLASH_LP_BlankCheck

```
ssp_err_t R_FLASH_LP_BlankCheck ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const address ,   uint32_t num_bytes ,  flash_result_t  * p_blank_check_result )
```

10.15.7.1 Brief description

Perform a blank check on the specified address area. Implements blankCheck.

10.15.7.2 Detailed description

The minimum/maximum number of bytes, as well as the invalid address and programming boundaries supported and enforced by this function are dependent on the MCU in use as well as the part package size. Please see the User manual and/or requirements document for additional information. The number of bytes for Data Flash blank checking must be between (1 and FLASH_DATA_BLANK_CHECK_MAX). The number of bytes for Code Flash blank checking must be between (1 and FLASH_CODE_BLANK_CHECK_MAX).

Table 1241:Return values

Name	Description
SSP_SUCCESS	Blankcheck operation completed with result in p_blank_check_result, or blankcheck started and in-progess (BGO mode).
SSP_ERR_INVALID_ADDRESS	Invalid data flash address was input
SSP_ERR_INVALID_SIZE	'num_bytes' was either too large or not aligned for the CF/DF boundary size.
SSP_ERR_IN_USE	Flash is busy with an on-going operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.7.3 Function steps

- Get the block information for this address
- Validate the address and length provided
- Update Flash state and enter the respective Code or Data Flash P/E mode, may return SSP_ERR_IN_USE
- SSP_ERR_IN_USE would indicate that a BGO operation is underway, so don't reset in that case

10.15.8 R_FLASH_LP_StatusGet

```
ssp_err_t R_FLASH_LP_StatusGet ( flash_ctrl_t  *const p_api_ctrl )
```

10.15.8.1 Brief description

Query the FLASH for its status. Implements statusGet.

10.15.8.2 Detailed description

Table 1242:Return values

Name	Description
SSP_SUCCESS	Flash is ready and available to accept commands.
SSP_ERR_IN_USE	Flash is busy with an on-going operation.

Table 1242:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.8.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Return flash status

10.15.9 R_FLASH_LP_AccessWindowSet

```
ssp_err_t R_FLASH_LP_AccessWindowSet ( flash_ctrl_t  *const p_api_ctrl ,
  uint32_t const start_addr ,   uint32_t const end_addr )
```

10.15.9.1 Brief description

Configure an access window for the Code Flash memory. Implements accessWindowSet.

10.15.9.2 Detailed description

An access window defines a contiguous area in Code Flash for which programming/erase is enabled. This area is on block boundaries. The block containing start_addr is the first block. The block containing end_addr is the last block. The access window then becomes first block > last block inclusive. Anything outside this range of Code Flash is then write protected. As an example, if you wanted to place an accesswindow on Code Flash Blocks 0 and 1, such that only those two blocks were writable, you would need to specify (address in block 0, address in block 2) as the respective start and end address. ***NOTE:*** *If the start address and end address are set to the same value, then the access window is effectively removed. This accomplishes the same functionality as R_FLASH_LP_AccessWindowClear.*The invalid address and programming boundaries supported and enforced by this function are dependent on the MCU in use as well as the part package size. Please see the User manual and/or requirements document for additional information.

Table 1243:Return values

Name	Description
SSP_SUCCESS	Access window successfully configured.
SSP_ERR_INVALID_ADDRESS	Invalid settings for start_addr and/or end_addr.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled.

Table 1243:Return values (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.9.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Remove warnings generated when Code Flash code is not compiled in.
- < For consistency with _LP API we return error if Code Flash not enabled

10.15.10 R_FLASH_LP_AccessWindowClear

```
ssp_err_t R_FLASH_LP_AccessWindowClear ( flash_ctrl_t  *const p_api_ctrl )
```

10.15.10.1 Brief description

Remove any access window that is configured in the Code Flash. Implements accessWindowClear. On successful return from this call all Code Flash is writable.

10.15.10.2 Detailed description

Table 1244:Return values

Name	Description
SSP_SUCCESS	Access window successfully removed.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled.
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.10.3 Function steps

- Eliminate warning if parameter checking is disabled.
- < Return error if Code Flash not enabled

10.15.11 R_FLASH_LP_Reset

```
ssp_err_t R_FLASH_LP_Reset ( flash_ctrl_t  *const p_api_ctrl )
```

10.15.11.1 Brief description

Reset the FLASH peripheral. Implements reset.

10.15.11.2 Detailed description

No attempt is made to grab the Flash software lock before executing the reset since the assumption is that a reset will terminate any existing operation.

Table 1245:Return values

Name	Description
SSP_SUCCESS	Flash circuit successfully reset.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.12 R_FLASH_LP_StartUpAreaSelect

```
ssp_err_t R_FLASH_LP_StartUpAreaSelect ( flash_ctrl_t  *const p_api_ctrl ,
 flash_startup_area_swap_t  swap_type ,   bool is_temporary )
```

10.15.12.1 Brief description

Select which block is used as the startup area block. Implements startupAreaSelect.

10.15.12.2 Detailed description

Selects which block - Default (Block 0) or Alternate (Block 1) is used as the startup area block. The provided parameters determine which block will become the active startup block and whether that action will be immediate (but temporary), or permanent subsequent to the next reset. Doing a temporary switch might appear to have limited usefulness. If there is an access window in place such that Block 0 is write protected, then one could do a temporary switch, update the block and switch them back without having to touch the access window.

Table 1246:Return values

Name	Description
SSP_SUCCESS	Start-up area successfully toggled.

Table 1246:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	Flash is busy with an on-going operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.12.3 Function steps

- This is already the normal reset behavior. There's nothing to do
- Update Flash state and enter the respective Code or Data Flash P/E mode, may return SSP_ERR_IN_USE
- Success?
- Return to read mode
- If there is an error, then reset the Flash. This will clear error flags and exit the P/E mode

10.15.13 R_FLASH_LP_UpdateFlashClockFreq

```
ssp_err_t R_FLASH_LP_UpdateFlashClockFreq ( flash_ctrl_t  *const p_api_ctrl )
```

10.15.13.1 Brief description

Indicate to the already open Flash API that the FCLK has changed. Implements r_flash_t::updateFlashClockFreq.

10.15.13.2 Detailed description

This could be the case if the application has changed the system clock, and therefore the FCLK. Failure to call this function subsequent to changing the FCLK could result in damage to the flash macro. This function uses R_CGC_SystemClockFreqGet to get the current FCLK frequency.

Table 1247:Return values

Name	Description
SSP_SUCCESS	Start-up area successfully toggled.
SSP_ERR_IN_USE	Flash is busy with an on-going operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.15.13.3 Function steps

- Grab the current flash state
- API already in this state
- < Check FCLK, calculate timeout values

10.15.14 R_FLASH_LP_InfoGet

```
ssp_err_t R_FLASH_LP_InfoGet ( flash_ctrl_t  *const p_api_ctrl ,  flash_info_t
*const p_info )
```

10.15.14.1 Brief description

Returns the information about the flash regions. Implements infoGet.

10.15.14.2 Detailed description

Table 1248:Return values

Name	Description
SSP_SUCCESS	Successful retrieved the request information.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_info.

10.15.15 R_FLASH_LP_Close

```
ssp_err_t R_FLASH_LP_Close ( flash_ctrl_t  *const p_api_ctrl )
```

10.15.15.1 Brief description

Release any resources that were allocated by the Flash API. Implements close.

10.15.15.2 Detailed description

Table 1249:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.

10.15.15.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Return the hardware lock for the Flash
- Restore the FLash Cache state to what it was before we opened the Flash API.
- API is now closed
- Release the lock

10.15.16 R_FLASH_LP_VersionGet

```
ssp_err_t R_FLASH_LP_VersionGet ( ssp_version_t  *const p_version )
```

10.15.16.1 Brief description

Get FLASH HAL driver version.

10.15.16.2 Detailed description

Table 1250:Return values

Name	Description
SSP_SUCCESS	- operation performed successfully

NOTE: This function is reentrant.

10.15.17 Extensions

10.15.17.1 flash_lp_instance_ctrl_t

flash_lp_instance_ctrl_t

Detailed description

Flash instance control block. DO NOT INITIALIZE.

Variables

- uint32_t opened

  To check whether api has been opened or not.
- void * p_reg

  Base address of flash registers.
- void(* p_callback)(*p_args)

- bsp_cache_state_t cache_state

  Used to disable and then restore Flash Cache while API is open.
- IRQn_Type irq

  Flash ready interrupt number.

10.16 High-performance Flash

Driver for the High-performance Flash Memory (S7G2 and S5D9).
Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
This module supports the Flash interface for the High Performance FLASH peripheral.

10.16.1 Functions

- R_FLASH_HP_Open
- R_FLASH_HP_Write
- R_FLASH_HP_Read
- R_FLASH_HP_Erase
- R_FLASH_HP_BlankCheck
- R_FLASH_HP_StatusGet
- R_FLASH_HP_AccessWindowSet
- R_FLASH_HP_AccessWindowClear
- R_FLASH_HP_Reset
- R_FLASH_HP_StartUpAreaSelect
- R_FLASH_HP_UpdateFlashClockFreq
- R_FLASH_HP_InfoGet
- R_FLASH_HP_Close
- R_FLASH_HP_VersionGet
- flash_lock_state
- flash_ReleaseState
- setup_for_pe_mode
- flash_get_block_info
- flash_setup
- flash_open_setup
- flash_write_initiate
- flash_operation_complete

- flash_blank_check_address_checking
- flash_blank_check_setup
- flash_blank_check_initiate
- flash_fmi_setup

10.16.2 Defines

- #define FLASH_HP_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define FLASH_HP_CODE_VERSION_MINOR

  Initial value:`(4U)`
- #define FLASH_HP_VERSION_PHASE_2

  Initial value:

  S5D9 and S5D5 MCUs uses RV40F Phase 2 Flash technology. This macro will eventually be migrated to bsp_feature.h.
- #define PLACE_IN_RAM_SECTION

  Initial value:

10.16.3 R_FLASH_HP_Open

```
ssp_err_t R_FLASH_HP_Open ( flash_ctrl_t  *const p_api_ctrl ,  flash_cfg_t  const
*const p_cfg )
```

10.16.3.1 Brief description

Initializes the flash peripheral. Implements open.

10.16.3.2 Detailed description

The Open function initializes the Flash. It first copies the FCU firmware to FCURAM and sets the FCU Clock based on the current FCLK frequency. In addition, if Code Flash programming is enabled, the API code responsible for Code Flash programming will be copied to RAM.

This function must be called once prior to calling any other FLASH API functions. If a user supplied callback function is supplied, then the Flash Ready and Error interrupts will be configured to call the users callback routine with an Event type describing the source of the interrupt.

NOTE: *Providing a callback function in the supplied p_cfg->callback field, automatically configures the Flash for Data Flash to operate in non-blocking (BGO) mode.*Subsequent to a successful Open(), the Flash is ready to process additional Flash commands.

Table 1251:Return values

Name	Description
SSP_SUCCESS	Initialization was successful and timer has started.
SSP_FLASH_ERR_FAILURE	Failed to successfully enter Programming/Erase mode.
SSP_ERR_TIMEOUT	Timed out waiting for FCU to be ready.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg or problem getting FMI info.
SSP_ERR_IRQ_BSP_DISABLED	Caller is requesting BGO but the Flash interrupts are not enabled.
SSP_ERR_FCLK	FCLK must be a minimum of 4 MHz for Flash operations.
SSP_ERR_HW_LOCKED	FLASH peripheral has already been initialized and is in use.

10.16.3.3 Function steps

- g_flash_hp_version is accessed by the ASSERT macro only, and so compiler toolchain can issue a warning that it is not accessed. The code below eliminates this warning and also ensures data structures are not optimized away.
- While the Flash API is in use we will disable the FLash Cache.
- Allow Initialization if not initialized or if no operation is ongoing and re-initialization is desired
- Check that API is not already Open
- Set the parameters struct based on the user supplied settings
- Setup the Flash interrupt callback based on the caller's info. If both Flash interrupts are not enabled in the BSP then this will return SSP_ERR_IRQ_BSP_DISABLED
- Make sure Flash interrupts are disabled, they are only used in BGO mode
- Save callback function pointer

10.16.4 R_FLASH_HP_Write

```
ssp_err_t R_FLASH_HP_Write ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const src_address ,   uint32_t flash_address ,   uint32_t const num_bytes )
```

10.16.4.1 Brief description

Writes to the specified Code or Data Flash memory area. Implements write.

10.16.4.2 Detailed description

Table 1252:Return values

Name	Description
SSP_SUCCESS	Operation successful. If BGO is enabled this means the operation was started successfully.
SSP_ERR_IN_USE	The Flash peripheral is busy with a prior on-going transaction.
SSP_ERR_NOT_OPEN	The Flash API is not Open.
SSP_ERR_CMD_LOCKED	FCU is in locked state, typically as a result of attempting to Write an area that is protected by an Access Window.
SSP_ERR_WRITE_FAILED	Status is indicating a Programming error for the requested operation. This may be returned if the requested Flash area is not blank.
SSP_ERR_TIMEOUT	Timed out waiting for FCU operation to complete.
SSP_ERR_INVALID_SIZE	Number of bytes provided was not a multiple of the programming size or exceeded the maximum range.
SSP_ERR_INVALID_ADDRESS	Invalid address was input or address not on programming boundary.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled and a request to write CF was requested.

10.16.5 R_FLASH_HP_Read

```
ssp_err_t R_FLASH_HP_Read ( flash_ctrl_t  *const p_api_ctrl ,   uint8_t
* p_dest_address ,   uint32_t const flash_address ,   uint32_t const num_bytes )
```

10.16.5.1 Brief description

Reads the requested number of bytes from the supplied Data or Code Flash memory address. Implements read.

10.16.5.2 Detailed description

NOTE: This function is provided simply for the purposes of maintaining a complete interface. It is possible (and recommended), to read Flash memory directly.

Table 1253:Return values

Name	Description
SSP_SUCCESS	Operation successful.
SSP_ERR_INVALID_ADDRESS	Invalid Flash address was supplied.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_dest_address

10.16.5.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.16.6 R_FLASH_HP_Erase

```
ssp_err_t R_FLASH_HP_Erase ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const address ,   uint32_t const num_blocks )
```

10.16.6.1 Brief description

Erases the specified Code or Data Flash blocks. Implements erase by the block_erase_address.

10.16.6.2 Detailed description

Table 1254:Return values

Name	Description
SSP_SUCCESS	Successful open.
SSP_ERR_INVALID_BLOCKS	Invalid number of blocks specified
SSP_ERR_INVALID_ADDRESS	Invalid address specified
SSP_ERR_IN_USE	Other flash operation in progress, or API not initialized
SSP_ERR_CMD_LOCKED	FCU is in locked state, typically as a result of attempting to Erase an area that is protected by an Access Window.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	The Flash API is not Open.

Table 1254:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled and a request to erase CF was requested.

10.16.6.3 Function steps

- Update Flash state and enter the respective Code or Data Flash P/E mode, may return SSP_ERR_IN_USE

10.16.7 R_FLASH_HP_BlankCheck

```
ssp_err_t R_FLASH_HP_BlankCheck ( flash_ctrl_t  *const p_api_ctrl ,   uint32_t
const address ,   uint32_t num_bytes ,  flash_result_t  * p_blank_check_result )
```

10.16.7.1 Brief description

Performs a blank check on the specified address area. Implements blankCheck.

10.16.7.2 Detailed description

Table 1255:Return values

Name	Description
SSP_SUCCESS	Blankcheck operation completed with result in p_blank_check_result, or blankcheck started and in-progess (BGO mode).
SSP_ERR_INVALID_ADDRESS	Invalid data flash address was input.
SSP_ERR_INVALID_SIZE	'num_bytes' was either too large or not aligned for the CF/DF boundary size.
SSP_ERR_IN_USE	Other flash operation in progress or API not initialized.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.

10.16.7.3 Function steps

- Validate the address and length provided

10.16.8 R_FLASH_HP_StatusGet

```
ssp_err_t R_FLASH_HP_StatusGet ( flash_ctrl_t  *const p_api_ctrl )
```

10.16.8.1 Brief description

Query the FLASH peripheral for its status. Implements statusGet.

10.16.8.2 Detailed description

Table 1256:Return values

Name	Description
SSP_SUCCESS	FLASH peripheral is ready to use.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.

10.16.8.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.16.9 R_FLASH_HP_AccessWindowSet

```
ssp_err_t R_FLASH_HP_AccessWindowSet ( flash_ctrl_t  *const p_api_ctrl ,
  uint32_t const start_addr ,   uint32_t const end_addr )
```

10.16.9.1 Brief description

Configure an access window for the Code Flash memory using the provided start and end address. An access window defines a contiguous area in Code Flash for which programming/erase is enabled. This area is on block boundaries. The block containing start_addr is the first block. The block containing end_addr is the last block. The access window then becomes first block > last block inclusive. Anything outside this range of Code Flash is then write protected.

10.16.9.2 Detailed description

NOTE: *If the start address and end address are set to the same value, then the access window is effectively removed. This accomplishes the same functionality as R_FLASH_HP_AccessWindowClear.*Implements accessWindowSet.

Table 1257:Return values

Name	Description
SSP_SUCCESS	Access window successfully configured.
SSP_ERR_INVALID_ADDRESS	Invalid settings for start_addr and/or end_addr.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled.
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.16.9.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Remove warnings generated when Code Flash code is not compiled in.
- < For consistency with _LP API we return error if Code Flash not enabled

10.16.10 R_FLASH_HP_AccessWindowClear

```
ssp_err_t R_FLASH_HP_AccessWindowClear ( flash_ctrl_t  *const p_api_ctrl )
```

10.16.10.1 Brief description

Remove any access window that is currently configured in the Code Flash. Subsequent to this call all Code Flash is writable. Implements accessWindowClear.

10.16.10.2 Detailed description

Table 1258:Return values

Name	Description
SSP_SUCCESS	Access window successfully removed.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.

Table 1258:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Code Flash Programming is not enabled.
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.16.10.3 Function steps

- Eliminate warning if parameter checking is disabled.
- < For consistency with _LP API we return error if Code Flash not enabled

10.16.11 R_FLASH_HP_Reset

```
ssp_err_t R_FLASH_HP_Reset ( flash_ctrl_t  *const p_api_ctrl )
```

10.16.11.1 Brief description

Resets the FLASH peripheral. Implements reset. No attempt is made to grab the Flash software lock before executing the reset since the assumption is that a reset will terminate any existing operation.

10.16.11.2 Detailed description

Table 1259:Return values

Name	Description
SSP_SUCCESS	Flash circuit successfully reset.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.

10.16.11.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.16.12 R_FLASH_HP_StartUpAreaSelect

```
ssp_err_t R_FLASH_HP_StartUpAreaSelect ( flash_ctrl_t  *const p_api_ctrl ,
 flash_startup_area_swap_t  swap_type ,   bool is_temporary )
```

10.16.12.1 Brief description

Selects which block - Default (Block 0) or Alternate (Block 1) is used as the startup area block. The provided parameters determine which block will become the active startup block and whether that action will be immediate (but temporary), or permanent subsequent to the next reset. Doing a temporary switch might appear to have limited usefulness. If there is an access window in place such that Block 0 is write protected, then one could do a temporary switch, update the block and switch them back without having to touch the access window. Implements startupAreaSelect.

10.16.12.2 Detailed description

Table 1260:Return values

Name	Description
SSP_SUCCESS	Start-up area successfully toggled.
SSP_ERR_IN_USE	FLASH peripheral is busy with a prior operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl.

10.16.12.3 Function steps

- Eliminate warning if parameter checking is disabled.
- This is already the normal reset behavior. There's nothing to do
- Update Flash state and enter the respective Code or Data Flash P/E mode based on whether or not we will be c calling Configuration Set

10.16.13 R_FLASH_HP_UpdateFlashClockFreq

```
ssp_err_t R_FLASH_HP_UpdateFlashClockFreq ( flash_ctrl_t  *const p_api_ctrl )
```

10.16.13.1 Brief description

Indicate to the already open Flash API, that the FCLK has changed since the Open(). This could be the case if the application has changed the system clock, and therefore the FCLK. Failure to call this function subsequent to changing the FCLK could result in damage to the flash macro. This function uses R_CGC_SystemClockFreqGet to get the current FCLK frequency. Implements updateFlashClockFreq.

10.16.13.2 Detailed description

Table 1261:Return values

Name	Description
SSP_SUCCESS	Start-up area successfully toggled.
SSP_ERR_IN_USE	Flash is busy with an on-going operation.
SSP_ERR_ASSERTION	NULL provided for p_ctrl
SSP_ERR_NOT_OPEN	Flash API has not yet been opened.

10.16.13.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Grab the current flash state
- API already in this state
- < Check FCLK, calculate timeout values

10.16.14 R_FLASH_HP_InfoGet

```
ssp_err_t R_FLASH_HP_InfoGet ( flash_ctrl_t  *const p_api_ctrl ,  flash_info_t
*const p_info )
```

10.16.14.1 Brief description

Returns the information about the flash regions. Implements infoGet.

10.16.14.2 Detailed description

Table 1262:

Name	Description
SSP_SUCCESS	Successful retrieved the request information.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_info.

10.16.14.3 Function steps

- Eliminate warning if parameter checking is disabled.

10.16.15 R_FLASH_HP_Close

```
ssp_err_t R_FLASH_HP_Close ( flash_ctrl_t  *const p_api_ctrl )
```

10.16.15.1 Brief description

Releases any resources that were allocated by the Open() or any subsequent Flash operations. Implements close.

10.16.15.2 Detailed description

Table 1263:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	NULL provided for p_ctrl or p_cfg.

10.16.15.3 Function steps

- Eliminate warning if parameter checking is disabled.
- Return the hardware lock for the Flash
- Restore the FLash Cache state to what it was before we opened the Flash API.
- Release the lock
- Close the API

10.16.16 R_FLASH_HP_VersionGet

```
ssp_err_t R_FLASH_HP_VersionGet ( ssp_version_t  *const p_version )
```

10.16.16.1 Brief description

This function gets FLASH HAL driver version.

10.16.16.2 Detailed description

Table 1264:

Name	Description
SSP_SUCCESS	- operation performed successfully

NOTE: This function is reentrant.

10.16.17 flash_lock_state

```
ssp_err_t flash_lock_state (  flash_states_t new_state )
```

10.16.17.1 Brief description

This function attempts to change the flash state to the new requested state. This can only happen if we are able to take the FLASH lock. If the lock is currently in use then we will return FLASH_ERR_BUSY, otherwise we will take the lock and change the state.

10.16.17.2 Detailed description

Internal functions.

Table 1265:

Name	Direction	Description
new_state	in	Which state to attempt to transition to.

Table 1266:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_IN_USE	Flash is busy with another operation.

10.16.18 flash_ReleaseState

```
flash_ReleaseState (  void  )
```

10.16.18.1 Brief description

This function releases the flash state so another flash operation can take place. This function is called by both the HLD and LLD layers (interrupt routines).

10.16.18.2 Detailed description

Table 1267:

Name	Description
None	

10.16.19 setup_for_pe_mode

```
ssp_err_t setup_for_pe_mode ( flash_hp_instance_ctrl_t  *const p_ctrl ,
  flash_type_t flash_type ,   flash_states_t flash_state )
```

10.16.19.1 Brief description

This function places the flash in the requested Code or Data P/E mode.

10.16.19.2 Detailed description

Table 1268:

Name	Direction	Description
p_ctrl	in	Pointer to the Flash control block.
flash_type	in	FLASH_TYPE_CODE_FLASH or FLASH_TYPE_DATA_FLASH.
flash_state	in	Desired Flash state to transition into (ie FLASH_STATE_WRITING).

Table 1269:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_IN_USE	Flash is busy with another operation.

10.16.20 flash_get_block_info

```
flash_get_block_info (  uint32_t addr ,   flash_block_info_t * p_block_info )
```

10.16.20.1 Brief description

This function takes the supplied address and fills in the supplied block info structure with the respective details about the block in which it resides.

10.16.20.2 Detailed description

Table 1270:

Name	Description
false	Supplied address is valid flash address on this MCU.
true	Supplied address is valid and p_block_info contains the details on this addresses block

10.16.20.3 Function steps

- Determine if this is a Code or Data Flash address, or something else (invalid)
- Is this address within this section of blocks?
- Make sure we know the end of the Code or Data Flash memory space.
- add in # of blocks in this section

10.16.21 flash_setup

```
ssp_err_t flash_setup (  R_FACI_Type * p_faci_reg )
```

10.16.21.1 Brief description

This function verifies that FCLK falls within the allowable range and calculates the timeout values based on the current FCLK frequency.

10.16.21.2 Detailed description

Table 1271:

Name	Description
SSP_SUCCESS	Setup complete
SSP_ERR_FCLK	FCLK must be a minimum of 4 MHz for Flash operations.

10.16.21.3 Function steps

- We need clock frequencies to calculate the worst case timeout values.
- FCLK must be a minimum of 4 MHz for Flash operations

10.16.22 flash_open_setup

```
ssp_err_t flash_open_setup ( flash_hp_instance_ctrl_t  *const p_ctrl ,
 ssp_feature_t  * p_ssp )
```

10.16.22.1 Brief description

This function does the completion setup for the R_FLASH_HP_Open function.

10.16.22.2 Function steps

- Check FCLK, calculate timeout values
- If we failed to successfully open then return the hardware lock
- API is now open

10.16.23 flash_write_initiate

```
ssp_err_t flash_write_initiate ( flash_hp_instance_ctrl_t  *const p_ctrl ,
  uint32_t const * src_start_address ,   uint32_t * dest_start_address ,
  uint32_t num_bytes )
```

10.16.23.1 Brief description

This function initiates the write sequence for the R_FLASH_HP_Write function.

10.16.23.2 Function steps

- Update Flash state and enter the respective Code or Data Flash P/E mode

10.16.24 flash_operation_complete

```
flash_operation_complete ( flash_hp_instance_ctrl_t  *const p_ctrl ,
ssp_err_t  err )
```

10.16.24.1 Brief description

This function performs the final cleanup for the erase, write and blankcheck functions.

10.16.24.2 Function steps

- SSP_ERR_IN_USE will be returned if we are in the process of executing a BGO operation. In this case we do not want to take any action as that will terminate the in process operation

10.16.25 flash_blank_check_address_checking

```
ssp_err_t flash_blank_check_address_checking (  uint32_t const address ,
  uint32_t num_bytes )
```

10.16.25.1 Brief description

This function performs the address checking required by the R_FLASH_HP_BlankCheck function.

10.16.25.2 Detailed description

Table 1272:

Name	Description
SSP_SUCCESS	Parameter checking completed without error.
SSP_ERR_INVALID_ADDRESS	Invalid data flash address was input.
SSP_ERR_INVALID_SIZE	'num_bytes' was either too large or not aligned for the CF/DF boundary size.

10.16.25.3 Function steps

- Check if the range is valid, num_bytes is larger than 0 and up to and including the last byte in the block
- Is this a request to Blank Check Data Flash?. If so, num_bytes must be a multiple of the programming size

10.16.26 flash_blank_check_setup

```
ssp_err_t flash_blank_check_setup ( flash_ctrl_t  *const p_api_ctrl ,   uint8_t
* p_address ,   uint32_t num_bytes ,  flash_result_t  * p_blank_check_result )
```

10.16.26.1 Brief description

This function performs the setup phase required by the R_FLASH_HP_BlankCheck function.

10.16.26.2 Detailed description

Table 1273:

Name	Description
SSP_SUCCESS	Setup completed completed without error.
SSP_ERR_PE_FAILURE	Failed to enter P/E mode

10.16.26.3 Function steps

- Blank checking for Code Flash does not require any FCU operations. The specified address area Can simply be checked for non 0xFF.
- Assume blank until we know otherwise

10.16.27 flash_blank_check_initiate

```
ssp_err_t flash_blank_check_initiate ( flash_hp_instance_ctrl_t  *const p_ctrl ,
  uint32_t const address ,   uint32_t num_bytes ,  flash_result_t
* p_blank_check_result )
```

10.16.27.1 Brief description

This function performs the Blank check phase required by the R_FLASH_HP_BlankCheck function.

10.16.27.2 Detailed description

Table 1274:

Name	Description
SSP_SUCCESS	Setup completed completed without error.

Table 1274: (Continued)

Name	Description
SSP_ERR_PE_FAILURE	Failed to enter P/E mode

10.16.28 flash_fmi_setup

```
ssp_err_t flash_fmi_setup ( flash_hp_instance_ctrl_t  *const p_ctrl ,
 flash_cfg_t  const *const p_cfg ,  ssp_feature_t  * p_ssp )
```

10.16.28.1 Brief description

This function initializes data required by the Flash based on information read from the FMI.

10.16.28.2 Detailed description

Table 1275:

Name	Description
SSP_SUCCESS	FMI based setup success.
SSP_ERR_IN_USE	The Flash peripheral is busy with a prior on-going transaction.
SSP_ERR_ASSERTION	Problem getting FMI information.
SSP_ERR_HW_LOCKED	FLASH peripheral has already been initialized and is in use.

10.16.29 Extensions

10.16.29.1 flash_hp_instance_ctrl_t

flash_hp_instance_ctrl_t

Detailed description

Flash HP instance control block. DO NOT INITIALIZE.

Variables

- uint32_t opened

  To check whether api has been opened or not.

- R_FACI_Type * p_reg

  Base address of flash registers.

- void(* p_callback)(*p_args)
- bsp_cache_state_t cache_state

  User Callback function.

  Used to disable and then restore Flash Cache while API is open.
- IRQn_Type irq

  Flash ready interrupt number.
- IRQn_Type err_irq

  Flash error interrupt number.

10.17 FMI

Driver for accessing Factory MCU Information (FMI).
Read Factory MCU Information flash memory.

10.17.1 Functions

- R_FMI_Init
- R_FMI_ProductInfoGet
- R_FMI_UniqueIdGet
- R_FMI_FeatureGet
- R_FMI_EventInfoGet
- R_FMI_VersionGet

10.17.2 Variables

- g_fmi_on_fmi

10.17.3 Defines

- #define FMI_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define FMI_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.17.4 R_FMI_Init

```
ssp_err_t R_FMI_Init (  void  )
```

10.17.4.1 Detailed description

Initializes factory flash base pointer. Implements init.

Table 1276:

Name	Description
SSP_SUCCESS	Factory flash initialization successful.
SSP_ERR_INVALID_FMI_DATA	The FMI data table provided is not valid, or at least one field in the part number is not compatible with the FMI data table provided.

10.17.4.2 Function steps

- Verify the provided MCU information data table is valid. This table is required to use the SSP. If the data in this table is not valid, log an unrecoverable error.
- Check to see if the factory flash is valid.
- The upper 16 bits of the base address of the factory flash must be 0x0100.
- If the factory flash is valid, store the base address for later and compare the part number to the part number mask match.

10.17.5 R_FMI_ProductInfoGet

```
ssp_err_t R_FMI_ProductInfoGet ( fmi_product_info_t  ** pp_product_info )
```

10.17.5.1 Detailed description

Get pointer to Factory MCU Information product information record. Implements productInfoGet.

Table 1277:

Name	Description
SSP_SUCCESS	Caller's pointer set to Product Information Record.
SSP_ERR_ASSERTION	Caller's pointer is NULL.

10.17.5.2 Function steps

- Use the factory flash if it is valid. Otherwise use the SSP MCU information.

10.17.6 R_FMI_UniqueIdGet

```
ssp_err_t R_FMI_UniqueIdGet ( fmi_unique_id_t  * p_unique_id )
```

10.17.6.1 Detailed description

Get unique ID for this device. Implements uniqueIdGet.

Table 1278:

Name	Description
SSP_SUCCESS	Unique ID stored in p_unique_id
SSP_ERR_ASSERTION	p_unique_id was NULL
SSP_ERR_INVALID_FACTORY_FLASH	Factory flash is not valid

10.17.6.2 Function steps

- If the factory flash is not valid, return an error.
- Store unique ID in p_unique_id.

10.17.7 R_FMI_FeatureGet

```
ssp_err_t R_FMI_FeatureGet ( ssp_feature_t  const *const p_feature ,
 fmi_feature_info_t  *const p_info )
```

10.17.7.1 Detailed description

Get feature information for the requested feature. Implements productFeatureGet.

Table 1279:

Name	Description
SSP_SUCCESS	Feature information stored in p_info
SSP_ERR_ASSERTION	p_feature was NULL or p_info was NULL

Table 1279: (Continued)

Name	Description
SSP_ERR_IP_CHANNEL_NOT_PRESENT	Requested channel does not exist on this MCU
SSP_ERR_IP_UNIT_NOT_PRESENT	Requested unit does not exist on this MCU
SSP_ERR_INTERNAL	Requested feature is in a format not supported at this time

10.17.7.2 Function steps

- Get the factory flash base address for this IP.
- Find the factory flash base address for this unit.
- Populate the feature information.

10.17.8 R_FMI_EventInfoGet

```
ssp_err_t R_FMI_EventInfoGet ( ssp_feature_t  const *const p_feature ,
  ssp_signal_t signal ,  fmi_event_info_t  *const p_info )
```

10.17.8.1 Brief description

Get event information for the requested feature and signal. Implements eventInfoGet.

10.17.8.2 Detailed description

Table 1280:

Name	Description
SSP_SUCCESS	Event information stored in p_info
SSP_ERR_ASSERTION	p_feature was NULL or p_info was NULL
SSP_ERR_IRQ_BSP_DISABLED	Event information could not be found. p_info::irq is set to SSP_INVALID_VECTOR and p_info::event is set to 0xFF.

NOTE: The ssp_signal_t enums and ssp_ip_t are collectively defined for different modules. When used with the FMI API, the signal and IP should correspond as follows:

1) IP name (ssp_ip_t enum : SSP_IP_<peripheral>)
2) Signal name (ssp_signal_t enum: SSP_SIGNAL_<peripheral>_ <signal>), where <peripheral> is the same as the one used in IP name above.

10.17.9 R_FMI_VersionGet

```
ssp_err_t R_FMI_VersionGet ( ssp_version_t  *const p_version )
```

10.17.9.1 Detailed description

Get the driver version based on compile time macros. Implements versionGet.

Table 1281:

Name	Description
SSP_SUCCESS	Caller's structure written.
SSP_ERR_ASSERTION	Caller's pointer is NULL.

10.17.10 g_fmi_on_fmi

```
fmi_api_t::g_fmi_on_fmi
```

10.18 GLCDC

Driver for the Graphics LCD Controller (GLCDC).

10.18.1 Summary

Implements Display Interface. This module supports the Graphics LCD Controller (GLCDC). It implements the display interface and drives LCD panels connected to the GLCDC pins.

10.18.2 Functions

- r_glcd_sync_signal_set
- r_glcd_background_screen_set
- r_glcd_graphics_layer_set
- r_glcd_output_block_set
- r_glcd_gamma_correction
- r_glcd_hsync_set
- r_glcd_vsync_set
- r_glcd_data_enable_set

- r_glcd_clock_set
- r_glcd_brightness_correction
- r_glcd_contrast_correction
- r_glcd_get_bit_size
- r_glcd_graphics_plane_format_set
- r_glcd_interrupt_enable
- r_glcd_output_data_order_set
- r_glcd_pixel_size_recalculate
- glcdc_line_detect_isr
- glcdc_underflow_1_isr
- glcdc_underflow_2_isr
- R_GLCD_Open
- R_GLCD_Close
- R_GLCD_Start
- R_GLCD_Stop
- R_GLCD_LayerChange
- R_GLCD_ColorCorrection
- R_GLCD_ClutUpdate
- R_GLCD_StatusGet
- R_GLCD_VersionGet

10.18.3 Variables

- module_version
- g_display_on_glcd
- default_glcd_cfg
- ctrl_blk

10.18.4 Defines

- #define GLCD_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &module_version)`
- #define GLCD_ADDRESS_ALIGNMENT_64B

  Initial value:`(64U)`

  The macro to use for 64-byte alignment checking, calculation

- #define OFFSET_MARGIN_MINUS_64PIX

  Initial value:(64U)

  This enables the GLCD to locate the foreground/background layer image, at the physical start of the left side of the display, if the offset of layer start position is a negative value, compared to the active video area. Shifting the base address of the layer image can be used in this operation, but it must be aligned to 64 byte boundary, so that the layer position adjustment can be made.

10.18.5 r_glcd_sync_signal_set

```
 r_glcd_sync_signal_set (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t  const
*const p_cfg )
```

10.18.5.1 Brief description

Subroutine to configure the sync signal setting (TCON block setting)

10.18.5.2 Detailed description

Table 1282:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for display interface
p_glcd_reg	in	Pointer to GLCD registers

Table 1283:

Name	Description
void	

10.18.5.3 Function steps

- Applied default configuration if GLCD HAL configuration is NULL

10.18.6 r_glcd_background_screen_set

```
 r_glcd_background_screen_set (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t
const *const p_cfg )
```

10.18.6.1 Brief description

Subroutine to configure the background screen setting.

10.18.6.2 Detailed description

- Panel timing setting
- Color setting for the background screen

Table 1284:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for display interface
p_glcd_reg	in	Pointer to GLCD registers

Table 1285:

Name	Description
void	

10.18.6.3 Function steps

- Set number of total cycle for a line including Sync & Back poach, Front poach
- Set the start position of Background screen
- Set the width of Background screen
- Set the Background color

10.18.7 r_glcd_graphics_layer_set

```
 r_glcd_graphics_layer_set (  R_GLCDC_Type * p_glcd_reg ,  display_input_cfg_t
const *const p_input ,  display_layer_t  const *const p_layer ,
 display_frame_layer_t  const frame )
```

10.18.7.1 Brief description

Subroutine to configure the graphics layer register settings which includes...

10.18.7.2 Detailed description

- Blend setting of foreground or background plane on background plane
- Rectangle area blending settings

Table 1286:

Name	Direction	Description
p_input	in	The input frame buffer configuration
p_layer	in	The layer configuration
frame	in	The number of input frame buffer
p_glcd_reg	in	Pointer to GLCD registers

Table 1287:

Name	Description
void	

NOTE: This function does not perform parameter check and it would be expected to be done in the caller function.

10.18.7.3 Function steps

- If the base address is NULL, just set the later transparent and disable read memory access
- Set the base address of graphics plane
- Set the background color on graphics plane
- Set the number of data transfer times per line, 64 bytes are transferred in each transfer
- If line number descending mode is enable, change its sign
- Set the line offset address for accessing the graphics data on graphics plane
- When line repeating mode, always read data on same line(s)
- Set the line offset address for accessing the graphics data
- Set the line offset address for accessing the graphics data on graphics plane
- Set the frame number of the graphics plane
- Set the frame offset for accessing the graphics data on the graphics plane
- Set the start position of the graphics layers
- Set the start position of the rectangle area in the graphics layers

- Set the width of the graphics layers
- Set the alpha blending condition

10.18.8 r_glcd_output_block_set

```
 r_glcd_output_block_set (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t  const
*const p_cfg )
```

10.18.8.1 Brief description

Subroutine to configure the output control block register settings which includes...

10.18.8.2 Detailed description

- Bit endian / color order setting
- Output color setting
- Color correction setting

Table 1288:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for display interface
p_glcd_reg	in	Pointer to GLCD registers

Table 1289:

Name	Description
void	

10.18.8.3 Function steps

- Applied default configuration if the GLCD HAL configuration is NULL
- selects the output format
- In case of serial RGB, set as RGB888 format
- sets the pixel clock (the GLCD internal signal) frequency in case that the output format is 8-bit serial RGB
- sets the Brightness/contrast and Gamma Correction processing order
- Set the dithering mode

10.18.9 r_glcd_gamma_correction

```
r_glcd_gamma_correction (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t  const
*const p_cfg )
```

10.18.9.1 Brief description

Subroutine to configure the gamma correction register setting.

10.18.9.2 Detailed description

Table 1290:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for the display interface
p_glcd_reg	in	Pointer to GLCD registers

Table 1291:

Name	Description
void	

10.18.10 r_glcd_hsync_set

```
r_glcd_hsync_set (  R_GLCDC_Type * p_glcd_reg ,  glcd_tcon_pin_t  tcon ,
display_timing_t  const * timing )
```

10.18.10.1 Brief description

Subroutine to configure the horizontal signal setting.

10.18.10.2 Detailed description

Table 1292:

Name	Direction	Description
p_glcd_reg	in	Pointer to GLCD registers

Table 1292: (Continued)

Name	Direction	Description
tcon	in	TCON pin select(GLCD_TCON_PIN_0|GLCD_TCON_PIN_1|GLCD_TCON_PIN_2|GLCD_TCON_PIN_3)
timing	in	Hsync signal timing

Table 1293:

Name	Description
void	

10.18.11 r_glcd_vsync_set

```
r_glcd_vsync_set (  R_GLCDC_Type * p_glcd_reg ,  glcd_tcon_pin_t  tcon ,
display_timing_t  const *const timing )
```

10.18.11.1 Brief description

Subroutine to configure the vertical signal setting.

10.18.11.2 Detailed description

Table 1294:

Name	Direction	Description
p_glcd_reg	in	Pointer to GLCD registers
tcon	in	TCON pin select(GLCD_TCON_PIN_0|GLCD_TCON_PIN_1|GLCD_TCON_PIN_2|GLCD_TCON_PIN_3)
timing	in	Vsync signal timing

Table 1295:

Name	Description
void	

10.18.12 r_glcd_data_enable_set

```
 r_glcd_data_enable_set (  R_GLCDC_Type * p_glcd_reg ,  glcd_tcon_pin_t
const tcon ,  display_timing_t  const *const vtiming ,  display_timing_t  const
*const htiming ,  display_signal_polarity_t  const polarity )
```

10.18.12.1 Brief description

Subroutine to configure the data enable(DE) signal setting.

10.18.12.2 Detailed description

Table 1296:

Name	Direction	Description
p_glcd_reg	in	Pointer to GLCD registers
tcon	in	TCON pin select(GLCD_TCON_PIN_0|GLCD_TCON_PIN_1|GLCD_TCON_PIN_2|GLCD_TCON_PIN_3)
vtiming	in	DE signal vertical timing
htiming	in	DE signal horizontal timing
polarity	in	DE signal porarity(DISPLAY_SIGNAL_POLARITY_LOACTIVE|DISPLAY_SIGNAL_POLARITY_HIACTIVE)

Table 1297:

Name	Description
void	

10.18.13 r_glcd_clock_set

```
r_glcd_clock_set (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t  const
*const p_cfg )
```

10.18.13.1 Brief description

Subroutine to configure dot clock setting.

10.18.13.2 Detailed description

Table 1298:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for display interface
p_glcd_reg	in	Pointer to GLCD registers

Table 1299:

Name	Description
void	

10.18.13.3 Function steps

- Selects input source for dot clock
- Sets division ratio
- Selects pixel clock output

10.18.14 r_glcd_brightness_correction

```
r_glcd_brightness_correction (  R_GLCDC_Type * p_glcd_reg ,
glcd_instance_ctrl_t  const *const p_ctrl ,  display_brightness_t  const
*const p_brightness )
```

10.18.14.1 Brief description

Subroutine to configure the brightness register settings. Pixel color output comes to be the value shown below processed by the brightness control block.

10.18.14.2 Detailed description

- Gout = Gin + p_cfg->output.brightness.g - 512 (output.brightness.g must be 10 bits value; up to 512)
- Bout = Bin + p_cfg->output.brightness.b - 512 (output.brightness.b must be 10 bits value; up to 512)
- Rout = Rin + p_cfg->output.brightness.r - 512 (output.brightness.r must be 10 bits value; up to 512)

Table 1300:

Name	Direction	Description
p_ctrl	in	Pointer to the control block for Display Interface
p_brightness	in	Pointer to brightness configuration structure
p_glcd_reg	in	Pointer to GLCD registers

Table 1301:

Name	Description
void	

10.18.14.3 Function steps

- Sets brightness correction register for each color in a pixel.
- If brightness setting in configuration is 'off', apply default value

10.18.15 r_glcd_contrast_correction

```
 r_glcd_contrast_correction (  R_GLCDC_Type * p_glcd_reg ,
 glcd_instance_ctrl_t  const *const p_ctrl ,  display_contrast_t  const
*const p_contrast )
```

10.18.15.1 Brief description

Subroutine to configure the contrast register settings. Pixel color output becomes the value shown below, processed by the contrast control block. Contrast can be changed between x0.000 to x1.992 (0x0:x0.000 / 0x80:x1.000 / 0xFF:x1.992).

10.18.15.2 Detailed description

- Gout = (Gin + p_contrast->g)/128

- Bout = (Bin + p_contrast->b)/128
- Rout = (Rin + p_contrast->r)/128

Table 1302:

Name	Direction	Description
p_ctrl	in	Pointer to the control block for the Display Interface
p_contrast	in	Pointer to the contrast configuration structure
p_glcd_reg	in	Pointer to GLCD registers

Table 1303:

Name	Description
void	

10.18.15.3 Function steps

- Sets the contrast correction register for each color in a pixel.
- If the contrast setting in the configuration is set to 'off', apply default value

10.18.16 r_glcd_get_bit_size

```
r_glcd_get_bit_size ( display_in_format_t  const format )
```

10.18.16.1 Brief description

Subroutine to get the bit size of the specified format.

10.18.16.2 Detailed description

Table 1304:

Name	Direction	Description
format	in	Color format (specify display_in_format_t type enumeration value)

Table 1305:

Name	Description
Bit	size

10.18.16.3 Function steps

- Get bit size and set color format
- < ARGB8888, 32bits
- < RGB888, 32bits
- < RGB565, 16bits
- < ARGB1555, 16bits
- < ARGB4444, 16bits
- < CLUT8
- < CLUT4
- < CLUT1

10.18.17 r_glcd_graphics_plane_format_set

```
r_glcd_graphics_plane_format_set (  R_GLCDC_Type * p_glcd_reg ,
display_in_format_t  const format ,  display_frame_layer_t  const frame )
```

10.18.17.1 Brief description

Subroutine to set the color format of graphics plane to the GLCD register.

10.18.17.2 Detailed description

Table 1306:

Name	Direction	Description
format	in	Color format (specify display_in_format_t type enumeration value)
frame	in	The number of input frame buffer
p_glcd_reg	in	Pointer to GLCD registers

Table 1307:

Name	Description
void	

10.18.17.3 Function steps

- < ARGB8888, 32bits
- < RGB888, 32bits
- < RGB565, 16bits
- < ARGB1555, 16bits
- < ARGB4444, 16bits
- < CLUT8
- < CLUT4
- < CLUT1

10.18.18 r_glcd_interrupt_enable

```
 r_glcd_interrupt_enable (  R_GLCDC_Type * p_glcd_reg ,   IRQn_Type
* line_detect_irq ,   IRQn_Type * underflow_1_irq ,   IRQn_Type
* underflow_2_irq )
```

10.18.18.1 Brief description

Enable the glcd interrupt.

10.18.18.2 Detailed description

Table 1308:

Name	Direction	Description
p_glcd_reg	in	Pointer to GLCD registers
line_detect_irq	in	Pointer to IRQ
underflow_1_irq	in	Pointer to IRQ
underflow_2_irq	in	Pointer to IRQ

Table 1309:

Name	Description
void	

10.18.18.3 Function steps

- Enable the GLCD interrupts

10.18.19 r_glcd_output_data_order_set

```
 r_glcd_output_data_order_set (  R_GLCDC_Type * p_glcd_reg ,  display_cfg_t
const *const p_cfg )
```

10.18.19.1 Brief description

Configure endianness for output data and output byte order swapping.

10.18.19.2 Detailed description

Table 1310:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for the display interface

Table 1310: (Continued)

Name	Direction	Description
p_glcd_reg	in	Pointer to GLCD registers

Table 1311:

Name	Description
void	

10.18.19.3 Function steps

- selects big or little endian for output data
- selects the output byte order swapping

10.18.20 r_glcd_pixel_size_recalculate

```
 r_glcd_pixel_size_recalculate ( display_input_cfg_t  const *const p_input ,
 display_layer_t  const *const p_layer ,   recalculated_param_t
* p_recalculated ,   uint16_t * bit_size )
```

10.18.20.1 Brief description

Calculate the pixels to be displayed on window based on the offset of the graphic layer.

10.18.20.2 Detailed description

Table 1312:

Name	Direction	Description
p_input	in	The input frame buffer configuration
p_layer	in	The layer configuration
p_recalculated	inout	Pointer to store recalculated parameter
bit_size	in	Pointer to bit size of the color format

Table 1313:

Name	Description
void	

10.18.20.3 Function steps

- If the offset of the graphics layer is greater than or equal to zero and it is less than or equal to the size of the display window, calculate actual pixel size to display.
- Calculate actual pixel size, the pixels to be displayed is less than the display window size
- Actual pixel size to display is same as the display window size
- If the offset of the graphics layer is less than zero, calculate the actual pixel size to display.
- If coordinate.x is a minus value, the layer image position can be adjusted not only by adjusting the horizontal offset but also by changing the base address of layer image. Since the base address has to be aligned to 64 bytes, we need to adjust the horizontal offset, which is the cycles from the 'internal zero' to the start cycle of active video region to achieve 1 pixel unit offset. We need to adjust the size of image to display as well.
- Base address must be aligned to a 64-bit address
- If graphics layer offset is beyond the display window size, set the pixel size to display to zero

10.18.21 glcdc_line_detect_isr

```
glcdc_line_detect_isr (  void  )
```

10.18.21.1 Brief description

The line detection interrupt service routine. This ISR is called when the number of the display line reaches the designated number of lines. If a callback function is registered in R_GLCD_Open, it is called from this ISR and the DISPLAY_EVENT_LINE_DETECTION event code is set as its argument.

10.18.21.2 Detailed description

Table 1314:

Name	Description
none	

10.18.21.3 Function steps

- Call back callback function if it is registered

- Clear interrupt flag in the register of the GLCD module
- Clear interrupt flag in the register of the NVIC module

10.18.22 glcdc_underflow_1_isr

```
glcdc_underflow_1_isr (  void  )
```

10.18.22.1 Brief description

The graphics plane 1 underflow detection interrupt service routine. This ISR is called when the underflow occurs in the graphics plane 1 control block. If a callback function is registered in R_GLCD_Open, it is called back from this ISR and the DISPLAY_EVENT_GR1_UNDERFLOW event code is set as its argument.

10.18.22.2 Detailed description

Table 1315:

Name	Description
none	

10.18.22.3 Function steps

- Call back callback function if it is registered
- Clear interrupt flag in the register of the GLCD module
- Clear interrupt flag in the register of the NVIC module

10.18.23 glcdc_underflow_2_isr

```
glcdc_underflow_2_isr (  void  )
```

10.18.23.1 Brief description

The graphics plane 2 underflow detection interrupt service routine. This ISR is called when the underflow occurs in the graphics plane 2 control block. If a callback function is registered in R_GLCD_Open, it is called from this ISR and the DISPLAY_EVENT_GR2_UNDERFLOW event code is set as its argument.

10.18.23.2 Detailed description

Table 1316:

Name	Description
none	

10.18.23.3 Function steps

- Call the callback function if it is registered
- Clear interrupt flag in the register of the GLCD module
- Clear interrupt flag in the register of the NVIC module

10.18.24 R_GLCD_Open

```
ssp_err_t R_GLCD_Open ( display_ctrl_t  *const p_api_ctrl ,  display_cfg_t  const
*const p_cfg )
```

10.18.24.1 Brief description

Open GLCDC module.

10.18.24.2 Detailed description

Implements

- open.

Table 1317:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_HW_LOCKED	GLCDC resource is locked.
SSP_ERR_CLOCK_GENERATION	Dot clock cannot be generated from clock source.

Table 1317: (Continued)

Name	Description
SSP_ERR_INVALID_TIMING_SETTING	Invalid panel timing parameter.
SSP_ERR_INVALID_LAYER_SETTING	Invalid layer setting found.
SSP_ERR_INVALID_LAYER_FORMAT	Invalid format is specified.
SSP_ERR_INVALID_GAMMA_SETTING	Invalid gamma correction setting found.

NOTE: PCLKA must be supplied to Graphics LCD Controller (GLCDC) and GLCDC pins must be set in IOPORT before calling this API.

10.18.24.3 Function steps

- Lock the GLCD resource
- Supply the peripheral clock to the GLCD module
- Release GLCD from a SW reset status.
- Set the dot clock frequency
- Set the panel signal timing
- Configure the background screen
- Store back poach position to the control block (needed to define the layer blending position later)
- Configure the graphics plane layers
- Configure the output control block
- Configure the color correction setting (brightness, brightness and gamma correction)
- Change GLCD driver state
- Save callback function
- Save user defined context
- Save the display interface context into GLCD HAL control block
- Set the line number which is suppose to happen the line detect interrupt

10.18.25 R_GLCD_Close

```
ssp_err_t R_GLCD_Close ( display_ctrl_t  *const p_api_ctrl )
```

10.18.25.1 Brief description

Close GLCDC module.

10.18.25.2 Detailed description

Implements

- close.

Table 1318:

Name	Description
SSP_SUCCESS	Device was closed successfully.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_NOT_OPEN	The function call is performed when the driver state is not equal to DISPLAY_STATE_CLOSED.
SSP_ERR_INVALID_UPDATE_TIMING	A function call is performed when the GLCD is updating register values internally.

NOTE: This API can be called when the driver is not in DISPLAY_STATE_CLOSED state. It returns an error if the register update operation for the background screen generation block is being held.

10.18.25.3 Function steps

- Disable the GLCD interrupts
- Reset the GLCD hardware
- Halt the peripheral clock to the GLCD module
- Unlock the GLCD resource

10.18.26 R_GLCD_Start

```
ssp_err_t R_GLCD_Start ( display_ctrl_t  *const p_api_ctrl )
```

10.18.26.1 Brief description

Start GLCDC module.

10.18.26.2 Detailed description

Implements

- start.

Table 1319:

Name	Description
SSP_SUCCESS	Device was started successfully.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_INVALID_MODE	Function call is performed when the driver state is not DISPLAY_STATE_OPENED.

NOTE: This API can be called when the driver is not in DISPLAY_STATE_OPENED status.

10.18.26.3 Function steps

- Start to output the vertical and horizontal synchronization signals and screen data.

10.18.27 R_GLCD_Stop

```
ssp_err_t R_GLCD_Stop ( display_ctrl_t  *const p_api_ctrl )
```

10.18.27.1 Brief description

Stop GLCDC module.

10.18.27.2 Detailed description

Implements

- stop.

Table 1320:

Name	Description
SSP_SUCCESS	Device was stopped successfully
SSP_ERR_ASSERTION	Pointer to the control block is NULL
SSP_ERR_INVALID_MODE	Function call is performed when the driver state is not DISPLAY_STATE_DISPLAYING.
SSP_ERR_INVALID_UPDATE_TIMING	The function call is performed while the GLCD is updating register values internally.

NOTE: This API can be called when the driver is in the DISPLAY_STATE_DISPLAYING state. It returns an error if the register update operation for the background screen generation blocks, the graphics data I/F blocks, or the output control block is being held.

10.18.27.3 Function steps

- Stop outputting the vertical and horizontal synchronization signals and screen data.

10.18.28 R_GLCD_LayerChange

```
ssp_err_t R_GLCD_LayerChange ( display_ctrl_t  const *const p_api_ctrl ,
 display_runtime_cfg_t  const *const p_cfg ,  display_frame_layer_t  frame )
```

10.18.28.1 Brief description

Change layer parameters of GLCDC module at runtime.

10.18.28.2 Detailed description

Implements

- layerChange.

Table 1321:

Name	Description
SSP_SUCCESS	Changed layer parameters of GLCDC module successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_MODE	A function call is performed when the driver state is not DISPLAY_STATE_DISPLAYING.
SSP_ERR_INVALID_ARGUMENT	An invalid parameter is found in the argument.
SSP_ERR_INVALID_UPDATE_TIMING	A function call is performed while the GLCD is updating register values internally.

NOTE: This API can be called when the driver is in DISPLAY_STATE_DISPLAYING state. It returns an error if the register update operation for the background screen generation blocks or the graphics data I/F block is being held.

10.18.28.3 Function steps

- Configure the graphics plane layers

- Reflect the graphics module register value to the GLCD internal operations (at the timing of the next Vsync assertion)

10.18.29 R_GLCD_ColorCorrection

```
ssp_err_t R_GLCD_ColorCorrection ( display_ctrl_t  const *const p_api_ctrl ,
 display_correction_t  const *const p_correction )
```

10.18.29.1 Brief description

Perform color correction by GLCDC module.

10.18.29.2 Detailed description

Implements

- correction.

Table 1322:

Name	Description
SSP_SUCCESS	Color correction by GLCDC module was performed successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the display correction structure is NULL.
SSP_ERR_INVALID_MODE	Function call is performed when the driver state is not DISPLAY_STATE_DISPLAYING.
SSP_ERR_INVALID_UPDATE_TIMING	A function call is performed while the GLCDC is updating registers internally.

NOTE: This API can be called when the driver is in the DISPLAY_STATE_DISPLAYING state. It returns an error if the register update operation for the background screen generation blocks or the output control block is being held.

10.18.29.3 Function steps

- Configure the brightness and contrast correction register setting.
- Update the Output block register setting.

10.18.30 R_GLCD_ClutUpdate

```
ssp_err_t R_GLCD_ClutUpdate ( display_ctrl_t  const *const p_api_ctrl ,
 display_clut_cfg_t  const *const p_clut_cfg ,  display_frame_layer_t  frame )
```

10.18.30.1 Brief description

Update Color Look Up Table of GLCDC module.

10.18.30.2 Detailed description

Implements

- clut.

Table 1323:

Name	Description
SSP_SUCCESS	CLUT updated successfully.
SSP_ERR_ASSERTION	Pointer to the control block or CLUT source data is NULL.
SSP_ERR_INVALID_CLUT_ACCESS	Illegal CLUT entry or size is specified.

NOTE: This API can be called any time.

10.18.30.3 Function steps

- Check the CLUT table current used
- Copy the new CLUT data on the source memory to the CLUT SRAM in the GLCD module
- Make the GLCD module read the new CLUT table data from the next frame

10.18.31 R_GLCD_StatusGet

```
ssp_err_t R_GLCD_StatusGet ( display_ctrl_t  const *const p_api_ctrl ,
 display_status_t  *const p_status )
```

10.18.31.1 Brief description

Get status of GLCDC module.

10.18.31.2 Detailed description

Implements

- statusGet.

Table 1324:

Name	Description
SSP_SUCCESS	Got status successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the status structure is NULL.

NOTE: The GLCD hardware starts the fading processing at the first Vsync after the previous LayerChange() call is held. Due to this behavior of the hardware, this API may not return DISPLAY_FADE_STATUS_FADING_UNDERWAY as the fading status, if it is called before the first Vsync after LayerChange() is called. In this case, the API returns DISPLAY_FADE_STATUS_UNCERTAIN, instead of DISPLAY_FADE_STATUS_NOT_UNDERWAY.

10.18.31.3 Function steps

- Return the GLCD HAL driver state
- Return the fading status for the layers

10.18.32 R_GLCD_VersionGet

```
ssp_err_t R_GLCD_VersionGet ( ssp_version_t  * p_version )
```

10.18.32.1 Brief description

Get version of R_GLCDC module.

10.18.32.2 Detailed description

Implements

- versionGet.

Table 1325:

Name	Description
p_version.	The version number.

NOTE: This function is re-entrant.

10.18.33 g_display_on_glcd

```
display_api_t::g_display_on_glcd
```

10.18.33.1 Detailed description

GLCD HAL module API function pointer list

10.18.33.2 Initialized as

```
g_display_on_glcd=
{
    .open        = R_GLCD_Open,
    .close       = R_GLCD_Close,
    .start       = R_GLCD_Start,
    .stop        = R_GLCD_Stop,
    .layerChange = R_GLCD_LayerChange,
    .clut        = R_GLCD_ClutUpdate,
    .correction  = R_GLCD_ColorCorrection,
    .statusGet   = R_GLCD_StatusGet,
    .versionGet  = R_GLCD_VersionGet
}
```

10.18.34 API Data

10.18.34.1 glcd_clk_src_t

`glcd_clk_src_t`

Detailed description

Clock source select

Enumerated values

Name	Description
GLCD_CLK_SRC_INTERNAL	Internal.
GLCD_CLK_SRC_EXTERNAL	External.

10.18.34.2 glcd_panel_clk_div_t

`glcd_panel_clk_div_t`

Detailed description

Clock frequency division ratio

Enumerated values

Name	Description
GLCD_PANEL_CLK_DIVISOR_1	Division Ratio 1/1.
GLCD_PANEL_CLK_DIVISOR_2	Division Ratio 1/2.
GLCD_PANEL_CLK_DIVISOR_3	Division Ratio 1/3.
GLCD_PANEL_CLK_DIVISOR_4	Division Ratio 1/4.
GLCD_PANEL_CLK_DIVISOR_5	Division Ratio 1/5.
GLCD_PANEL_CLK_DIVISOR_6	Division Ratio 1/6.
GLCD_PANEL_CLK_DIVISOR_7	Division Ratio 1/7.
GLCD_PANEL_CLK_DIVISOR_8	Division Ratio 1/8.
GLCD_PANEL_CLK_DIVISOR_9	Division Ratio 1/9.
GLCD_PANEL_CLK_DIVISOR_12	Division Ratio 1/12.
GLCD_PANEL_CLK_DIVISOR_16	Division Ratio 1/16.
GLCD_PANEL_CLK_DIVISOR_24	Division Ratio 1/24.
GLCD_PANEL_CLK_DIVISOR_32	Division Ratio 1/32.

10.18.34.3 glcd_tcon_pin_t

`glcd_tcon_pin_t`

Detailed description

LCD TCON output pin select

Enumerated values

Name	Description
GLCD_TCON_PIN_NONE	No output.
GLCD_TCON_PIN_0	LCD_TCON0.
GLCD_TCON_PIN_1	LCD_TCON1.
GLCD_TCON_PIN_2	LCD_TCON2.

Name	Description
GLCD_TCON_PIN_3	LCD_TCON3.
GLCD_TCON_PIN_NUM	

10.18.34.4 glcd_bus_arbitration_t

`glcd_bus_arbitration_t`

Detailed description

Bus Arbitration setting

Enumerated values

Name	Description
GLCD_BUS_ARBITRATION_ROUNDROBIN	Round robin.
GLCD_BUS_ARBITRATION_FIX_PRIORITY	Fixed.

10.18.34.5 glcd_correction_proc_order_t

`glcd_correction_proc_order_t`

Detailed description

Correction circuit sequence control

Enumerated values

Name	Description
GLCD_CORRECTION_PROC_ORDER_BRIGHTNESS_CONTRAST2GAMMA	Brightness -> contrast -> gamma correction.
GLCD_CORRECTION_PROC_ORDER_GAMMA2BRIGHTNESS_CONTRAST	Gamma correction -> brightness -> contrast.

10.18.34.6 glcd_tcon_signal_select_t

`glcd_tcon_signal_select_t`

Detailed description

Timing signals for driving the LCD panel

Enumerated values

Name	Description
GLCD_TCON_SIGNAL_SELECT_STVA_VS	STVA/VS.
GLCD_TCON_SIGNAL_SELECT_STVB_VE	STVB/VE.
GLCD_TCON_SIGNAL_SELECT_STHA_HS	STH/SP/HS.
GLCD_TCON_SIGNAL_SELECT_STHB_HE	STB/LP/HE.
GLCD_TCON_SIGNAL_SELECT_DE	DE.

10.18.34.7 glcd_clut_plane_t

`glcd_clut_plane_t`

Detailed description

Clock phase adjustment for serial RGB output

Enumerated values

Name	Description
GLCD_CLUT_PLANE_0	GLCD CLUT plane 0.
GLCD_CLUT_PLANE_1	GLCD CLUT plane 1.

10.18.34.8 glcd_dithering_mode_t

`glcd_dithering_mode_t`

Detailed description

Dithering mode

Enumerated values

Name	Description
GLCD_DITHERING_MODE_TRUNCATE	No dithering (truncate)
GLCD_DITHERING_MODE_ROUND_OFF	Dithering with round off.
GLCD_DITHERING_MODE_2X2PATTERN	Dithering with 2x2 pattern.
GLCD_DITHERING_MODE_SETTING_MAX	

10.18.34.9 glcd_dithering_pattern_t

`glcd_dithering_pattern_t`

Detailed description

Dithering mode

Enumerated values

Name	Description
GLCD_DITHERING_PATTERN_00	2x2 pattern '00'
GLCD_DITHERING_PATTERN_01	2x2 pattern '01'
GLCD_DITHERING_PATTERN_10	2x2 pattern '10'
GLCD_DITHERING_PATTERN_11	2x2 pattern '11'

10.18.34.10 glcd_input_interface_format_t

`glcd_input_interface_format_t`

Detailed description

Output interface format

Enumerated values

Name	Description
GLCD_INPUT_INTERFACE_FORMAT_RGB565	Input interface format RGB565.
GLCD_INPUT_INTERFACE_FORMAT_RGB888	Input interface format RGB888.
GLCD_INPUT_INTERFACE_FORMAT_ARGB1555	Input interface format ARGB1555.
GLCD_INPUT_INTERFACE_FORMAT_ARGB4444	Input interface format ARGB4444.
GLCD_INPUT_INTERFACE_FORMAT_ARGB8888	Input interface format ARGB8888.
GLCD_INPUT_INTERFACE_FORMAT_CLUT8	Input interface format CLUT8.
GLCD_INPUT_INTERFACE_FORMAT_CLUT4	Input interface format CLUT4.
GLCD_INPUT_INTERFACE_FORMAT_CLUT1	Input interface format CLUT1.

10.18.34.11 glcd_output_interface_format_t

`glcd_output_interface_format_t`

Detailed description

Output interface format

Enumerated values

Name	Description
GLCD_OUTPUT_INTERFACE_FORMAT_RGB888	Output interface format RGB888.
GLCD_OUTPUT_INTERFACE_FORMAT_RGB666	Output interface format RGB666.
GLCD_OUTPUT_INTERFACE_FORMAT_RGB565	Output interface format RGB565.
GLCD_OUTPUT_INTERFACE_FORMAT_SERIAL_RGB	Output interface format Serial RGB.

10.18.34.12 glcd_dithering_output_format_t

`glcd_dithering_output_format_t`

Detailed description

Dithering output format

Enumerated values

Name	Description
GLCD_DITHERING_OUTPUT_FORMAT_RGB888	Dithering output format RGB888.
GLCD_DITHERING_OUTPUT_FORMAT_RGB666	Dithering output format RGB666.
GLCD_DITHERING_OUTPUT_FORMAT_RGB565	Dithering output format RGB565.

10.18.34.13 display_plane_blend_t

`display_plane_blend_t`

Detailed description

RGB color order select

Enumerated values

Name	Description
DISPLAY_PLANE_BLEND_TRANSPARENT	Current graphics layer is transparent and the lower layer is.
DISPLAY_PLANE_BLEND_NON_TRANSPARENT	Current graphics layer is displayed.
DISPLAY_PLANE_BLEND_ON_LOWER_LAYER	Current graphics layer is blended with the lower layer.

10.18.34.14 glcd_fading_control_initial_alpha_t

`glcd_fading_control_initial_alpha_t`

Detailed description

RGB color order select

Enumerated values

Name	Description
GLCD_FADING_CONTROL_INITIAL_ALPHA_MIN	Initial alpha value setting for a graphics plane is zero.
GLCD_FADING_CONTROL_INITIAL_ALPHA_MAX	Initial alpha value setting for a graphics plane is maximum.

10.18.35 Extensions

10.18.35.1 glcd_instance_ctrl_t

glcd_instance_ctrl_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Display control block. DO NOT INITIALIZE.

Variables

- display_state_t state

  Status of GLCD module.

- void(* p_callback)(*p_args)

  Pointer to callback function.

- void const * p_context

  Pointer to the higher level device context.

- R_GLCDC_Type * p_reg

  Base register address.

10.18.35.2 glcd_cfg_t

glcd_cfg_t

Detailed description

GLCD hardware specific configuration

Variables

- glcd_tcon_pin_t tcon_hsync

  GLCD TCON output pin select.
- glcd_tcon_pin_t tcon_vsync

  GLCD TCON output pin select.
- glcd_tcon_pin_t tcon_de

  GLCD TCON output pin select.
- glcd_correction_proc_order_t correction_proc_order

  Correction control route select.
- glcd_clk_src_t clksrc

  Clock Source selection.
- glcd_panel_clk_div_t clock_div_ratio

  Clock divide ratio for dot clock.
- glcd_dithering_mode_t dithering_mode

  Dithering mode.
- glcd_dithering_pattern_t dithering_pattern_A

  Dithering pattern A.
- glcd_dithering_pattern_t dithering_pattern_B

  Dithering pattern B.
- glcd_dithering_pattern_t dithering_pattern_C

  Dithering pattern C.
- glcd_dithering_pattern_t dithering_pattern_D

  Dithering pattern D.

10.18.35.3 glcd_ctrl_t

glcd_ctrl_t

Detailed description

GLCD hardware specific control block

Variables

- display_coordinate_t back_porch

  Zero coordinate for graphics plane(Bach porch End)
- uint16_t hsize

  Horizontal pixel size in a line.
- uint16_t vsize

  Vertical pixel size in a frame.
- bsp_lock_t resource_lock

  Resource lock.
- void * p_context

  Pointer to the function level device context (e.g. display_ctrl_t type data)

10.18.35.4 tcon_func_t

tcon_func_t

Detailed description

Function pointers to control TCON pin settings

Variables

- void(* tcon_select)(R_GLCDC_Type *p_glcd_reg,)
- void(* invert)(R_GLCDC_Type *p_glcd_reg)

10.18.35.5 recalculated_param_t

recalculated_param_t

Detailed description

The structure for the layer parameter recalculation

Variables

- uint16_t hpix_size
- uint16_t vpix_size
- int16_t hpix_offset
- int16_t vpix_offset
- uint32_t hread_size
- uint32_t base_address

10.19 GPT

Driver for the General PWM Timer (GPT).

10.19.1 Summary

Extends Timer Interface.

This module implements the Timer Interface using the General PWM Timer (GPT) peripherals GPT32EH, GPT32E, GPT32. It also provides an output compare extension to output the timer signal to the GTIOC pin.

10.19.2 Functions

- R_GPT_TimerOpen
- R_GPT_Stop
- R_GPT_Start
- R_GPT_CounterGet
- R_GPT_Reset
- R_GPT_PeriodSet
- R_GPT_DutyCycleSet
- R_GPT_InfoGet
- R_GPT_Close
- R_GPT_VersionGet

10.19.3 Defines

- #define GPT_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define GPT_CODE_VERSION_MINOR

  Initial value:`(6U)`

10.19.4 R_GPT_TimerOpen

```
ssp_err_t R_GPT_TimerOpen ( timer_ctrl_t *const p_api_ctrl , timer_cfg_t const
*const p_cfg )
```

10.19.4.1 Brief description

Powers on GPT, handles required initialization described in hardware manual. Implements open.

10.19.4.2 Detailed description

The Open function configures a single GPT channel, starts the channel, and provides a handle for use with the GPT API Control and Close functions. This function must be called once prior to calling any other GPT API functions. After a channel is opened, the Open function should not be called again for the same channel without first calling the associated Close function.

GPT hardware does not support one-shot functionality natively. When using one-shot mode, the timer will be stopped in an ISR after the requested period has elapsed.

The GPT implementation of the general timer can accept a timer_on_gpt_cfg_t extension parameter.

Table 1326:

Name	Description
SSP_SUCCESS	Initialization was successful and timer has started.
SSP_ERR_ASSERTION	One of the following parameters is incorrect. Either • p_cfg is NULL, OR • p_ctrl is NULL, OR
SSP_ERR_INVALID_ARGUMENT	One of the following parameters is invalid: • p_cfg->period: must be in the following range: – Lower bound: (1 / (PCLK frequency) – Upper bound: (0xFFFFFFFF * 1024 / (PCLK frequency)) • p_cfg->p_callback not NULL, but ISR is not enabled. ISR must be enabled to use callback function. Enable channel's overflow ISR in bsp_irq_cfg.h.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.
SSP_ERR_IRQ_BSP_DISABLED	- p_cfg->mode is TIMER_MODE_ONE_SHOT, but ISR is not enabled. ISR must be enabled to use one-shot mode.
SSP_ERR_IP_CHANNEL_NOT_PRESENT	- The channel requested in the p_cfg parameter is not available on this device.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.19.4.3 Function steps

- Calculate period and store internal variables

- Calculate duty cycle
- Verify channel is not already used
- Power on GPT before setting any hardware registers. Make sure the counter is stopped before setting mode register, PCLK divisor register, and counter register.

10.19.5 R_GPT_Stop

```
ssp_err_t R_GPT_Stop ( timer_ctrl_t  *const p_api_ctrl )
```

10.19.5.1 Brief description

Stops timer. Implements stop.

10.19.5.2 Detailed description

Table 1327:

Name	Description
SSP_SUCCESS	Timer successfully stopped.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.5.3 Function steps

- Stop timer

10.19.6 R_GPT_Start

```
ssp_err_t R_GPT_Start ( timer_ctrl_t  *const p_api_ctrl )
```

10.19.6.1 Brief description

Starts timer. Implements start.

10.19.6.2 Detailed description

Table 1328:

Name	Description
SSP_SUCCESS	Timer successfully started.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.6.3 Function steps

- Start timer

10.19.7 R_GPT_CounterGet

```
ssp_err_t R_GPT_CounterGet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
*const p_value )
```

10.19.7.1 Brief description

Sets counter value in provided p_value pointer. Implements counterGet.

10.19.7.2 Detailed description

Table 1329:

Name	Description
SSP_SUCCESS	Counter value read, p_value is valid.
SSP_ERR_ASSERTION	The p_ctrl or p_value parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.7.3 Function steps

- Read counter value

10.19.8 R_GPT_Reset

```
ssp_err_t R_GPT_Reset ( timer_ctrl_t  *const p_api_ctrl )
```

10.19.8.1 Brief description

Resets the counter value to 0. Implements reset.

10.19.8.2 Detailed description

Table 1330:

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.8.3 Function steps

- Write the counter value

10.19.9 R_GPT_PeriodSet

```
ssp_err_t R_GPT_PeriodSet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
const period ,  timer_unit_t  const unit )
```

10.19.9.1 Brief description

Sets period value provided. Implements periodSet.

10.19.9.2 Detailed description

Table 1331:

Name	Description
SSP_SUCCESS	Period value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.

Table 1331: (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	One of the following is invalid: • p_period->unit: must be one of the options from timer_unit_t • p_period->value: must result in a period in the following range: – Lower bound: (1 / (PCLK frequency)) – Upper bound: (0xFFFFFFFF * 1024 / (PCLK frequency))
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.9.3 Function steps

- Delay must be converted to PCLK counts before it can be set in registers
- Make sure period is valid.
- Store current status, then stop timer before setting divisor register
- Reset counter in case new cycle is less than current count value, then restore state (counting or stopped).

10.19.10 R_GPT_DutyCycleSet

```
ssp_err_t R_GPT_DutyCycleSet ( timer_ctrl_t  *const p_api_ctrl ,  timer_size_t
const duty_cycle ,  timer_pwm_unit_t  const unit ,   uint8_t const pin )
```

10.19.10.1 Brief description

Sets status in provided p_status pointer. Implements pwm_api_t::dutyCycleSet.

10.19.10.2 Detailed description

Table 1332:

Name	Description
SSP_SUCCESS	Counter value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.10.3 Function steps

- Converted duty cycle to PCLK counts before it can be set in registers
- Set duty cycle.

10.19.11 R_GPT_InfoGet

```
ssp_err_t R_GPT_InfoGet ( timer_ctrl_t  *const p_api_ctrl ,  timer_info_t
*const p_info )
```

10.19.11.1 Brief description

Get timer information and store it in provided pointer p_info. Implements infoGet.

10.19.11.2 Detailed description

Table 1333:

Name	Description
SSP_SUCCESS	Period, count direction, frequency, and status value written to caller's structure successfully.
SSP_ERR_ASSERTION	The p_ctrl or p_info parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.11.3 Function steps

- Get and store period
- Get and store clock frequency
- Get and store clock counting direction

10.19.12 R_GPT_Close

```
ssp_err_t R_GPT_Close ( timer_ctrl_t  *const p_api_ctrl )
```

10.19.12.1 Brief description

Stops counter, disables interrupts, disables output pins, and clears internal driver data.

10.19.12.2 Detailed description

Table 1334:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.19.12.3 Function steps

- Cleanup. Disable interrupts, stop counter, and disable output.
- Unlock channel
- Clear stored internal driver data

10.19.13 R_GPT_VersionGet

```
ssp_err_t R_GPT_VersionGet ( ssp_version_t  *const p_version )
```

10.19.13.1 Brief description

Sets driver version based on compile time macros.

10.19.13.2 Detailed description

Table 1335:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.19.14 API Data

10.19.14.1 gpt_pin_level_t

```
gpt_pin_level_t
```

Detailed description

Level of GPT pin

Enumerated values

Name	Description
GPT_PIN_LEVEL_LOW	Pin level low.
GPT_PIN_LEVEL_HIGH	Pin level high.
GPT_PIN_LEVEL_RETAINED	Pin level retained.

10.19.14.2 gpt_trigger_t

`gpt_trigger_t`

Detailed description

Sources can be used to start the timer, stop the timer, count up, or count down.

Enumerated values

Name	Description
GPT_TRIGGER_NONE	No action performed.
GPT_TRIGGER_GTIOCA_RISING_WHILE_GTIOCB_LOW	Action performed when GTIOCA input rises while GTIOCB is low.
GPT_TRIGGER_GTIOCA_RISING_WHILE_GTIOCB_HIGH	Action performed when GTIOCA input rises while GTIOCB is high.
GPT_TRIGGER_GTIOCA_FALLING_WHILE_GTIOCB_LOW	Action performed when GTIOCA input falls while GTIOCB is low.
GPT_TRIGGER_GTIOCA_FALLING_WHILE_GTIOCB_HIGH	Action performed when GTIOCA input falls while GTIOCB is high.
GPT_TRIGGER_GTIOCB_RISING_WHILE_GTIOCA_LOW	Action performed when GTIOCB input rises while GTIOCA is low.
GPT_TRIGGER_GTIOCB_RISING_WHILE_GTIOCA_HIGH	Action performed when GTIOCB input rises while GTIOCA is high.
GPT_TRIGGER_GTIOCB_FALLING_WHILE_GTIOCA_LOW	Action performed when GTIOCB input falls while GTIOCA is low.

Name	Description
GPT_TRIGGER_GTIOCB_FALLING_WHILE_GTIOCA_HIGH	Action performed when GTIOCB input falls while GTIOCA is high.
GPT_TRIGGER_SOURCE_REGISTER_ENABLE	Enables settings in the Source Select Register.

10.19.14.3 gpt_output_t

`gpt_output_t`

Detailed description

Output level used when selecting what happens at compare match or cycle end.

Enumerated values

Name	Description
GPT_OUTPUT_RETAINED	Output retained.
GPT_OUTPUT_LOW	Output low.
GPT_OUTPUT_HIGH	Output high.
GPT_OUTPUT_TOGGLED	Output toggled.

10.19.15 Extensions

10.19.15.1 gpt_instance_ctrl_t

gpt_instance_ctrl_t

Detailed description

Channel control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- void(* p_callback)(*p_args)

  Callback provided when a timer ISR occurs. NULL indicates no CPU interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user callback in timer_callback_args_t.
- void * p_reg

  Base register for this channel.

- uint32_t open

  Whether or not channel is open.
- uint8_t channel

  Channel number.
- bool one_shot

  Whether or not timer is in one shot mode.
- IRQn_Type irq

  Counter overflow IRQ number.
- timer_variant_t variant

  Timer variant.

10.19.15.2 gpt_output_pin_t

gpt_output_pin_t

Detailed description

Configurations for output pins.

Variables

- bool output_enabled

  Set to true to enable output, false to disable output.
- gpt_pin_level_t stop_level

  Select a stop level from gpt_pin_level_t.

10.19.15.3 timer_on_gpt_cfg_t

timer_on_gpt_cfg_t

Detailed description

GPT extension configures the output pins for GPT.

Variables

- gpt_output_pin_t gtioca

  Configuration for GPT I/O pin A.
- gpt_output_pin_t gtiocb

  Configuration for GPT I/O pin B.

10.20 GPT Input Capture

Driver for the General PWM Timer (GPT) with Input Capture.

10.20.1 Summary

Extends Input Capture Interface.

This module implements the Input Capture Interface for the General PWM Timer (GPT) peripherals GPT32EH, GPT32E, GPT32.

10.20.2 Functions

- R_GPT_InputCaptureOpen
- R_GPT_InputCaptureClose
- R_GPT_InputCaptureVersionGet
- R_GPT_InputCaptureDisable
- R_GPT_InputCaptureEnable
- R_GPT_InputCaptureInfoGet
- R_GPT_InputCaptureLastCaptureGet

10.20.3 Defines

- #define GPT_INPUT_CAPTURE_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  IncludesCommon macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define GPT_INPUT_CAPTURE_CODE_VERSION_MINOR

  Initial value:`(6U)`
- #define GPT_INPUT_CAPTURE_MAX_COUNT

  Initial value:`(0xFFFFFFFFUL)`

  Maximum value of GPT counter.

10.20.4 R_GPT_InputCaptureOpen

```
ssp_err_t R_GPT_InputCaptureOpen ( input_capture_ctrl_t  *const p_api_ctrl ,
 input_capture_cfg_t  const *const p_cfg )
```

10.20.4.1 Brief description

Open a GPT Timer for Input Capture. Implements open.

10.20.4.2 Detailed description

The Open function configures a single GPT channel for input capture and provides a handle for use with the other Input Capture API functions. This function must be called once prior to calling any other Input Capture API function. After a channel is opened, the Open function should not be called again for the same channel without first calling the associated Close function.

Table 1336:

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the parameters is NULL: p_cfg, p_ctrl, p_extend. Or the channel requested in the p_cfg parameter may not be available on the device selected in r_bsp_cfg.h. Or p_cfg->mode is invalid.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.20.4.3 Function steps

- Verify channel is not already used

10.20.5 R_GPT_InputCaptureClose

```
ssp_err_t R_GPT_InputCaptureClose ( input_capture_ctrl_t  *const p_api_ctrl )
```

10.20.5.1 Brief description

Close a GPT Timer Channel for Input Capture. Implements close.

10.20.5.2 Detailed description

Clears Timer settings, disables interrupts, and clears internal driver data.

Table 1337:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.20.5.3 Function steps

- Cleanup. Disable interrupts and stop measurements.
- Unlock channel
- Clear stored internal driver data

10.20.6 R_GPT_InputCaptureVersionGet

```
ssp_err_t R_GPT_InputCaptureVersionGet ( ssp_version_t  *const p_version )
```

10.20.6.1 Brief description

Gets driver version based on compile time macros. Implements versionGet.

10.20.6.2 Detailed description

Table 1338:

Name	Description
SSP_SUCCESS	Success.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.20.7 R_GPT_InputCaptureDisable

```
ssp_err_t R_GPT_InputCaptureDisable ( input_capture_ctrl_t  const
*const p_api_ctrl )
```

10.20.7.1 Brief description

Disables GPT Input Capture RegA interrupt for specified channel at NVIC. Implements disable.

10.20.7.2 Detailed description

Table 1339:

Name	Description
SSP_SUCCESS	Interrupt disabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.20.7.3 Function steps

- Disable interrupts

10.20.8 R_GPT_InputCaptureEnable

```
ssp_err_t R_GPT_InputCaptureEnable ( input_capture_ctrl_t  const
*const p_api_ctrl )
```

10.20.8.1 Brief description

Enables GPT Input Capture RegA interrupt for specified channel at NVIC. Implements enable.

10.20.8.2 Detailed description

Table 1340:

Name	Description
SSP_SUCCESS	Interrupt enabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.20.9 R_GPT_InputCaptureInfoGet

```
ssp_err_t R_GPT_InputCaptureInfoGet ( input_capture_ctrl_t  const
*const p_api_ctrl ,  input_capture_info_t  *const p_info )
```

10.20.9.1 Brief description

Gets status into provided p_info pointer. Implements infoGet.

10.20.9.2 Detailed description

Table 1341:

Name	Description
SSP_SUCCESS	Success.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.20.10 R_GPT_InputCaptureLastCaptureGet

```
ssp_err_t R_GPT_InputCaptureLastCaptureGet ( input_capture_ctrl_t  const
*const p_api_ctrl ,  input_capture_capture_t  *const p_capture )
```

10.20.10.1 Brief description

Update the last captured value and overflow count, in provided p_capture pointer. Implements lastCaptureGet.

10.20.10.2 Detailed description

Table 1342:

Name	Description
SSP_SUCCESS	Period value written successfully.
SSP_ERR_ASSERTION	The p_ctrl or p_value parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.20.10.3 Function steps

- Set capture value

10.20.11 API Data

10.20.11.1 gpt_input_capture_signal_t

`gpt_input_capture_signal_t`

Detailed description

Input capture signal selection

Enumerated values

Name	Description
GPT_INPUT_CAPTURE_SIGNAL_PIN_GTIOCA	GTIOCxA pin, where x is channel number.
GPT_INPUT_CAPTURE_SIGNAL_PIN_GTIOCB	GTIOCxB pin, where x is channel number.

10.20.11.2 gpt_input_capture_signal_filter_t

`gpt_input_capture_signal_filter_t`

Detailed description

Input capture signal noise filter (debounce) setting. Only available for input signals GTIOCxA and GTIOCxB. The noise filter samples the external signal at intervals of the PCLK divided by one of the values. When 3 consecutive samples are at the same level (high or low), then that level is passed on as the observed state of the signal. See "Noise Filter Function" in the hardware manual, GPT section.

Enumerated values

Name	Description
GPT_INPUT_CAPTURE_SIGNAL_FILTER_NONE	NONE - no filtering.
GPT_INPUT_CAPTURE_SIGNAL_FILTER_1	PCLK/1 - fast sampling.
GPT_INPUT_CAPTURE_SIGNAL_FILTER_4	PCLK/4.
GPT_INPUT_CAPTURE_SIGNAL_FILTER_16	PCLK/16.
GPT_INPUT_CAPTURE_SIGNAL_FILTER_64	PCLK/64 - slow sampling.

10.20.11.3 gpt_input_capture_clock_divider_t

`gpt_input_capture_clock_divider_t`

Detailed description

Input capture PCLK divider. Used to scale the timer counter.

Enumerated values

Name	Description
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_1	/ 1
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_4	/ 4
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_16	/ 16
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_64	/ 64
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_256	/ 256
GPT_INPUT_CAPTURE_CLOCK_DIVIDER_1024	/ 1024

10.20.12 Extensions

10.20.12.1 gpt_input_capture_extend_t

gpt_input_capture_extend_t

Brief description

Extension configuration struct for TU Input Capture.

Detailed description

Pointed to by p_extend

Variables

- gpt_input_capture_signal_t signal

  One of gpt_input_capture_signal_t.

- gpt_input_capture_signal_filter_t signal_filter

  One of gpt_input_capture_signal_filter_t.

- gpt_input_capture_clock_divider_t clock_divider

  One of gpt_input_capture_clock_divider_t.

- input_capture_signal_level_t enable_level

  The unused GTIOCx pin can be used as an enable signal to enable captures. If the Input Capture Signal Pin is GTIOCA, then the enable pin is GTIOCB. The enable level is set here if used.

- gpt_input_capture_signal_filter_t enable_filter

  One of gpt_input_capture_signal_filter_t.

10.20.12.2 gpt_input_capture_instance_ctrl_t

gpt_input_capture_instance_ctrl_t

Detailed description

Channel control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- uint32_t open

  Whether or not channel is open.

- uint8_t channel

  The channel in use.

- input_capture_mode_t mode

  The mode of measurement being performed.

- input_capture_repetition_t repetition

  One-shot or periodic measurement.

- uint32_t capture_count

  The value of the timer captured at the time of interrupt.

- uint32_t overflows_last

  Overflow count that occurred during last measurement.

- uint32_t overflows_current

  Running count of overflows in current measurement.

- void(* p_callback)(*p_args)

  Pointer to user callback.

- void const * p_context

  Pointer to user's context data, to be passed to the callback function.

- void * p_reg

  GPT base register for this channel.

- IRQn_Type capture_irq

  Capture IRQ number.

- IRQn_Type overflow_irq

  Overflow IRQ number.

- input_capture_variant_t variant

  Timer variant.

10.21 ICU

Driver for the Interrupt Controller Unit (ICU) External pin interrupts function.

10.21.1 Summary

Extends External IRQ Interface.

This module implements the External IRQ Interface using the external input pins in the Interrupt Controller Unit (ICU).

10.21.2 Functions

- R_ICU_ExternalIrqOpen
- R_ICU_ExternalIrqEnable
- R_ICU_ExternalIrqDisable
- R_ICU_ExternalIrqTriggerSet
- R_ICU_ExternalIrqFilterEnable
- R_ICU_ExternalIrqFilterDisable
- R_ICU_ExternalIrqVersionGet
- R_ICU_ExternalIrqClose

10.21.3 Defines

- #define ICU_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define ICU_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.21.4 R_ICU_ExternalIrqOpen

```
ssp_err_t R_ICU_ExternalIrqOpen ( external_irq_ctrl_t  *const p_api_ctrl ,
 external_irq_cfg_t  const *const p_cfg )
```

10.21.4.1 Brief description

Configure an external input pin for use with the button interface. Implements open.

10.21.4.2 Detailed description

The Open function is responsible for preparing an external input pin for operation. After completion of the Open function the external input pin shall be enabled and ready to service interrupts. This function must be called once prior to calling any other external input pin API functions. Once successfully completed, the status of the selected external input pin will be set to "open". After that this function should not be called again for the same external input pin without first performing a "close" by calling R_ICU_ExternalIrqClose.

Table 1343:

Name	Description
SSP_SUCCESS	Open successful.
SSP_ERR_ASSERTION	One of the following is invalid: • p_ctrl or p_cfg is NULL • The channel requested in p_cfg is not available on the device selected in r_bsp_cfg.h.
SSP_ERR_INVALID_ARGUMENT	p_cfg->p_callback is not NULL, but ISR is not enabled. ISR must be enabled to use callback function. Enable channel's overflow ISR in bsp_irq_cfg.h.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function to reconfigure the channel.
SSP_ERR_IP_CHANNEL_NOT_PRESENT	Requested channel does not exist on this device.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.21.5 R_ICU_ExternalIrqEnable

```
ssp_err_t R_ICU_ExternalIrqEnable ( external_irq_ctrl_t  *const p_api_ctrl )
```

10.21.5.1 Brief description

Enable external interrupt for specified channel at NVIC. Implements enable.

10.21.5.2 Detailed description

Table 1344:

Name	Description
SSP_SUCCESS	Interrupt Enabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INTERNAL	Requested IRQ is not defined in this system

10.21.6 R_ICU_ExternalIrqDisable

```
ssp_err_t R_ICU_ExternalIrqDisable ( external_irq_ctrl_t  *const p_api_ctrl )
```

10.21.6.1 Brief description

Disable external interrupt for specified channel at NVIC. Implements disable.

10.21.6.2 Detailed description

Table 1345:

Name	Description
SSP_SUCCESS	Interrupt disabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_INTERNAL	Requested IRQ is not defined in this system

10.21.7 R_ICU_ExternalIrqTriggerSet

```
ssp_err_t R_ICU_ExternalIrqTriggerSet ( external_irq_ctrl_t  *const p_api_ctrl ,
 external_irq_trigger_t  hw_trigger )
```

10.21.7.1 Brief description

Set trigger value provided. Implements triggerSet.

10.21.7.2 Detailed description

Table 1346:

Name	Description
SSP_SUCCESS	Period value written successfully.
SSP_ERR_ASSERTION	The p_ctrl or p_period parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.21.8 R_ICU_ExternalIrqFilterEnable

```
ssp_err_t R_ICU_ExternalIrqFilterEnable ( external_irq_ctrl_t
*const p_api_ctrl )
```

10.21.8.1 Brief description

Enable external interrupt digital filter for specified channel. Implements filterEnable.

10.21.8.2 Detailed description

Table 1347:

Name	Description
SSP_SUCCESS	External interrupt digital filter enabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.21.9 R_ICU_ExternalIrqFilterDisable

```
ssp_err_t R_ICU_ExternalIrqFilterDisable ( external_irq_ctrl_t
*const p_api_ctrl )
```

10.21.9.1 Brief description

Enable external interrupt digital filter for specified channel. Implements filterDisable.

10.21.9.2 Detailed description

Table 1348:

Name	Description
SSP_SUCCESS	External interrupt digital filter disabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.21.10 R_ICU_ExternalIrqVersionGet

```
ssp_err_t R_ICU_ExternalIrqVersionGet ( ssp_version_t  *const p_version )
```

10.21.10.1 Brief description

Set driver version based on compile time macros. Implements versionGet.

10.21.10.2 Detailed description

Table 1349:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.21.11 R_ICU_ExternalIrqClose

```
ssp_err_t R_ICU_ExternalIrqClose ( external_irq_ctrl_t  *const p_api_ctrl )
```

10.21.11.1 Brief description

Disable external interrupt. Implements close.

10.21.11.2 Detailed description

Table 1350:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.21.11.3 Function steps

- Cleanup. Disable interrupt
- Release BSP hardware lock

10.21.12 Extensions

10.21.12.1 icu_instance_ctrl_t

icu_instance_ctrl_t

Detailed description

Channel instance control block. DO NOT INITIALIZE. Initialization occurs in open.

Variables

- uint32_t open

  Used to determine if channel control block is in use.
- R_ICU_Type * p_reg

  Pointer to register base address.
- void(* p_callback)(*p_args)

  Callback provided when a external IRQ ISR occurs. Set to NULL for no CPU interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user callback in external_irq_callback_args_t.
- IRQn_Type irq

  NVIC interrupt number.
- uint8_t channel

  Channel.

10.22 IOPORT

Driver for the I/O Ports.

The IOPort HAL drivers provide the ability to access the I/O Ports of a device at both bit and port level. Port and pin direction can be changed. In addition a number of configuration APIs are provided to change the functionality of individual pins.

10.22.1 Functions

- R_IOPORT_Init
- R_IOPORT_PinsCfg
- R_IOPORT_PinCfg
- R_IOPORT_PinRead
- R_IOPORT_PortRead
- R_IOPORT_PortWrite
- R_IOPORT_PinWrite
- R_IOPORT_PortDirectionSet
- R_IOPORT_PinDirectionSet
- R_IOPORT_PortEventInputRead
- R_IOPORT_PinEventInputRead
- R_IOPORT_PortEventOutputWrite
- R_IOPORT_PinEventOutputWrite
- R_IOPORT_VersionGet
- R_IOPORT_EthernetModeCfg

10.22.2 Defines

- #define IOPORT_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define IOPORT_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.22.3 R_IOPORT_Init

```
ssp_err_t R_IOPORT_Init ( ioport_cfg_t const * p_cfg )
```

10.22.3.1 Brief description

Initializes internal driver data, then calls R_IOPORT_PinsCfg to configure pins.

10.22.3.2 Detailed description

Table 1351:

Name	Description
SSP_SUCCESS	Pin configuration data written to PFS register(s)
SSP_ERR_ASSERTION	NULL pointer

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.22.4 R_IOPORT_PinsCfg

```
ssp_err_t R_IOPORT_PinsCfg ( ioport_cfg_t const * p_cfg )
```

10.22.4.1 Brief description

Configures the functions of multiple pins by loading configuration data into pin PFS registers. Implements pinsCfg.

10.22.4.2 Detailed description

This function initializes the supplied list of PmnPFS registers with the supplied values. This data can be generated by the ISDE pin configurator or manually by the developer. Different pin configurations can be loaded for different situations such as low power modes and test.*

Table 1352:

Name	Description
SSP_SUCCESS	Pin configuration data written to PFS register(s)
SSP_ERR_ASSERTION	NULL pointer

10.22.5 R_IOPORT_PinCfg

```
ssp_err_t R_IOPORT_PinCfg ( ioport_port_pin_t  pin ,   uint32_t cfg )
```

10.22.5.1 Brief description

Configures the settings of a pin. Implements pinCfg.

10.22.5.2 Detailed description

Table 1353:

Name	Description
SSP_SUCCESS	Pin configured.
SSP_ERR_INVALID_ARGUMENT	Invalid pin

NOTE: This function is re-entrant for different pins. This function will change the configuration of the pin with the new configuration. For example it is not possible with this function to change the drive strength of a pin while leaving all the other pin settings unchanged. To achieve this the original settings with the required change will need to be written using this function.

10.22.6 R_IOPORT_PinRead

```
ssp_err_t R_IOPORT_PinRead ( ioport_port_pin_t  pin ,  ioport_level_t
* p_pin_value )
```

10.22.6.1 Brief description

Reads the level on a pin. Implements pinRead.

10.22.6.2 Detailed description

Table 1354:

Name	Description
SSP_SUCCESS	Pin read.
SSP_ERR_INVALID_ARGUMENT	Invalid argument
SSP_ERR_ASSERTION	NULL pointer

NOTE: This function is re-entrant for different pins.

10.22.7 R_IOPORT_PortRead

```
ssp_err_t R_IOPORT_PortRead ( ioport_port_t  port ,  ioport_size_t
* p_port_value )
```

10.22.7.1 Brief description

Reads the value on an IO port. Implements portRead.

10.22.7.2 Detailed description

The specified port will be read, and the levels for all the pins will be returned. Each bit in the returned value corresponds to a pin on the port. For example, bit 7 corresponds to pin 7, bit 6 to pin 6, and so on. *

Table 1355:

Name	Description
SSP_SUCCESS	Port read.
SSP_ERR_INVALID_ARGUMENT	Port not valid.
SSP_ERR_ASSERTION	NULL pointer

NOTE: This function is re-entrant for different ports.

10.22.8 R_IOPORT_PortWrite

```
ssp_err_t R_IOPORT_PortWrite ( ioport_port_t  port ,  ioport_size_t  value ,
 ioport_size_t  mask )
```

10.22.8.1 Brief description

Writes to multiple pins on a port. Implements portWrite.

10.22.8.2 Detailed description

The input value will be written to the specified port. Each bit in the value parameter corresponds to a bit on the port. For example, bit 7 corresponds to pin 7, bit 6 to pin 6, and so on. Each bit in the mask parameter corresponds to a pin on the port.

Only the bits with the corresponding bit in the mask value set will be updated. e.g. value = 0xFFFF, mask = 0x0003 results in only bits 0 and 1 being updated.

Table 1356:

Name	Description
SSP_SUCCESS	Port written to.
SSP_ERR_INVALID_ARGUMENT	The port and/or mask not valid.

NOTE: This function is re-entrant for different ports. This function makes use of the PCNTR3 register to atomically modify the levels on the specified pins on a port.

10.22.8.3 Function steps

- High bits
- Low bits

10.22.9 R_IOPORT_PinWrite

```
ssp_err_t R_IOPORT_PinWrite ( ioport_port_pin_t  pin ,  ioport_level_t  level )
```

10.22.9.1 Brief description

Sets a pin's output either high or low. Implements pinWrite.

10.22.9.2 Detailed description

Table 1357:

Name	Description
SSP_SUCCESS	Pin written to.
SSP_ERR_INVALID_ARGUMENT	The pin and/or level not valid.

NOTE: This function is re-entrant for different pins. This function makes use of the PCNTR3 register to atomically modify the level on the specified pin on a port.

10.22.10 R_IOPORT_PortDirectionSet

```
ssp_err_t R_IOPORT_PortDirectionSet ( ioport_port_t  port ,
 ioport_size_t  direction_values ,  ioport_size_t  mask )
```

10.22.10.1 Brief description

Sets the direction of individual pins on a port. Implements portDirectionSet.

10.22.10.2 Detailed description

Multiple pins on a port can be set to inputs or outputs at once. Each bit in the mask parameter corresponds to a pin on the port. For example, bit 7 corresponds to pin 7, bit 6 to pin 6, and so on. If a bit is set to 1 then the corresponding pin will be changed to an input or an output as specified by the direction values. If a mask bit is set to 0 then the direction of the pin will not be changed.

Table 1358:

Name	Description
SSP_SUCCESS	Port direction updated.
SSP_ERR_INVALID_ARGUMENT	The port and/or mask not valid.

NOTE: This function is re-entrant for different ports.

10.22.10.3 Function steps

- High bits
- Low bits
- New value to write to port direction register

10.22.11 R_IOPORT_PinDirectionSet

```
ssp_err_t R_IOPORT_PinDirectionSet ( ioport_port_pin_t  pin ,
 ioport_direction_t  direction )
```

10.22.11.1 Brief description

Sets the direction of an individual pin on a port. Implements pinDirectionSet.

10.22.11.2 Detailed description

Table 1359:

Name	Description
SSP_SUCCESS	Pin direction updated.

Table 1359: (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	The pin and/or direction not valid.

NOTE: This function is re-entrant for different pins.

10.22.12 R_IOPORT_PortEventInputRead

```
ssp_err_t R_IOPORT_PortEventInputRead ( ioport_port_t  port ,  ioport_size_t
* p_event_data )
```

10.22.12.1 Brief description

Reads the value of the event input data. Implements portEventInputRead.

10.22.12.2 Detailed description

The event input data for the port will be read. Each bit in the returned value corresponds to a pin on the port. For example, bit 7 corresponds to pin 7, bit 6 to pin 6, and so on.

The port event data is captured in response to a trigger from the ELC. This function enables this data to be read. Using the event system allows the captured data to be stored when it occurs and then read back at a later time.

Table 1360:

Name	Description
SSP_SUCCESS	Port read.
SSP_ERR_INVALID_ARGUMENT	Port not valid.
SSP_ERR_ASSERTION	NULL pointer

NOTE: This function is re-entrant for different ports.

10.22.13 R_IOPORT_PinEventInputRead

```
ssp_err_t R_IOPORT_PinEventInputRead ( ioport_port_pin_t  pin ,  ioport_level_t
* p_pin_event )
```

10.22.13.1 Brief description

Reads the value of the event input data of a specific pin. Implements pinEventInputRead.

10.22.13.2 Detailed description

The pin event data is captured in response to a trigger from the ELC. This function enables this data to be read. Using the event system allows the captured data to be stored when it occurs and then read back at a later time.

Table 1361:

Name	Description
SSP_SUCCESS	Pin read.
SSP_ERR_INVALID_ARGUMENT	Pin not valid.
SSP_ERR_ASSERTION	NULL pointer

NOTE: This function is re-entrant.

10.22.14 R_IOPORT_PortEventOutputWrite

```
ssp_err_t R_IOPORT_PortEventOutputWrite ( ioport_port_t  port ,
 ioport_size_t  event_data ,  ioport_size_t  mask_value )
```

10.22.14.1 Brief description

This function writes the set and reset event output data for a port. Implements portEventOutputWrite.

10.22.14.2 Detailed description

Using the event system enables a port state to be stored by this function in advance of being output on the port. The output to the port will occur when the ELC event occurs.

The input value will be written to the specified port when an ELC event configured for that port occurs. Each bit in the value parameter corresponds to a bit on the port. For example, bit 7 corresponds to pin 7, bit 6 to pin 6, and so on. Each bit in the mask parameter corresponds to a pin on the port.

Table 1362:

Name	Description
SSP_SUCCESS	Port event data written.
SSP_ERR_INVALID_ARGUMENT	Port and/or mask not valid. •

NOTE: This function is re-entrant for different ports.

10.22.15 R_IOPORT_PinEventOutputWrite

```
ssp_err_t R_IOPORT_PinEventOutputWrite ( ioport_port_pin_t  pin ,
 ioport_level_t  pin_value )
```

10.22.15.1 Brief description

This function writes the event output data value to a pin. Implements pinEventOutputWrite.

10.22.15.2 Detailed description

Using the event system enables a pin state to be stored by this function in advance of being output on the pin. The output to the pin will occur when the ELC event occurs.

Table 1363:

Name	Description
SSP_SUCCESS	Pin event data written.
SSP_ERR_INVALID_ARGUMENT	Pin or value not valid.

NOTE: This function is re-entrant for different ports.

10.22.16 R_IOPORT_VersionGet

```
ssp_err_t R_IOPORT_VersionGet ( ssp_version_t  * p_data )
```

10.22.16.1 Brief description

Returns IOPort HAL driver version. Implements versionGet.

10.22.16.2 Detailed description

Table 1364:

Name	Description
SSP_SUCCESS	Version information read.
SSP_ERR_ASSERTION	The parameter p_data is NULL.

NOTE: This function is reentrant.

10.22.17 R_IOPORT_EthernetModeCfg

```
ssp_err_t R_IOPORT_EthernetModeCfg ( ioport_ethernet_channel_t  channel ,
 ioport_ethernet_mode_t  mode )
```

10.22.17.1 Brief description

Configures Ethernet channel PHY mode. Implements ioport_api_t::ethModeCfg.

10.22.17.2 Detailed description

Table 1365:

Name	Description
SSP_SUCCESS	Ethernet PHY mode set.
SSP_ERR_INVALID_ARGUMENT	Channel or mode not valid.
SSP_ERR_UNSUPPORTED	Ethernet configuration not supported on this device.

NOTE: This function is not re-entrant.

10.23 IWDT

Driver for the Independent Watchdog Timer (IWDT).

10.23.1 Summary

This module supports the Independent Watchdog Timer (IWDT). It implements the WDT Interface. Extends WDT_API HAL layer drivers for interfacing with the Independent Watchdog Timer (IWDT) peripheral.

The IWDT HAL APIs provide the ability to refresh the independent watchdog, read the timer value and read and clear status flags. When used in NMI output mode the callback to be called by the NMI ISR can be registered.

10.23.2 Functions

- R_IWDT_Open
- R_IWDT_CfgGet
- R_IWDT_Refresh
- R_IWDT_StatusGet
- R_IWDT_StatusClear

- R_IWDT_CounterGet
- R_IWDT_TimeoutGet
- R_IWDT_VersionGet

10.23.3 Defines

- #define IWDT_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define IWDT_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.23.4 R_IWDT_Open

```
ssp_err_t R_IWDT_Open ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_cfg_t  const
*const p_cfg )
```

10.23.4.1 Brief description

Register the IWDT NMI callback.

10.23.4.2 Detailed description

Table 1366:

Name	Description
SSP_SUCCESS	IWDT NMI callback successfully configured.
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_INVALID_MODE	An attempt to open the IWDT when the OFS0 register is not configured for auto-start mode.
SSP_ERR_HW_LOCKED	IWDT module has already been called.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

NOTE: This function is not reentrant.

10.23.4.3 Function steps

- g_iwdt_version is accessed by the ASSERT macro only and so compiler toolchain can issue a warning that it is not accessed. The code below eliminates this warning and also ensures these data structures are not optimized away.
- Eliminate toolchain warning when NMI output is not being used.
- Lock the IWDT Hardware Resource
- Initialize global pointer to WDT for NMI callback use.
- Check for NMI output mode
- NMI output mode
- Enable the IWDT underflow/refresh error interrupt (will generate an NMI).

10.23.5 R_IWDT_CfgGet

```
ssp_err_t R_IWDT_CfgGet ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_cfg_t
*const p_cfg )
```

10.23.5.1 Brief description

Read the configuration of the IWDT. Implements cfgGet.

10.23.5.2 Detailed description

Table 1367:

Name	Description
SSP_SUCCESS	IWDT configuration successfully read.
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid.

NOTE: This function is reentrant.

10.23.5.3 Function steps

- Get timeout value from OFS0 register.

10.23.6 R_IWDT_Refresh

```
ssp_err_t R_IWDT_Refresh ( wdt_ctrl_t  *const p_api_ctrl )
```

10.23.6.1 Brief description

Refresh the Independent Watchdog Timer. Implements refresh.

10.23.6.2 Detailed description

Table 1368:

Name	Description
SSP_SUCCESS	IWDT successfully refreshed.

NOTE: This function is reentrant. This function only returns SSP_SUCCESS. If the refresh fails due to being performed outside of the permitted refresh period the device will either reset or trigger an NMI ISR to run.

10.23.7 R_IWDT_StatusGet

```
ssp_err_t R_IWDT_StatusGet ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_status_t
*const p_status )
```

10.23.7.1 Brief description

Read the IWDT status flags.

10.23.7.2 Detailed description

Indicates both status and error conditions.

Table 1369:

Name	Description
SSP_SUCCESS	IWDT status successfully read.
SSP_ERR_ASSERTION	Null pointer as a parameter.

NOTE: This function is reentrant. When the IWDT is configured to output a reset on underflow or refresh error reading the status and error flags can be read after reset to establish if the IWDT caused the reset. Reading the status and error flags in NMI output mode indicates whether the IWDT generated the NMI interrupt.

10.23.8 R_IWDT_StatusClear

```
ssp_err_t R_IWDT_StatusClear ( wdt_ctrl_t  *const p_api_ctrl ,
 wdt_status_t const  status )
```

10.23.8.1 Brief description

Clear the IWDT status and error flags. Implements statusClear.

10.23.8.2 Detailed description

Table 1370:

Name	Description
SSP_SUCCESS	IWDT flag(s) successfully cleared.
SSP_ERR_ASSERTION	Null pointer as a parameter.

NOTE: This function is reentrant.

10.23.8.3 Function steps

- Write zero to clear flags

10.23.9 R_IWDT_CounterGet

```
ssp_err_t R_IWDT_CounterGet ( wdt_ctrl_t  *const p_api_ctrl ,   uint32_t
*const p_count )
```

10.23.9.1 Brief description

Read the current count value of the IWDT. Implements counterGet.

10.23.9.2 Detailed description

Table 1371:

Name	Description
SSP_SUCCESS	IWDT current count successfully read.
SSP_ERR_ASSERTION	Null pointer passed as a parameter.

NOTE: This function is reentrant.

10.23.10 R_IWDT_TimeoutGet

```
ssp_err_t R_IWDT_TimeoutGet ( wdt_ctrl_t  *const p_api_ctrl ,
 wdt_timeout_values_t  *const p_timeout )
```

10.23.10.1 Brief description

Read timeout information for the watchdog timer. Implements timeoutGet.

10.23.10.2 Detailed description

Table 1372:

Name	Description
SSP_SUCCESS	WDT successfully refreshed.
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_ABORTED	Invalid clock divider for this watchdog

NOTE: This function is reentrant. This function must not be called before calling R_WDT_Open.

10.23.11 R_IWDT_VersionGet

```
ssp_err_t R_IWDT_VersionGet ( ssp_version_t  *const p_data )
```

10.23.11.1 Brief description

Return IWDT HAL driver version. Implements versionGet.

10.23.11.2 Detailed description

Table 1373:

Name	Description
SSP_SUCCESS	Call successful.
SSP_ERR_ASSERTION	Null pointer passed as a parameter.

NOTE: This function is reentrant.

10.23.12 Extensions

10.23.12.1 iwdt_instance_ctrl_t

iwdt_instance_ctrl_t

Detailed description

WDT control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- bool wdt_open

  Indicates whether the open() API has been successfully called.

- void const * p_context

  Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

- R_IWDT_Type * p_reg

  Pointer to register base address.

- void(* p_callback)(*p_args)

  Callback provided when a WDT NMI ISR occurs.

10.24 JPEG CODEC

Driver for the JPEG CODEC.

10.24.1 Functions

- R_JPEG_Decode_Open
- R_JPEG_Decode_OutputBufferSet
- R_JPEG_Decode_LinesDecodedGet
- R_JPEG_Decode_InputBufferSet
- R_JPEG_Decode_ImageSubsampleSet
- R_JPEG_Decode_HorizontalStrideSet
- R_JPEG_Decode_Close
- R_JPEG_Decode_ImageSizeGet
- R_JPEG_Decode_StatusGet
- R_JPEG_Decode_PixelFormatGet
- R_JPEG_Decode_VersionGet

10.24.2 Defines

- #define JPEG_DECODE_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define JPEG_DECODE_CODE_VERSION_MINOR

  Initial value:`(5U)`

10.24.3 R_JPEG_Decode_Open

```
ssp_err_t R_JPEG_Decode_Open ( jpeg_decode_ctrl_t  *const p_api_ctrl ,
 jpeg_decode_cfg_t  const *const p_cfg )
```

10.24.3.1 Brief description

Initialize the JPEG Codec module. This function configures the JPEG Codec for decoding operation, sets up the registers for data format and pixel format based on user-supplied configuration parameters. Interrupts are enabled to support image size read operation and callback functions.

10.24.3.2 Detailed description

Table 1374:

Name	Description
SSP_SUCCESS	JPEG Codec module is properly configured and is ready to take input data.
SSP_ERR_IN_USE	JPEG Codec is already in use.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_HW_LOCKED	JPEG Codec resource is locked.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

10.24.3.3 Function steps

- Verify JPEG Codec is not already used.
- Record the configuration settings.
- Initialize horizontal stride value.

- Initialize output buffer size.
- Initialize total_lines_decoded
- Initialize horizontal sub-sample setting.
- Provide power to the JPEG module.
- Perform bus reset
- Reset the destination buffer address.
- Reset the source buffer address.
- Reset the horizontal stride.
- Configure the JPEG module for decode operation.
- Set image format for the decoded image.
- If the output pixel format is ARGB8888, also configure the alpha value.
- Set the alpha value for the decoded image.
- Set the output data format.
- The following interrupts are enabled: Interrupt on all errors Interrupt on Image Size
- Record user supplied callback routine.
- Set the driver status.
- All done. Return success.

10.24.4 R_JPEG_Decode_OutputBufferSet

```
ssp_err_t R_JPEG_Decode_OutputBufferSet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
  void * p_output_buffer ,   uint32_t output_buffer_size )
```

10.24.4.1 Brief description

Assign output buffer to the JPEG Codec for storing output data.

10.24.4.2 Detailed description

NOTE: *The number of image lines to be decoded depends on the size of the buffer and the horizontal stride settings. Once the output buffer size is known, the horizontal stride value is known, and the input pixel format is known (the input pixel format is obtained by the JPEG decoder from the JPEG headers), the driver automatically computes the number of lines that can be decoded into the output buffer. After these lines are decoded, the JPEG engine pauses and a callback function is triggered, so the application is able to provide the next buffer for the JPEG module to resume the operation.*The JPEG decoding operation automatically starts after both the input buffer and the output buffer are set, and the output buffer is big enough to hold at least eight lines of decoded image data.

Table 1375:

Name	Description
SSP_SUCCESS	The output buffer is properly assigned to JPEG codec device.
SSP_ERR_ASSERTION	Pointer to the control block is NULL, or the pointer to the output_buffer. is NULL, or the output_buffer_size is 0.
SSP_ERR_INVALID_ALIGNMENT	Buffer starting address is not 8-byte aligned.
SSP_ERR_NOT_OPEN	JPEG not opened.
SSP_ERR_JPEG_BUFFERSIZE_NOT_ENOUGH	Invalid buffer size

10.24.4.3 Function steps

- Set the decoding destination address.
- Record the size of the output buffer.
- If the image size is not ready yet, the driver does not know the input pixel format. Without that information, the driver is unable to compute the number of lines of image to decode. In this case, the driver would record the output buffer size. Once all the information is ready, the driver would attempt to start the decoding process.
- For a given buffer size, compute number of lines to decode if the image size acquisition is known.
- If the driver status is IMAGE_SIZE_READY with no other flags, that means the driver just received IMAGE_SIZE. It has not started the decoding process yet.
- If Input buffer is set, output buffer is set, and horizontal stride is set, the driver is able to determine the number of lines to decode, and start the decoding operation.
- If the current status is OUTPUT_PAUSE, the driver needs to resume the operation.

10.24.5 R_JPEG_Decode_LinesDecodedGet

```
ssp_err_t R_JPEG_Decode_LinesDecodedGet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
  uint32_t * p_lines )
```

10.24.5.1 Brief description

Returns the number of lines decoded into the output buffer.

10.24.5.2 Detailed description

NOTE: Use this function to retrieve number of image lines written to the output buffer after JPEG decoded a partial image. Combined with the horizontal stride settings and the output pixel format, the application can compute the amount of data to read from the output buffer.

Table 1376:

Name	Description
SSP_SUCCESS	The output buffer is properly assigned to JPEG codec device.
SSP_ERR_ASSERTION	Pointer to the control block is NULL, or the pointer to the output_buffer. is NULL, or the output_buffer_size is 0.
SSP_ERR_NOT_OPEN	JPEG not opened.

10.24.6 R_JPEG_Decode_InputBufferSet

```
ssp_err_t R_JPEG_Decode_InputBufferSet ( jpeg_decode_ctrl_t  *const p_api_ctrl ,
  void * p_data_buffer ,   uint32_t data_buffer_size )
```

10.24.6.1 Brief description

Assign input data buffer to JPEG codec for processing.

10.24.6.2 Detailed description

NOTE: *After the amount of data is processed, the JPEG driver triggers a callback function with the flag JPEG_OPERATION_INPUT_PAUSE set. The application supplies the next chunk of data to the driver so JPEG decoding can resume.*The JPEG decoding operation automatically starts after both the input buffer and the output buffer are set, and the output buffer is big enough to hold at least one line of decoded image data.

Table 1377:

Name	Description
SSP_SUCCESS	The input data buffer is properly assigned to JPEG Codec device.
SSP_ERR_ASSERTION	Pointer to the control block is NULL, or the pointer to the input_buffer is NULL, or the input_buffer_size is 0.
SSP_ERR_INVALID_ALIGNMENT	Buffer starting address is not 8-byte aligned.

Table 1377: (Continued)

Name	Description
SSP_ERR_NOT_OPEN	JPEG not opened.

10.24.6.3 Function steps

- Configure the input buffer address.
- If the system is idle, start the JPEG engine. This allows the system to obtain image information (image size and input pixel format). This information is needed to drive the decode process later on.
- Based on buffer size, detect the in count mode setting. The driver is able to read input data in chunks. However the size of each chunk is limited to BUFFER_MAX_SIZE. Therefore, if the input data size is larger than BUFFER_MAX_SIZE, the driver assumes the entire input data is present, and can be decoded without additional input data. Otherwise, the driver enables input stream feature. This works even if the entire input size is smaller than BUFFER_MAX_SIZE.

10.24.7 R_JPEG_Decode_ImageSubsampleSet

```
ssp_err_t R_JPEG_Decode_ImageSubsampleSet ( jpeg_decode_ctrl_t
*const p_api_ctrl ,  jpeg_decode_subsample_t  horizontal_subsample ,
 jpeg_decode_subsample_t  vertical_subsample )
```

10.24.7.1 Brief description

Configure horizontal and vertical subsample.

10.24.7.2 Detailed description

NOTE: Use for scaling the decoded image.

Table 1378:

Name	Description
SSP_SUCCESS	Horizontal Stride value is properly configured.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_INVALID_ARGUMENT	Sub-sample setting is invalid.
SSP_ERR_NOT_OPEN	JPEG not opened.

10.24.7.3 Function steps

- Update horizontal sub-sample setting.

10.24.8 R_JPEG_Decode_HorizontalStrideSet

```
ssp_err_t R_JPEG_Decode_HorizontalStrideSet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
  uint32_t horizontal_stride )
```

10.24.8.1 Brief description

Configure horizontal stride setting.

10.24.8.2 Detailed description

NOTE: Use when the horizontal stride needs to match the image width and the image size is unknown when opening the JPEG driver. (If the image size is known prior to the open call, pass the horizontal stride value in the jpef_cfg_t structure.) After the image size becomes available, use this function to update the horizontal stride value. If the driver must decode one line at a time, the horizontal stride can be set to zero.

Table 1379:

Name	Description
SSP_SUCCESS	Horizontal Stride value is properly configured.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_INVALID_ALIGNMENT	Horizontal stride is zero or not 8-byte aligned.
SSP_ERR_NOT_OPEN	JPEG not opened.

10.24.8.3 Function steps

- Record the horizontal stride value in the control block
- Set the horizontal stride.
- If the parameters all are set, resume the core to decode.
- For the given buffer size, compute number of lines to decode.

10.24.9 R_JPEG_Decode_Close

```
ssp_err_t R_JPEG_Decode_Close ( jpeg_decode_ctrl_t  * p_api_ctrl )
```

10.24.9.1 Brief description

Cancel an outstanding JPEG codec operation and close the device.

10.24.9.2 Detailed description

Table 1380:

Name	Description
SSP_SUCCESS	The input data buffer is properly assigned to JPEG Codec device.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_NOT_OPEN	JPEG not opened.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- eventInfoGet

10.24.9.3 Function steps

- Clear JPEG JINTE0 interrupt and JINTE1 interrupt.
- Disable JEDI and JDTI at NVIC
- Power off the JPEG codec.
- Reset the jpeg status flag in the driver.
- Unlock module at BSP level.

10.24.10 R_JPEG_Decode_ImageSizeGet

```
ssp_err_t R_JPEG_Decode_ImageSizeGet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
  uint16_t * p_horizontal_size ,   uint16_t * p_vertical_size )
```

10.24.10.1 Brief description

Obtain the size of the image. This operation is valid during JPEG decoding operation.

10.24.10.2 Detailed description

Table 1381:

Name	Description
SSP_SUCCESS	The image size is available and the horizontal and vertical values are stored in the memory pointed to by p_horizontal_size and p_vertical_size.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_IMAGE_SIZE_UNKNOWN	The image size is unknown. More input data may be needed.
SSP_ERR_INVALID_MODE	JPEG Codec module is not decoding.
SSP_ERR_NOT_OPEN	JPEG is not opened.

10.24.11 R_JPEG_Decode_StatusGet

```
ssp_err_t R_JPEG_Decode_StatusGet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
 jpeg_decode_status_t  * p_status )
```

10.24.11.1 Brief description

Get the status of the JPEG codec. This function can also be used to poll the device.

10.24.11.2 Detailed description

Table 1382:

Name	Description
SSP_SUCCESS	The status information is successfully retrieved.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_NOT_OPEN	JPEG is not opened.

10.24.11.3 Function steps

- HW does not report error. Return internal status information.

10.24.12 R_JPEG_Decode_PixelFormatGet

```
ssp_err_t R_JPEG_Decode_PixelFormatGet ( jpeg_decode_ctrl_t  * p_api_ctrl ,
 jpeg_decode_color_space_t  * p_color_space )
```

10.24.12.1 Brief description

Get the input pixel format.

10.24.12.2 Detailed description

Table 1383:

Name	Description
SSP_SUCCESS	The status information is successfully retrieved.
SSP_ERR_ASSERTION	Pointer to the control block is NULL.
SSP_ERR_NOT_OPEN	JPEG is not opened.

10.24.12.3 Function steps

- HW does not report error. Return internal status information.

10.24.13 R_JPEG_Decode_VersionGet

```
ssp_err_t R_JPEG_Decode_VersionGet ( ssp_version_t  * p_version )
```

10.24.13.1 Brief description

Get version of the display interface and GLCD HAL code.

10.24.13.2 Detailed description

Table 1384:

Name	Description
SSP_SUCCESS	Version number
SSP_ERR_ASSERTION	The parameter p_version is NULL.

NOTE: This function is reentrant.

10.24.14 Extensions

10.24.14.1 jpeg_decode_instance_ctrl_t

jpeg_decode_instance_ctrl_t

Detailed description

JPEG Codec module control block. DO NOT INITIALIZE. Initialization occurs when jpep_api_t::open is called.

Variables

- jpeg_decode_status_t status

  JPEG Codec module status.
- ssp_err_t error_code

  JPEG Codec error code (if any).
- void(* p_callback)(*p_args)

  User-supplied callback functions.
- void const * p_extend

  JPEG Codec hardware dependent configuration */.
- void const * p_context

  Placeholder for user data. Passed to user callback in jpeg_decode_callback_args_t.
- R_JPEG_Type * p_reg

  Pointer to register base address.
- jpeg_decode_pixel_format_t pixel_format

  Pixel format.
- uint32_t horizontal_stride

  Horizontal Stride settings.
- uint32_t outbuffer_size

  out buffer size
- uint16_t total_lines_decoded

  Track the number of lines decoded so far.
- jpeg_decode_subsample_t horizontal_subsample

  Horizontal sub-sample setting.

10.25 Key Interrupts

Driver for the Key Interrupt Function.

The Key input driver can be used for one to eight channels or in a matrix format. This module implements the following interface: Key Matrix Interface.

10.25.1 Functions

- R_KINT_KEYMATRIX_Open
- R_KINT_KEYMATRIX_Close
- R_KINT_KEYMATRIX_Enable
- R_KINT_KEYMATRIX_Disable
- R_KINT_KEYMATRIX_TriggerSet
- R_KINT_VersionGet

10.25.2 Defines

- #define KINT_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define KINT_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.25.3 R_KINT_KEYMATRIX_Open

```
ssp_err_t R_KINT_KEYMATRIX_Open ( keymatrix_ctrl_t  *const p_api_ctrl ,
 keymatrix_cfg_t  const *const p_cfg )
```

10.25.3.1 Brief description

Power on KINT, handle required initialization described in hardware manual. Implements open.

10.25.3.2 Detailed description

The Open function configures all the Key Input (KINT) channels and provides a handle for use with the rest of the KINT API functions. This function must be called at least once prior to calling any other KINT API functions. After the peripheral is initialized, the Open function should not be called again without first calling the Close function.

Table 1385:

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters may be NULL: p_cfg, or p_ctrl or the callback.
SSP_ERR_INVALID_ARGUMENT	One of the following may be invalid: • p_cfg->channel: must be between 0 and KINT_MAX_NUM_CHANNELS • p_cfg->trigger not a valid value.
SSP_ERR_HW_LOCKED	The API has already been opened. It must be closed before it can be opened again.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: This function is not reentrant.

10.25.3.3 Function steps

- Check to see that the arguments passed are not null pointers
- Grab the hardware lock. If successful this indicates that the open was not previously called.
- Disable interrupts in the KINT peripheral
- Clear any pending interrupt requests in the KINT peripheral
- Clear the Interrupt Request in the ICU
- Configure the trigger edge. The trigger edge can be modified in the TriggerSet function later if desired
- KEYMATRIX_TRIG_RISING condition
- Configure the usage of key interrupt flags
- Store the callback and context information
- If interrupts are to be enabled now, set it up for the selected channels. The channels can be changed later in the enable function but to modify the callback and context, the API has to be closed and reopened with the new callback and context. ***NOTE:*** *The KINT hardware only supports a single interrupt for all channels*Enable interrupt for the selected channels after casting since KRM is an 8 bit register
- Enable interrupt
- Store number of channels for use to the control block to use it later
- Mark KINT as initialized

10.25.4 R_KINT_KEYMATRIX_Close

```
ssp_err_t R_KINT_KEYMATRIX_Close ( keymatrix_ctrl_t  *const p_api_ctrl )
```

10.25.4.1 Brief description

Disable KINT. Implements close.

10.25.4.2 Detailed description

End of function R_KINT_KEYMATRIX_Open The Close function disables the interrupts in the peripheral and the NVIC and clears any pending interrupt requests. It also releases the hardware lock on the API.

Table 1386:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.

10.25.4.3 Function steps

- Disable interrupt in ICU
- Disable interrupts in the KINT peripheral
- Clear any pending interrupt requests in the KINT peripheral
- Clear the Interrupt Request bit
- Clear stored internal driver data
- Mark KINT as un-initialized
- Release the lock

10.25.5 R_KINT_KEYMATRIX_Enable

```
ssp_err_t R_KINT_KEYMATRIX_Enable ( keymatrix_ctrl_t  *const p_api_ctrl )
```

10.25.5.1 Brief description

Enable external irq for all the specified channel by R_KINT_KEYMATRIX_Open. Implements enable.

10.25.5.2 Detailed description

This function enables interrupts for the KINT peripheral both at the interrupt level and in the NVIC after clearing any pending requests in the KINT and ICU peripheral. All the channels specified by R_KINT_KEYMATRIX_Open are enabled.

Table 1387:

Name	Description
SSP_SUCCESS	Interrupt disabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_NOT_OPEN	The peripheral is not opened.

10.25.5.3 Function steps

- Clear any pending interrupt requests in the KINT peripheral
- Clear the Interrupt Request Flag in the ICU
- Enable interrupt for the selected channels after casting since KRM is an 8 bit register
- Enable interrupt

10.25.6 R_KINT_KEYMATRIX_Disable

```
ssp_err_t R_KINT_KEYMATRIX_Disable ( keymatrix_ctrl_t  *const p_api_ctrl )
```

10.25.6.1 Brief description

Disable external irq for all the specified channel by R_KINT_KEYMATRIX_Open. Implements disable.

10.25.6.2 Detailed description

This function disables interrupts for the KINT peripheral both at the interrupt level and in the NVIC. All the channels specified by R_KINT_KEYMATRIX_Open are disabled.

Table 1388:

Name	Description
SSP_SUCCESS	Interrupt disabled successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.

Table 1388: (Continued)

Name	Description
SSP_ERR_NOT_OPEN	The channel is not opened.

10.25.6.3 Function steps

- Disable interrupts in the KINT peripheral
- Clear any pending interrupt requests in the KINT peripheral
- Disable interrupt in the ICU

10.25.7 R_KINT_KEYMATRIX_TriggerSet

```
ssp_err_t R_KINT_KEYMATRIX_TriggerSet ( keymatrix_ctrl_t  *const p_api_ctrl ,
 keymatrix_trigger_t  hw_trigger )
```

10.25.7.1 Brief description

Set trigger edge (falling or rising) provided. Implements triggerSet.

10.25.7.2 Detailed description

This function changes trigger sense at run-time. The change is applied to all the channels specified by R_KINT_KEYMATRIX_Open.

Table 1389:

Name	Description
SSP_SUCCESS	Trigger value written successfully.
SSP_ERR_ASSERTION	The p_ctrl parameter was null.
SSP_ERR_INVALID_ARGUMENT	Trigger input invalid. hw_trigger must be one of the options from button_trigger_t.
SSP_ERR_NOT_OPEN	The channel is not opened.

NOTE: This function expects to be called when the driver is disabled (the driver state before R_KINT_KEYMATRIX-_Enable is called if the driver is opened in the non-auto start mode, or after R_KINT_KEYMATRIX_Disable is called if the driver is opened in the auto start mode). The driver does not restrict to call this API when the driver is enabled but users need to make sure the edge detection sense is instantly changed by this API call.

10.25.7.3 Function steps

- Configure the trigger edge
- KEYMATRIX_TRIG_RISING condition
- Configure the usage of key interrupt flags

10.25.8 R_KINT_VersionGet

```
ssp_err_t R_KINT_VersionGet ( ssp_version_t  *const p_version )
```

10.25.8.1 Brief description

Set driver version based on compile time macros.

10.25.8.2 Detailed description

Table 1390:

Name	Description
SSP_SUCCESS	Successful return.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.25.9 Extensions

10.25.9.1 kint_instance_ctrl_t

kint_instance_ctrl_t

Detailed description

Channel instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- R_KINT_Type * p_reg

  Pointer to register base address.
- keymatrix_channels_t channels

  Channel bitmask.
- IRQn_Type irq

  Interrupt priority number.

10.26 LPM

Driver for Low Power Modes.

The LPM (Low Power Mode) Driver provides several functions for controlling power consumption including stopping modules, selecting the operating mode, configuring low power modes, and transitioning to low power modes. The LPM driver supports configuration of MCU operating modes and MCU low power modes using the LPM hardware peripheral. The LPM driver supports operating modes low-voltage, low-speed, middle-speed, high-speed, and suboscillator mode.

The LPM driver supports low power modes deep standby, standby, sleep, and snooze. The LPM driver supports reducing power consumption when in deep standby mode via internal power supply control and reseting the states of IO ports. The LPM driver supports disabling and enabling of the MCU's other hardware peripherals.

NOTE: *Not all operating modes are available on all MCU's. Not all low power modes are available on all MCU's.*

10.26.1 Functions

- R_LPM_Init
- R_LPM_MSTPCRSet
- R_LPM_MSTPCRGet
- R_LPM_ModuleStop
- R_LPM_ModuleStart
- R_LPM_OperatingPowerModeSet
- R_LPM_SnoozeEnable
- R_LPM_SnoozeDisable
- R_LPM_LowPowerConfigure
- R_LPM_WUPENSet
- R_LPM_WUPENGet
- R_LPM_DeepStandbyCancelRequestEnable
- R_LPM_DeepStandbyCancelRequestDisable
- R_LPM_LowPowerModeEnter
- R_LPM_VersionGet

10.26.2 Defines

- #define LPM_CODE_VERSION_MAJOR

  Initial value:`(2U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define LPM_CODE_VERSION_MINOR

  Initial value:`(6U)`

10.26.3 R_LPM_Init

```
ssp_err_t R_LPM_Init ( lpm_cfg_t  const *const p_cfg )
```

10.26.3.1 Brief description

Initialize the mcu operating power mode.

10.26.3.2 Detailed description

Initialize the mcu operating power mode according to the passed in config structure.

Table 1391:

Name	Description
SSP_SUCCESS	Configuration was successful
SSP_ERR_ASSERTION	p_cfg was NULL

10.26.4 R_LPM_MSTPCRSet

```
ssp_err_t R_LPM_MSTPCRSet (  uint32_t mstpcra_value ,   uint32_t mstpcrb_value ,
  uint32_t mstpcrc_value ,   uint32_t mstpcrd_value )
```

10.26.4.1 Brief description

Set the value of all the Module Stop Control Registers.

10.26.4.2 Detailed description

Table 1392:

Name	Description
SSP_SUCCESS	Value was successfully set

10.26.5 R_LPM_MSTPCRGet

```
ssp_err_t R_LPM_MSTPCRGet (  uint32_t * mstpcra_value ,   uint32_t
* mstpcrb_value ,   uint32_t * mstpcrc_value ,   uint32_t * mstpcrd_value )
```

10.26.5.1 Brief description

Get the values of all the Module Stop Control Registers.

10.26.5.2 Detailed description

Table 1393:

Name	Description
SSP_SUCCESS	Value was successfully retrieved
SSP_ERR_ASSERTION	One of the passed in pointers was NULL

10.26.6 R_LPM_ModuleStop

```
ssp_err_t R_LPM_ModuleStop ( lpm_mstp_t  module )
```

10.26.6.1 Brief description

Stop a module.

10.26.6.2 Detailed description

Stop a module from running by setting the corresponding bit in the module stop register.

Table 1394:

Name	Description
SSP_SUCCESS	Configuration was successful
SSP_ERR_ASSERTION	Module stop bit not available on this MCU.

10.26.7 R_LPM_ModuleStart

```
ssp_err_t R_LPM_ModuleStart ( lpm_mstp_t  module )
```

10.26.7.1 Brief description

Run a module.

10.26.7.2 Detailed description

Start a module running by clearing the corresponding bit in the module stop register.

Table 1395:

Name	Description
SSP_SUCCESS	Configuration was successful
SSP_ERR_ASSERTION	Module stop bit not available on this MCU.

10.26.8 R_LPM_OperatingPowerModeSet

```
ssp_err_t R_LPM_OperatingPowerModeSet ( lpm_operating_power_t  power_mode ,
 lpm_subosc_t  subosc )
```

10.26.8.1 Brief description

Set the operating power mode.

10.26.8.2 Detailed description

Table 1396:

Name	Description
SSP_SUCCESS	Operating power mode successfully set
SSP_ERR_ASSERTION	Operating power control mode not available on this MCU.

10.26.9 R_LPM_SnoozeEnable

```
ssp_err_t R_LPM_SnoozeEnable ( lpm_snooze_rxd0_t  rdx0_mode ,
 lpm_snooze_dtc_t  dtc_mode ,  lpm_snooze_request_t  requests ,
 lpm_snooze_end_t  triggers )
```

10.26.9.1 Brief description

Configure and enable snooze mode.

10.26.9.2 Detailed description

NOTE: this function must be called before entering Software Standby mode, otherwise snooze mode will not be entered

Table 1397:

Name	Description
SSP_SUCCESS	Snooze mode successfully enabled
SSP_ERR_ASSERTION	One or more snooze settings are not available on this MCU.

10.26.10 R_LPM_SnoozeDisable

```
ssp_err_t R_LPM_SnoozeDisable (  void  )
```

10.26.10.1 Brief description

Disable snooze mode.

10.26.10.2 Detailed description

Table 1398:

Name	Description
SSP_SUCCESS	Snooze mode successfully disabled

10.26.11 R_LPM_LowPowerConfigure

```
ssp_err_t R_LPM_LowPowerConfigure ( lpm_low_power_mode_t  power_mode ,
 lpm_output_port_enable_t  output_port_enable ,
 lpm_power_supply_t  power_supply ,  lpm_io_port_t  io_port_state )
```

10.26.11.1 Brief description

Configure a low power mode.

10.26.11.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1399:

Name	Description
SSP_SUCCESS	Sleep mode entered successfully
SSP_ERR_ASSERTION	Low power mode are not available on this MCU.
SSP_ERR_ASSERTION	Invalid deep standby power supply configuration.

10.26.12 R_LPM_WUPENSet

```
ssp_err_t R_LPM_WUPENSet (  uint32_t wupen_value )
```

10.26.12.1 Brief description

Set the value of the Wake Up Interrupt Enable Register WUPEN.

10.26.12.2 Detailed description

Table 1400:

Name	Description
SSP_SUCCESS	Value was successfully set

10.26.13 R_LPM_WUPENGet

```
ssp_err_t R_LPM_WUPENGet (  uint32_t * wupen_value )
```

10.26.13.1 Brief description

Get the value of the Wake Up Interrupt Enable Register WUPEN.

10.26.13.2 Detailed description

Table 1401:

Name	Description
SSP_SUCCESS	Value was successfully retrieved

Table 1401: (Continued)

Name	Description
SSP_ERR_ASSERTION	One of the passed in pointers was NULL

10.26.14 R_LPM_DeepStandbyCancelRequestEnable

```
ssp_err_t R_LPM_DeepStandbyCancelRequestEnable ( lpm_deep_standby_t  pin_signal
,  lpm_cancel_request_edge_t  rising_falling )
```

10.26.14.1 Brief description

Enable a Deep Standby Cancel Request.

10.26.14.2 Detailed description

Enable a pin or signal that will cancel Deep Software Standby mode

Table 1402:

Name	Description
SSP_SUCCESS	Deep Software Standby cancel request enabled

10.26.15 R_LPM_DeepStandbyCancelRequestDisable

```
ssp_err_t R_LPM_DeepStandbyCancelRequestDisable ( lpm_deep_standby_t  pin_signal
 )
```

10.26.15.1 Brief description

Disable a Deep Standby Cancel Request.

10.26.15.2 Detailed description

Disable a pin or signal that will cancel Deep Software Standby mode

Table 1403:

Name	Description
SSP_SUCCESS	Deep Software Standby cancel request disabled

10.26.16 R_LPM_LowPowerModeEnter

```
ssp_err_t R_LPM_LowPowerModeEnter (  void  )
```

10.26.16.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.26.16.2 Detailed description

Function will return after waking from low power mode.

Table 1404:

Name	Description
SSP_SUCCESS	Successful.

10.26.16.3 Function steps

- DSB should be last instruction executed before WFI
  infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0321a/BIHICBGB.html

10.26.17 R_LPM_VersionGet

```
ssp_err_t R_LPM_VersionGet ( ssp_version_t  *const p_version )
```

10.26.17.1 Brief description

Get the driver version based on compile time macros.

10.26.17.2 Detailed description

Table 1405:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_version is NULL.

10.27 LPMV2 S124

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.27.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.27.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t

10.27.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.27.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.27.4.1 Brief description

Get the driver version based on compile time macros.

10.27.4.2 Detailed description

Table 1406:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.27.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.27.5.1 Brief description

Perform any necessary initialization.

10.27.5.2 Detailed description

Table 1407:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.27.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.27.6.1 Brief description

Configure a low power mode.

10.27.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1408:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.27.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.27.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.27.7.2 Detailed description

Function will return after waking from low power mode.

Table 1409:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.27.8 API Data

10.27.8.1 lpmv2_snooze_request_t

```
lpmv2_snooze_request_t
```

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.27.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.27.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.27.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.27.8.5 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.27.8.6 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.27.9 Extensions

10.27.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.27.10 Modules

- Build Time Configurations

10.27.10.1 Build Time Configurations

Defines

- #define LPMV2_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking

10.28 LPMV2 S128

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.28.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.28.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t

10.28.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`

- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`

- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.28.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.28.4.1 Brief description

Get the driver version based on compile time macros.

10.28.4.2 Detailed description

Table 1410:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.28.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.28.5.1 Brief description

Perform any necessary initialization.

10.28.5.2 Detailed description

Table 1411:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.28.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.28.6.1 Brief description

Configure a low power mode.

10.28.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1412:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.28.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.28.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.28.7.2 Detailed description

Function will return after waking from low power mode.

Table 1413:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.28.8 API Data

10.28.8.1 lpmv2_snooze_request_t

```
lpmv2_snooze_request_t
```

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.28.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.28.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.28.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.28.8.5 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.28.8.6 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.28.9 Extensions

10.28.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.28.10 Modules

- Build Time Configurations

10.28.10.1 Build Time Configurations

Defines

- #define LPMV2_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking

10.29 LPMV2 S3A3

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.29.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.29.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t

10.29.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`

- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`

- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.29.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.29.4.1 Brief description

Get the driver version based on compile time macros.

10.29.4.2 Detailed description

Table 1414:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.29.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.29.5.1 Brief description

Perform any necessary initialization.

10.29.5.2 Detailed description

Table 1415:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.29.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.29.6.1 Brief description

Configure a low power mode.

10.29.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1416:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.29.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.29.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.29.7.2 Detailed description

Function will return after waking from low power mode.

Table 1417:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.29.8 API Data

10.29.8.1 lpmv2_snooze_request_t

```
lpmv2_snooze_request_t
```

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.29.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.29.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.29.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.29.8.5 lpmv2_output_port_enable_t

`lpmv2_output_port_enable_t`

Detailed description

Output port enable

Enumerated values

Name	Description
LPMV2_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPMV2_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

10.29.8.6 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.29.8.7 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.29.9 Extensions

10.29.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby

- lpmv2_snooze_request_t snooze_request_source

  Snooze request source

- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.29.10 Modules

- Build Time Configurations

10.29.10.1 Build Time Configurations

Defines

- #define LPMV2_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking

10.30 LPMV2 S3A6

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.30.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.30.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t

10.30.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.30.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.30.4.1 Brief description

Get the driver version based on compile time macros.

10.30.4.2 Detailed description

Table 1418:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.30.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.30.5.1 Brief description

Perform any necessary initialization.

10.30.5.2 Detailed description

Table 1419:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.30.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.30.6.1 Brief description

Configure a low power mode.

10.30.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1420:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.30.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.30.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.30.7.2 Detailed description

Function will return after waking from low power mode.

Table 1421:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.30.8 API Data

10.30.8.1 lpmv2_snooze_request_t

```
lpmv2_snooze_request_t
```

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.30.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.30.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.30.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.30.8.5 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.30.8.6 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.30.9 Extensions

10.30.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources
  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source
  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources
  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze
  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable
  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state
  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state
  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source
  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge
  Signal edges for the sources that can trigger exit from deep standby

10.31 LPMV2 S3A7

Driver for Low Power Modes.
The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.
NOTE: *Not all low power modes are available on all MCU's.*

10.31.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.31.2 Typedefs

- lpmv2_snooze_end_bits_t

- lpmv2_standby_wake_source_bits_t

10.31.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.31.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.31.4.1 Brief description

Get the driver version based on compile time macros.

10.31.4.2 Detailed description

Table 1422:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.31.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.31.5.1 Brief description

Perform any necessary initialization.

10.31.5.2 Detailed description

Table 1423:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.31.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.31.6.1 Brief description

Configure a low power mode.

10.31.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1424:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.31.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.31.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.31.7.2 Detailed description

Function will return after waking from low power mode.

Table 1425:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.31.8 API Data

10.31.8.1 lpmv2_snooze_request_t

`lpmv2_snooze_request_t`

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.31.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.

Name	Description
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.31.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.31.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.31.8.5 lpmv2_output_port_enable_t

```
lpmv2_output_port_enable_t
```

Detailed description

Output port enable

Enumerated values

Name	Description
LPMV2_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPMV2_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

10.31.8.6 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.31.8.7 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.31.9 Extensions

10.31.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.31.10 Modules

- Build Time Configurations

10.31.10.1 Build Time Configurations

Defines

- #define LPMV2_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking

10.32 LPMV2 S5D5

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.32.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.32.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t
- lpmv2_deep_standby_cancel_edge_bits_t
- lpmv2_deep_standby_cancel_source_bits_t

10.32.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`

- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`

- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.32.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.32.4.1 Brief description

Get the driver version based on compile time macros.

10.32.4.2 Detailed description

Table 1426:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.32.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.32.5.1 Brief description

Perform any necessary initialization.

10.32.5.2 Detailed description

Table 1427:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.32.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.32.6.1 Brief description

Configure a low power mode.

10.32.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1428:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.32.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.32.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.32.7.2 Detailed description

Function will return after waking from low power mode.

Table 1429:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.32.8 API Data

10.32.8.1 lpmv2_snooze_request_t

```
lpmv2_snooze_request_t
```

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.32.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.32.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.32.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.32.8.5 lpmv2_io_port_t

`lpmv2_io_port_t`

Detailed description

I/O port state after Deep Software Standby mode

Enumerated values

Name	Description
LPMV2_IO_PORT_RESET	When the Deep Software Standby mode is canceled, the I/O ports are in the reset state

Name	Description
LPMV2_IO_PORT_NO_CHANGE	When the Deep Software Standby mode is canceled, the I/O ports are in the same state as in the Deep Software Standby mode

10.32.8.6 lpmv2_power_supply_t

`lpmv2_power_supply_t`

Detailed description

Power supply control

Enumerated values

Name	Description
LPMV2_POWER_SUPPLY_DEEPCUT0	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is supplied in deep software standby mode
LPMV2_POWER_SUPPLY_DEEPCUT1	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode
LPMV2_POWER_SUPPLY_DEEPCUT3	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode. In addition, LVD is disabled and the low power function in a poweron reset circuit is enabled

10.32.8.7 lpmv2_deep_standby_cancel_edge_t

`lpmv2_deep_standby_cancel_edge_t`

Detailed description

Deep Standby Interrupt Edge

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_EDGE_NONE	No options for a deep standby cancel source.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_RISING	IRQ0-DS Pin Rising Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_FALLING	IRQ0-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_RISING	IRQ1-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_FALLING	IRQ1-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_RISING	IRQ2-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_FALLING	IRQ2-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_RISING	IRQ3-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_FALLING	IRQ3-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_RISING	IRQ4-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_FALLING	IRQ4-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_RISING	IRQ5-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_FALLING	IRQ5-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_RISING	IRQ6-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_FALLING	IRQ6-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_RISING	IRQ7-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_FALLING	IRQ7-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_RISING	IRQ8-DS Pin Rising Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_FALLING	IRQ8-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_RISING	IRQ9-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_FALLING	IRQ9-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_RISING	IRQ10-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_FALLING	IRQ10-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_RISING	IRQ11-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_FALLING	IRQ11-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_RISING	IRQ12-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_FALLING	IRQ12-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_RISING	IRQ13-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_FALLING	IRQ13-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_RISING	LVD1 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_FALLING	LVD1 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_RISING	LVD2 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_FALLING	LVD2 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_RISING	NMI Pin Rising Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_FALLING	NMI Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_RISING	IRQ14-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_FALLING	IRQ14-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_RISING	IRQ15-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_FALLING	IRQ15-DS Pin Falling Edge.

10.32.8.8 lpmv2_deep_standby_cancel_source_t

`lpmv2_deep_standby_cancel_source_t`

Detailed description

Deep Standby cancel sources

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RESET_ONLY	Cancel deep standby only by reset.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0	IRQ0.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1	IRQ1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2	IRQ2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3	IRQ3.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4	IRQ4.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5	IRQ5.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6	IRQ6.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7	IRQ7.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8	IRQ8.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9	IRQ9.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10	IRQ10.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11	IRQ11.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12	IRQ12.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13	IRQ13.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1	LVD1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2	LVD2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_INTERVAL	RTC Interval Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_ALARM	RTC Alarm Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI	NMI.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBFS	USBFS Suspend/Resume.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_AGT1	AGT1 Underflow.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14	IRQ14.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBHS	USBHS Suspend/Resume.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15	IRQ15.

10.32.8.9 lpmv2_output_port_enable_t

`lpmv2_output_port_enable_t`

Detailed description

Output port enable

Enumerated values

Name	Description
LPMV2_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPMV2_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

10.32.8.10 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.32.8.11 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.32.8.12 lpmv2_deep_standby_cancel_edge_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_edge_bits_t
```

10.32.8.13 lpmv2_deep_standby_cancel_source_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_source_bits_t
```

10.32.9 Extensions

10.32.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.33 LPMV2 S5D9

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.33.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.33.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t
- lpmv2_deep_standby_cancel_edge_bits_t
- lpmv2_deep_standby_cancel_source_bits_t

10.33.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.33.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.33.4.1 Brief description

Get the driver version based on compile time macros.

10.33.4.2 Detailed description

Table 1430:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.33.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.33.5.1 Brief description

Perform any necessary initialization.

10.33.5.2 Detailed description

Table 1431:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.33.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.33.6.1 Brief description

Configure a low power mode.

10.33.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1432:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.33.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.33.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.33.7.2 Detailed description

Function will return after waking from low power mode.

Table 1433:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.33.8 API Data

10.33.8.1 lpmv2_snooze_request_t

`lpmv2_snooze_request_t`

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.33.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.

Name	Description
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.33.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.33.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.33.8.5 lpmv2_io_port_t

`lpmv2_io_port_t`

Detailed description

I/O port state after Deep Software Standby mode

Enumerated values

Name	Description
LPMV2_IO_PORT_RESET	When the Deep Software Standby mode is canceled, the I/O ports are in the reset state
LPMV2_IO_PORT_NO_CHANGE	When the Deep Software Standby mode is canceled, the I/O ports are in the same state as in the Deep Software Standby mode

10.33.8.6 lpmv2_power_supply_t

`lpmv2_power_supply_t`

Detailed description

Power supply control

Enumerated values

Name	Description
LPMV2_POWER_SUPPLY_DEEPCUT0	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is supplied in deep software standby mode

Name	Description
LPMV2_POWER_SUPPLY_DEEPCUT1	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode
LPMV2_POWER_SUPPLY_DEEPCUT3	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode. In addition, LVD is disabled and the low power function in a poweron reset circuit is enabled

10.33.8.7 lpmv2_deep_standby_cancel_edge_t

`lpmv2_deep_standby_cancel_edge_t`

Detailed description

Deep Standby Interrupt Edge

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_EDGE_NONE	No options for a deep standby cancel source.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_RISING	IRQ0-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_FALLING	IRQ0-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_RISING	IRQ1-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_FALLING	IRQ1-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_RISING	IRQ2-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_FALLING	IRQ2-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_RISING	IRQ3-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_FALLING	IRQ3-DS Pin Falling Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_RISING	IRQ4-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_FALLING	IRQ4-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_RISING	IRQ5-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_FALLING	IRQ5-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_RISING	IRQ6-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_FALLING	IRQ6-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_RISING	IRQ7-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_FALLING	IRQ7-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_RISING	IRQ8-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_FALLING	IRQ8-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_RISING	IRQ9-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_FALLING	IRQ9-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_RISING	IRQ10-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_FALLING	IRQ10-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_RISING	IRQ11-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_FALLING	IRQ11-DS Pin Falling Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_RISING	IRQ12-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_FALLING	IRQ12-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_RISING	IRQ13-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_FALLING	IRQ13-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_RISING	LVD1 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_FALLING	LVD1 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_RISING	LVD2 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_FALLING	LVD2 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_RISING	NMI Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_FALLING	NMI Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_RISING	IRQ14-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_FALLING	IRQ14-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_RISING	IRQ15-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_FALLING	IRQ15-DS Pin Falling Edge.

10.33.8.8 lpmv2_deep_standby_cancel_source_t

`lpmv2_deep_standby_cancel_source_t`

Detailed description

Deep Standby cancel sources

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RESET_ONLY	Cancel deep standby only by reset.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0	IRQ0.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1	IRQ1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2	IRQ2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3	IRQ3.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4	IRQ4.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5	IRQ5.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6	IRQ6.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7	IRQ7.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8	IRQ8.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9	IRQ9.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10	IRQ10.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11	IRQ11.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12	IRQ12.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13	IRQ13.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1	LVD1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2	LVD2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_INTERVAL	RTC Interval Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_ALARM	RTC Alarm Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI	NMI.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBFS	USBFS Suspend/Resume.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_AGT1	AGT1 Underflow.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14	IRQ14.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBHS	USBHS Suspend/Resume.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15	IRQ15.

10.33.8.9 lpmv2_output_port_enable_t

```
lpmv2_output_port_enable_t
```

Detailed description

Output port enable

Enumerated values

Name	Description
LPMV2_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPMV2_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

10.33.8.10 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.33.8.11 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.33.8.12 lpmv2_deep_standby_cancel_edge_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_edge_bits_t
```

10.33.8.13 lpmv2_deep_standby_cancel_source_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_source_bits_t
```

10.33.9 Extensions

10.33.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby

- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.33.10 Modules

- Build Time Configurations

10.33.10.1 Build Time Configurations

Defines

- #define LPMV2_CFG_PARAM_CHECKING_ENABLE

  Initial value:`(BSP_CFG_PARAM_CHECKING_ENABLE)`

  Specify whether to include code for API parameter checking. Valid settings include:BSP_CFG_PARAM_-CHECKING_ENABLE : Utilizes the system default setting from bsp_cfg.h1 : Includes parameter checking0 : Compiles out parameter checking

10.34 LPMV2 S7G2

Driver for Low Power Modes.

The LPMV2 driver supports low power modes deep standby, standby, sleep, and snooze.

NOTE: *Not all low power modes are available on all MCU's.*

10.34.1 Functions

- R_LPMV2_VersionGet
- R_LPMV2_Init
- R_LPMV2_LowPowerConfigure
- R_LPMV2_LowPowerModeEnter

10.34.2 Typedefs

- lpmv2_snooze_end_bits_t
- lpmv2_standby_wake_source_bits_t
- lpmv2_deep_standby_cancel_edge_bits_t
- lpmv2_deep_standby_cancel_source_bits_t

10.34.3 Defines

- #define LPMV2_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define LPMV2_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define LPMV2_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &(g_lpmv2_module_name[0]), &g_lpmv2_version)`

10.34.4 R_LPMV2_VersionGet

```
ssp_err_t R_LPMV2_VersionGet ( ssp_version_t  *const p_version )
```

10.34.4.1 Brief description

Get the driver version based on compile time macros.

10.34.4.2 Detailed description

Table 1434:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_INVALID_POINTER	p_version is NULL.

10.34.5 R_LPMV2_Init

```
ssp_err_t R_LPMV2_Init (  void  )
```

10.34.5.1 Brief description

Perform any necessary initialization.

10.34.5.2 Detailed description

Table 1435:

Name	Description
SSP_SUCCESS	LPMV2 Software lock initialized

10.34.6 R_LPMV2_LowPowerConfigure

```
ssp_err_t R_LPMV2_LowPowerConfigure ( lpmv2_cfg_t  const *const p_cfg )
```

10.34.6.1 Brief description

Configure a low power mode.

10.34.6.2 Detailed description

NOTE: This function does not enter the low power mode, it only configures parameters of the mode. Execution of the WFI instruction is what causes the low power mode to be entered.

Table 1436:

Name	Description
SSP_SUCCESS	Low power mode successfuly applied
SSP_ERR_INVALID_POINTER	p_cfg is NULL
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_INVALID_HW_CONDITION	Operating mode change transition was detected (OPCMTSF, SOPCMTSF bits)

10.34.7 R_LPMV2_LowPowerModeEnter

```
ssp_err_t R_LPMV2_LowPowerModeEnter (  void  )
```

10.34.7.1 Brief description

Enter low power mode (sleep/standby/deep standby) using WFI macro.

10.34.7.2 Detailed description

Function will return after waking from low power mode.

Table 1437:

Name	Description
SSP_SUCCESS	Successful.
SSP_ERR_INVALID_HW_CONDITION	HOCO was unstable during attempt to revert operating mode.

10.34.8 API Data

10.34.8.1 lpmv2_snooze_request_t

`lpmv2_snooze_request_t`

Detailed description

Snooze request sources

Enumerated values

Name	Description
LPMV2_SNOOZE_REQUEST_RXD0_FALLING	Enable RXD0 falling edge snooze request.
LPMV2_SNOOZE_REQUEST_IRQ0	Enable IRQ0 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ1	Enable IRQ1 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ2	Enable IRQ2 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ3	Enable IRQ3 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ4	Enable IRQ4 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ5	Enable IRQ5 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ6	Enable IRQ6 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ7	Enable IRQ7 pin snooze request.
LPMV2_SNOOZE_REQUEST_KEY	Enable KR snooze request.
LPMV2_SNOOZE_REQUEST_ACMPLP	Enable Low-speed analog comparator snooze request.

Name	Description
LPMV2_SNOOZE_REQUEST_RTC_ALARM	Enable RTC alarm snooze request.
LPMV2_SNOOZE_REQUEST_RTC_PERIOD	Enable RTC period snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_UNDERFLOW	Enable AGT1 underflow snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_A	Enable AGT1 compare match A snooze request.
LPMV2_SNOOZE_REQUEST_AGT1_COMPARE_B	Enable AGT1 compare match B snooze request.
LPMV2_SNOOZE_REQUEST_IRQ8	Enable IRQ8 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ9	Enable IRQ9 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ10	Enable IRQ10 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ11	Enable IRQ11 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ12	Enable IRQ12 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ13	Enable IRQ13 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ14	Enable IRQ14 pin snooze request.
LPMV2_SNOOZE_REQUEST_IRQ15	Enable IRQ15 pin snooze request.
LPMV2_SNOOZE_REQUEST_ACMPHS0	Enable High-speed analog comparator 0 snooze request.

10.34.8.2 lpmv2_snooze_end_t

`lpmv2_snooze_end_t`

Detailed description

Snooze end control

Enumerated values

Name	Description
LPMV2_SNOOZE_END_STANDBY_WAKE_SOURCES	Transition from Snooze to Normal mode directly.
LPMV2_SNOOZE_END_AGT1_UNDERFLOW	AGT1 underflow.
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE	Last DTC transmission completion.

Name	Description
LPMV2_SNOOZE_END_DTC_TRANS_COMPLETE_NEGATED	Not Last DTC transmission completion.
LPMV2_SNOOZE_END_ADC0_COMPARE_MATCH	ADC Channel 0 compare match.
LPMV2_SNOOZE_END_ADC0_COMPARE_MISMATCH	ADC Channel 0 compare mismatch.
LPMV2_SNOOZE_END_SCI0_ADDRESS_MATCH	SCI0 address mismatch.
LPMV2_SNOOZE_END_ADC1_COMPARE_MATCH	ADC 1 compare match.
LPMV2_SNOOZE_END_ADC1_COMPARE_MISMATCH	ADC 1 compare mismatch.

10.34.8.3 lpmv2_snooze_dtc_t

`lpmv2_snooze_dtc_t`

Detailed description

DTC Enable in Snooze Mode

Enumerated values

Name	Description
LPMV2_SNOOZE_DTC_DISABLE	Disable DTC operation
LPMV2_SNOOZE_DTC_ENABLE	Enable DTC operation

10.34.8.4 lpmv2_standby_wake_source_t

`lpmv2_standby_wake_source_t`

Detailed description

Wake from standby mode sources, does not apply to sleep or deep standby modes

Enumerated values

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ0	IRQ0.
LPMV2_STANDBY_WAKE_SOURCE_IRQ1	IRQ1.
LPMV2_STANDBY_WAKE_SOURCE_IRQ2	IRQ2.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ3	IRQ3.
LPMV2_STANDBY_WAKE_SOURCE_IRQ4	IRQ4.
LPMV2_STANDBY_WAKE_SOURCE_IRQ5	IRQ5.
LPMV2_STANDBY_WAKE_SOURCE_IRQ6	IRQ6.
LPMV2_STANDBY_WAKE_SOURCE_IRQ7	IRQ7.
LPMV2_STANDBY_WAKE_SOURCE_IWDT	Independent watchdog interrupt.
LPMV2_STANDBY_WAKE_SOURCE_KEY	Key interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD1	Low Voltage Detection 1 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_LVD2	Low Voltage Detection 2 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPLP0	Analog Comparator Low-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCALM	RTC Alarm interrupt.
LPMV2_STANDBY_WAKE_SOURCE_RTCPRD	RTC Period interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBFS	USB Full-speed interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1UD	AGT1 underflow interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CA	AGT1 compare match A interrupt.
LPMV2_STANDBY_WAKE_SOURCE_AGT1CB	AGT1 compare match B interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IIC0	I2C 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_IRQ8	IRQ8.
LPMV2_STANDBY_WAKE_SOURCE_IRQ9	IRQ9.
LPMV2_STANDBY_WAKE_SOURCE_IRQ10	IRQ10.
LPMV2_STANDBY_WAKE_SOURCE_IRQ11	IRQ11.
LPMV2_STANDBY_WAKE_SOURCE_IRQ12	IRQ12.
LPMV2_STANDBY_WAKE_SOURCE_IRQ13	IRQ13.
LPMV2_STANDBY_WAKE_SOURCE_IRQ14	IRQ14.

Name	Description
LPMV2_STANDBY_WAKE_SOURCE_IRQ15	IRQ15.
LPMV2_STANDBY_WAKE_SOURCE_VBATT	VBATT Monitor interrupt.
LPMV2_STANDBY_WAKE_SOURCE_ACMPHS0	Analog Comparator High-speed 0 interrupt.
LPMV2_STANDBY_WAKE_SOURCE_USBHS	USB High-speed interrupt.

10.34.8.5 lpmv2_io_port_t

`lpmv2_io_port_t`

Detailed description

I/O port state after Deep Software Standby mode

Enumerated values

Name	Description
LPMV2_IO_PORT_RESET	When the Deep Software Standby mode is canceled, the I/O ports are in the reset state
LPMV2_IO_PORT_NO_CHANGE	When the Deep Software Standby mode is canceled, the I/O ports are in the same state as in the Deep Software Standby mode

10.34.8.6 lpmv2_power_supply_t

`lpmv2_power_supply_t`

Detailed description

Power supply control

Enumerated values

Name	Description
LPMV2_POWER_SUPPLY_DEEPCUT0	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is supplied in deep software standby mode
LPMV2_POWER_SUPPLY_DEEPCUT1	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode

Name	Description
LPMV2_POWER_SUPPLY_DEEPCUT3	Power to the standby RAM, Low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit is not supplied in deep software standby mode. In addition, LVD is disabled and the low power function in a poweron reset circuit is enabled

10.34.8.7 lpmv2_deep_standby_cancel_edge_t

`lpmv2_deep_standby_cancel_edge_t`

Detailed description

Deep Standby Interrupt Edge

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_EDGE_NONE	No options for a deep standby cancel source.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_RISING	IRQ0-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0_FALLING	IRQ0-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_RISING	IRQ1-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1_FALLING	IRQ1-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_RISING	IRQ2-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2_FALLING	IRQ2-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_RISING	IRQ3-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3_FALLING	IRQ3-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_RISING	IRQ4-DS Pin Rising Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4_FALLING	IRQ4-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_RISING	IRQ5-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5_FALLING	IRQ5-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_RISING	IRQ6-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6_FALLING	IRQ6-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_RISING	IRQ7-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7_FALLING	IRQ7-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_RISING	IRQ8-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8_FALLING	IRQ8-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_RISING	IRQ9-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9_FALLING	IRQ9-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_RISING	IRQ10-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10_FALLING	IRQ10-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_RISING	IRQ11-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11_FALLING	IRQ11-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_RISING	IRQ12-DS Pin Rising Edge.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12_FALLING	IRQ12-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_RISING	IRQ13-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13_FALLING	IRQ13-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_RISING	LVD1 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1_FALLING	LVD1 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_RISING	LVD2 Rising Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2_FALLING	LVD2 Falling Slope.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_RISING	NMI Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI_FALLING	NMI Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_RISING	IRQ14-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14_FALLING	IRQ14-DS Pin Falling Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_RISING	IRQ15-DS Pin Rising Edge.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15_FALLING	IRQ15-DS Pin Falling Edge.

10.34.8.8 lpmv2_deep_standby_cancel_source_t

`lpmv2_deep_standby_cancel_source_t`

Detailed description

Deep Standby cancel sources

Enumerated values

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RESET_ONLY	Cancel deep standby only by reset.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ0	IRQ0.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ1	IRQ1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ2	IRQ2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ3	IRQ3.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ4	IRQ4.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ5	IRQ5.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ6	IRQ6.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ7	IRQ7.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ8	IRQ8.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ9	IRQ9.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ10	IRQ10.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ11	IRQ11.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ12	IRQ12.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ13	IRQ13.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD1	LVD1.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_LVD2	LVD2.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_INTERVAL	RTC Interval Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_RTC_ALARM	RTC Alarm Interrupt.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_NMI	NMI.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBFS	USBFS Suspend/Resume.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_AGT1	AGT1 Underflow.

Name	Description
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ14	IRQ14.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_USBHS	USBHS Suspend/Resume.
LPMV2_DEEP_STANDBY_CANCEL_SOURCE_IRQ15	IRQ15.

10.34.8.9 lpmv2_output_port_enable_t

```
lpmv2_output_port_enable_t
```

Detailed description

Output port enable

Enumerated values

Name	Description
LPMV2_OUTPUT_PORT_ENABLE_HIGH_IMPEDANCE	0: In Software Standby Mode or Deep Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins are set to the high-impedance state. In Snooze, the status of the address bus and bus control signals are same as before entering Software Standby Mode.
LPMV2_OUTPUT_PORT_ENABLE_RETAIN	1: In Software Standby Mode, the address output pins, data output pins, and other bus control signal output pins retain the output state.

10.34.8.10 lpmv2_snooze_end_bits_t

```
typedef uint8_t lpmv2_snooze_end_bits_t
```

10.34.8.11 lpmv2_standby_wake_source_bits_t

```
typedef uint32_t lpmv2_standby_wake_source_bits_t
```

10.34.8.12 lpmv2_deep_standby_cancel_edge_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_edge_bits_t
```

10.34.8.13 lpmv2_deep_standby_cancel_source_bits_t

```
typedef uint32_t lpmv2_deep_standby_cancel_source_bits_t
```

10.34.9 Extensions

10.34.9.1 lpmv2_mcu_cfg_t

lpmv2_mcu_cfg_t

Detailed description

S124 specific configuration structure

S128 specific configuration structure

S3A3 specific configuration structure

S3A6 specific configuration structure

S3A7 specific configuration structure

S5D5 specific configuration structure

S5D9 specific configuration structure

S7G2 specific configuration structure

Variables

- lpmv2_standby_wake_source_bits_t standby_wake_sources

  Bitwise list of sources to wake from standby
- lpmv2_snooze_request_t snooze_request_source

  Snooze request source
- lpmv2_snooze_end_bits_t snooze_end_sources

  Bitwise list of snooze end sources
- lpmv2_snooze_dtc_t dtc_state_in_snooze

  State of DTC in snooze mode, enabled or disabled
- lpmv2_output_port_enable_t output_port_enable

  Output port enabled/disabled in standby and deep standby
- lpmv2_io_port_t io_port_state

  IO port state in deep standby (maintained or reset)
- lpmv2_power_supply_t power_supply_state

  Internal power supply state in standby and deep standby (deepcut)
- lpmv2_deep_standby_cancel_source_bits_t deep_standby_cancel_source

  Sources that can trigger exit from deep standby
- lpmv2_deep_standby_cancel_edge_bits_t deep_standby_cancel_edge

  Signal edges for the sources that can trigger exit from deep standby

10.35 LVD

Driver for Low Voltage Detection.

Implementation of the LVD API.For a detailed description see the Low Voltage Detection Interface.

10.35.1 Functions

- R_LVD_Open
- R_LVD_Close
- R_LVD_StatusGet
- R_LVD_StatusClear
- R_LVD_VersionGet
- lvd_common_isr_handler
- lvd_lvd_isr
- lvd_nmi_handler
- detection_response_configure
- detection_response_check
- negation_delay_check
- interrupt_type_configure
- lvd_open_parameter_check

10.35.2 Defines

- #define LVD_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define LVD_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.35.3 R_LVD_Open

```
ssp_err_t R_LVD_Open ( lvd_ctrl_t  *const p_api_ctrl ,  lvd_cfg_t  const
*const p_cfg )
```

10.35.3.1 Brief description

Initializes a low voltage detection driver according to the passed in configuration structure. Enables an LVD peripheral based on configuration structure.

10.35.3.2 Detailed description

Implements

- open.

NOTE: Digital filter is not to be used with standby modes

Table 1438:

Name	Description
SSP_SUCCESS	Successful
SSP_ERR_ASSERTION	One or more pointers passed as function argument point to NULL.
SSP_ERR_ASSERTION	Requested configuration of a voltage monitor was invalid.
SSP_ERR_ASSERTION	Invalid detection response.
SSP_ERR_ASSERTION	Invalid voltage threshold.
SSP_ERR_ASSERTION	Invalid monitor number.
SSP_ERR_ASSERTION	Invalid sample clock configuration.
SSP_ERR_IN_USE	Unable to acquire the hardware lock.
SSP_ERR_INVALID_MODE	MOCO must be running for LVD_NEGATION_DELAY_FROM_RESET negation delay option

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

10.35.3.3 Function steps

- Select the detection voltage by setting the LVDLVLR.LVDnLVL[m:0] bits.
- Enable voltage detection LVCMPCR.LVDnE = 1
- Enable output of the results of comparison by voltage monitor LVDnCR0.CMPE = 1
- Mark driver as opened by initializing it to "LVD" in its ASCII equivalent.

10.35.4 R_LVD_Close

```
ssp_err_t R_LVD_Close ( lvd_ctrl_t  *const p_api_ctrl )
```

10.35.4.1 Brief description

Disables the LVD peripheral. Closes the driver instance.

10.35.4.2 Detailed description

Implements

- close.

Table 1439:

Name	Description
SSP_SUCCESS	Successful
SSP_ERR_ASSERTION	Pointers passed as function argument point to NULL.
SSP_ERR_ASSERTION	Invalid monitor number

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- eventInfoGet

10.35.4.3 Function steps

- Disable voltage monitor comparison result output
- Disable reset/interrupt event
- Clear low voltage detection flag LVDnSR.DET = 0
- Disable digital filtering
- Disable voltage monitor

10.35.5 R_LVD_StatusGet

```
ssp_err_t R_LVD_StatusGet ( lvd_ctrl_t  *const p_api_ctrl ,  lvd_status_t
* p_lvd_status )
```

10.35.5.1 Brief description

Get the current state of the monitor, (threshold crossing detected, voltage currently within range)

10.35.5.2 Detailed description

Implements

- statusGet.

Table 1440:

Name	Description
SSP_SUCCESS	Successful
SSP_ERR_ASSERTION	One or more pointers passed as function argument point to NULL, invalid LVD monitor number
SSP_ERR_NOT_OPEN	Driver is not open, status will not be valid

10.35.6 R_LVD_StatusClear

```
ssp_err_t R_LVD_StatusClear ( lvd_ctrl_t  *const p_api_ctrl )
```

10.35.6.1 Brief description

Clears the latched status of the monitor.

10.35.6.2 Detailed description

Implements

- statusClear.

Table 1441:

Name	Description
SSP_SUCCESS	Successful
SSP_ERR_ASSERTION	Pointers passed as function argument point to NULL, invalid LVD monitor number
SSP_ERR_NOT_OPEN	Driver is not open, status will not be valid

10.35.7 R_LVD_VersionGet

```
ssp_err_t R_LVD_VersionGet ( ssp_version_t  *const p_version )
```

10.35.7.1 Brief description

Returns the LVD driver version based on compile time macros.

10.35.7.2 Detailed description

Implements

- versionGet.

Table 1442:

Name	Description
SSP_SUCCESS	Successful
SSP_ERR_ASSERTION	p_version was NULL

10.35.8 lvd_common_isr_handler

```
lvd_common_isr_handler ( lvd_instance_ctrl_t  * p_ctrl ,
 uint32_t monitor_number )
```

10.35.8.1 Brief description

Common code needed for all LVD ISRs.

10.35.8.2 Detailed description

Table 1443:

Name	Direction	Description
p_ctrl	in	Pointer to the control block structure for the driver instance
monitor_number	in	Identifier for the monitor which generated the interrupt

NOTE: Do not call this function directly. To be used internally by LVD driver.

10.35.8.3 Function steps

- If a callback is specified, call it

10.35.9 lvd_lvd_isr

```
lvd_lvd_isr (  void  )
```

10.35.9.1 Brief description

ISR for LVD.

10.35.9.2 Detailed description

NOTE: *Do not call this function directly. To be used internally by LVD driver.*

10.35.9.3 Function steps

- Save context if RTOS is used
- Clear the Interrupt Request
- Restore context if RTOS is used

10.35.10 lvd_nmi_handler

```
lvd_nmi_handler (  bsp_grp_irq_t irq )
```

10.35.10.1 Brief description

NMI ISR handler for LVD 1 and 2.

10.35.10.2 Detailed description

Table 1444:

Name	Direction	Description
irq	in	BSP group IRQ identifier

NOTE: Do not call this function directly. To be used internally by LVD driver.

10.35.10.3 Function steps

- Save context if RTOS is used
- Restore context if RTOS is used

10.35.11 detection_response_configure

```
 detection_response_configure ( lvd_instance_ctrl_t  * p_ctrl ,  lvd_cfg_t  const
*const p_cfg ,   IRQn_Type irq )
```

10.35.11.1 Brief description

Configure LVD monitor detection response. Internal function, do not use directly.

10.35.11.2 Detailed description

Table 1445:

Name	Direction	Description
p_ctrl	in	Pointer to the control block structure for the driver instance
p_cfg	in	Pointer to the configuration structure for the driver instance
irq	in	IRQ associated with the monitor

NOTE: Do not call this function directly. To be used internally by LVD driver.

10.35.12 detection_response_check

```
ssp_err_t detection_response_check ( lvd_cfg_t  const *const p_cfg )
```

10.35.12.1 Brief description

Verify the detection response. Internal function, do not use directly.

10.35.12.2 Detailed description

Table 1446:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for the driver instance

Table 1447:

Name	Description
SSP_SUCCESS	detection response configuration is valid
SSP_ERR_ASSERTION	detection response configuration is invalid

10.35.13 negation_delay_check

```
ssp_err_t negation_delay_check ( lvd_cfg_t  const *const p_cfg )
```

10.35.13.1 Brief description

Verify the LVD signal negation delay. Internal function, do not use directly.

10.35.13.2 Detailed description

Table 1448:

Name	Direction	Description
p_cfg	in	Pointer to the configuration structure for the driver instance

Table 1449:

Name	Description
SSP_SUCCESS	signal negation delay is valid
SSP_ERR_ASSERTION	signal negation delay is invalid

10.35.14 interrupt_type_configure

```
 interrupt_type_configure ( lvd_instance_ctrl_t  * p_ctrl ,  lvd_cfg_t  const
*const p_cfg )
```

10.35.14.1 Brief description

Configure the interrupt response type. Internal function, do not use directly.

10.35.14.2 Detailed description

Table 1450:

Name	Direction	Description
p_ctrl	in	Pointer to the control block structure for the driver instance
p_cfg	in	Pointer to the configuration structure for the driver instance

Table 1451:

Name	Description
none	

10.35.15 lvd_open_parameter_check

```
ssp_err_t lvd_open_parameter_check ( lvd_instance_ctrl_t  * p_ctrl ,  lvd_cfg_t
const *const p_cfg )
```

10.35.15.1 Brief description

Helper function to do parameter checking for the Open function for LVD module.

10.35.15.2 Detailed description

Table 1452:

Name	Direction	Description
p_ctrl	in	Pointer to the control block structure for the driver instance
p_cfg	in	Pointer to the configuration structure for the driver instance

Table 1453:

Name	Description
SSP_SUCCESS	If all parameter checks are success
SSP_ERR_ASSERTION	If any of the parameter checks fails
SSP_ERR_INVALID_MODE	If the attempted mode is invalid for this configuration

10.35.16 API Data

10.35.16.1 lvd_sample_clock_t

`lvd_sample_clock_t`

Detailed description

Sample clock divider, use LVD_SAMPLE_CLOCK_DISABLED to disable digital filtering

Enumerated values

Name	Description
LVD_SAMPLE_CLOCK_LOCO_DIV_2	Digital filter sample clock is LOCO divided by 2.
LVD_SAMPLE_CLOCK_LOCO_DIV_4	Digital filter sample clock is LOCO divided by 4.
LVD_SAMPLE_CLOCK_LOCO_DIV_8	Digital filter sample clock is LOCO divided by 8.
LVD_SAMPLE_CLOCK_LOCO_DIV_16	Digital filter sample clock is LOCO divided by 16.
LVD_SAMPLE_CLOCK_DISABLED	Digital filter is disabled.

10.35.16.2 lvd_negation_delay_t

`lvd_negation_delay_t`

Detailed description

Negation of LVD signal follows reset or voltage in range

Enumerated values

Name	Description
LVD_NEGATION_DELAY_FROM_VOLTAGE	Negation follows a stabilization time (tLVDn) after VCC > Vdet1 is detected. If a transition to software standby or deep software standby is to be made, the only possible value for the RN bit is LVD_NEGATION_DELAY_FROM_VOLTAGE
LVD_NEGATION_DELAY_FROM_RESET	Negation follows a stabilization time (tLVDn) after assertion of the LVDn reset. If a transition to software standby or deep software standby is to be made, the only possible value for the RN bit is LVD_NEGATION_DELAY_FROM_VOLTAGE

10.35.17 Extensions

10.35.17.1 lvd_instance_ctrl_t

lvd_instance_ctrl_t

Detailed description

LVD instance control structure

Variables

- uint32_t monitor_number

  Monitor number, 1, 2, ...
- R_SYSTEM_Type * p_reg

  Pointer to LVD register base address.
- void(* p_callback)(*p_args)
- lvd_callback_args_t lvd_callback_args
- uint32_t opened

10.35.17.2 lvd_extend_t

lvd_extend_t

Detailed description

Hardware extend structure

Variables

- lvd_negation_delay_t negation_delay

  Negation of LVD signal follows reset or voltage in range
- lvd_sample_clock_t sample_clock_divisor

  Sample clock divider, use LVD_SAMPLE_CLOCK_DISABLED to disable digital filtering

10.36 PDC

Driver for the Parallel Data Capture Unit (PDC).

10.36.1 Summary

extends PDC Interface The PDC interface allows the capturing of an image from a camera module.

10.36.2 Functions

- R_PDC_Open
- R_PDC_Close
- R_PDC_CaptureStart
- R_PDC_StateGet
- R_PDC_VersionGet

10.36.3 Defines

- #define PDC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define PDC_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.36.4 R_PDC_Open

```
ssp_err_t R_PDC_Open ( pdc_ctrl_t  *const p_api_ctrl ,  pdc_cfg_t  const
*const p_cfg )
```

10.36.4.1 Brief description

Powers on PDC, handles required initialization described in the hardware manual. Implements open.

10.36.4.2 Detailed description

The Open function provides initial configuration for the PDC module. It powers on the module and enables the PCLKO output and the PIXCLK input. Further initialization requires the PIXCLK input to be running in order to be able to reset the PDC as part of its initialization. This clock is input from a camera module and so the reset and further initialization is performed in captureStart. This function should be called once prior to calling any other PDC API functions. After the PDC is opened the Open function should not be called again without first calling the Close function.

Table 1454:

Name	Description
SSP_SUCCESS	Initialization was successful.
SSP_ERR_ASSERTION	One of the following parameters is incorrect. Either • p_cfg is NULL, OR • p_api_ctrl is NULL, OR • The pointer to the transfer interface in the p_cfg parameter is NULL
SSP_ERR_INVALID_ARGUMENT	One of the following configuration parameters is incorrect. Either • bytes_per_pixel is zero, OR • x_capture_pixels is zero, OR • y_capture_pixels is zero, OR • x_capture_start_pixel + x_capture_pixels is greater than 4095, OR • y_capture_start_pixel + y_capture_pixels is greater than 4095
SSP_ERR_HW_LOCKED	Unable to reserve BSP hardware lock for this module.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet

NOTE: This function is not reentrant.

10.36.4.3 Function steps

- Lock the PDC Hardware Resource
- Disable module stop mode for PDC
- Set PCLKB divider
- Enable PCLKO output
- Enable the PIXCLK input
- Mark driver as open by initializing it to "PDC" in its ASCII equivalent

10.36.5 R_PDC_Close

```
ssp_err_t R_PDC_Close ( pdc_ctrl_t  *const p_api_ctrl )
```

10.36.5.1 Brief description

Stops and closes the transfer interface, disables the PDC, powers off the PDC, clears internal driver data and disables interrupts. Implements close.

10.36.5.2 Detailed description

Table 1455:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	One of the following parameters is incorrect. Either • p_api_ctrl is NULL, OR • low level transfer is not assigned, OR • low level transfer APIs are not assigned
SSP_ERR_NOT_OPEN	Open has not been successfully called.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- disable
- close

NOTE: This API will close the PDC driver. If a capture is in progress it will be stopped. This function is reentrant.

10.36.5.3 Function steps

- Disable all interrupts.
- Enable module stop mode for PDC
- Unlock the PDC Hardware Resource

10.36.6 R_PDC_CaptureStart

```
ssp_err_t R_PDC_CaptureStart ( pdc_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_buffer )
```

10.36.6.1 Brief description

Starts a capture. Enables interrupts. Implements captureStart.

10.36.6.2 Detailed description

Sets up the transfer interface to transfer data from the PDC into the specified buffer. Configures the PDC settings as previously set by the open API. These settings are configured here as the PIXCLK input must be active for the PDC reset operation. When a capture is complete the callback registered during open API call will be called.

Table 1456:

Name	Description
SSP_SUCCESS	Capture start successful.
SSP_ERR_ASSERTION	One of the following parameters is incorrect. Either • p_api_ctrl is NULL, OR • low level transfer is not assigned, OR • low level transfer APIs are not assigned • buffer is not assigned, assign buffer
SSP_ERR_NOT_OPEN	Open has not been successfully called.
SSP_ERR_IN_USE	Pdc transfer is already in progress, wait for transfer to complete.
SSP_ERR_TIMEOUT	Reset operation timed out.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- open
- enable

NOTE: If the PIXCLK is being generated by a camera module the camera must be configured after the call to open and before the call to captureStart. This function is not reentrant.

The user is responsible to ensuring that the memory pointed to by p_buffer is both valid and large enough to store a complete image. The amount of space required, in bytes can be calculated as shown:

size (bytes) = image width (pixels) * image height (lines) * number of bytes per pixel

10.36.6.3 Function steps

- Set up transfer interface
- Configure the transfer interface
- Open transfer interface

- Wait for reset to complete
- Set horizontal capture range
- Set horizontal capture size
- Set vertical capture range
- Set vertical capture size
- Set VSYNC polarity
- Set HSYNC polarity
- Set endianess of capture data
- Enable interrupts: Receive data ready interrupt, Underrun interrupt, Overrun interrupt, Frame end interrupt, Vertical line number setting error interrupt, Horizontal byte number setting error interrupt

10.36.7 R_PDC_StateGet

```
ssp_err_t R_PDC_StateGet ( pdc_ctrl_t  *const p_api_ctrl ,  pdc_state_t
* p_state )
```

10.36.7.1 Brief description

Returns the state of the VSYNC and HSYNC pins.Implements stateGet.

10.36.7.2 Detailed description

Table 1457:

Name	Description
SSP_SUCCESS	State read successful.
SSP_ERR_ASSERTION	p_api_ctrl is NULL OR p_state is NULL
SSP_ERR_NOT_OPEN	Open has not been successfully called.

NOTE: This function is reentrant.

10.36.8 R_PDC_VersionGet

```
ssp_err_t R_PDC_VersionGet ( ssp_version_t  *const p_data )
```

10.36.8.1 Brief description

Return PDC HAL driver version. Implements versionGet.

10.36.8.2 Detailed description

Table 1458:

Name	Description
SSP_SUCCESS	Version information successfully read.
SSP_ERR_ASSERTION	Null pointer passed as a parameter

NOTE: This function is reentrant.

10.36.9 Extensions

10.36.9.1 pdc_instance_ctrl_t

pdc_instance_ctrl_t

Detailed description

PDC instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- R_PDC_Type * p_reg

  Pointer to PDC base address.
- uint32_t open

  Indicates whether or not the driver is open called.
- uint8_t bytes_per_pixel

  Number of bytes per pixel.
- uint16_t x_resolution_pixels

  Total number of horizontal pixels input to PDC.
- uint16_t y_resolution_pixels

  Total number of lines input to the PDC.
- uint16_t x_capture_start_pixel

  Horizontal position to start capture.
- uint16_t x_capture_pixels

  Number of horizontal pixels to capture.
- uint16_t y_capture_start_pixel

  Vertical position to start capture.

- uint16_t y_capture_pixels

  Number of vertical lines/pixels to capture.
- pdc_endian_t endian

  Endian of capture data.
- pdc_hsync_polarity_t hsync_polarity

  Polarity of HSYNC input.
- pdc_vsync_polarity_t vsync_polarity

  Polarity of VSYNC input.
- uint8_t * p_current_buffer

  Pointer to buffer currently in use.
- bool transfer_in_progress

  Indicates if a PDC transfer is already in progress.
- transfer_instance_t const * p_lower_lvl_transfer

  Pointer to the transfer instance the PDC should use.
- transfer_info_t info_transfer

  Transfer info structure for low level Transfer interface.
- void const * p_context

  Placeholder for user data. Passed to the user callback in pdc_callback_args_t.
- void(* p_callback)(*p_args)

  Callback provided when a PDC transfer ISR occurs.
- IRQn_Type frame_end_irq

  Frame end interrupt number.
- IRQn_Type irq

  PDC interrupt number.

10.37 QSPI

Driver for the Quad Serial Peripheral Interface (QSPI).

This is a driver for the Quad-SPI module (QSPI) which is a memory controller for connecting a serial ROM (non-volatile memory such as a serial flash memory, serial EEPROM, or serial FeRAM) that has an SPI-compatible interface.

10.37.1 Summary

Extends Quad SPI Flash Interface.

10.37.2 Functions

- R_QSPI_Open
- R_QSPI_Close
- R_QSPI_Read
- R_QSPI_PageProgram
- R_QSPI_Erase
- R_QSPI_InfoGet
- R_QSPI_SectorErase
- R_QSPI_StatusGet
- R_QSPI_BankSelect
- R_QSPI_VersionGet

10.37.3 Variables

- g_qspi_on_qspi

10.37.4 Defines

- #define QSPI_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define QSPI_CODE_VERSION_MINOR

  Initial value:`(5U)`

10.37.5 R_QSPI_Open

```
ssp_err_t R_QSPI_Open ( qspi_ctrl_t  * p_api_ctrl ,  qspi_cfg_t  const
*const p_cfg )
```

10.37.5.1 Brief description

Open the QSPI driver module.

10.37.5.2 Detailed description

Open the QSPI module driver in direct communication mode for the purposes of reading and writing flash memory via SPI protocols.

Table 1459:

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL.
SSP_ERR_UNSUPPORTED	Driver not able to query the following information from the flash manufacturer id,memory capacity and memory type.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.37.6 R_QSPI_Close

```
ssp_err_t R_QSPI_Close ( qspi_ctrl_t  * p_api_ctrl )
```

10.37.6.1 Brief description

Close the QSPI driver module.

10.37.6.2 Detailed description

Return the QSPI module back to ROM access mode.

Table 1460:

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.7 R_QSPI_Read

```
ssp_err_t R_QSPI_Read ( qspi_ctrl_t  * p_api_ctrl ,   uint8_t
* p_device_address ,   uint8_t * p_memory_address ,   uint32_t byte_count )
```

10.37.7.1 Brief description

Read data from the flash.

10.37.7.2 Detailed description

Read a block of data from a particular address on the SPI flash device.

Table 1461:

Name	Description
SSP_SUCCESS	The flash was programmed successfully.
SSP_ERR_ASSERTION	p_ctrl,p_device_address or p_memory_address is NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.8 R_QSPI_PageProgram

```
ssp_err_t R_QSPI_PageProgram ( qspi_ctrl_t  * p_api_ctrl ,   uint8_t
* p_device_address ,   uint8_t * p_memory_address ,   uint32_t byte_count )
```

10.37.8.1 Brief description

Program a page of data to the flash.

10.37.8.2 Detailed description

Table 1462:

Name	Description
SSP_SUCCESS	The flash was programmed successfully.
SSP_ERR_ASSERTION	p_ctrl,p_device_address or p_memory_address is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter is passed.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.9 R_QSPI_Erase

```
ssp_err_t R_QSPI_Erase ( qspi_ctrl_t  * p_api_ctrl ,   uint8_t
* p_device_address ,   uint32_t byte_count )
```

10.37.9.1 Brief description

Erase a number of byte from the flash.

10.37.9.2 Detailed description

Table 1463:

Name	Description
SSP_SUCCESS	The command to erase the flash was executed successfully.
SSP_ERR_ASSERTION	p_ctrl or p_device_address is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid byte_count entered.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.9.3 Function steps

- Get the information of underline flash
- If requested byte_count is supported by underline flash, assign the value of size_index to cmd_index for searching the command
- Send command to enable writing
- Close the SPI bus cycle
- Get the erase command
- Send command to erase
- Send command to write data
- Close the SPI bus cycle
- Send command to disable writing
- Close the SPI bus cycle

10.37.10 R_QSPI_InfoGet

```
ssp_err_t R_QSPI_InfoGet ( qspi_ctrl_t  * p_api_ctrl ,  qspi_info_t
*const p_info )
```

10.37.10.1 Brief description

Returns the information about the flash.

10.37.10.2 Detailed description

Table 1464:

Name	Description
SSP_SUCCESS	
SSP_ERR_ASSERTION	p_ctrl or p_info is NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.10.3 Function steps

- Get the information of underline flash

10.37.11 R_QSPI_SectorErase

```
ssp_err_t R_QSPI_SectorErase ( qspi_ctrl_t  * p_api_ctrl ,   uint8_t
* p_device_address )
```

10.37.11.1 Brief description

Erase a sector on the flash.

10.37.11.2 Detailed description

Erase one sector on the SPI flash device. Any passed in address within the sector to be erased is acceptable.

Table 1465:

Name	Description
SSP_SUCCESS	The command to erase the sector of flash was executed successfully.
SSP_ERR_ASSERTION	p_ctrl or p_device_address is NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.12 R_QSPI_StatusGet

```
ssp_err_t R_QSPI_StatusGet ( qspi_ctrl_t  * p_api_ctrl ,   bool
* p_write_in_progress )
```

10.37.12.1 Brief description

Get the write or erase status of the flash.

10.37.12.2 Detailed description

Return the write status of the flash. This is most useful for determining if erases are complete.

Table 1466:

Name	Description
SSP_SUCCESS	The write status is correct.
SSP_ERR_ASSERTION	p_ctrl or p_write_in_progress is NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.

10.37.13 R_QSPI_BankSelect

```
ssp_err_t R_QSPI_BankSelect (  uint32_t bank )
```

10.37.13.1 Brief description

Select the bank to access.

10.37.13.2 Detailed description

A bank is a 64MB sliding access window into the flash memory space. This function sets the current bank.

Table 1467:

Name	Description
SSP_SUCCESS	Bank successfully selected.

10.37.14 R_QSPI_VersionGet

```
ssp_err_t R_QSPI_VersionGet ( ssp_version_t  *const p_version )
```

10.37.14.1 Brief description

Get the driver version based on compile time macros.

10.37.14.2 Detailed description

Table 1468:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	p_version is NULL.

10.37.15 g_qspi_on_qspi

qspi_api_t::g_qspi_on_qspi

10.37.15.1 Initialized as

```
g_qspi_on_qspi=
{
    .open           = R_QSPI_Open,
    .close          = R_QSPI_Close,
    .read           = R_QSPI_Read,
    .pageProgram    = R_QSPI_PageProgram,
    .erase          = R_QSPI_Erase,
    .infoGet        = R_QSPI_InfoGet,
    .sectorErase    = R_QSPI_SectorErase,
    .statusGet      = R_QSPI_StatusGet,
    .bankSelect     = R_QSPI_BankSelect,
    .versionGet     = R_QSPI_VersionGet
}
```

10.37.16 Extensions

10.37.16.1 qspi_instance_ctrl_t

qspi_instance_ctrl_t

Detailed description

Instance control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- R_QSPI_Type * p_reg

  Pointer to QSPI base register.
- uint32_t max_eraseable_size

  Largest eraseable sector size in kbytes. Used to determine buffer size for.
- uint32_t num_address_bytes

  Number of bytes used to represent the address.
- uint32_t spi_mode

  SPI mode - 0 = Extended, 1 = Dual, 2 = Quad.
- uint32_t page_size

  Number of bytes in a programmable page.
- uint8_t manufacturer_id

  Manufacturer ID.
- uint8_t memory_type

  Memory type.
- uint8_t memory_capacity

  Memory capacity (in MByte)
- bool xip_mode

  0 = run in read mode, 1 = run in XIP mode

10.38 IIC

Driver for the I2C Bus Interface (IIC).

This module supports the Renesas Inter-Integrated Circuit (IIC) peripheral. It implements the following interfaces:

- I2C Interface r_i2c_api.h

10.38.1 Functions

- riic_notify
- riic_clock_settings
- riic_abort_seq_master
- riic_transfer_open
- r_riic_irq_cfg
- riic_param_check
- riic_open_hw_master

- riic_close_hw_master
- riic_configure_interrupts_master
- riic_abort_hw_master
- riic_enable_transfer_support_tx
- riic_enable_transfer_support_rx
- riic_run_hw_master
- riic_rxi_read_data
- riic_txi_send_address
- riic_set_valid_interrupts_priority
- riic_transfer_configure_rx
- riic_transfer_configure_tx
- riic_rxi_master
- riic_txi_master
- riic_tei_master
- riic_err_master
- R_RIIC_MasterVersionGet
- R_RIIC_MasterOpen
- R_RIIC_MasterClose
- R_RIIC_MasterRead
- R_RIIC_MasterWrite
- R_RIIC_MasterReset
- R_RIIC_MasterSlaveAddressSet

10.38.2 Variables

- g_riic_master_version
- g_i2c_master_on_riic

10.38.3 Defines

- #define RIIC_MASTER_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define RIIC_MASTER_CODE_VERSION_MINOR

  Initial value:`(8U)`

- #define RIIC_OPEN

  Initial value:`(0x52494943ULL)`

  "RIIC" in ASCII, used to determine if channel is open.
- #define VARIANT_FMPLUS_MASK

  Initial value:`0x04U`
- #define RIIC_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &g_riic_master_version)`

  Macro for error logger.
- #define I2C_CODE_READ

  Initial value:`(0x01U)`
- #define I2C_CODE_10BIT

  Initial value:`(0xF0U)`

10.38.4 riic_notify

```
riic_notify ( riic_instance_ctrl_t  *const p_ctrl ,  i2c_event_t  const event )
```

10.38.4.1 Brief description

Single point for managing the logic around notifying a transfer has finished.

10.38.4.2 Detailed description

Internal helper functions
(end addtogroup RIIC)

Table 1469:

Name	Direction	Description
p_ctrl	in	Pointer to transfer that is ending.
event	in	The event code to pass to the callback.

10.38.4.3 Function steps

- Set the flag indicating that the transaction is completed

10.38.5 riic_clock_settings

```
ssp_err_t riic_clock_settings ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.5.1 Brief description

Configures the clock and filter settings.

10.38.5.2 Detailed description

Table 1470:

Name	Direction	Description
p_ctrl	in	Pointer to RIIC software control block.

Table 1471:

Name	Description
SSP_SUCCESS	Successfully configured the clock
SSP_ERR_INVALID_RATE	Failed to configure the clock settings for this rate

10.38.6 riic_abort_seq_master

```
ssp_err_t riic_abort_seq_master ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.6.1 Brief description

Single point for managing the logic around aborting a transfer when operating as a master.

10.38.6.2 Detailed description

Table 1472:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

Table 1473:

Name	Description
SSP_SUCCESS	Stopped any DTC assisted transfer.
SSP_ERR_ABORTED	If a transfer is in-progress.

10.38.7 riic_transfer_open

```
ssp_err_t riic_transfer_open ( riic_instance_ctrl_t * p_ctrl , i2c_cfg_t const
*const p_cfg )
```

10.38.7.1 Brief description

Configures RIIC related transfer drivers (if enabled).

10.38.7.2 Detailed description

Table 1474:

Name	Direction	Description
p_ctrl	in	Pointer to IIC specific control structure
p_cfg	in	Pointer to IIC specific configuration structure

Table 1475:

Name	Description
SSP_SUCCESS	Transfer interface initialized successfully.
SSP_ERR_ASSERTION	Pointer to transfer instance for I2C receive in p_cfg is NULL.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.

See Common Error Codes for other possible return codes. This function internally calls

- productFeatureGet

10.38.8 r_riic_irq_cfg

```
ssp_err_t r_riic_irq_cfg ( ssp_feature_t  * p_feature ,   ssp_signal_t signal ,
  uint8_t ipl ,   void * p_ctrl ,   IRQn_Type * p_irq )
```

10.38.8.1 Brief description

Sets interrupt priority and initializes vector info.

10.38.8.2 Detailed description

Table 1476:

Name	Direction	Description
p_feature	in	SSP feature
signal	in	SSP signal ID
ipl	in	Interrupt priority level
p_ctrl	in	Pointer to driver control block
p_irq	out	Pointer to IRQ for this signal, set here

Table 1477:

Name	Description
SSP_SUCCESS	Interrupt enabled
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table

See Common Error Codes for other possible return codes. This function calls

- eventInfoGet

10.38.9 riic_param_check

```
ssp_err_t riic_param_check ( riic_instance_ctrl_t  *const p_ctrl ,  i2c_cfg_t
const *const p_cfg )
```

10.38.9.1 Brief description

Parameter check.

10.38.9.2 Detailed description

Functions that manipulate hardware

Table 1478:

Name	Direction	Description
p_ctrl	in	Pointer to IIC specific control structure
p_cfg	in	Pointer to IIC specific configuration structure

Table 1479:

Name	Description
SSP_SUCCESS	Provided parameters not NULL.
SSP_ERR_ASSERTION	The parameter p_api_ctrl or p_cfg is NULL or clock rate is greater than 1MHz. or p_cfg->p_extend not equal to NULL.

10.38.9.3 Function steps

- p_extend not supported currently

10.38.10 riic_open_hw_master

```
ssp_err_t riic_open_hw_master ( riic_instance_ctrl_t  *const p_ctrl ,
 ssp_feature_t  * p_feature )
```

10.38.10.1 Brief description

Performs the hardware initialization sequence when operating as a master.

10.38.10.2 Detailed description

Table 1480:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

Table 1480: (Continued)

Name	Direction	Description
p_feature	in	SSP Feature

Table 1481:

Name	Description
SSP_SUCCESS	Hardware initialized with proper configurations.
SSP_ERR_INVALID_RATE	The requested rate could not be set.

10.38.11 riic_close_hw_master

```
riic_close_hw_master ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.11.1 Brief description

Performs the hardware initialization sequence when operating as a master.

10.38.11.2 Detailed description

Table 1482:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

10.38.12 riic_configure_interrupts_master

```
riic_configure_interrupts_master ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.12.1 Brief description

Enables and assigns the interrupts to be used in master mode.

10.38.12.2 Detailed description

Table 1483:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

10.38.13 riic_abort_hw_master

```
riic_abort_hw_master ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.13.1 Brief description

Safely stops the current data transfer when operating as a master.

10.38.13.2 Detailed description

Table 1484:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

10.38.14 riic_enable_transfer_support_tx

```
riic_enable_transfer_support_tx ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.14.1 Brief description

Enables the dtc transfer interface for the transmit operation.

10.38.14.2 Detailed description

Table 1485:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.38.15 riic_enable_transfer_support_rx

```
riic_enable_transfer_support_rx ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.15.1 Brief description

Enables the dtc transfer interface for the receive operation.

10.38.15.2 Detailed description

Table 1486:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.38.16 riic_run_hw_master

```
ssp_err_t riic_run_hw_master ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.16.1 Brief description

Performs the data transfer described by the parameters when operating as a master.

10.38.16.2 Detailed description

Table 1487:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device.

Table 1488:

Name	Description
SSP_SUCCESS	Data transfer success.
SSP_ERR_IN_USE	If data transfer is in progress.

Table 1488: (Continued)

Name	Description
SSP_ERR_ABORTED	If an error occurred while data transfer.

10.38.17 riic_rxi_read_data

```
riic_rxi_read_data ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.17.1 Brief description

Check valid receive data and set WAIT, NACK and STOP/RESTART bit in RXI handler.

10.38.17.2 Detailed description

Table 1489:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.38.18 riic_txi_send_address

```
riic_txi_send_address ( riic_instance_ctrl_t  *const p_ctrl )
```

10.38.18.1 Brief description

Write the address byte to the riic bus.

10.38.18.2 Detailed description

Table 1490:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.38.19 riic_set_valid_interrupts_priority

```
ssp_err_t riic_set_valid_interrupts_priority ( riic_instance_ctrl_t  * p_ctrl ,
 i2c_cfg_t  const *const p_cfg )
```

10.38.19.1 Brief description

Set valid interrupts priority.

10.38.19.2 Detailed description

Table 1491:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device
p_cfg	in	Pointer to IIC specific configuration structure

Table 1492:

Name	Description
SSP_SUCCESS	Interrupts enabled.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet

10.38.20 riic_transfer_configure_rx

```
ssp_err_t riic_transfer_configure_rx ( riic_instance_ctrl_t  * p_ctrl ,
 i2c_cfg_t  const *const p_cfg )
```

10.38.20.1 Brief description

Configures RIIC RX related transfer.

10.38.20.2 Detailed description

Table 1493:

Name	Direction	Description
p_ctrl	in	Pointer to IIC specific control structure
p_cfg	in	Pointer to IIC specific configuration structure

Table 1494:

Name	Description
SSP_SUCCESS	rx transfer interface in configured with valid parameters.
SSP_ERR_ASSERTION	Pointer to transfer instance for I2C receive in p_cfg is NULL.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet

10.38.20.3 Function steps

- Set default transfer info and open receive transfer module, if enabled.

10.38.21 riic_transfer_configure_tx

```
ssp_err_t riic_transfer_configure_tx ( riic_instance_ctrl_t  * p_ctrl ,
 i2c_cfg_t  const *const p_cfg )
```

10.38.21.1 Brief description

Configures RIIC TX related transfer.

10.38.21.2 Detailed description

Table 1495:

Name	Direction	Description
p_ctrl	in	Pointer to IIC specific control structure
p_cfg	in	Pointer to IIC specific configuration structure

Table 1496:

Name	Description
SSP_SUCCESS	tx transfer interface in configured with valid parameters.
SSP_ERR_ASSERTION	Pointer to transfer instance for I2C transmit in p_cfg is NULL.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet

10.38.21.3 Function steps

- Set default transfer info and open transmit transfer module, if enabled.

10.38.22 riic_rxi_master

```
riic_rxi_master ( riic_instance_ctrl_t  * p_ctrl )
```

10.38.22.1 Brief description

Handles the receive data full interrupt when operating as a master.

10.38.22.2 Detailed description

Interrupt handlers

Table 1497:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.38.23 riic_txi_master

```
riic_txi_master ( riic_instance_ctrl_t  * p_ctrl )
```

10.38.23.1 Brief description

Handles the transmit data empty interrupt when operating as a master.

10.38.23.2 Detailed description

Table 1498:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.38.24 riic_tei_master

```
riic_tei_master ( riic_instance_ctrl_t  * p_ctrl )
```

10.38.24.1 Brief description

Handles the transmit end interrupt when operating as a master.

10.38.24.2 Detailed description

NOTE: This interrupt is configured to be generated at the end of last byte of the requested transfer.

Table 1499:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.38.25 riic_err_master

```
riic_err_master ( riic_instance_ctrl_t  * p_ctrl )
```

10.38.25.1 Brief description

Handles the error interrupts when operating as a master.

10.38.25.2 Detailed description

Table 1500:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.38.26 R_RIIC_MasterVersionGet

```
ssp_err_t R_RIIC_MasterVersionGet ( ssp_version_t  *const p_version )
```

10.38.26.1 Brief description

Gets version information and stores it in the provided version struct.

10.38.26.2 Detailed description

Table 1501:

Name	Description
SSP_SUCCESS	Successful version get.
SSP_ERR_ASSERTION	p_version is NULL.

10.38.27 R_RIIC_MasterOpen

```
ssp_err_t R_RIIC_MasterOpen ( i2c_ctrl_t  *const p_api_ctrl ,  i2c_cfg_t  const
*const p_cfg )
```

10.38.27.1 Brief description

Opens the I2C device. May power on IIC peripheral and perform initialization described in hardware manual.

10.38.27.2 Detailed description

This function will reconfigure the clock settings of the peripheral when a device with a lower rate than previously configured is opened.

Table 1502:

Name	Description
SSP_SUCCESS	Requested clock rate was set exactly.
SSP_ERR_ASSERTION	The parameter p_api_ctrl or p_cfg is NULL or clock rate is greater than 1MHz. or p_cfg->p_extend not equal to NULL.
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_INVALID_ARGUMENT	If fast mode plus is configured and the channel does not support it
SSP_ERR_INVALID_RATE	The requested rate cannot be set.

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet
- g_cgc_on_cgc.systemClockFreqGet

10.38.28 R_RIIC_MasterClose

```
ssp_err_t R_RIIC_MasterClose ( i2c_ctrl_t  *const p_api_ctrl )
```

10.38.28.1 Brief description

Closes the I2C device. May power down IIC peripheral.

10.38.28.2 Detailed description

This function will safely terminate any in-progress I2C transfer with the device. If a transfer is aborted, the user will be notified via callback with an abort event. Since the callback is optional, this function will also return a specific error code in this situation.

Table 1503:

Name	Description
SSP_SUCCESS	Device closed without issue.

Table 1503: (Continued)

Name	Description
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.

10.38.29 R_RIIC_MasterRead

```
ssp_err_t R_RIIC_MasterRead ( i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   bool const restart )
```

10.38.29.1 Brief description

Performs a read from the I2C device.

10.38.29.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C read operation will begin. When no callback is provided by the user, this function performs a blocking read. Otherwise, the read operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_RX_COMPLETE in the callback.

Table 1504:

Name	Description
SSP_SUCCESS	Function executed without issue, if no callback was provided, the process was kicked off.
SSP_ERR_ASSERTION	p_api_ctrl, p_dest or bytes is NULL.
SSP_ERR_INVALID_SIZE	Provided number of bytes more than uint16_t size(65535) while DTC is used for data transfer.
SSP_ERR_IN_USE	Another transfer was in progress.
SSP_ERR_ABORTED	The transfer failed.

10.38.30 R_RIIC_MasterWrite

```
ssp_err_t R_RIIC_MasterWrite ( i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_src ,   uint32_t const bytes ,   bool const restart )
```

10.38.30.1 Brief description

Performs a write to the I2C device.

10.38.30.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C write operation will begin. When no callback is provided by the user, this function performs a blocking write. Otherwise, the write operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_TX_COMPLETE in the callback.

Table 1505:

Name	Description
SSP_SUCCESS	Function executed without issue, if no callback was provided, the process was kicked off.
SSP_ERR_ASSERTION	p_api_ctrl or p_src is NULL.
SSP_ERR_INVALID_SIZE	Provided number of bytes more than uint16_t size(65535) while DTC is used for data transfer.
SSP_ERR_IN_USE	Another transfer was in progress.
SSP_ERR_ABORTED	The transfer failed.

10.38.31 R_RIIC_MasterReset

```
ssp_err_t R_RIIC_MasterReset ( i2c_ctrl_t  *const p_api_ctrl )
```

10.38.31.1 Brief description

Aborts any in-progress transfer and forces the IIC peripheral into a ready state.

10.38.31.2 Detailed description

This function will safely terminate any in-progress I2C transfer with the device. If a transfer is aborted, the user will be notified via callback with an abort event. Since the callback is optional, this function will also return a specific error code in this situation.

Table 1506:

Name	Description
SSP_SUCCESS	Channel was reset without issue.

Table 1506: (Continued)

Name	Description
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_ABORTED	A transfer was aborted while resetting the hardware.

10.38.32 R_RIIC_MasterSlaveAddressSet

```
ssp_err_t R_RIIC_MasterSlaveAddressSet ( i2c_ctrl_t  *const p_api_ctrl ,
  uint16_t const slave ,  i2c_addr_mode_t  const addr_mode )
```

10.38.32.1 Brief description

Sets address and addressing mode of the slave device.

10.38.32.2 Detailed description

This function is used to set the device address and addressing mode of the slave without reconfiguring the entire bus.

Table 1507:

Name	Description
SSP_SUCCESS	Address of the slave is set correctly.
SSP_ERR_ASSERTION	Pointer to control structure is NULL.
SSP_ERR_IN_USE	Another transfer was in-progress.
SSP_ERR_NOT_OPEN	Device was not even opened.

10.38.33 g_riic_master_version

```
ssp_version_t::g_riic_master_version
```

10.38.33.1 Detailed description

Name of module used by error logger macro Version data structure used by error logger macro.

10.38.33.2 Initialized as

```
g_riic_master_version=
{
```

```
    .api_version_minor  = I2C_MASTER_API_VERSION_MINOR,
    .api_version_major  = I2C_MASTER_API_VERSION_MAJOR,
    .code_version_major = RIIC_MASTER_CODE_VERSION_MAJOR,
    .code_version_minor = RIIC_MASTER_CODE_VERSION_MINOR
}
```

10.38.34 g_i2c_master_on_riic

i2c_api_master_t::g_i2c_master_on_riic

10.38.34.1 Detailed description

RIIC Implementation of I2C device master interface

10.38.34.2 Initialized as

```
g_i2c_master_on_riic=
{
    .open            = R_RIIC_MasterOpen,
    .close           = R_RIIC_MasterClose,
    .read            = R_RIIC_MasterRead,
    .write           = R_RIIC_MasterWrite,
    .reset           = R_RIIC_MasterReset,
    .versionGet      = R_RIIC_MasterVersionGet,
    .slaveAddressSet = R_RIIC_MasterSlaveAddressSet
}
```

10.38.35 Extensions

10.38.35.1 riic_instance_ctrl_t

riic_instance_ctrl_t

Detailed description

I2C control structure. DO NOT INITIALIZE.

Variables

- i2c_cfg_t info

  Information describing I2C device.

- uint32_t open

  Flag to determine if the device is open.

- void * p_reg

  Base register for this channel.

- IRQn_Type rxi_irq

  Receive IRQ number.
- IRQn_Type txi_irq

  Transmit IRQ number.
- IRQn_Type tei_irq

  Transmit end IRQ number.
- IRQn_Type eri_irq

  Error IRQ number.
- uint8_t * p_buff

  Holds the data associated with the transfer
- uint32_t total

  Holds the total number of data bytes to transfer
- uint32_t remain

  Tracks the remaining data bytes to transfer
- uint32_t loaded

  Tracks the number of data bytes written to the register
- uint8_t addr_low

  Holds the last address byte to issue
- uint8_t addr_high

  Holds the first address byte to issue in 10-bit mode
- uint8_t addr_total

  Holds the total number of address bytes to transfer
- uint8_t addr_remain

  Tracks the remaining address bytes to transfer
- uint8_t addr_loaded

  Tracks the number of address bytes written to the register
- bool read

  Holds the direction of the data byte transfer
- bool restart

  Holds whether or not the restart should be issued when done
- bool err

  Tracks whether or not an error occurred during processing
- bool restarted

  Tracks whether or not a restart was issued during the previous transfer

- bool transaction_completed

  Tracks if the transaction started earlier was completed
- bool dummy_read_completed

  Tracks whether the dummy read is performed
- bool activation_on_rxi

  < Tracks whether the transfer is activated on RXI interrupt
- bool activation_on_txi

  < Tracks whether the transfer is activated on TXI interrupt
- bool address_restarted

  < Tracks whether the restart condition is send on 10 bit read

10.39 IIC Slave

Driver for the I2C Bus Slave Interface (IIC Slave).

This module supports the Renesas Inter-Integrated Circuit (IIC) peripheral. It implements the following interfaces:

- I2C Interface r_i2c_api.h

10.39.1 Functions

- riic_notify
- riic_slave_set_irq_parameters
- riic_open_hw_slave
- riic_close_hw_slave
- riic_run_hw_slave
- riic_slave_clock_settings
- riic_clear_all_pending_interrupts
- riic_slave_configure_interrupts
- riic_rxi_slave
- riic_txi_slave
- riic_err_slave
- R_RIIC_SlaveVersionGet
- R_RIIC_SlaveOpen
- R_RIIC_SlaveClose
- R_RIIC_MasterWriteSlaveRead

- R_RIIC_MasterReadSlaveWrite

10.39.2 Variables

- g_riic_slave_version
- g_i2c_slave_on_riic

10.39.3 Defines

- #define RIIC_SLAVE_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define RIIC_SLAVE_CODE_VERSION_MINOR

  Initial value:`(3U)`
- #define RIIC_SLAVE_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], &g_riic_slave_version)`

  Macro for error logger.
- #define RIIC_SLAVE_OPEN

  Initial value:`(0x49324353ULL)`

  "I2CS" in ASCII, used to determine if channel is open.
- #define MAX_CKS_DIVISOR

  Initial value:`7U`
- #define MAX_BRCL_VALUE

  Initial value:`31U`

10.39.4 riic_notify

```
 riic_notify ( riic_slave_instance_ctrl_t  *const p_ctrl ,  i2c_event_t
const event )
```

10.39.4.1 Brief description

Single point for managing the logic around notifying a transfer has finished.

10.39.4.2 Detailed description

Internal helper functions

(end addtogroup RIIC_SLAVE)

Table 1508:

Name	Direction	Description
p_ctrl	in	Pointer to transfer that is ending.
event	in	The event code to pass to the callback.

10.39.5 riic_slave_set_irq_parameters

```
ssp_err_t riic_slave_set_irq_parameters ( ssp_feature_t  * p_feature ,
  ssp_signal_t signal ,   uint8_t ipl ,   void * p_ctrl ,   IRQn_Type * p_irq )
```

10.39.5.1 Detailed description

Sets interrupt priority and initializes vector info

Table 1509:

Name	Direction	Description
p_feature	in	SSP feature
signal	in	SSP signal ID
ipl	in	Interrupt priority level
p_ctrl	in	Pointer to driver control block
p_irq	out	Pointer to IRQ for this signal, set here

Table 1510:

Name	Description
SSP_SUCCESS	Interrupt enabled
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table

See Common Error Codes for other possible return codes. This function calls

- eventInfoGet

10.39.6 riic_open_hw_slave

```
riic_open_hw_slave ( riic_slave_instance_ctrl_t  *const p_ctrl ,
ssp_feature_t  * p_feature )
```

10.39.6.1 Brief description

Performs the hardware initialization sequence when operating as a slave.

10.39.6.2 Detailed description

Functions that manipulate hardware

Table 1511:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device
p_feature	in	

10.39.7 riic_close_hw_slave

```
riic_close_hw_slave ( riic_slave_instance_ctrl_t  *const p_ctrl )
```

10.39.7.1 Brief description

Performs the hardware initialization sequence when operating as a slave.

10.39.7.2 Detailed description

Table 1512:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

10.39.8 riic_run_hw_slave

```
ssp_err_t riic_run_hw_slave ( riic_slave_instance_ctrl_t  *const p_ctrl )
```

10.39.8.1 Brief description

Performs the data transfer described by the parameters when operating as a slave.

10.39.8.2 Detailed description

Table 1513:

Name	Direction	Description
p_ctrl	in	Pointer to transfer that needs to be done.

Table 1514:

Name	Description
SSP_SUCCESS	Transaction completed successfully.
SSP_ERR_ABORTED	If transaction encounter an error.

10.39.9 riic_slave_clock_settings

```
riic_slave_clock_settings ( riic_slave_instance_ctrl_t  *const p_ctrl )
```

10.39.9.1 Brief description

Sets the I2C clock and BRCL counter to a value greater than the operation mode setup time.

10.39.9.2 Detailed description

Table 1515:

Name	Direction	Description
p_ctrl	in	Pointer to driver control block

10.39.10 riic_clear_all_pending_interrupts

```
riic_clear_all_pending_interrupts ( riic_slave_instance_ctrl_t  *const p_ctrl )
```

10.39.10.1 Brief description

Enables and assigns the interrupts to be used in slave mode.

10.39.10.2 Detailed description

Table 1516:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

10.39.11 riic_slave_configure_interrupts

```
ssp_err_t riic_slave_configure_interrupts ( ssp_feature_t  * ssp_feature ,
 riic_slave_instance_ctrl_t  *const p_api_ctrl ,  i2c_cfg_t  const *const p_cfg )
```

10.39.11.1 Detailed description

Sets interrupt priority and initializes vector info

Table 1517:

Name	Direction	Description
ssp_feature	in	SSP feature
p_ctrl	in	Pointer to driver control block
p_cfg	out	Pointer to driver configuration block

Table 1518:

Name	Description
SSP_SUCCESS	Interrupt enabled.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.

10.39.12 riic_rxi_slave

```
riic_rxi_slave ( riic_slave_instance_ctrl_t  * p_ctrl )
```

10.39.12.1 Brief description

Handles the receive data full interrupt when operating as a slave.

10.39.12.2 Detailed description

Interrupt handlers

Table 1519:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.39.13 riic_txi_slave

```
riic_txi_slave ( riic_slave_instance_ctrl_t  * p_ctrl )
```

10.39.13.1 Brief description

Handles the transmit data empty interrupt when operating as a slave.

10.39.13.2 Detailed description

Table 1520:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.39.14 riic_err_slave

```
riic_err_slave ( riic_slave_instance_ctrl_t  * p_ctrl )
```

10.39.14.1 Brief description

Handles the error interrupts when operating as a slave.

10.39.14.2 Detailed description

Table 1521:

Name	Direction	Description
p_ctrl	in	The target RIIC block's control block.

10.39.15 R_RIIC_SlaveVersionGet

```
ssp_err_t R_RIIC_SlaveVersionGet ( ssp_version_t  *const p_version )
```

10.39.15.1 Brief description

Gets version information and stores it in the provided version struct.

10.39.15.2 Detailed description

Table 1522:

Name	Description
SSP_SUCCESS	Successful version get.
SSP_ERR_ASSERTION	p_version is NULL.

10.39.16 R_RIIC_SlaveOpen

```
ssp_err_t R_RIIC_SlaveOpen ( i2c_ctrl_t  *const p_api_ctrl ,  i2c_cfg_t  const
*const p_cfg )
```

10.39.16.1 Brief description

Opens the I2C device. May power on IIC peripheral and perform initialization described in hardware manual.

10.39.16.2 Detailed description

Table 1523:

Name	Description
SSP_SUCCESS	Opened identical configuration of already open instance.
SSP_ERR_ASSERTION	p_api_ctrl or p_cfg is NULL.
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table.
SSP_ERR_INVALID_ARGUMENT	If fast mode plus is configured and the channel does not support it

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet
- g_cgc_on_cgc.systemClockFreqGet

10.39.17 R_RIIC_SlaveClose

```
ssp_err_t R_RIIC_SlaveClose ( i2c_ctrl_t  *const p_api_ctrl )
```

10.39.17.1 Brief description

Closes the I2C device. Power down IIC peripheral.

10.39.17.2 Detailed description

Table 1524:

Name	Description
SSP_SUCCESS	Device closed without issue.
SSP_ERR_NOT_OPEN	Device not opened.
SSP_ERR_ASSERTION	p_api_ctrl is NULL.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.

10.39.18 R_RIIC_MasterWriteSlaveRead

```
ssp_err_t R_RIIC_MasterWriteSlaveRead ( i2c_ctrl_t  *const p_api_ctrl ,
  uint8_t *const p_dest ,   uint32_t const bytes )
```

10.39.18.1 Brief description

Performs a read from the I2C Master device.

10.39.18.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C read operation will begin. When no callback is provided by the user, this function performs a blocking read. Otherwise, the read operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_RX_COMPLETE in the callback.

Table 1525:

Name	Description
SSP_SUCCESS	Function executed without issue; if no callback was provided, the process was kicked off
SSP_ERR_ASSERTION	p_api_ctrl, bytes or p_dest is NULL.
SSP_ERR_IN_USE	Another transfer was in progress.
SSP_ERR_NOT_OPEN	device is not open.
SSP_ERR_ABORTED	If transaction encounter an error.

10.39.19 R_RIIC_MasterReadSlaveWrite

```
ssp_err_t R_RIIC_MasterReadSlaveWrite ( i2c_ctrl_t  *const p_api_ctrl ,
  uint8_t *const p_src ,   uint32_t const bytes )
```

10.39.19.1 Brief description

Performs a write to the I2C Master device.

10.39.19.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C write operation will begin. When no callback is provided by the user, this function performs a blocking write. Otherwise, the write operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_TX_COMPLETE in the callback.

Table 1526:

Name	Description
SSP_SUCCESS	Function executed without issue; if no callback was provided, the process was kicked off
SSP_ERR_ASSERTION	p_api_ctrl or p_src is NULL.
SSP_ERR_IN_USE	Another transfer was in progress.
SSP_ERR_NOT_OPEN	device is not open.
SSP_ERR_ABORTED	If transaction encounter an error.

10.39.20 g_riic_slave_version

`ssp_version_t::g_riic_slave_version`

10.39.20.1 Detailed description

Name of module used by error logger macro Version data structure used by error logger macro.

10.39.20.2 Initialized as

```
g_riic_slave_version=
{
    .api_version_minor  = I2C_MASTER_API_VERSION_MINOR,
    .api_version_major  = I2C_MASTER_API_VERSION_MAJOR,
    .code_version_major = RIIC_SLAVE_CODE_VERSION_MAJOR,
    .code_version_minor = RIIC_SLAVE_CODE_VERSION_MINOR
}
```

10.39.21 g_i2c_slave_on_riic

`i2c_api_slave_t::g_i2c_slave_on_riic`

10.39.21.1 Detailed description

RIIC Implementation of I2C device slave interface

10.39.21.2 Initialized as

```
g_i2c_slave_on_riic=
{
    .open                       = R_RIIC_SlaveOpen,
    .close                      = R_RIIC_SlaveClose,
    .masterWriteSlaveRead       = R_RIIC_MasterWriteSlaveRead,
    .masterReadSlaveWrite       = R_RIIC_MasterReadSlaveWrite,
    .versionGet                 = R_RIIC_SlaveVersionGet
}
```

10.39.22 Extensions

10.39.22.1 riic_slave_instance_ctrl_t

riic_slave_instance_ctrl_t

Detailed description

I2C control structure. DO NOT INITIALIZE.

Variables

- i2c_cfg_t info

  Information describing I2C device.

- uint32_t open

  Flag to determine if the device is open.

- void * p_reg

  Base register for this channel.

- IRQn_Type rxi_irq

  Receive IRQ number.

- IRQn_Type txi_irq

  Transmit IRQ number.

- IRQn_Type eri_irq

  Error IRQ number.

- uint8_t * p_buff

  Holds the data associated with the transfer

- uint32_t total

  Holds the total number of data bytes to transfer

- uint32_t remain

  Tracks the remaining data bytes to transfer

- uint32_t loaded

  Tracks the number of data bytes written to the register
- uint32_t transaction_count

  Tracks the actual number of transactions
- bool read

  Holds the direction of the data byte transfer
- bool err

  Tracks whether or not an error occurred during processing
- bool slave_busy

  Tracks if the slave is busy performing a transaction
- bool do_dummy_read
- bool start_interrupt_enabled

  << Tracks whether a dummy read is issued on the first RX < Tracks whether the start interrupt is enabled
- bool restarted

10.40 SPI

Driver for the Serial Peripheral Interface (SPI).

This module supports SPI serial communication for the SPI module. The SPI Interface is defined in r_spi_api.h

10.40.1 Functions

- R_RSPI_Open
- R_RSPI_Read
- R_RSPI_Write
- R_RSPI_WriteRead
- rspi_write_read_common
- R_RSPI_Close
- rspi_baud_set
- R_RSPI_VersionGet

10.40.2 Defines

- #define RSPI_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define RSPI_CODE_VERSION_MINOR

  Initial value:`(8U)`

10.40.3 R_RSPI_Open

```
ssp_err_t R_RSPI_Open ( spi_ctrl_t  * p_api_ctrl ,  spi_cfg_t  const
*const p_cfg )
```

10.40.3.1 Brief description

This functions initializes a channel for SPI communication mode.

10.40.3.2 Detailed description

Implements open This function performs the following tasks: Performs parameter checking and processes error conditions. Applies power to the SPI channel. Disables interrupts. Initializes the associated registers with some default value and the user-configurable options. Provides the channel control for use with other API functions. Updates user-configurable file if necessary.

Table 1527:

Name	Description
SSP_SUCCESS	Channel initialized successfully.
SSP_ERR_ASSERTION	NULL pointer to following parameters p_ctrl, p_cfg, p_cfg::p_transfer_rx::p_api, p_cfg::p_transfer_rx::p_ctrl, p_cfg::p_transfer_rx::p_cfg, p_cfg::p_transfer_rx::p_cfg::p_info. or failed to set the baud rate,
SSP_ERR_INVALID_ARGUMENT	An element of the r_spi_cfg_t structure contains an invalid value. The parameters is out of range. Both transfer modules need to be present or absent.
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- open
- productFeatureGet

- eventInfoGet

NOTE: This function is reentrant.

10.40.4 R_RSPI_Read

```
ssp_err_t R_RSPI_Read ( spi_ctrl_t  *const p_api_ctrl ,   void const * p_dest ,
  uint32_t const length ,  spi_bit_width_t  const bit_width )
```

10.40.4.1 Brief description

This function receives data from a SPI device.

10.40.4.2 Detailed description

Implements read The function performs the following tasks: Performs parameter checking and processes error conditions. Disable Interrupts. Disable the SPI bus. Setup data bit width per user request. Enable the SPI bus. Enable interrupts. Start data transmission with dummy data via transmit buffer empty interrupt. Copy data from source buffer to the SPI data register for transmission. Receive data from receive buffer full interrupt occurs and copy data to the buffer of destination. Complete data reception via receive buffer full interrupt and transmitting dummy data.

Table 1528:

Name	Description
SSP_SUCCESS	Read operation successfully completed.
SSP_ERR_ASSERTION	NULL pointer to control or destination parameters or transfer length is zero.
SSP_ERR_UNSUPPORTED	With DTC transfer mode, bit_width must match configured DTC transfer width
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open channel first.
SSP_ERR_INVALID_HW_CONDITION	Failed to clear errors in the module

NOTE: This function is reentrant.

10.40.5 R_RSPI_Write

```
ssp_err_t R_RSPI_Write ( spi_ctrl_t  *const p_api_ctrl ,   void const * p_src ,
  uint32_t const length ,  spi_bit_width_t  const bit_width )
```

10.40.5.1 Brief description

This function transmits data to a SPI device using the TX Only Communications Operation Mode.

10.40.5.2 Detailed description

Implements write The function performs the following tasks: Performs parameter checking and processes error conditions. Disable Interrupts. Disable the SPI bus. Setup data bit width per user request. Enable the SPI bus. Enable interrupts. Start data transmission with dummy data via transmit buffer empty interrupt. Copy data from source buffer to the SPI data register for transmission. Receive data from receive buffer full interrupt occurs and do nothing with the received data. Complete data transmission via receive buffer full interrupt.

Table 1529:

Name	Description
SSP_SUCCESS	Write operation successfully completed.
SSP_ERR_ASSERTION	NULL pointer to control or source parameters or transfer length is zero.
SSP_ERR_UNSUPPORTED	With DTC transfer mode, bit_width must match configured DTC transfer width
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.
SSP_ERR_INVALID_HW_CONDITION	Failed to clear errors in the module

NOTE: This function is reentrant.

10.40.6 R_RSPI_WriteRead

```
ssp_err_t R_RSPI_WriteRead ( spi_ctrl_t  *const p_api_ctrl ,   void const
* p_src ,   void const * p_dest ,   uint32_t const length ,  spi_bit_width_t
const bit_width )
```

10.40.6.1 Brief description

This function simultaneously transmits data to a SPI device while receiving data from a SPI device (full duplex).

10.40.6.2 Detailed description

Implements spi_api_t::writeread The function performs the following tasks: Performs parameter checking and processes error conditions. Disable Interrupts. Disable the SPI bus. Setup data bit width per user request. Enable the SPI bus. Enable interrupts. Start data transmission using transmit buffer empty interrupt. Copy data from source buffer to the SPI data

register for transmission. Receive data from receive buffer full interrupt occurs and copy data to the buffer of destination. Complete data transmission and reception via receive buffer full interrupt.

Table 1530:

Name	Description
SSP_SUCCESS	Write operation successfully completed.
SSP_ERR_ASSERTION	NULL pointer to control, source or destination parameters or transfer length is zero.
SSP_ERR_UNSUPPORTED	With DTC transfer mode, bit_width must match configured DTC transfer width
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.
SSP_ERR_INVALID_HW_CONDITION	Failed to clear errors in the module

NOTE: This function is reentrant.

10.40.7 rspi_write_read_common

```
ssp_err_t rspi_write_read_common ( rspi_instance_ctrl_t  *const p_ctrl ,   void
const * p_src ,   void const * p_dest ,   uint32_t const length ,
 spi_bit_width_t  const bit_width ,  spi_operation_t  tx_rx_mode )
```

10.40.8 R_RSPI_Close

```
ssp_err_t R_RSPI_Close ( spi_ctrl_t  *const p_api_ctrl )
```

10.40.8.1 Brief description

This function manages the closing of a channel by the following task.

10.40.8.2 Detailed description

Implements close Disables SPI operations by disabling the SPI bus. Power off the channel. Disables all the associated interrupts. Update channel status.

Table 1531:

Name	Description
SSP_SUCCESS	Channel successfully closed.
SSP_ERR_ASSERTION	A required pointer argument is NULL.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.

NOTE: This function is reentrant.

10.40.9 rspi_baud_set

```
rspi_baud_set ( rspi_instance_ctrl_t  * p_ctrl ,   uint32_t baud_target )
```

10.40.10 R_RSPI_VersionGet

```
ssp_err_t R_RSPI_VersionGet ( ssp_version_t  * p_version )
```

10.40.10.1 Brief description

This function gets the version information of the underlying driver.

10.40.10.2 Detailed description

Implements spi_api_t::versionget

Table 1532:

Name	Description
void	

NOTE: This function is reentrant.

10.40.11 API Data

10.40.11.1 rspi_spcmd_bit_length_t

`rspi_spcmd_bit_length_t`

Detailed description

Frame data length

Enumerated values

Name	Description
RSPI_SPCMD_BIT_LENGTH_8	0100 to 0111 = 8 bits data length
RSPI_SPCMD_BIT_LENGTH_16	1111 = 16 bits data length
RSPI_SPCMD_BIT_LENGTH_32	0011 = 32 bits data length

10.40.11.2 rspi_operation_t

`rspi_operation_t`

Detailed description

SPCR (RSPI Control register) SPMS (RSPI mode) select

Enumerated values

Name	Description
RSPI_OPERATION_SPI	SPI operation (4-wire method)
RSPI_OPERATION_CLK_SYN	Clock Synchronous operation (3-wire method)

10.40.11.3 rspi_communication_t

`rspi_communication_t`

Detailed description

SPCR (RSPI Control register) TXMD (communication operating mode) select

Enumerated values

Name	Description
RSPI_COMMUNICATION_FULL_DUPLEX	Full-Duplex synchronous serial communication
RSPI_COMMUNICATION_TRANSMIT_ONLY	Transit only serial communication

10.40.11.4 rspi_sslp_t

`rspi_sslp_t`

Detailed description

Definition for SSLP (RSPI Slave Select Polarity register) select

Enumerated values

Name	Description
RSPI_SSLP_LOW	SSLP signal polarity active low
RSPI_SSLP_HIGH	SSLP signal polarity active high

10.40.11.5 rspi_loopback1_t

`rspi_loopback1_t`

Detailed description

SPPCR (RSPI Pin Control Register) Loopback1 select

Enumerated values

Name	Description
RSPI_LOOPBACK1_NORMAL_DATA	Loopback1 normal mode
RSPI_LOOPBACK1_INVERTED_DATA	Loopback1 with inverted data

10.40.11.6 rspi_loopback2_t

`rspi_loopback2_t`

Detailed description

SPPCR (RSPI Pin Control Register) Loopback2 select

Enumerated values

Name	Description
RSPI_LOOPBACK2_NORMAL_DATA	Loopback2 normal mode
RSPI_LOOPBACK2_NOT_INVERTED_DATA	Loopback2 with not inverted data

10.40.11.7 rspi_mosi_idle_fixed_val_t

`rspi_mosi_idle_fixed_val_t`

Detailed description

SPPCR (RSPI Pin Control Register) MOIFV select

Enumerated values

Name	Description
RSPI_MOSI_IDLE_FIXED_VAL_LOW	MOSIn level low during MOSI idling
RSPI_MOSI_IDLE_FIXED_VAL_HIGH	MOSIn level high during MOSI idling

10.40.11.8 rspi_mosi_idle_val_fixing_t

`rspi_mosi_idle_val_fixing_t`

Detailed description

SPPCR (RSPI Pin Control Register) MOIFE (MOSI idle value fixing) select

Enumerated values

Name	Description
RSPI_MOSI_IDLE_VAL_FIXING_ENABLE	MOSI output value=final data from previous transfer
RSPI_MOSI_IDLE_VAL_FIXING_DISABLE	MOSI output value=value set in MOIFV bit

10.40.11.9 rspi_parity_state_t

`rspi_parity_state_t`

Detailed description

SPCR2 (RSPI Control Register 2) Parity Enable select

Enumerated values

Name	Description
RSPI_PARITY_STATE_DISABLE	Disable parity
RSPI_PARITY_STATE_ENABLE	Enable parity

10.40.11.10 rspi_parity_mode_t

`rspi_parity_mode_t`

Detailed description

SPCR2 (RSPI Control Register 2) Parity select

Enumerated values

Name	Description
RSPI_PARITY_MODE_ODD	Select even parity
RSPI_PARITY_MODE_EVEN	Select odd parity

10.40.11.11 rspi_ssl_select_t

`rspi_ssl_select_t`

Detailed description

SPCMD (RSPI Command) Register SSL Signal Assertion select

Enumerated values

Name	Description
RSPI_SSL_SELECT_SSL0	Select SSL0 as slave
RSPI_SSL_SELECT_SSL1	Select SSL1 as slave
RSPI_SSL_SELECT_SSL2	Select SSL2 as slave
RSPI_SSL_SELECT_SSL3	Select SSL3 as slave

10.40.11.12 rspi_ssl_level_keep_t

`rspi_ssl_level_keep_t`

Detailed description

SPCMD (RSPI Command) Register SSL Signal Level Keeping select

Enumerated values

Name	Description
RSPI_SSL_LEVEL_KEEP_NOT	Negates all SSL signals upon transfer completion
RSPI_SSL_LEVEL_KEEP	Keeps the SSL level upon transfer completion

10.40.11.13 rspi_clock_delay_count_t

`rspi_clock_delay_count_t`

Detailed description

SPCKD (RSPI Clock Delay) Register Clock Delay Count select

Enumerated values

Name	Description
RSPI_CLOCK_DELAY_COUNT_1	Set RSPCK Clock delay to 1 RSPCK
RSPI_CLOCK_DELAY_COUNT_2	Set RSPCK Clock delay to 2 RSPCK
RSPI_CLOCK_DELAY_COUNT_3	Set RSPCK Clock delay to 3 RSPCK
RSPI_CLOCK_DELAY_COUNT_4	Set RSPCK Clock delay to 4 RSPCK
RSPI_CLOCK_DELAY_COUNT_5	Set RSPCK Clock delay to 5 RSPCK
RSPI_CLOCK_DELAY_COUNT_6	Set RSPCK Clock delay to 6 RSPCK
RSPI_CLOCK_DELAY_COUNT_7	Set RSPCK Clock delay to 7 RSPCK
RSPI_CLOCK_DELAY_COUNT_8	Set RSPCK Clock delay to 8 RSPCK

10.40.11.14 rspi_clock_delay_state_t

`rspi_clock_delay_state_t`

Detailed description

SPCMD (RSPI Command) Register RSPCK Delay Enable/Disable select SCKDEN

Enumerated values

Name	Description
RSPI_CLOCK_DELAY_STATE_DISABLE	RSPCK delay=1 RSPCK
RSPI_CLOCK_DELAY_STATE_ENABLE	RSPCK delay=SPCKD register setting

10.40.11.15 rspi_ssl_negation_delay_count_t

`rspi_ssl_negation_delay_count_t`

Detailed description

SSLND (RSPI Slave Select Negation Delay) Register Slave Select Negation Delay Count select

Enumerated values

Name	Description
RSPI_SSL_NEGATION_DELAY_1	Set SSL negation delay to 1 RSPCK
RSPI_SSL_NEGATION_DELAY_2	Set SSL negation delay to 2 RSPCK
RSPI_SSL_NEGATION_DELAY_3	Set SSL negation delay to 3 RSPCK
RSPI_SSL_NEGATION_DELAY_4	Set SSL negation delay to 4 RSPCK
RSPI_SSL_NEGATION_DELAY_5	Set SSL negation delay to 5 RSPCK
RSPI_SSL_NEGATION_DELAY_6	Set SSL negation delay to 6 RSPCK
RSPI_SSL_NEGATION_DELAY_7	Set SSL negation delay to 7 RSPCK
RSPI_SSL_NEGATION_DELAY_8	Set SSL negation delay to 8 RSPCK

10.40.11.16 rspi_ssl_negation_delay_state_t

`rspi_ssl_negation_delay_state_t`

Detailed description

SPCMD (RSPI Command) Register SSL Negation Delay select SLNDEN

Enumerated values

Name	Description
RSPI_SSL_NEGATION_DELAY_DISABLE	SSL negation delay=1 RSPCK
RSPI_SSL_NEGATION_DELAY_ENABLE	SSL negation delay=SSLND register setting

10.40.11.17 rspi_next_access_delay_count_t

`rspi_next_access_delay_count_t`

Detailed description

SPND (RSPI Next-Access Delay) Register Next Access Delay Count select

Enumerated values

Name	Description
RSPI_NEXT_ACCESS_DELAY_COUNT_1	Set next access delay to 1 RSPCK+2PCLK

Name	Description
RSPI_NEXT_ACCESS_DELAY_COUNT_2	Set next access delay to 2 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_3	Set next access delay to 3 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_4	Set next access delay to 4 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_5	Set next access delay to 5 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_6	Set next access delay to 6 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_7	Set next access delay to 7 RSPCK+2PCLK
RSPI_NEXT_ACCESS_DELAY_COUNT_8	Set next access delay to 8 RSPCK+2PCLK

10.40.11.18 rspi_next_access_delay_state_t

`rspi_next_access_delay_state_t`

Detailed description

SPCMD (RSPI Command) Register Next Access Delay select SPNDEN

Enumerated values

Name	Description
RSPI_NEXT_ACCESS_DELAY_STATE_DISABLE	Next access delay=1 RSPCK + 2 PCLK
RSPI_NEXT_ACCESS_DELAY_STATE_ENABLE	Next access delay=SPND register setting

10.40.11.19 rspi_spcmd_br_div_t

`rspi_spcmd_br_div_t`

Enumerated values

Name	Description
RSPI_SPCMD_BR_DIV_1	
RSPI_SPCMD_BR_DIV_2	
RSPI_SPCMD_BR_DIV_4	
RSPI_SPCMD_BR_DIV_8	

10.40.11.20 rspi_spcmd_assert_ssl_t

`rspi_spcmd_assert_ssl_t`

Detailed description

Slave select to be asserted during transfer operation.

Enumerated values

Name	Description
RSPI_SPCMD_ASSERT_SSL0	
RSPI_SPCMD_ASSERT_SSL1	
RSPI_SPCMD_ASSERT_SSL2	
RSPI_SPCMD_ASSERT_SSL3	

10.40.12 Extensions

10.40.12.1 rspi_ssl_polarity_t

rspi_ssl_polarity_t

Detailed description

SSLP (RSPI Slave Select Polarity register) SSLnP select

Variables

- rspi_sslp_t rspi_ssl2
- rspi_sslp_t rspi_ssl3
- rspi_sslp_t rspi_ssl0
- rspi_sslp_t rspi_ssl1

10.40.12.2 rspi_loopback_t

rspi_loopback_t

Detailed description

SPPCR (RSPI Pin Control Register) Loopback select

Variables

- rspi_loopback1_t rspi_loopback1
- rspi_loopback2_t rspi_loopback2

10.40.12.3 rspi_mosi_idle_t

rspi_mosi_idle_t

Detailed description

SPPCR (RSPI Pin Control Register) MOIFV (mosi idle value) select

Variables

- rspi_mosi_idle_fixed_val_t rspi_mosi_idle_fixed_val
- rspi_mosi_idle_val_fixing_t rspi_mosi_idle_val_fixing

10.40.12.4 rspi_parity_t

rspi_parity_t

Detailed description

SPCR2 (RSPI Control Register 2) Parity select

Variables

- rspi_parity_state_t rspi_parity
- rspi_parity_mode_t rspi_parity_mode

10.40.12.5 rspi_clock_delay_t

rspi_clock_delay_t

Detailed description

Select RSPI Clock Delay Register (SPCKD) and SPCMD (RSPI Command) Register-Clock Delay state(SCKDEN)

Variables

- rspi_clock_delay_count_t rspi_clock_delay_count
- rspi_clock_delay_state_t rspi_clock_delay_state

10.40.12.6 rspi_ssl_negation_delay_t

rspi_ssl_negation_delay_t

Detailed description

Select SSL Negation Delay(SSLND) and SPCMD Register-SSL negation Delay state(SLNDEN)

Variables

- rspi_ssl_negation_delay_count_t rspi_ssl_neg_delay_count
- rspi_ssl_negation_delay_state_t rspi_ssl_neg_delay_state

10.40.12.7 rspi_access_delay_t

rspi_access_delay_t

Detailed description

Select Next Access Delay(SPND) and SPCMD Register-Next Access Delay state(SPNDEN)

Variables

- rspi_next_access_delay_count_t rspi_next_access_delay_count
- rspi_next_access_delay_state_t rspi_next_access_delay_state

10.40.12.8 spi_on_rspi_cfg_t

spi_on_rspi_cfg_t

Detailed description

Extended SPI interface configuration

Variables

- rspi_operation_t rspi_clksyn

  Select spi or clock syn mode operation
- rspi_communication_t rspi_comm

  Select full-duplex or transmit-only communication
- rspi_ssl_polarity_t ssl_polarity

  Select SSLn signal polarity
- rspi_loopback_t loopback

  Select loopback1 and loopback2
- rspi_mosi_idle_t mosi_idle

  Select mosi idle fixed value and selection
- rspi_parity_t parity

  Select parity and enable/disable parity
- rspi_ssl_select_t ssl_select

  Select which slave to use;0-SSL0;1-SSL1;2-SSL2;3-SSL3
- rspi_ssl_level_keep_t ssl_level_keep

  Select SSL level after transfer completion;0-negate;1-keeps
- rspi_clock_delay_t clock_delay

  Select clock delay from 0 to 7
- rspi_ssl_negation_delay_t ssl_neg_delay

  Select Slave elect negation delay from 0 to 7
- rspi_access_delay_t access_delay

  Select next access delay from 0 to 7

10.40.12.9 rspi_instance_ctrl_t

rspi_instance_ctrl_t

Detailed description

SPI instance control block. DO NOT INITIALIZE.

Variables

- uint8_t channel

  Channel number to be used.

- uint8_t current_slave

  Number of the currently assigned slave.

- uint32_t channel_opened

  Internal flag to indicate the peripheral was initialized.

- transfer_instance_t const * p_transfer_tx

  To use SPI DTC/DMA write transfer.

- transfer_instance_t const * p_transfer_rx

  To use SPI DTC/DMA read transfer.

- void(* p_callback)(*p_args)

  Pointer to user callback function.

- void const * p_context

  Pointer to the higher level device context.

- void * p_reg

  Base register for this channel.

- IRQn_Type rxi_irq

  Receive IRQ number.

- IRQn_Type txi_irq

  Transmit IRQ number.

- IRQn_Type tei_irq

  Transmit end IRQ number.

- IRQn_Type eri_irq

  Error IRQ number.

- void * p_src
- void * p_dest
- uint32_t tx_count
- uint32_t rx_count
- uint32_t xfr_length

- uint8_t bytes_per_transfer
- bool do_tx
- bool using_dtc
- transfer_addr_mode_t tx_dtc_addr_mode
- transfer_addr_mode_t rx_dtc_addr_mode
- spi_operation_t transfer_mode
- uint32_t rx_data
- bsp_lock_t resource_lock_tx_rx

  Resource lock for transmission/reception

10.41 RTC

Driver for the Realtime Clock (RTC).

This module supports the Real Time Clock (RTC). It implements the following interfaces:

- RTC Interface

10.41.1 Functions

- R_RTC_Open
- R_RTC_Close
- R_RTC_CalendarTimeSet
- R_RTC_CalendarTimeGet
- R_RTC_CalendarAlarmSet
- R_RTC_CalendarAlarmGet
- R_RTC_CalendarCounterStart
- R_RTC_CalendarCounterStop
- R_RTC_IrqEnable
- R_RTC_IrqDisable
- R_RTC_PeriodicIrqRateSet
- R_RTC_InfoGet
- R_RTC_VersionGet
- r_rtc_start_bit_clear
- r_rtc_start_bit_set
- r_rtc_software_reset

- r_rtc_set_clock_source
- r_rtc_nvic_enable_irq
- r_rtc_config_rtc_interrupts
- r_rtc_enable_alarm_irq
- r_rtc_validate_time
- r_rtc_validate_time_fields
- r_rtc_validate_date_fields
- r_rtc_enable_irq
- r_rtc_disable_irq
- rtc_dec_to_bcd
- rtc_bcd_to_dec

10.41.2 Defines

- #define RTC_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define RTC_CODE_VERSION_MINOR

  Initial value:`(5U)`

10.41.3 R_RTC_Open

```
ssp_err_t R_RTC_Open ( rtc_ctrl_t  *const p_api_ctrl ,  rtc_cfg_t  const
*const p_cfg )
```

10.41.3.1 Brief description

Open the RTC driver.

10.41.3.2 Detailed description

Implements open.

Opens and configures the RTC driver module. Configuration includes clock source, and interrupt callback function. If the sub-clock oscillator is the clock source it is started in this function.

Table 1533:

Name	Description
SSP_SUCCESS	Initialization was successful and RTC has started.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_cfg pointer.
SSP_ERR_HW_LOCKED	Hardware in use
SSP_ERR_TIMEOUT	Status check for counter mode or reset timed out

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

10.41.3.3 Function steps

- Mark driver as open by initializing it to "RTC" in its ASCII equivalent.

10.41.4 R_RTC_Close

```
ssp_err_t R_RTC_Close ( rtc_ctrl_t  *const p_api_ctrl )
```

10.41.4.1 Brief description

Close the RTC driver.

10.41.4.2 Detailed description

Implements close

Table 1534:

Name	Description
SSP_SUCCESS	De-Initialization was successful and RTC driver closed.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for close.

10.41.5 R_RTC_CalendarTimeSet

```
ssp_err_t R_RTC_CalendarTimeSet ( rtc_ctrl_t  *const p_api_ctrl ,  rtc_time_t
* p_time ,   bool clock_start )
```

10.41.5.1 Brief description

Set the calendar time.

10.41.5.2 Detailed description

Implements calendarTimeSet.

Table 1535:

Name	Description
SSP_SUCCESS	Calendar time set operation was successful.
SSP_ERR_ASSERTION	Invalid p_api_ctrl, p_time or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.
SSP_ERR_INVALID_ARGUMENT	Invalid time parameter field.
SSP_ERR_TIMEOUT	Software reset status check failed.

10.41.6 R_RTC_CalendarTimeGet

```
ssp_err_t R_RTC_CalendarTimeGet ( rtc_ctrl_t  *const p_api_ctrl ,  rtc_time_t
* p_time )
```

10.41.6.1 Brief description

Get the calendar time.

10.41.6.2 Detailed description

Implements calendarTimeGet

Table 1536:

Name	Description
SSP_SUCCESS	Calendar time get operation was successful.
SSP_ERR_ASSERTION	Invalid p_api_ctrl, p_time or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.7 R_RTC_CalendarAlarmSet

```
ssp_err_t R_RTC_CalendarAlarmSet ( rtc_ctrl_t  *const p_api_ctrl ,
 rtc_alarm_time_t  * p_alarm ,   bool interrupt_enable_flag )
```

10.41.7.1 Brief description

Set the calendar alarm time.

10.41.7.2 Detailed description

Implements calendarAlarmSet.

NOTE: The calendar counter must be running before the alarm can be set.

Table 1537:

Name	Description
SSP_SUCCESS	Calendar alarm time set operation was successful.
SSP_ERR_INVALID_ARGUMENT	Invalid time parameter field.
SSP_ERR_ASSERTION	Invalid p_api_ctrl, p_alarm or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.8 R_RTC_CalendarAlarmGet

```
ssp_err_t R_RTC_CalendarAlarmGet ( rtc_ctrl_t  *const p_api_ctrl ,
 rtc_alarm_time_t  * p_alarm )
```

10.41.8.1 Brief description

Get the calendar alarm time.

10.41.8.2 Detailed description

Implements calendarAlarmGet

Table 1538:

Name	Description
SSP_SUCCESS	Calendar alarm time get operation was successful.
SSP_ERR_ASSERTION	Invalid p_api_ctrl, p_alarm or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.9 R_RTC_CalendarCounterStart

```
ssp_err_t R_RTC_CalendarCounterStart ( rtc_ctrl_t  *const p_api_ctrl )
```

10.41.9.1 Brief description

Start the calendar counter.

10.41.9.2 Detailed description

Implements calendarCounterStart.

Table 1539:

Name	Description
SSP_SUCCESS	Calendar counter started.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.10 R_RTC_CalendarCounterStop

```
ssp_err_t R_RTC_CalendarCounterStop ( rtc_ctrl_t  *const p_api_ctrl )
```

10.41.10.1 Brief description

Stop the calendar counter.

10.41.10.2 Detailed description

Implements calendarCounterStop.

Table 1540:

Name	Description
SSP_SUCCESS	Calendar counter stopped.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.11 R_RTC_IrqEnable

```
ssp_err_t R_RTC_IrqEnable ( rtc_ctrl_t  *const p_api_ctrl ,
 rtc_event_t  event )
```

10.41.11.1 Brief description

Enable the alarm interrupt.

10.41.11.2 Detailed description

Implements rtc_api_t::interruptEnable.

Table 1541:

Name	Description
SSP_SUCCESS	Alarm interrupt enabled.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_IRQ_BSP_DISABLED	User IRQ parameter not valid.
SSP_ERR_INVALID_ARGUMENT	Invalid IRQ event.
SSP_ERR_TIMEOUT	IRQ enable operation timed out.

Table 1541: (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.12 R_RTC_IrqDisable

```
ssp_err_t R_RTC_IrqDisable ( rtc_ctrl_t  *const p_api_ctrl ,
 rtc_event_t  event )
```

10.41.12.1 Brief description

Disable the alarm interrupt.

10.41.12.2 Detailed description

Implements rtc_api_t::interruptDisable.

Table 1542:

Name	Description
SSP_SUCCESS	Alarm interrupt disabled.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_IRQ_BSP_DISABLED	User IRQ parameter not valid
SSP_ERR_INVALID_ARGUMENT	Invalid IRQ event
SSP_ERR_TIMEOUT	IRQ disable operation timed out.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.13 R_RTC_PeriodicIrqRateSet

```
ssp_err_t R_RTC_PeriodicIrqRateSet ( rtc_ctrl_t  *const p_api_ctrl ,
 rtc_periodic_irq_select_t  rate )
```

10.41.13.1 Brief description

Set the periodic interrupt rate.

10.41.13.2 Detailed description

Implements rtc_api_t::periodicInterruptRateSet.

Table 1543:

Name	Description
SSP_SUCCESS	The periodic interrupt rate was successfully set.
SSP_ERR_ASSERTION	Invalid p_api_ctrl or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_TIMEOUT	Periodic interrupt rate get query timed out.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.14 R_RTC_InfoGet

```
ssp_err_t R_RTC_InfoGet ( rtc_ctrl_t  * p_api_ctrl ,  rtc_info_t  * p_rtc_info )
```

10.41.14.1 Brief description

This function returns information about the driver clock source.

10.41.14.2 Detailed description

Implements infoGet

Table 1544:

Name	Description
SSP_SUCCESS	Get information Successful.
SSP_ERR_ASSERTION	Invalid p_api_ctrl, p_rtc_info or p_ctrl->p_reg member pointed by p_api_ctrl pointer.
SSP_ERR_NOT_OPEN	Driver not open already for operation.

10.41.15 R_RTC_VersionGet

```
ssp_err_t R_RTC_VersionGet ( ssp_version_t  * p_version )
```

10.41.15.1 Brief description

Get driver version based on compile time macros.

10.41.15.2 Detailed description

Implements versionGet

Table 1545:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.41.16 r_rtc_start_bit_clear

```
ssp_err_t r_rtc_start_bit_clear (  R_RTC_Type * p_rtc_reg )
```

10.41.16.1 Brief description

Clear the start bit.

10.41.16.2 Detailed description

Table 1546:

Name	Description
SSP_SUCCESS	start bit cleared
SSP_ERR_TIMEOUT	start bit not cleared

10.41.17 r_rtc_start_bit_set

```
ssp_err_t r_rtc_start_bit_set (  R_RTC_Type * p_rtc_reg )
```

10.41.17.1 Brief description

Set the start bit.

10.41.17.2 Detailed description

Table 1547:

Name	Description
SSP_SUCCESS	start bit set
SSP_ERR_TIMEOUT	start bit not set

10.41.18 r_rtc_software_reset

```
ssp_err_t r_rtc_software_reset (  R_RTC_Type * p_rtc_reg )
```

10.41.18.1 Brief description

Perform a software reset.

10.41.18.2 Detailed description

Table 1548:

Name	Description
SSP_SUCCESS	software reset complete
SSP_ERR_TIMEOUT	software reset not complete

10.41.19 r_rtc_set_clock_source

```
ssp_err_t r_rtc_set_clock_source (  R_RTC_Type * p_rtc_reg ,  rtc_cfg_t  const
*const p_cfg )
```

10.41.19.1 Brief description

Set the RTC clock source.

10.41.19.2 Detailed description

Table 1549:

Name	Description
SSP_SUCCESS	RTC clock source set
SSP_ERR_TIMEOUT	status check for counter mode or reset timed out

10.41.20 r_rtc_nvic_enable_irq

```
ssp_err_t r_rtc_nvic_enable_irq (  IRQn_Type nvic_interrupt ,
  uint32_t irq_en )
```

10.41.20.1 Brief description

check if timeout set else enable passed NVIC interrupt.

10.41.20.2 Detailed description

Implements a helper function

Table 1550:

Name	Description
SSP_SUCCESS	Calendar counter stopped.

10.41.21 r_rtc_config_rtc_interrupts

```
ssp_err_t r_rtc_config_rtc_interrupts ( fmi_event_info_t  * event_info ,
 rtc_instance_ctrl_t  * p_ctrl ,  rtc_cfg_t  const *const p_cfg ,  ssp_feature_t
* ssp_feature )
```

10.41.21.1 Brief description

get IRQ from event info to set IRQ priority and control info for IRQ handler .

10.41.21.2 Detailed description

Implements a helper function

Table 1551:

Name	Description
SSP_SUCCESS	Successful configuration

10.41.22 r_rtc_enable_alarm_irq

```
ssp_err_t r_rtc_enable_alarm_irq ( rtc_instance_ctrl_t  * p_ctrl ,
  bool interrupt_enable_flag )
```

10.41.22.1 Brief description

enable alarm irq if valid

10.41.22.2 Detailed description

Implements a helper function

Table 1552:

Name	Description
SSP_SUCCESS	Alarm IRQ enabled
SSP_ERR_TIMEOUT	check for Alarm IRQ enable bit timed out

10.41.23 r_rtc_validate_time

```
ssp_err_t r_rtc_validate_time ( rtc_time_t  * p_time )
```

10.41.23.1 Brief description

validate time parameter fields

10.41.23.2 Detailed description

Implements a helper function

Table 1553:

Name	Description
SSP_SUCCESS	validation successful
SSP_ERR_INVALID_ARGUMENT	invalid field in rtc_time_t structure

10.41.24 r_rtc_validate_time_fields

```
ssp_err_t r_rtc_validate_time_fields ( rtc_time_t  * p_time )
```

10.41.24.1 Brief description

validate time fields of time type parameter

10.41.24.2 Detailed description

Implements a helper function

Table 1554:

Name	Description
SSP_SUCCESS	validation successful
SSP_ERR_INVALID_ARGUMENT	invalid field in rtc_time_t structure

10.41.25 r_rtc_validate_date_fields

```
ssp_err_t r_rtc_validate_date_fields ( rtc_time_t  * p_time )
```

10.41.25.1 Brief description

validate date fields of time type parameter Day of week between 0 to 6 Day between 1 to 31 Month between 0 to 11 as per standard time.h, There's a mismatch between hardware configuration, UM indicates that "A value from 01 through 12 (in BCD) can be specified" for Month Counter register in the RTC. This difference will be taken care in the Set and Get functions.

10.41.25.2 Detailed description

As per HW manual, value of Year is between 0 to 99, the RTC has a 100 year calendar from 2000 to 2099. But as per C standards, tm_year is years since 1900. A sample year set in an application would be like time.tm_year = 2017-1900; (to

set year 2017) Since RTC API follows the Date and time structure defined in C standard library <time.h>, the valid value of year is between 100 and 199, which will be internally converted to HW required value.

Implements a helper function

Table 1555:

Name	Description
SSP_SUCCESS	validation successful
SSP_ERR_INVALID_ARGUMENT	invalid field in rtc_time_t structure

10.41.26 r_rtc_enable_irq

```
ssp_err_t r_rtc_enable_irq (  R_RTC_Type * p_rtc_reg ,   IRQn_Type irq ,
 rtc_event_t  event ,   uint32_t timeout )
```

10.41.26.1 Brief description

enable alarm irq if valid

10.41.26.2 Detailed description

Implements a helper function

Table 1556:

Name	Description
SSP_SUCCESS	enable IRQ successful
SSP_ERR_TIMEOUT	check for IRQ enable bit for the event timed out

10.41.27 r_rtc_disable_irq

```
ssp_err_t r_rtc_disable_irq (  R_RTC_Type * p_rtc_reg ,   IRQn_Type irq ,
 rtc_event_t  event ,   uint32_t timeout )
```

10.41.27.1 Brief description

enable alarm irq if valid

10.41.27.2 Detailed description

Implements a helper function

Table 1557:

Name	Description
SSP_SUCCESS	disable IRQ successful
SSP_ERR_TIMEOUT	check for IRQ disable bit for the event timed out

10.41.28 rtc_dec_to_bcd

```
rtc_dec_to_bcd (  uint8_t to_convert )
```

10.41.28.1 Brief description

RTC Alarm ISR.

10.41.28.2 Detailed description

Saves context if RTOS is used, stops the timer if one-shot mode, clears interrupts, calls callback if one was provided in the open function, and restores context if RTOS is used. Convert decimal to BCD

10.41.29 rtc_bcd_to_dec

```
rtc_bcd_to_dec (  uint8_t to_convert )
```

10.41.29.1 Brief description

Convert BCD to decimal.

10.41.30 API Data

10.41.30.1 rtc_count_mode_t

`rtc_count_mode_t`

Detailed description

Counting mode

Enumerated values

Name	Description
RTC_CALENDAR_MODE	
RTC_BINARY_MODE	

10.41.31 Extensions

10.41.31.1 rtc_instance_ctrl_t

rtc_instance_ctrl_t

Detailed description

Channel control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- void * p_reg

  Pointer to register base address.
- void(* p_callback)(*cb_data)

  Called from the ISR.
- void const * p_context

  Passed to the callback.
- uint32_t open

  Whether or not driver is open.
- IRQn_Type alarm_irq

  Alarm IRQ number.
- IRQn_Type periodic_irq

  Periodic IRQ number.
- IRQn_Type carry_irq

  Carry IRQ number.
- rtc_clock_source_t clock_source

  Clock source for the RTC block.

10.42 Simple I2C on SCI

Driver for the Simple IIC on SCI.

This module supports the SCI in I2C mode. It implements the following interfaces:

- I2C Interface r_i2c_api.h

10.42.1 Functions

- sci_siic_notify
- sci_siic_clock_settings
- sci_siic_sda_delay_settings
- sci_siic_abort_seq_master
- sci_siic_open_hw_master
- sci_siic_close_hw_master
- sci_siic_abort_hw_master
- sci_siic_run_hw_master
- sci_siic_set_irq_parameters
- sci_siic_configure_irqs
- sci_i2c_rxi_isr
- sci_i2c_txi_isr
- sci_i2c_tei_isr
- sci_siic_open_transfer_interface
- sci_siic_reconfigure_interrupts_for_transfer
- sci_siic_stop_transfer_interface
- sci_siic_transfer_configure_rx
- sci_siic_transfer_configure_tx
- sci_siic_enable_transfer_support_tx
- sci_siic_enable_transfer_support_rx
- sci_siic_txi_send_data
- sci_siic_tei_send_address
- sci_siic_open_parameter_check
- sci_siic_tei_handler
- sci_siic_txi_handler
- sci_siic_rxi_handler
- sci_siic_txi_process_nack
- sci_siic_dtc_max_length_check
- R_SCI_SIIC_MasterVersionGet
- R_SCI_SIIC_MasterOpen

- R_SCI_SIIC_MasterClose
- R_SCI_SIIC_MasterRead
- R_SCI_SIIC_MasterWrite
- R_SCI_SIIC_MasterReset
- R_SCI_SIIC_MasterSlaveAddressSet

10.42.2 Variables

- g_sci_siic_master_version
- g_dummy_write_data_for_read_op
- g_i2c_master_on_sci
- sync_baud

10.42.3 Defines

- #define SCI_SIIC_MASTER_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCI_SIIC_MASTER_CODE_VERSION_MINOR

  Initial value:`(6U)`
- #define SIIC_OPEN

  Initial value:`(0x83737343ULL)`

  "SIIC" in ASCII, used to determine if channel is open.
- #define MDDR_MIN

  Initial value:`128U`
- #define MDDR_MAX

  Initial value:`256U`
- #define SCI_SIIC_ERROR_RETURN

  Initial value:`SSP_ERROR_RETURN((a), (err), &g_module_name[0], NULL)`

  Macro for error logger.
- #define I2C_CODE_READ

  Initial value:`(0x01U)`
- #define I2C_CODE_10BIT

  Initial value:`(0xF0U)`

- #define SCI_SIIC_NUM_DIVISORS_SYNC
  Initial value:(4U)
- #define SCI_SIIC_BRR_MAX
  Initial value:(255U)
- #define SCI_SIIC_BRR_MIN
  Initial value:(0U)
- #define SCI_SIIC_MAX_PCLK
  Initial value:(0xFFFFFFFFFFFFFFFFULL)
- #define SCI_I2C_NANOSECONDS_PER_SECOND
  Initial value:(1000000000U)
- #define SCI_I2C_SIMR1_IICDL_MAXIMUM_VALUE
  Initial value:(0x1FU)

10.42.4 sci_siic_notify

```
 sci_siic_notify ( sci_i2c_instance_ctrl_t  *const p_ctrl ,  i2c_event_t
const event )
```

10.42.4.1 Brief description

Single point for managing the logic around notifying a transfer has finished.

10.42.4.2 Detailed description

(end defgroup SIIC)

Table 1558:

Name	Direction	Description
p_ctrl	in	Pointer to transfer that is ending.
event	in	The event code to pass to the callback.

10.42.4.3 Function steps

- Set the flag indicating that the transaction is completed

10.42.5 sci_siic_clock_settings

```
sci_siic_clock_settings (  uint32_t *const p_rate ,   uint8_t *const p_brr ,
 uint8_t *const p_divisor ,   uint32_t *const p_mddr ,   uint16_t sda_delay ,
 uint32_t * cycles )
```

10.42.6 sci_siic_sda_delay_settings

```
sci_siic_sda_delay_settings (  uint32_t const clk_divisor ,
 uint16_t sda_delay ,   uint32_t *const p_cycles )
```

10.42.7 sci_siic_abort_seq_master

```
ssp_err_t sci_siic_abort_seq_master ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.7.1 Brief description

Single point for managing the logic around aborting a transfer when operating as a master.

10.42.7.2 Detailed description

Table 1559:

Name	Direction	Description
p_ctrl	in	Pointer to control struct of specific device

Table 1560:

Name	Description
SSP_ERR_ABORTED	If there is an in-progress transfer

10.42.8 sci_siic_open_hw_master

```
ssp_err_t sci_siic_open_hw_master ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.8.1 Brief description

Performs the hardware initialization sequence when operating as a master.

10.42.8.2 Detailed description

Table 1561:

Name	Direction	Description
p_ctrl	in	Pointer to control structure of specific device

Table 1562:

Name	Description
SSP_SUCCESS	Hardware initialized with proper configurations
SSP_ERR_INVALID_RATE	The requested rate cannot be set.

10.42.8.3 Function steps

- enables or disable the bitrate modulation function

10.42.9 sci_siic_close_hw_master

```
sci_siic_close_hw_master ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.9.1 Brief description

Performs the hardware initialization sequence when operating as a master.

10.42.9.2 Detailed description

Table 1563:

Name	Direction	Description
p_ctrl	in	Pointer to control structure of specific device

10.42.10 sci_siic_abort_hw_master

```
sci_siic_abort_hw_master ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.10.1 Brief description

Safely stops the current data transfer when operating as a master.

10.42.10.2 Detailed description

Table 1564:

Name	Direction	Description
p_ctrl	in	Pointer to control structure of specific device

10.42.10.3 Function steps

- Disable channel interrupts

10.42.11 sci_siic_run_hw_master

```
ssp_err_t sci_siic_run_hw_master ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.11.1 Brief description

Performs the data transfer described by the parameters when operating as a master.

10.42.11.2 Detailed description

Table 1565:

Name	Direction	Description
p_ctrl	in	Pointer to Control structure of specific device.

Table 1566:

Name	Description
SSP_SUCCESS	Data transfered when operating as a master.
SSP_ERR_IN_USE	If a transfer is in progress.

Table 1566: (Continued)

Name	Description
SSP_ERR_ABORTED	If there is an in-progress transfer.

10.42.12 sci_siic_set_irq_parameters

```
 sci_siic_set_irq_parameters (  uint8_t ipl ,   void * p_ctrl ,   IRQn_Type
* p_irq )
```

10.42.12.1 Detailed description

Sets interrupt priority and initializes vector info for the callback

Table 1567:

Name	Direction	Description
ipl	in	Interrupt priority level
p_ctrl	in	Pointer to driver control block
p_irq	in	Pointer to IRQ for this signal

10.42.12.2 Function steps

- Set the interrupt priority for the specified interrupt number(*p_irq)
- Set the p_ctrl into the vector_info structure for the element. This information will be used by the ISR to determine the p_ctrl to be used in the ISR processing.

10.42.13 sci_siic_configure_irqs

```
ssp_err_t sci_siic_configure_irqs ( ssp_feature_t  * p_feature ,  i2c_ctrl_t
*const p_api_ctrl ,  i2c_cfg_t  const *const p_cfg )
```

10.42.13.1 Detailed description

Disables and clears interrupts in the NVIC and the peripheral. Then, sets interrupt priority and initializes vector info to be used in the callback.

Table 1568:

Name	Direction	Description
p_feature	in	SSP feature
p_api_ctrl	in	Pointer to driver control block
p_cfg	in	Pointer to driver configuration block

Table 1569:

Name	Description
SSP_SUCCESS	Interrupt enabled
SSP_ERR_IRQ_BSP_DISABLED	Interrupt does not exist in the vector table

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- eventInfoGet

10.42.13.2 Function steps

- Get the interrupt vector number for SSP_SIGNAL_SCI_RXI
- Save the interrupt vector number to the control block
- Get the interrupt vector number for SSP_SIGNAL_SCI_TXI
- Save the interrupt vector number to the control block
- Get the interrupt vector number for SSP_SIGNAL_SCI_TEI
- Save the interrupt vector number to the control block
- Disable the interrupt in the NVIC and the peripheral
- Clear the IR flag in the ICU
- Clear interrupt flag in the NVIC
- Disable the interrupt in the NVIC and the peripheral
- Clear the IR flag in the ICU
- Clear interrupt flag in the NVIC
- Disable the interrupt in the NVIC and the peripheral
- Clear the IR flag in the ICU
- Clear interrupt flag in the NVIC

10.42.14 sci_i2c_rxi_isr

```
sci_i2c_rxi_isr (  void  )
```

10.42.14.1 Brief description

ISR for ACK/RXI interrupt.

10.42.15 sci_i2c_txi_isr

```
sci_i2c_txi_isr (  void  )
```

10.42.15.1 Brief description

ISR for NACK/TXI interrupt.

10.42.16 sci_i2c_tei_isr

```
sci_i2c_tei_isr (  void  )
```

10.42.16.1 Brief description

Handles the STI interrupt.

10.42.17 sci_siic_open_transfer_interface

```
ssp_err_t sci_siic_open_transfer_interface ( sci_i2c_instance_ctrl_t
*const p_ctrl ,  i2c_cfg_t  const *const p_cfg )
```

10.42.17.1 Brief description

Configures SCI I2C related transfer drivers (if enabled).

10.42.17.2 Detailed description

Table 1570:

Name	Direction	Description
p_ctrl	in	Pointer to SCI I2C specific control structure

Table 1570: (Continued)

Name	Direction	Description
p_cfg	in	Pointer to SCI I2C specific configuration structure

Table 1571:

Name	Description
SSP_SUCCESS	If configures SCI I2C related transfer drivers
SSP_ERR_ASSERTION	Transfer configuration for tx/rx not proper.

10.42.18 sci_siic_reconfigure_interrupts_for_transfer

```
ssp_err_t sci_siic_reconfigure_interrupts_for_transfer ( sci_i2c_instance_ctrl_t
  *const p_ctrl )
```

10.42.18.1 Brief description

Reconfigure the address mode for transfer interface.

10.42.18.2 Detailed description

Table 1572:

Name	Direction	Description
p_ctrl	in	transfer control block

Table 1573:

Name	Description
SSP_SUCCESS	Address mode for transfer interface reconfigured.
SSP_ERR_IN_USE	If a transfer is in progress.

10.42.19 sci_siic_stop_transfer_interface

```
sci_siic_stop_transfer_interface ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.19.1 Brief description

Stop any running instance of the transfer interface.

10.42.19.2 Detailed description

Table 1574:

Name	Direction	Description
p_ctrl	in	transfer control block

10.42.20 sci_siic_transfer_configure_rx

```
ssp_err_t sci_siic_transfer_configure_rx ( sci_i2c_instance_ctrl_t
*const p_ctrl ,  i2c_cfg_t  const *const p_cfg ,   R_SCI0_Type * p_sci_reg )
```

10.42.20.1 Brief description

Configures SCI I2C RX related transfer (if enabled).

10.42.20.2 Detailed description

Table 1575:

Name	Direction	Description
p_ctrl	in	Pointer to SCI I2C specific control structure
p_cfg	in	Pointer to SCI I2C specific configuration structure
p_sci_reg	in	Pointer to hardware registers for SCI I2C module

Table 1576:

Name	Description
SSP_SUCCESS	Configurations proper for RX related transfer.
SSP_ERR_ASSERTION	Pointer to the transfer instance for I2C receive in p_cfg is NULL.

10.42.21 sci_siic_transfer_configure_tx

```
ssp_err_t sci_siic_transfer_configure_tx ( sci_i2c_instance_ctrl_t
*const p_ctrl ,  i2c_cfg_t  const *const p_cfg ,   R_SCI0_Type * p_sci_reg )
```

10.42.21.1 Brief description

Configures SCI I2C TX related transfer (if enabled).

10.42.21.2 Detailed description

Table 1577:

Name	Direction	Description
p_ctrl	in	Pointer to SCI I2C specific control structure
p_cfg	in	Pointer to SCI I2C specific configuration structure
p_sci_reg	in	Pointer to hardware registers for SCI I2C module

Table 1578:

Name	Description
SSP_SUCCESS	Configurations proper for TX related transfer.
SSP_ERR_ASSERTION	Pointer to the transfer instance for I2C transmit in p_cfg is NULL.

10.42.22 sci_siic_enable_transfer_support_tx

```
sci_siic_enable_transfer_support_tx ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.22.1 Brief description

Enables the dtc transfer interface for the transmit operation.

10.42.22.2 Detailed description

Table 1579:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.42.23 sci_siic_enable_transfer_support_rx

```
sci_siic_enable_transfer_support_rx ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.23.1 Brief description

Enables the dtc transfer interface for the receive operation.

10.42.23.2 Detailed description

Table 1580:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.42.24 sci_siic_txi_send_data

```
sci_siic_txi_send_data ( sci_i2c_instance_ctrl_t  *const p_ctrl ,   R_SCI0_Type
* p_sci_reg )
```

10.42.24.1 Brief description

Check for the receive condition.

10.42.24.2 Detailed description

Table 1581:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block
p_sci_reg	in	Pointer to hardware registers for SCI I2C module

10.42.25 sci_siic_tei_send_address

```
sci_siic_tei_send_address ( sci_i2c_instance_ctrl_t  *const p_ctrl ,
 R_SCI0_Type * p_sci_reg )
```

10.42.25.1 Brief description

Enables transfer support while handling the tei interrupt.

10.42.25.2 Detailed description

Table 1582:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block
p_sci_reg	in	Pointer to hardware registers for SCI I2C module

10.42.26 sci_siic_open_parameter_check

```
ssp_err_t sci_siic_open_parameter_check ( sci_i2c_instance_ctrl_t
*const p_ctrl ,  i2c_cfg_t  const *const p_cfg )
```

10.42.26.1 Brief description

Parameter check for control block and configuration block.

10.42.26.2 Detailed description

Table 1583:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block
p_cfg	in	Pointer to driver configuration block

Table 1584:

Name	Description
SSP_SUCCESS	If parameter p_ctrl, p_cfg or clock rate not NULL.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL or if clock rate greater than 400KHz.

10.42.27 sci_siic_tei_handler

```
sci_siic_tei_handler ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.28 sci_siic_txi_handler

```
sci_siic_txi_handler ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.29 sci_siic_rxi_handler

```
sci_siic_rxi_handler ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.30 sci_siic_txi_process_nack

```
sci_siic_txi_process_nack ( sci_i2c_instance_ctrl_t  *const p_ctrl )
```

10.42.30.1 Brief description

Process NACK reception within TXI interrupt.

10.42.30.2 Detailed description

Table 1585:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block

10.42.31 sci_siic_dtc_max_length_check

```
ssp_err_t sci_siic_dtc_max_length_check ( sci_i2c_instance_ctrl_t
*const p_ctrl ,   const uint32_t bytes )
```

10.42.31.1 Brief description

Parameter check for Read and Write when DTC is used for transfer.

10.42.31.2 Detailed description

Table 1586:

Name	Direction	Description
p_ctrl	in	Pointer to transfer control block
bytes	in	number of bytes to be transferred

Table 1587:

Name	Description
SSP_SUCCESS	Provided length supported by DTC.
SSP_ERR_INVALID_SIZE	Provided number of bytes more than uint16_t size(65535) while DTC is used for data transfer.

10.42.32 R_SCI_SIIC_MasterVersionGet

```
ssp_err_t R_SCI_SIIC_MasterVersionGet ( ssp_version_t  *const p_version )
```

10.42.32.1 Brief description

Sets driver version based on compile time macros.

10.42.32.2 Detailed description

Table 1588:

Name	Description
SSP_SUCCESS	Successful version get.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.42.33 R_SCI_SIIC_MasterOpen

```
ssp_err_t R_SCI_SIIC_MasterOpen ( i2c_ctrl_t  *const p_api_ctrl ,  i2c_cfg_t
const *const p_cfg )
```

10.42.33.1 Brief description

Opens the I2C device. Power on I2C peripheral and perform initialization described in hardware manual.

10.42.33.2 Detailed description

This function will reconfigure the clock settings of the peripheral when a device with a lower rate than previously configured is opened.

Table 1589:

Name	Description
SSP_SUCCESS	Requested baud rate was valid.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL or if clock rate greater than 400KHz.
SSP_ERR_INVALID_RATE	The requested rate cannot be set.
SSP_ERR_IN_USE	Lock was not acquired
SSP_ERR_IRQ_BSP_DISABLED	Event information could not be found.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

- eventInfoGet

10.42.33.3 Function steps

- Disable, clear and configure interrupts

10.42.34 R_SCI_SIIC_MasterClose

```
ssp_err_t R_SCI_SIIC_MasterClose ( i2c_ctrl_t  *const p_api_ctrl )
```

10.42.34.1 Brief description

Closes the I2C device. Power down I2C peripheral.

10.42.34.2 Detailed description

This function will safely terminate any in-progress I2C transfer with the device. If a transfer is aborted, the user will be notified via callback with an abort event. Since the callback is optional, this function will also return a specific error code in this situation.

Table 1590:

Name	Description
SSP_SUCCESS	Device closed without issue.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_ABORTED	Device was closed while a transfer was in-progress.
SSP_ERR_NOT_OPEN	Device was not even opened.

10.42.35 R_SCI_SIIC_MasterRead

```
ssp_err_t R_SCI_SIIC_MasterRead ( i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const bytes ,   bool const restart )
```

10.42.35.1 Brief description

Performs a read from the I2C device.

10.42.35.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C read operation will begin. When no callback is provided by the user, this function performs a blocking read. Otherwise, the

read operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_RX_COMPLETE in the callback.

Table 1591:

Name	Description
SSP_SUCCESS	Function executed without issue.
SSP_ERR_ASSERTION	The parameter p_ctrl, p_dest is NULL, bytes is 0.
SSP_ERR_INVALID_SIZE	Provided number of bytes more than uint16_t size(65535) while DTC is used for data transfer.
SSP_ERR_IN_USE	Another transfer was in-progress.
SSP_ERR_NOT_OPEN	Device was not even opened.

10.42.36 R_SCI_SIIC_MasterWrite

```
ssp_err_t R_SCI_SIIC_MasterWrite ( i2c_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_src ,   uint32_t const bytes ,   bool const restart )
```

10.42.36.1 Brief description

Performs a write to the I2C device.

10.42.36.2 Detailed description

This function will fail if there is already an in-progress I2C transfer on the associated channel. Otherwise, the I2C write operation will begin. When no callback is provided by the user, this function performs a blocking write. Otherwise, the write operation is non-blocking and the caller will be notified when the operation has finished by an I2C_EVENT_TX_COMPLETE in the callback.

Table 1592:

Name	Description
SSP_SUCCESS	Function executed without issue.
SSP_ERR_ASSERTION	p_ctrl, p_src is NULL.
SSP_ERR_INVALID_SIZE	Provided number of bytes more than uint16_t size(65535) while DTC is used for data transfer.
SSP_ERR_IN_USE	Another transfer was in-progress.

Table 1592: (Continued)

Name	Description
SSP_ERR_NOT_OPEN	Device was not even opened.

10.42.37 R_SCI_SIIC_MasterReset

```
ssp_err_t R_SCI_SIIC_MasterReset ( i2c_ctrl_t  *const p_api_ctrl )
```

10.42.37.1 Brief description

Aborts any in-progress transfer and forces the I2C peripheral into a ready state.

10.42.37.2 Detailed description

This function will safely terminate any in-progress I2C transfer with the device. If a transfer is aborted, the user will be notified via callback with an abort event. Since the callback is optional, this function will also return a specific error code in this situation.

Table 1593:

Name	Description
SSP_SUCCESS	Channel was reset without issue.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_ABORTED	A transfer was aborted while resetting the hardware.
SSP_ERR_NOT_OPEN	Device was not even opened.

10.42.38 R_SCI_SIIC_MasterSlaveAddressSet

```
ssp_err_t R_SCI_SIIC_MasterSlaveAddressSet ( i2c_ctrl_t  *const p_api_ctrl ,
  uint16_t const slave ,  i2c_addr_mode_t  const addr_mode )
```

10.42.38.1 Brief description

Sets address and addressing mode of the slave device.

10.42.38.2 Detailed description

This function is used to set the device address and addressing mode of the slave without reconfiguring the entire bus.

Table 1594:

Name	Description
SSP_SUCCESS	Address of the slave is set correctly.
SSP_ERR_ASSERTION	p_ctrl or address is NULL.
SSP_ERR_IN_USE	Another transfer was in-progress.
SSP_ERR_NOT_OPEN	Device was not even opened.

10.42.39 g_sci_siic_master_version

`ssp_version_t::g_sci_siic_master_version`

10.42.39.1 Detailed description

Name of module used by error logger macro Version data structure used by error logger macro.

10.42.39.2 Initialized as

```
g_sci_siic_master_version=
{
    .api_version_major  = I2C_MASTER_API_VERSION_MAJOR,
.api_version_minor  = I2C_MASTER_API_VERSION_MINOR,
.code_version_minor = SCI_SIIC_MASTER_CODE_VERSION_MINOR,
    .code_version_major = SCI_SIIC_MASTER_CODE_VERSION_MAJOR,
}
```

10.42.40 g_dummy_write_data_for_read_op

`const uint8_t g_dummy_write_data_for_read_op`

10.42.40.1 Detailed description

constant used as the source location for the DTC dummy write

10.42.41 g_i2c_master_on_sci

`i2c_api_master_t::g_i2c_master_on_sci`

10.42.41.1 Detailed description

SCI SIIC Implementation of I2C device master interface

10.42.41.2 Initialized as

```
g_i2c_master_on_sci=
{
    .versionGet = R_SCI_SIIC_MasterVersionGet,
    .open       = R_SCI_SIIC_MasterOpen,
    .close      = R_SCI_SIIC_MasterClose,
    .read       = R_SCI_SIIC_MasterRead,
    .write      = R_SCI_SIIC_MasterWrite,
    .reset      = R_SCI_SIIC_MasterReset,
    .slaveAddressSet = R_SCI_SIIC_MasterSlaveAddressSet
}
```

10.42.42 Extensions

10.42.42.1 sci_i2c_instance_ctrl_t

sci_i2c_instance_ctrl_t

Detailed description

I2C control structure. DO NOT INITIALIZE.

Variables

- i2c_cfg_t info

  Information describing I2C device.

- uint32_t open

  Flag to determine if the device is open.

- void * p_reg

  Base register for this channel.

- transfer_instance_t const * p_transfer_tx

  DTC instance for I2C transmit.Set to NULL if unused.

  DTC/DMA support

- transfer_instance_t const * p_transfer_rx

  DTC instance for I2C receive. Set to NULL if unused.

- IRQn_Type rxi_irq

  Receive IRQ number.

- IRQn_Type txi_irq

  Transmit IRQ number.
- IRQn_Type tei_irq

  Transmit end IRQ number.
- uint8_t * p_buff

  Holds the data associated with the transfer
- uint32_t total

  Holds the total number of data bytes to transfer
- uint32_t remain

  Tracks the remaining data bytes to transfer
- uint32_t loaded

  Tracks the number of data bytes written to the register
- uint8_t addr_low

  Holds the last address byte to issue
- uint8_t addr_high

  Holds the first address byte to issue in 10-bit mode
- uint8_t addr_total

  Holds the total number of address bytes to transfer
- uint8_t addr_remain

  Tracks the remaining address bytes to transfer
- uint8_t addr_loaded

  Tracks the number of address bytes written to the register
- bool read

  Holds the direction of the data byte transfer
- bool restart

  Holds whether or not the restart should be issued when done
- bool err

  Tracks whether or not an error occurred during processing
- bool restarted

  Tracks whether or not a restart was issued during the previous transfer
- bool transaction_completed

  Tracks if the transaction started earlier was completed
- bool do_dummy_read

- bool activation_on_rxi

  << Tracks whether a dummy read is issued on the first RX < Tracks whether the transfer is activated on RXI interrupt

- bool activation_on_txi

  < Tracks whether the transfer is activated on TXI interrupt

10.42.42.2 sci_i2c_extended_cfg

sci_i2c_extended_cfg

Detailed description

SCI I2C extended configuration

Variables

- bool bitrate_modulation

  Bitrate Modulation Function enable or disable

10.43 Simple SPI on SCI

Driver for the Simple SPI on SCI.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

This module supports simple SPI serial communication using the microcontroller's SCI peripheral. The Interface is defined in r_spi_api.h. This module implements SPI Interface.

10.43.1 Functions

- R_SCI_SPI_Open
- R_SCI_SPI_Read
- R_SCI_SPI_Write
- R_SCI_SPI_WriteRead
- r_sci_spi_write_read_common
- R_SCI_SPI_Close
- R_SCI_SPI_VersionGet

10.43.2 Variables

- g_spi_on_sci

10.43.3 Defines

- #define SCI_SPI_CODE_VERSION_MAJOR
  Initial value:`(1U)`
- #define SCI_SPI_CODE_VERSION_MINOR
  Initial value:`(7U)`

10.43.4 R_SCI_SPI_Open

```
ssp_err_t R_SCI_SPI_Open ( spi_ctrl_t  * p_api_ctrl ,  spi_cfg_t  const
*const p_cfg )
```

10.43.4.1 Brief description

Initialize a channel for SPI communication mode. Implements open. This function performs the following tasks: Performs parameter checking and processes error conditions. Applies power to the SPI channel. Disables interrupts. Initializes the associated registers with default value and the user-configurable options. Provides the channel handle for use with other API functions. Updates user-configurable file if necessary.

10.43.4.2 Detailed description

Table 1595:

Name	Description
SSP_SUCCESS	Channel initialized successfully.
SSP_ERR_ASSERTION	One of the following invalid parameter passed. • Pointer p_api_ctrl is NULL • Pointer p_cfg is NULL
SSP_ERR_IN_USE	Channel currently in operation; Close channel first.
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
See	Common Error Codes or functions called by this function for other possible return codes. This function calls: • productFeatureGet

NOTE: This function is reentrant.

NOTE: The bit-rate argument in p_cfg ranges from 2500 to 7.5m for Simple SPI at PCLK=120 MHz. For RSPI, BRDV is fixed at 0 to get the maximum bit rate. The range is 10.0 mbps to 30.0 mbps at PCLK=120.0 MHz.

10.43.5 R_SCI_SPI_Read

```
ssp_err_t R_SCI_SPI_Read ( spi_ctrl_t  *const p_api_ctrl ,   void const
* p_dest ,   uint32_t const length ,  spi_bit_width_t  const bit_width )
```

10.43.5.1 Brief description

Receive data from an SPI device. Implements read. The function performs the following tasks:

10.43.5.2 Detailed description

- Performs parameter checking and processes error conditions.
- Disable Interrupts.
- Set-up data bit width per user request.
- Enable transmitter.
- Enable receiver.
- Enable interrupts.
- Start data transmission with dummy data via transmit buffer empty interrupt.
- Copy data from source buffer to the SPI data register for transmission.
- Receive data from receive buffer full interrupt occurs and copy data to the buffer of destination.
- Complete data reception via receive buffer full interrupt and transmitting dummy data.
- Disable transmitter.
- Disable receiver.
- Disable interrupts.

Table 1596:

Name	Description
SSP_SUCCESS	Read operation successfully completed.
SSP_ERR_ASSERTION	One of the following invalid parameters passed • Pointer p_api_ctrl is NULL • Bit width is not 8 bits • Length is equal to 0 • Pointer to destination is NULL
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.

Table 1596: (Continued)

Name	Description
See	Common Error Codes or functions called by this function for other possible return codes. This function calls: • reset

NOTE: This function is reentrant.

10.43.6 R_SCI_SPI_Write

```
ssp_err_t R_SCI_SPI_Write ( spi_ctrl_t  *const p_api_ctrl ,   void const
* p_src ,   uint32_t const length ,  spi_bit_width_t  const bit_width )
```

10.43.6.1 Brief description

Transmit data to a SPI device. Implements write.

10.43.6.2 Detailed description

- The function performs the following tasks:
- Performs parameter checking and processes error conditions.
- Disable Interrupts.
- Setup data bit width per user request.
- Enable transmitter.
- Enable receiver.
- Enable interrupts.
- Start data transmission with data via transmit buffer empty interrupt.
- Copy data from source buffer to the SPI data register for transmission.
- Receive data from receive buffer full interrupt occurs and do nothing with the received data.
- Complete data transmission via receive buffer full interrupt.
- Disable transmitter.
- Disable receiver.
- Disable interrupts.

Table 1597:

Name	Description
SSP_SUCCESS	Write operation successfully completed.
SSP_ERR_ASSERTION	One of the following invalid parameters passed • Pointer p_api_ctrl is NULL • Pointer to source is NULL • Length is equal to 0 • Bit width is not equal to 8 bits
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.
See	Common Error Codes or functions called by this function for other possible return codes. This function calls: • reset

NOTE: This function is reentrant.

10.43.7 R_SCI_SPI_WriteRead

```
ssp_err_t R_SCI_SPI_WriteRead ( spi_ctrl_t  *const p_api_ctrl ,   void const
* p_src ,   void const * p_dest ,   uint32_t const length ,  spi_bit_width_t
const bit_width )
```

10.43.7.1 Brief description

Simultaneously transmit data to SPI device while receiving data from SPI device (full duplex). Implements writeRead. The function performs the following tasks:

10.43.7.2 Detailed description

- Performs parameter checking and processes error conditions.
- Disable Interrupts. Setup data bit width per user request. Enable transmitter. Enable receiver. Enable interrupts. Start data transmission using transmit buffer empty interrupt. Copy data from source buffer to the SPI data register for transmission. Receive data from receive buffer full interrupt occurs and copy data to the buffer of destination. Complete data transmission and reception via receive buffer full interrupt. Disable transmitter. Disable receiver. Disable interrupts.

Table 1598:

Name	Description
SSP_SUCCESS	Write operation successfully completed.
SSP_ERR_ASSERTION	One of the following invalid parameters passed • Pointer p_api_ctrl is NULL • Pointer to source is NULL • Pointer to destination is NULL • Length is equal to 0 • Bit width is not equal to 8 bits
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.
See	Common Error Codes or functions called by this function for other possible return codes. This function calls: • reset

NOTE: This function is reentrant.

10.43.8 r_sci_spi_write_read_common

```
ssp_err_t r_sci_spi_write_read_common ( sci_spi_instance_ctrl_t  *const p_ctrl ,
  void const * p_src ,   void const * p_dest ,   uint32_t const length ,
 spi_bit_width_t  const bit_width ,  spi_operation_t  tx_rx_mode )
```

10.43.9 R_SCI_SPI_Close

```
ssp_err_t R_SCI_SPI_Close ( spi_ctrl_t  *const p_api_ctrl )
```

10.43.9.1 Brief description

Handle the closing of a channel by the following task. Implements close Power off the channel. Disables all the associated interrupts. Update channel status.

10.43.9.2 Detailed description

Table 1599:

Name	Description
SSP_SUCCESS	Channel successfully closed.
SSP_ERR_ASSERTION	The parameter p_api_ctrl is NULL.
SSP_ERR_NOT_OPEN	The channel has not been opened. Open the channel first.

NOTE: This function is reentrant.

10.43.10 R_SCI_SPI_VersionGet

```
ssp_err_t R_SCI_SPI_VersionGet ( ssp_version_t  * p_version )
```

10.43.10.1 Brief description

Get the version information of the underlying driver. Implements versionGet.

10.43.10.2 Detailed description

Table 1600:

Name	Description
SSP_SUCCESS	Successful version get.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

NOTE: This function is reentrant.

10.43.11 g_spi_on_sci

```
spi_api_t::g_spi_on_sci
```

10.43.11.1 Detailed description

Filled in Interface API structure for this Instance.

10.43.12 Extensions

10.43.12.1 sci_spi_instance_ctrl_t

sci_spi_instance_ctrl_t

Detailed description

SPI instance control block. DO NOT INITIALIZE.

Variables

- uint8_t channel

  Channel number to be used.
- uint32_t channel_opened

  Internal flag to indicate the peripheral was initialized.
- transfer_instance_t const * p_transfer_tx

  To use SPI DTC/DMA write transfer.
- transfer_instance_t const * p_transfer_rx

  To use SPI DTC/DMA read transfer.
- void(* p_callback)(*p_args)

  Pointer to user callback function.
- void const * p_context

  Pointer to the higher level device context.
- void * p_reg

  Base register for this channel.
- IRQn_Type rxi_irq

  Receive IRQ number.
- IRQn_Type txi_irq

  Transmit IRQ number.
- IRQn_Type tei_irq

  Transmit end IRQ number.
- IRQn_Type eri_irq

  Error IRQ number.
- void * p_src
- void * p_dest
- uint16_t tx_count
- uint16_t rx_count
- uint32_t xfr_length

- uint8_t bytes_per_transfer
- bool do_rx_now
- bool do_tx
- spi_operation_t transfer_mode
- uint8_t rx_data
- bsp_lock_t resource_lock_tx_rx

  Resource lock for transmission/reception

10.43.12.2 sci_spi_extended_cfg

sci_spi_extended_cfg

Detailed description

SCI SPI extended configuration

Variables

- bool bitrate_modulation

  Bitrate Modulation Function enable or disable

10.44 UART on SCI

Driver for the UART on SCI.

10.44.1 Summary

This module supports the UART on SCI. It implements the UART interface and drives SCI as a full-duplex UART communication port. This module can drive all SCI channels as UART ports.

Extends UART Interface.

NOTE: *This module can use either the 16-stage hardware FIFO or a DTC transfer implementation to write multiple bytes.*

10.44.2 Functions

- R_SCI_UartOpen
- R_SCI_UartClose
- R_SCI_UartRead
- R_SCI_UartWrite
- R_SCI_UartBaudSet
- R_SCI_UartInfoGet
- R_SCI_UartVersionGet

10.44.3 Defines

- #define SCI_UART_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCI_UART_CODE_VERSION_MINOR

  Initial value:`(5U)`

10.44.4 R_SCI_UartOpen

```
ssp_err_t R_SCI_UartOpen ( uart_ctrl_t  *const p_api_ctrl ,  uart_cfg_t  const
*const p_cfg )
```

10.44.4.1 Detailed description

Configures the UART driver based on the input configurations. If reception is enabled at compile time, reception is enabled at the end of this function.

Table 1601:

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_ASSERTION	Pointer to UART control block or configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter setting found in the configuration structure.
SSP_ERR_IN_USE	Control block has already been opened or channel is being used by another instance. Call close then open() to reconfigure.
SSP_ERR_IRQ_BSP_DISABLED	A required interrupt does not exist in the vector table

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet
- systemClockFreqGet
- open

10.44.4.2 Function steps

- Make sure this channel exists.
- Reserve the hardware lock.
- Determine if this channel has a FIFO.
- Calculate the baud rate register settings.
- Configure the interrupts.
- Configure the transfer interface for transmission and reception if provided.
- Enable the SCI channel and reset the registers to their initial state.
- Set the default level of the TX pin to 1.
- Set the baud rate registers.
- Set the UART configuration settings provided in uart_cfg_t and uart_on_sci_cfg_t.
- If reception is enabled at build time, enable reception.

10.44.5 R_SCI_UartClose

```
ssp_err_t R_SCI_UartClose ( uart_ctrl_t  *const p_api_ctrl )
```

10.44.5.1 Detailed description

Disables interrupts, receiver, and transmitter. Closes lower level transfer drivers if used. Removes power and releases hardware lock.

Table 1602:

Name	Description
SSP_SUCCESS	Channel successfully closed.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL.
SSP_ERR_NOT_OPEN	The control block has not been opened

10.44.5.2 Function steps

- Mark the channel not open so other APIs cannot use it.
- Disable interrupts, receiver, and transmitter.
- If reception is enabled at build time, disable reception irqs.
- If transmission is enabled at build time, disable transmission irqs.
- Disable baud clock output.

- Close the lower level transfer instances.
- Clear control block parameters.
- Remove power to the channel.
- Unlock the SCI channel.

10.44.6 R_SCI_UartRead

```
ssp_err_t R_SCI_UartRead ( uart_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_dest ,   uint32_t const bytes )
```

10.44.6.1 Detailed description

Receives user specified number of bytes into destination buffer pointer.

Table 1603:

Name	Description
SSP_SUCCESS	Data reception successfully ends.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL. Number of transfers outside the max or min boundary when transfer instance used
SSP_ERR_INVALID_ARGUMENT	Destination address or data size is not valid for 9-bit mode.
SSP_ERR_NOT_OPEN	The control block has not been opened
SSP_ERR_UNSUPPORTED	SCI_UART_CFG_RX_ENABLE is set to 0

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- reset

NOTE: This API is only valid when SCI_UART_CFG_RX_ENABLE is enabled. If 9-bit data length is specified at R_SCI_UartOpen call, p_dest must be aligned 16-bit boundary.

10.44.6.2 Function steps

- Configure transfer instance to receive the requested number of bytes if transfer is used for reception.
- Do nothing in non-transfer case. Bytes will come in through the callback.

10.44.7 R_SCI_UartWrite

```
ssp_err_t R_SCI_UartWrite ( uart_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_src ,   uint32_t const bytes )
```

10.44.7.1 Detailed description

Transmits user specified number of bytes from the source buffer pointer.

Table 1604:

Name	Description
SSP_SUCCESS	Data transmission finished successfully.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL. Number of transfers outside the max or min boundary when transfer instance used
SSP_ERR_INVALID_ARGUMENT	Source address or data size is not valid for 9-bit mode.
SSP_ERR_NOT_OPEN	The control block has not been opened
SSP_ERR_IN_USE	A UART transmission is in progress
SSP_ERR_UNSUPPORTED	SCI_UART_CFG_TX_ENABLE is set to 0

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- reset

NOTE: This API is only valid when SCI_UART_CFG_TX_ENABLE is enabled. If 9-bit data length is specified at R_SCI_UartOpen call, p_src must be aligned on a 16-bit boundary.

10.44.7.2 Function steps

- Transmit interrupts must be disabled to start with.
- Save data to transmit to the control block. It will be transmitted in the TXI ISR.
- If a transfer instance is used for transmission, reset the transfer instance to transmit the requested data.
- Trigger a TXI interrupt. This triggers the transfer instance or a TXI interrupt if the transfer instance is not used.

10.44.8 R_SCI_UartBaudSet

```
ssp_err_t R_SCI_UartBaudSet ( uart_ctrl_t  *const p_api_ctrl ,   uint32_t
const baudrate )
```

10.44.8.1 Detailed description

Updates the baud rate.

ATTENTION: This terminates any in-progress transmission.

Table 1605:

Name	Description
SSP_SUCCESS	Baud rate was successfully changed.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL or the UART is not configured to use the internal clock.
SSP_ERR_INVALID_ARGUMENT	Illegal baud rate value is specified.
SSP_ERR_NOT_OPEN	The control block has not been opened

10.44.8.2 Function steps

- Calculate new baud rate register settings.
- Disables transmitter and receiver. This terminates any in-progress transmission.
- Apply new baud rate register settings.
- Enable receiver.

10.44.9 R_SCI_UartInfoGet

```
ssp_err_t R_SCI_UartInfoGet ( uart_ctrl_t  *const p_api_ctrl ,  uart_info_t
*const p_info )
```

10.44.9.1 Detailed description

Provides the driver information, including the maximum number of bytes that can be received or transmitted at a time.

Table 1606:

Name	Description
SSP_SUCCESS	Information stored in provided p_info.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL.
SSP_ERR_NOT_OPEN	The control block has not been opened

10.44.9.2 Function steps

- Store number of bytes that can be read at a time.
- Store number of bytes that can be written at a time.

10.44.10 R_SCI_UartVersionGet

```
ssp_err_t R_SCI_UartVersionGet ( ssp_version_t  * p_version )
```

10.44.10.1 Detailed description

Provides API and code version in the user provided pointer.

Table 1607:

Name	Direction	Description
p_version	in	Version number set here

Table 1608:

Name	Description
SSP_SUCCESS	Version information stored in provided p_version.
SSP_ERR_ASSERTION	p_version is NULL.

10.44.11 API Data

10.44.11.1 sci_clk_src_t

`sci_clk_src_t`

Detailed description

Enumeration for SCI clock source

Enumerated values

Name	Description
SCI_CLK_SRC_INT	Use internal clock for baud generation.
SCI_CLK_SRC_EXT	Use external clock for baud generation.

Name	Description
SCI_CLK_SRC_EXT8X	Use external clock 8x baud rate.
SCI_CLK_SRC_EXT16X	Use external clock 16x baud rate.

10.44.11.2 sci_uart_rx_fifo_trigger_t

`sci_uart_rx_fifo_trigger_t`

Detailed description

Receive FIFO trigger configuration.

Enumerated values

Name	Description
SCI_UART_RX_FIFO_TRIGGER_1	Callback after each byte is received without buffering.
SCI_UART_RX_FIFO_TRIGGER_MAX	Callback when FIFO is full or after 15 bit times with no data (fewer interrupts)

10.44.12 Extensions

10.44.12.1 sci_uart_instance_ctrl_t

sci_uart_instance_ctrl_t

Detailed description

UART instance control block.

Variables

- uint8_t channel

  Channel number.
- uint8_t fifo_depth

  FIFO depth of the UART channel.
- uint8_t rx_transfer_in_progress

  Set to 1 if a receive transfer is in progress, 0 otherwise.
- uint8_t data_bytes

  1 byte for 7 or 8 bit data, 2 bytes for 9 bit data
- uint8_t bitrate_modulation

  1 if bit rate modulation is enabled, 0 otherwise

- uint32_t open

  Used to determine if the channel is configured.
- transfer_instance_t const * p_transfer_rx

  Optional transfer instance used to send or receive multiple bytes without interrupts.
- transfer_instance_t const * p_transfer_tx

  Optional transfer instance used to send or receive multiple bytes without interrupts.
- uint8_t const * p_tx_src

  Source buffer pointer used to fill hardware FIFO from transmit ISR.
- uint32_t tx_src_bytes

  Size of source buffer pointer used to fill hardware FIFO from transmit ISR.
- IRQn_Type rxi_irq

  Receive IRQ number.
- IRQn_Type txi_irq

  Transmit IRQ number.
- IRQn_Type tei_irq

  Transmit end IRQ number.
- IRQn_Type eri_irq

  Error IRQ number.
- void(* p_callback)(*p_args)

  Pointer to callback function.
- void const * p_context

  Pointer to user interrupt context data.
- void * p_reg

  Base register for this channel.
- void(* p_extpin_ctrl)(uint32_t channel, uint32_t level)

  External pin control.

10.44.12.2 uart_on_sci_cfg_t

uart_on_sci_cfg_t

Detailed description

UART on SCI device Configuration

Variables

- sci_clk_src_t clk_src

  Use SCI_CLK_SRC_INT/EXT8X/EXT16X.

- bool baudclk_out

  Baud rate clock output enable.

- bool rx_edge_start

  Start reception on falling edge.

- bool noisecancel_en

  Noise cancel enable.

- sci_uart_rx_fifo_trigger_t rx_fifo_trigger

  Receive FIFO trigger level, unused if channel has no FIFO or if DTC is used.

- void(* p_extpin_ctrl)(uint32_t channel, uint32_t level)

  Pointer to the user callback function to control external GPIO pin control used as RTS signal.channelThe UART channel used. levelWhen level is 0, assert RTS. When level is 1, deassert RTS.

- bool bitrate_modulation

  Bitrate Modulation Function enable or disable.

10.45 SDMMC

Driver for the SD/MMC Host Interface (SDHI).

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

SD/MMC driver to access SD, eMMC, and SDIO devices.

10.45.1 Functions

- R_SDMMC_Open
- R_SDMMC_Close
- R_SDMMC_Read
- R_SDMMC_Write
- R_SDMMC_Control
- R_SDMMC_ReadIo
- R_SDMMC_WriteIo
- R_SDMMC_ReadIoExt
- R_SDMMC_WriteIoExt
- R_SDMMC_IoIntEnable
- R_SDMMC_VersionGet
- R_SDMMC_InfoGet
- R_SDMMC_Erase

10.45.2 Defines

- #define SDMMC_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define SDMMC_CODE_VERSION_MINOR

  Initial value:`(6U)`

10.45.3 R_SDMMC_Open

```
ssp_err_t R_SDMMC_Open ( sdmmc_ctrl_t  *const p_api_ctrl ,  sdmmc_cfg_t  const
*const p_cfg )
```

10.45.3.1 Detailed description

Initializes the SDHI hardware and completes identification and configuration for the SD or eMMC device. This procedure requires several sequential commands. This API blocks until all identification and configuration commands are complete.

For SDIO, SDIO interrupts are enabled after card identification is complete. SDIO interrupts can be disabled using sdmmc_api_t::ioIntEnable().

Implements open.

Table 1609:

Name	Description
SSP_SUCCESS	Port is available and is now open for read/write/control access.
SSP_ERR_ASSERTION	Null Pointer or block size is not in the valid range of 1-512.
SSP_ERR_INVALID_ARGUMENT	Block size must be 512 bytes for SD cards and eMMC devices. It is configurable for SDIO only.
SSP_ERR_ALREADY_OPEN	Driver has already been opened with this instance of the control structure.
SSP_ERR_HW_LOCKED	The channel specified has already been opened.
SSP_ERR_CARD_INIT_FAILED	Hardware related failure occurred, with the MCU or with the card itself.
SSP_ERR_IRQ_BSP_DISABLED	Access interrupt is not enabled.
SSP_ERR_CARD_NOT_INSERTED	Card detection is enabled and no card is plugged in.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet
- systemClockFreqGet

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.3.2 Function steps

- Verify the requested hardware channel exists on the MCU.
- Configure interrupts.
- Perform the identification procedure for SD card or eMMC device.
- Configure bus clock, block size, and bus width.
- Check to see if the card is write protected (SD cards only).

10.45.4 R_SDMMC_Close

```
ssp_err_t R_SDMMC_Close ( sdmmc_ctrl_t  *const p_api_ctrl )
```

10.45.4.1 Detailed description

Closes an open SD/MMC device. Implements close.

Table 1610:

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.4.2 Function steps

- Disable SDHI interrupts.
- Close the transfer driver.
- Release hardware lock.

10.45.5 R_SDMMC_Read

```
ssp_err_t R_SDMMC_Read ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const start_sector ,   uint32_t const sector_count )
```

10.45.5.1 Detailed description

Reads data from an SD or eMMC device. Up to 0x10000 sectors can be read at a time. Implements read.

This function blocks until the command is sent and the response is received. A callback with the event SDMMC_EVENT_TRANSFER_COMPLETE is called when the read data is available.

Table 1611:

Name	Description
SSP_SUCCESS	Data read successfully.
SSP_ERR_ASSERTION	NULL pointer.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_READ_FAILED	Read operation failed.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.6 R_SDMMC_Write

```
ssp_err_t R_SDMMC_Write ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_source ,   uint32_t const start_sector ,   uint32_t const sector_count )
```

10.45.6.1 Detailed description

Writes data to an SD or eMMC device. Up to 0x10000 sectors can be written at a time. Implements write.

This function blocks until the command is sent and the response is received. A callback with the event SDMMC_EVENT_TRANSFER_COMPLETE is called when the all data has been written.

Table 1612:

Name	Description
SSP_SUCCESS	Card write finished successfully.

Table 1612: (Continued)

Name	Description
SSP_ERR_ASSERTION	Handle or Source address is NULL.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_WRITE_PROTECTED	SD card is Write Protected.
SSP_ERR_WRITE_FAILED	Write operation failed.

NOTE: This function is reentrant for different channels.

10.45.7 R_SDMMC_Control

```
ssp_err_t R_SDMMC_Control ( sdmmc_ctrl_t  *const p_api_ctrl ,  ssp_command_t
const command ,   void * p_data )
```

10.45.7.1 Detailed description

Sends control commands to and receives the status of the SD/MMC device. Implements control.

Table 1613:

Name	Description
SSP_SUCCESS	Command executed successfully.
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_INVALID_ARGUMENT	Command is invalid.
SSP_ERR_INVALID_SIZE	Block size not in valid range of 1-512 for SDIO or 512 only for SD cards and eMMC.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.8 R_SDMMC_ReadIo

```
ssp_err_t R_SDMMC_ReadIo ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_data ,   uint32_t const function ,   uint32_t const address )
```

10.45.8.1 Detailed description

The Read function reads a one byte register from an SDIO card. Implements readIo.

This function blocks until the command is sent and the response is received. p_data contains the register value read when this function returns.

Table 1614:

Name	Description
SSP_SUCCESS	Data read successfully.
SSP_ERR_ASSERTION	NULL pointer.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_READ_FAILED	Read operation failed.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.9 R_SDMMC_WriteIo

```
ssp_err_t R_SDMMC_WriteIo ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_data ,   uint32_t const function ,   uint32_t const address ,
 sdmmc_io_write_mode_t  const read_after_write )
```

10.45.9.1 Detailed description

Writes a one byte register to an SDIO card. Implements writeIo.

This function blocks until the command is sent and the response is received. The register has been written when this function returns. If read_after_write is true, p_data contains the register value read when this function returns.

Table 1615:

Name	Description
SSP_SUCCESS	Card write finished successfully.
SSP_ERR_ASSERTION	Handle or Source address is NULL.
SSP_ERR_NOT_OPEN	Driver has not been initialized.

Table 1615: (Continued)

Name	Description
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_WRITE_FAILED	Write operation failed.

NOTE: This function is reentrant for different channels.

10.45.10 R_SDMMC_ReadIoExt

```
ssp_err_t R_SDMMC_ReadIoExt ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t
*const p_dest ,   uint32_t const function ,   uint32_t const address ,   uint32_t
*const count ,  sdmmc_io_transfer_mode_t  transfer_mode ,
 sdmmc_io_address_mode_t  address_mode )
```

10.45.10.1 Detailed description

Reads data from an SDIO card function. Implements readIoExt.

This function blocks until the command is sent and the response is received. A callback with the event SDMMC_EVENT_TRANSFER_COMPLETE is called when the read data is available.

Table 1616:

Name	Description
SSP_SUCCESS	Data read successfully.
SSP_ERR_ASSERTION	NULL pointer, or count is not in the valid range of 1-512 for byte mode or 1-511 for block mode.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_READ_FAILED	Read operation failed.

NOTE: This function is reentrant for different channels. It is not reentrant for the same channel.

10.45.11 R_SDMMC_WriteIoExt

```
ssp_err_t R_SDMMC_WriteIoExt ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_source ,   uint32_t const function ,   uint32_t const address ,
  uint32_t const count ,  sdmmc_io_transfer_mode_t  transfer_mode ,
 sdmmc_io_address_mode_t  address_mode )
```

10.45.11.1 Detailed description

Writes data to an SDIO card function. Implements writeIoExt.

This function blocks until the command is sent and the response is received. A callback with the event SDMMC_EVENT_TRANSFER_COMPLETE is called when the all data has been written.

Table 1617:

Name	Description
SSP_SUCCESS	Card write finished successfully.
SSP_ERR_ASSERTION	NULL pointer, or count is not in the valid range of 1-512 for byte mode or 1-511 for block mode.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_WRITE_FAILED	Write operation failed.

NOTE: This function is reentrant for different channels.

10.45.12 R_SDMMC_IoIntEnable

```
ssp_err_t R_SDMMC_IoIntEnable ( sdmmc_ctrl_t  *const p_api_ctrl ,
  bool enable )
```

10.45.12.1 Detailed description

Enables or disables the SDIO Interrupt. Implements sdmmc_api_t::ioIntEnable().

Table 1618:

Name	Description
SSP_SUCCESS	Card enabled or disabled SDIO interrupts successfully.

Table 1618: (Continued)

Name	Description
SSP_ERR_ASSERTION	NULL pointer.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.

NOTE: This function is reentrant for different channels.

10.45.12.2 Function steps

- Enable or disable interrupt.

10.45.13 R_SDMMC_VersionGet

```
ssp_err_t R_SDMMC_VersionGet ( ssp_version_t  *const p_version )
```

10.45.13.1 Detailed description

Returns the version of the firmware and API. Implements versionGet.

Table 1619:

Name	Description
SSP_SUCCESS	Function executed successfully.
SSP_ERR_ASSERTION	Null Pointer.

NOTE: This function is reentrant.

10.45.14 R_SDMMC_InfoGet

```
ssp_err_t R_SDMMC_InfoGet ( sdmmc_ctrl_t  *const p_api_ctrl ,  sdmmc_info_t
*const p_info )
```

10.45.14.1 Detailed description

Provides information about the connected device and driver status. Implements infoGet.

NOTE: *This function is reentrant.*

10.45.14.2 Function steps

- Copy information stored during open.
- Check if the card is busy.

10.45.15 R_SDMMC_Erase

```
ssp_err_t R_SDMMC_Erase ( sdmmc_ctrl_t  *const p_api_ctrl ,   uint32_t
const start_sector ,   uint32_t const sector_count )
```

10.45.15.1 Detailed description

Erases sectors of an SD card or eMMC device. Implements erase.

This function blocks until erase is complete.

Table 1620:

Name	Description
SSP_SUCCESS	Erase operation requested.
SSP_ERR_ASSERTION	A required pointer is NULL.
SSP_ERR_NOT_OPEN	Driver has not been initialized.
SSP_ERR_CARD_NOT_READY	Card was unplugged.
SSP_ERR_TRANSFER_BUSY	Driver is busy with a previous operation.
SSP_ERR_WRITE_PROTECTED	SD card is Write Protected.
SSP_ERR_ERASE_FAILED	Erase operation unsuccessful.

NOTE: This function is reentrant for different channels.

10.45.15.2 Function steps

- Send command to set start erase address (CMD35 for eMMC, CMD32 for SD).
- Send command to set end erase address (CMD36 for eMMC, CMD33 for SD).
- Send erase command (CMD38).

10.45.16 API Data

10.45.16.1 sdmmc_card_detect_t

`sdmmc_card_detect_t`

Detailed description

Card detection configuration options.

Enumerated values

Name	Description
SDMMC_CARD_DETECT_NONE	Card detection unused.
SDMMC_CARD_DETECT_CD	Card detection using the SDnCD pin.

10.45.16.2 sdmmc_transfer_dir_t

`sdmmc_transfer_dir_t`

Enumerated values

Name	Description
SDMMC_TRANSFER_DIR_NONE	
SDMMC_TRANSFER_DIR_READ	
SDMMC_TRANSFER_DIR_WRITE	

10.45.16.3 sdmmc_io_command_t

`sdmmc_io_command_t`

Enumerated values

Name	Description
SDMMC_IO_COMMAND_DIRECT_IO	
SDMMC_IO_COMMAND_DIRECT_IO_EXT	

10.45.17 Extensions

10.45.17.1 sdhi_event_t

Detailed description

Variables

word

bit

10.45.17.2 sdhi_sdio_event_t

Detailed description

Variables

word

bit

10.45.17.3 sdhi_int_status_t

sdhi_int_status_t

Detailed description

Variables

- uint32_t info2_mask

10.45.17.4 sdmmc_io_mode_t

sdmmc_io_mode_t

Detailed description

Variables

- sdmmc_io_command_t command
- sdmmc_io_transfer_mode_t transfer_mode
- sdmmc_io_address_mode_t address_mode
- sdmmc_io_write_mode_t write_mode

10.45.17.5 sdmmc_instance_ctrl_t

sdmmc_instance_ctrl_t

Detailed description

SDMMC instance control block. This is private to the SSP and should not be used or modified by the application.

Variables

- uint32_t open
- sdmmc_hw_t hw
- transfer_instance_t const * p_lower_lvl_transfer
- sdmmc_info_t status
- bool transfer_in_progress
- void(* p_callback)(*p_args)
- void const * p_context
- R_SDHI0_Type * p_reg
- sdhi_event_t sdhi_event
- IRQn_Type transfer_irq
- sdmmc_transfer_dir_t transfer_dir
- uint8_t * p_transfer_data
- uint32_t transfer_blocks_total
- uint32_t transfer_block_current
- uint32_t transfer_block_size
- uint32_t aligned_buff[SDMMC_MAX_BLOCK_SIZE/sizeof(uint32_t)]

10.45.17.6 sdmmc_extended_cfg_t

sdmmc_extended_cfg_t

Detailed description

Extended SDMMC configuration, to be pointed to p_extend.

Variables

- uint32_t block_size

  Block size in bytes. Block size must be 512 bytes for SD cards and eMMC devices. Block size can be 1-512 bytes for SDIO.

- sdmmc_card_detect_t card_detect

  Whether or not card detection is used.

10.46 SLCDC

Driver for the Segment LCD Controller (SLCDC).

10.46.1 Summary

Extends SLCDC Interface.

10.46.2 Functions

- R_SLCDC_Open
- R_SLCDC_Write
- R_SLCDC_Modify
- R_SLCDC_Start
- R_SLCDC_Stop
- R_SLCDC_ContrastIncrease
- R_SLCDC_ContrastDecrease
- R_SLCDC_SetDisplayArea
- R_SLCDC_Close
- R_SLCDC_VersionGet

10.46.3 R_SLCDC_Open

```
ssp_err_t R_SLCDC_Open ( slcdc_ctrl_t  *const p_api_ctrl ,  slcdc_cfg_t  const
*const p_cfg )
```

10.46.3.1 Detailed description

Table 1621:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_HW_LOCKED	SLCDC resource is locked.

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet

10.46.4 R_SLCDC_Write

```
ssp_err_t R_SLCDC_Write ( slcdc_ctrl_t  *const p_api_ctrl ,  slcdc_size_t
const start_segment ,  slcdc_size_t  const *const p_data ,  slcdc_size_t
const segment_count )
```

10.46.4.1 Detailed description

Table 1622:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized

10.46.5 R_SLCDC_Modify

```
ssp_err_t R_SLCDC_Modify ( slcdc_ctrl_t  *const p_api_ctrl ,  slcdc_size_t
const segment_number ,  slcdc_size_t  const data ,  slcdc_size_t
const data_mask )
```

10.46.5.1 Detailed description

Table 1623:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized

10.46.6 R_SLCDC_Start

```
ssp_err_t R_SLCDC_Start ( slcdc_ctrl_t  *const p_api_ctrl )
```

10.46.6.1 Detailed description

Table 1624:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized

10.46.7 R_SLCDC_Stop

```
ssp_err_t R_SLCDC_Stop ( slcdc_ctrl_t  *const p_api_ctrl )
```

10.46.7.1 Detailed description

Table 1625:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized

10.46.8 R_SLCDC_ContrastIncrease

```
ssp_err_t R_SLCDC_ContrastIncrease ( slcdc_ctrl_t  *const p_api_ctrl )
```

10.46.8.1 Detailed description

Table 1626:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized
SSP_ERR_UNSUPPORTED	Unsupported operation

10.46.9 R_SLCDC_ContrastDecrease

```
ssp_err_t R_SLCDC_ContrastDecrease ( slcdc_ctrl_t  *const p_api_ctrl )
```

10.46.9.1 Detailed description

Table 1627:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized
SSP_ERR_UNSUPPORTED	Unsupported operation

10.46.10 R_SLCDC_SetDisplayArea

```
ssp_err_t R_SLCDC_SetDisplayArea ( slcdc_ctrl_t  *const p_api_ctrl ,
 slcdc_display_area_t  const display_area )
```

10.46.10.1 Detailed description

Table 1628:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_UNSUPPORTED	Unsupported operation
SSP_ERR_NOT_OPEN	Device is not opened or initialized
SSP_ERR_NOT_ENABLED	RTC not enabled for blink operation

See Common Error Codes for other possible return codes. This function calls

- productFeatureGet

10.46.11 R_SLCDC_Close

```
ssp_err_t R_SLCDC_Close ( slcdc_ctrl_t  *const p_api_ctrl )
```

10.46.11.1 Detailed description

Table 1629:

Name	Description
SSP_SUCCESS	Device was opened successfully.
SSP_ERR_ASSERTION	Pointer to the control block or the configuration structure is NULL.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_NOT_OPEN	Device is not opened or initialized

10.46.12 R_SLCDC_VersionGet

```
ssp_err_t R_SLCDC_VersionGet ( ssp_version_t  *const p_version )
```

10.46.12.1 Brief description

Retrieve the API version number.

10.46.12.2 Detailed description

Table 1630:

Name	Description
SSP_SUCCESS	Successful return.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.46.13 Extensions

10.46.13.1 slcdc_instance_ctrl_t

slcdc_instance_ctrl_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. SLCDC control block. DO NOT INITIALIZE. Initialization occurs when open is called

Variables

- slcdc_display_state_t state

  Status of SLCD module.

- slcdc_cfg_t info

  SLCDC config info.

- void const * p_context

  Pointer to the higher level device context.

- R_LCD_Type * p_reg

  Pointer to register base address.

10.47 SSI

Driver for the Serial Sound Interface (SSI).

10.47.1 Summary

Extends I2S Interface.

10.47.2 Functions

- R_SSI_Open
- R_SSI_Stop
- R_SSI_Close
- R_SSI_Write
- R_SSI_Read
- R_SSI_WriteRead
- R_SSI_Mute
- R_SSI_InfoGet
- R_SSI_VersionGet

10.47.3 Defines

- #define SSI_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SSI_CODE_VERSION_MINOR

  Initial value:`(5U)`

10.47.4 R_SSI_Open

```
ssp_err_t R_SSI_Open ( i2s_ctrl_t  *const p_api_ctrl ,  i2s_cfg_t  const
*const p_cfg )
```

10.47.4.1 Brief description

Opens the SSI. Implements open.

10.47.4.2 Detailed description

This function calculates the clock divisor based on the input audio clock frequency and the requested sampling frequency. It sets this clock divisor and the configurations specified in i2s_cfg_t. It also opens the timer and transfer interfaces if they are provided.

Table 1631:Return values

Name	Description
SSP_SUCCESS	Ready for I2S communication.
SSP_ERR_ASSERTION	The pointer to p_ctrl or p_cfg is null.
SSP_ERR_IN_USE	The requested channel has already been opened or hardware has been locked.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet
- eventInfoGet
- open
- open

10.47.4.3 Function steps

- Configure dependent timer and transfer drivers.
- Configure interrupts.
- Configure the audio clock.
- Mark driver as open by initializing it to "SSI" in its ASCII equivalent.

10.47.5 R_SSI_Stop

```
ssp_err_t R_SSI_Stop ( i2s_ctrl_t  *const p_api_ctrl ,  i2s_dir_t  const dir )
```

10.47.5.1 Brief description

Stops SSI. Implements stop.

10.47.5.2 Detailed description

This function disables the transfer if the transfer interface is used, or sends a stop signal to stop writing data in the ISR if interrupt driven mode is used.

Table 1632:Return values

Name	Description
SSP_SUCCESS	I2S communication stop request issued.
SSP_ERR_ASSERTION	The pointer to p_ctrl null.
SSP_ERR_NOT_OPEN	The channel is not opened.

See Common Error Codes or lower level drivers for other possible return codes.

10.47.5.3 Function steps

- Stop will occur when the peripheral is idle.

10.47.6 R_SSI_Close

```
ssp_err_t R_SSI_Close ( i2s_ctrl_t  *const p_api_ctrl )
```

10.47.6.1 Brief description

Closes SSI. Implements close.

10.47.6.2 Detailed description

This function powers down the SSI and closes the lower level timer and transfer drivers if they are used.

Table 1633:Return values

Name	Description
SSP_SUCCESS	Device closed successfully.
SSP_ERR_ASSERTION	The pointer to p_ctrl null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.47.6.3 Function steps

- Stop feeding clock to SSI peripheral to deactivate it.
- If a timer instance is provided, close the timer instance.
- If a transfer instance is provided for write, close the transfer instance.

- If a transfer instance is provided for read, close the transfer instance.
- Release HW lock.

10.47.7 R_SSI_Write

```
ssp_err_t R_SSI_Write ( i2s_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_src ,   uint16_t const bytes )
```

10.47.7.1 Brief description

Writes data buffer to SSI. Implements write.

10.47.7.2 Detailed description

This function resets the transfer if the transfer interface is used, or writes the length of data that fits in the FIFO then stores the remaining write buffer in the control block to be written in the ISR.

Write() cannot be called if another write(), read() or writeRead() operation is in progress. Write can be called when the SSI is idle, or after the I2S_EVENT_TX_EMPTY event.

Table 1634:Return values

Name	Description
SSP_SUCCESS	Write initiated successfully.
SSP_ERR_ASSERTION	The pointer to p_ctrl or p_src was null, or bytes requested was 0.
SSP_ERR_IN_USE	Another transfer is in progress, data was not written.
SSP_ERR_NOT_OPEN	The channel is not opened.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- reset
- start

10.47.7.3 Function steps

- Make sure transmission is enabled.
- If a transfer instance is provided for write, reset the transfer.
- Otherwise, write to the FIFO directly.

10.47.8 R_SSI_Read

```
ssp_err_t R_SSI_Read ( i2s_ctrl_t  *const p_api_ctrl ,   uint8_t *const p_dest ,
  uint16_t const bytes )
```

10.47.8.1 Brief description

Reads data into provided buffer. Implements read.

10.47.8.2 Detailed description

This function resets the transfer if the transfer interface is used, or reads the length of data available in the FIFO then stores the remaining read buffer in the control block to be filled in the ISR.

Read() cannot be called if another write(), read() or writeRead() operation is in progress. Read can be called when the SSI is idle, or after the I2S_EVENT_RX_FULL event.

Table 1635:Return values

Name	Description
SSP_SUCCESS	Read initiated successfully.
SSP_ERR_ASSERTION	The pointer to p_ctrl or p_dest was null, or bytes requested was 0.
SSP_ERR_NOT_OPEN	The channel is not opened.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- reset
- start

10.47.8.3 Function steps

- Make sure reception is enabled.
- If a transfer instance is provided for read, reset the transfer.
- Otherwise, read from the FIFO directly.

10.47.9 R_SSI_WriteRead

```
ssp_err_t R_SSI_WriteRead ( i2s_ctrl_t  *const p_api_ctrl ,   uint8_t const
*const p_src ,   uint8_t *const p_dest ,   uint16_t const bytes )
```

10.47.9.1 Brief description

Writes from source buffer and reads data into destination buffer. Implements writeRead.

10.47.9.2 Detailed description

This function calls R_SSI_Write and R_SSI_Read.

writeRead() cannot be called if another write(), read() or writeRead() operation is in progress. writeRead() can be called when the SSI is idle, or after the I2S_EVENT_RX_FULL event.

Table 1636:Return values

Name	Description
SSP_SUCCESS	Write and read initiated successfully.
SSP_ERR_ASSERTION	The pointer to p_ctrl, p_src, or p_dest was null, or bytes requested was 0.
SSP_ERR_NOT_OPEN	The channel is not opened.

See Common Error Codes or lower level drivers for other possible return codes.

10.47.9.3 Function steps

- Call read, then write.

10.47.10 R_SSI_Mute

```
ssp_err_t R_SSI_Mute ( i2s_ctrl_t  *const p_api_ctrl ,  i2s_mute_t
const mute_enable )
```

10.47.10.1 Brief description

Mutes SSI. Implements mute.

10.47.10.2 Detailed description

Data is still written while mute is enabled, but the transmit line outputs zeros.

Table 1637:Return values

Name	Description
SSP_SUCCESS	Transmission is muted.

Table 1637:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	The pointer to p_ctrl was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.47.11 R_SSI_InfoGet

```
ssp_err_t R_SSI_InfoGet ( i2s_ctrl_t  *const p_api_ctrl ,  i2s_info_t
*const p_info )
```

10.47.11.1 Brief description

Get I2S information and store it in provided pointer p_info. Implements infoGet.

10.47.11.2 Detailed description

Table 1638:Return values

Name	Description
SSP_SUCCESS	Information stored successfully.
SSP_ERR_ASSERTION	The p_ctrl or p_info parameter was null.
SSP_ERR_NOT_OPEN	The channel is not opened.

10.47.12 R_SSI_VersionGet

```
ssp_err_t R_SSI_VersionGet ( ssp_version_t  *const p_version )
```

10.47.12.1 Brief description

Sets driver version based on compile time macros.

10.47.12.2 Detailed description

Table 1639:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.47.13 API Data

10.47.13.1 ssi_audio_clock_t

`ssi_audio_clock_t`

Enumerated values

Name	Description
SSI_AUDIO_CLOCK_EXTERNAL	Audio clock source is the AUDIO_CLK input pin.
SSI_AUDIO_CLOCK_GTIOC1A	Audio clock source is internal connection to GPT channel 1 output.

10.47.14 Extensions

10.47.14.1 ssi_instance_ctrl_t

ssi_instance_ctrl_t

Detailed description

Channel instance control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- void(* p_callback)(*p_args)

  Callback provided when an I2S ISR occurs. NULL indicates no CPU interrupt.
- void const * p_context

  Placeholder for user data. Passed to the user callback in i2s_callback_args_t.
- R_SSI0_Type * p_reg

  Pointer to SSI register base address.

- timer_instance_t const * p_timer

  Timer used to generate audio clock.
- transfer_instance_t const * p_transfer_tx

  Transfer used for hardware acceleration during write.
- transfer_instance_t const * p_transfer_rx

  Transfer used for hardware acceleration during read.
- uint32_t const * p_tx_src

  Source buffer pointer used to fill hardware FIFO from transmit ISR.
- uint32_t tx_src_bytes

  Size of source buffer used to fill hardware FIFO from transmit ISR.
- uint32_t * p_rx_dest

  Destination buffer pointer used to fill from hardware FIFO in receive ISR.
- uint32_t rx_dest_bytes

  Size of destination buffer used to fill from hardware FIFO in receive ISR.
- uint32_t sampling_freq_hz

  Sampling frequency in Hertz.
- uint8_t channel

  Channel number.
- uint8_t fifo_access_bytes

  Byte access to FIFO.
- uint8_t stop_requested_tx

  Stops I2S after transmit is complete.
- uint8_t stop_requested_rx

  Stops I2S after receive is complete.
- uint8_t tx_in_progress

  True if a transmit transfer is in progress.
- uint8_t tx_in_use

  True if a transmission is in progress.
- uint8_t rx_in_use

  True if a reception is in progress.
- uint8_t zeros_written

  True if zeros have been transmitted.
- IRQn_Type txi_irq

  Transmit IRQ number.

- IRQn_Type rxi_irq

  Receive IRQ number.
- IRQn_Type int_irq

  Idle/Error IRQ number.
- uint32_t open

  Whether or not this control block is initialized.

10.47.14.2 i2s_on_ssi_cfg_t

i2s_on_ssi_cfg_t

Detailed description

SSI configuration extension. This extension is optional.

Variables

- ssi_audio_clock_t audio_clock

  Audio clock source, default is SSI_AUDIO_CLOCK_EXTERNAL.

10.48 WDT

Driver for the Watchdog Timer (WDT).

10.48.1 Summary

This module supports the Watchdog Timer (WDT). It implements the WDT Interface. The WDT HAL APIs provide the ability to configure the operation of the WDT (when used in register start mode), refresh the watchdog, read the timer value and read and clear status flags.

10.48.2 Functions

- R_WDT_Open
- R_WDT_CfgGet
- R_WDT_TimeoutGet
- R_WDT_Refresh
- R_WDT_StatusGet
- R_WDT_StatusClear
- R_WDT_CounterGet
- R_WDT_VersionGet

10.48.3 Defines

- #define WDT_CODE_VERSION_MAJOR

  Initial value:`(1U)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

- #define WDT_CODE_VERSION_MINOR

  Initial value:`(4U)`

10.48.4 R_WDT_Open

```
ssp_err_t R_WDT_Open ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_cfg_t  const
*const p_cfg )
```

10.48.4.1 Brief description

Configure the WDT in register start mode. In auto-start_mode the NMI callback can be registered. Implements open.

10.48.4.2 Detailed description

This function should only be called once as WDT configuration registers can only be written to once so subsequent calls will have no effect.

Table 1640:Return values

Name	Description
SSP_SUCCESS	WDT successfully configured.
SSP_ERR_ASSERTION	Null Pointer(s).
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid.
SSP_ERR_INVALID_MODE	An attempt to open the WDT in register-start mode when the OFS0 register is configured for auto-start mode. Or to open the WDT in auto-start mode when the OSF0 is configured for register start mode.

See Common Error Codes or functions called by this function for other possible return codes. This function calls:

- productFeatureGet

NOTE: This function is reentrant. In auto-start mode the only valid configuration option is for registering the callback for the NMI ISR if NMI output has been selected.

10.48.4.3 Function steps

- g_wdt_version is accessed by the ASSERT macro only and so compiler toolchain can issue a warning that it is not accessed. The code below eliminates this warning and also ensures this data structure is not optimised away.
- Eliminate toolchain warning when NMI output is not being used.
- Check the expected start mode matches the OSF0 configuration.
- Lock the IWDT Hardware Resource
- Initialize global pointer to WDT for NMI callback use.
- Configuration only valid when WDT operating in register-start mode.
- Register-start mode.
- Register callback with BSP NMI ISR.
- Enable the WDT underflow/refresh error interrupt (will generate an NMI).
- Start the timer by performing a refresh.

10.48.5 R_WDT_CfgGet

```
ssp_err_t R_WDT_CfgGet ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_cfg_t
*const p_cfg )
```

10.48.5.1 Brief description

Read the configuration of the WDT in both register-start and auto-start modes. Implements cfgGet.

10.48.5.2 Detailed description

Table 1641:Return values

Name	Description
SSP_SUCCESS	WDT successfully configured.
SSP_ERR_ASSERTION	Null Pointer.

NOTE: This function is reentrant.

10.48.5.3 Function steps

- Register-start mode.
- Get timeout value from WDTCR register.

10.48.6 R_WDT_TimeoutGet

```
ssp_err_t R_WDT_TimeoutGet ( wdt_ctrl_t  *const p_api_ctrl ,
 wdt_timeout_values_t  *const p_timeout )
```

10.48.6.1 Brief description

Read timeout information for the watchdog timer. Implements timeoutGet.

10.48.6.2 Detailed description

Table 1642:Return values

Name	Description
SSP_SUCCESS	WDT successfully refreshed.
SSP_ERR_ASSERTION	Null Pointer.
SSP_ERR_ABORTED	Invalid clock divider for this watchdog

NOTE: This function is reentrant. This function must not be called before calling R_WDT_Open.

10.48.7 R_WDT_Refresh

```
ssp_err_t R_WDT_Refresh ( wdt_ctrl_t  *const p_api_ctrl )
```

10.48.7.1 Brief description

Refresh the watchdog timer. Implements refresh.

10.48.7.2 Detailed description

In addition to refreshing the watchdog counter this function can be used, in register start mode to start the counter.

Table 1643:Return values

Name	Description
SSP_SUCCESS	WDT successfully refreshed.

NOTE: This function is reentrant. This function only returns SSP_SUCCESS. If the refresh fails due to being performed outside of the permitted refresh period the device will either reset or trigger an NMI ISR to run. This function must not be called before calling R_WDT_Open.

10.48.8 R_WDT_StatusGet

```
ssp_err_t R_WDT_StatusGet ( wdt_ctrl_t  *const p_api_ctrl ,  wdt_status_t
*const p_status )
```

10.48.8.1 Brief description

Read the WDT status flags. Implements statusGet.

10.48.8.2 Detailed description

Indicates both status and error conditions.

Table 1644:Return values

Name	Description
SSP_SUCCESS	WDT status successfully read.
SSP_ERR_ASSERTION	Null pointer as a parameter.

NOTE: This function is reentrant. When the WDT is configured to output a reset on underflow or refresh error reading the status and error flags serves no purpose as they will always indicate that no underflow has occurred and there is no refresh error. Reading the status and error flags is only valid when interrupt request output is enabled.

10.48.9 R_WDT_StatusClear

```
ssp_err_t R_WDT_StatusClear ( wdt_ctrl_t  *const p_api_ctrl ,
 wdt_status_t const  status )
```

10.48.9.1 Brief description

Clear the WDT status and error flags. Implements statusClear.

10.48.9.2 Detailed description

Table 1645:Return values

Name	Description
SSP_SUCCESS	WDT flag(s) successfully cleared.
SSP_ERR_ASSERTION	Null pointer as a parameter.

NOTE: This function is reentrant.

10.48.9.3 Function steps

- Write zero to clear flags.

10.48.10 R_WDT_CounterGet

```
ssp_err_t R_WDT_CounterGet ( wdt_ctrl_t  *const p_api_ctrl ,   uint32_t
*const p_count )
```

10.48.10.1 Brief description

Read the current count value of the WDT. Implements counterGet.

10.48.10.2 Detailed description

Table 1646:Return values

Name	Description
SSP_SUCCESS	WDT current count successfully read.
SSP_ERR_ASSERTION	Null pointer passed as a parameter.

NOTE: This function is reentrant.

10.48.11 R_WDT_VersionGet

```
ssp_err_t R_WDT_VersionGet ( ssp_version_t  *const p_data )
```

10.48.11.1 Brief description

Return WDT HAL driver version. Implements versionGet.

10.48.11.2 Detailed description

Table 1647:Return values

Name	Description
SSP_SUCCESS	Version information successfully read.

Table 1647:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	Null pointer passed as a parameter

NOTE: This function is reentrant.

10.48.12 Extensions

10.48.12.1 wdt_instance_ctrl_t

wdt_instance_ctrl_t

Detailed description

WDT control block. DO NOT INITIALIZE. Initialization occurs when open is called.

Variables

- bool wdt_open

  Indicates whether the open() API has been successfully called.
- void const * p_context

  Placeholder for user data. Passed to the user callback in wdt_callback_args_t.
- R_WDT_Type * p_reg

  Pointer to register base address.
- void(* p_callback)(*p_args)

  Callback provided when a WDT NMI ISR occurs.

10.49 SCE Module

Primitive cryptographic functions.

Initializes the SCE module cryptographic operations.

Crypto API interface on SCE

10.49.1 Functions

- R_SCE_Open
- R_SCE_VersionGet
- R_SCE_StatusGet
- R_SCE_Close

10.49.2 Variables

- g_sce_crypto_api
- sceInitializationStatus
- s_sce_version
- crypto_bsp_lock

10.49.3 R_SCE_Open

```
R_SCE_Open ( crypto_ctrl_t  *const p_ctrl ,  crypto_cfg_t  const *const p_cfg )
```

10.49.3.1 Detailed description

SCE Initialization

Table 1648:Parameters

Name	Direction	Description
p_ctrl	inout	control structure for the SCE module
p_cfg	in	configuration structure for the SCE module

an uint32_t integer indicating

- PASS if the initialization is successful
- FAIL if the initialization failed
- RESOURCE_CONFLICT if the SCE hardware resource is busy

10.49.4 R_SCE_VersionGet

```
R_SCE_VersionGet ( ssp_version_t  *const p_version )
```

10.49.4.1 Brief description

Sets driver version based on compile time macros.

10.49.4.2 Detailed description

Table 1649:Parameters

Name	Direction	Description
p_version	out	version info for the SCE implementation

Table 1650:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.49.5 R_SCE_StatusGet

```
R_SCE_StatusGet (  uint32_t * p_status )
```

10.49.5.1 Brief description

This function indicates if the SCE has been initialized or not.

10.49.5.2 Detailed description

SCE Get status of SCE module initialization

Table 1651:Parameters

Name	Direction	Description
p_status	in	pointer to uint32_t. This memory location is updated with the SCE module initialization

Table 1652:Return values

Name	Description
SSP_SUCCESS	Successful close.

Table 1652:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	The parameter p_version is NULL.

10.49.6 R_SCE_Close

```
R_SCE_Close ( crypto_ctrl_t  *const p_ctrl )
```

10.49.6.1 Brief description

Close SCE driver. Does nothing. Always returns success.

10.49.6.2 Detailed description

Table 1653:Parameters

Name	Direction	Description
p_ctrl	in	control structure for SCE block

Table 1654:Return values

Name	Description
SSP_SUCCESS	Successful close.

10.49.7 g_sce_crypto_api

```
crypto_api_t::g_sce_crypto_api
```

10.49.7.1 Initialized as

```
g_sce_crypto_api=
{
    .open           = R_SCE_Open,
    .close          = R_SCE_Close,
    .interfaceGet   = R_SCE_InterfaceGet,
    .statusGet      = R_SCE_StatusGet,
    .versionGet     = R_SCE_VersionGet
}
```

10.49.8 Modules

- SCE AES
- SCE HRK AES
- SCE ARC4
- SCE DSA
- SCE HASH
- SCE_RSA
- SCE_TDES
- SCE_TRNG

10.49.8.1 SCE AES

Primitive cryptographic functions.

AES key generation, encryption and decryption functions for plain text/raw keys

AES encryption and decryption functions

AES 256-bit CBC mode implementation for encryption and decryption functions

AES 256-bit CTR mode implementation for encryption and decryption functions

AES 256-bit ECB mode implementation for encryption and decryption functions

AES 256-bit XTS mode implementation for encryption and decryption functions

Functions

- R_SCE_AES_Open
- R_SCE_AES_VersionGet
- R_SCE_AES_Close
- R_SCE_AES_128CbcEncrypt
- R_SCE_AES_128CbcDecrypt
- R_SCE_AES_128CtrEncrypt
- R_SCE_AES_128EcbEncrypt
- R_SCE_AES_128EcbDecrypt
- R_SCE_AES_128GcmInitialize
- R_SCE_AES_128GcmTagCompute
- r_sce_aes_128gcmtag_compute_and_verify
- R_SCE_AES_128GcmOpen
- R_SCE_AES_128GcmEncrypt
- R_SCE_AES_128GcmGetGcmTag
- R_SCE_AES_128GcmSetGcmTag

- R_SCE_AES_128GcmDecrypt
- R_SCE_AES_128GcmZeroPaddingEncrypt
- R_SCE_AES_128GcmZeroPaddingDecrypt
- R_SCE_AES_128XtsEncrypt
- R_SCE_AES_128XtsDecrypt
- R_SCE_AES_192CbcEncrypt
- R_SCE_AES_192CbcDecrypt
- R_SCE_AES_192CtrEncrypt
- R_SCE_AES_192EcbEncrypt
- R_SCE_AES_192EcbDecrypt
- R_SCE_AES_192GcmInitialize
- R_SCE_AES_192GcmTagCompute
- r_sce_aes_192gcmtag_compute_and_verify
- R_SCE_AES_192GcmOpen
- R_SCE_AES_192GcmEncrypt
- R_SCE_AES_192GcmGetGcmTag
- R_SCE_AES_192GcmSetGcmTag
- R_SCE_AES_192GcmDecrypt
- R_SCE_AES_192GcmZeroPaddingEncrypt
- R_SCE_AES_192GcmZeroPaddingDecrypt
- R_SCE_AES_256CbcEncrypt
- R_SCE_AES_256CbcDecrypt
- R_SCE_AES_256CtrEncrypt
- R_SCE_AES_256EcbEncrypt
- R_SCE_AES_256EcbDecrypt
- R_SCE_AES_256GcmInitialize
- R_SCE_AES_256GcmTagCompute
- r_sce_aes_256gcmtag_compute_and_verify
- R_SCE_AES_256GcmOpen
- R_SCE_AES_256GcmEncrypt
- R_SCE_AES_256GcmGetGcmTag
- R_SCE_AES_256GcmSetGcmTag
- R_SCE_AES_256GcmDecrypt
- R_SCE_AES_256GcmZeroPaddingEncrypt

- R_SCE_AES_256GcmZeroPaddingDecrypt
- R_SCE_AES_256XtsEncrypt
- R_SCE_AES_256XtsDecrypt
- R_SCE_AES_EImpl_CreateKey
- R_SCE_AES_EImpl_EncryptFinal
- R_SCE_AES_EImpl_ZeroPaddingEncrypt
- R_SCE_AES_EImpl_ZeroPaddingDecrypt
- R_SCE_AES_EImpl_SetGcmTag
- R_SCE_AES_EImpl_GetGcmTag
- R_SCE_AES_EImpl_AddAdditionalAuthenticationData

Variables

- s_sce_aes_version
- g_aes128cbc_on_sce
- g_aes128ctr_on_sce
- g_aes128ecb_on_sce
- g_aes128gcm_on_sce
- g_aes128xts_on_sce
- g_aes192cbc_on_sce
- g_aes192ctr_on_sce
- g_aes192ecb_on_sce
- g_aes192gcm_on_sce
- g_aes256cbc_on_sce
- g_aes256ctr_on_sce
- g_aes256ecb_on_sce
- g_aes256gcm_on_sce
- g_aes256xts_on_sce

Defines

- #define SCE_AES_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_AES_CODE_VERSION_MINOR

  Initial value:`(00)`

- #define SIZE_GCMTAG_BITS
  Initial value:(128)
- #define SIZE_GCMTAG_BYTES
  Initial value:((SIZE_GCMTAG_BITS) / 8)
- #define SIZE_GCMTAG_WORDS
  Initial value:((SIZE_GCMTAG_BITS) / 32)
- #define SIZE_AES_BLOCK_BITS
  Initial value:(128)
- #define SIZE_AES_BLOCK_BYTES
  Initial value:((SIZE_AES_BLOCK_BITS) / 8)
- #define SIZE_AES_BLOCK_WORDS
  Initial value:((SIZE_AES_BLOCK_BITS) / 32)
- #define SIZE_AES_128BIT_KEYLEN_BITS
  Initial value:(128)
- #define SIZE_AES_128BIT_KEYLEN_BYTES
  Initial value:((SIZE_AES_128BIT_KEYLEN_BITS) / 8)
- #define SIZE_AES_128BIT_KEYLEN_WORDS
  Initial value:((SIZE_AES_128BIT_KEYLEN_BITS) / 32)
- #define SIZE_AES_192BIT_KEYLEN_BITS
  Initial value:(192)
- #define SIZE_AES_192BIT_KEYLEN_BYTES
  Initial value:((SIZE_AES_192BIT_KEYLEN_BITS) / 8)
- #define SIZE_AES_192BIT_KEYLEN_WORDS
  Initial value:((SIZE_AES_192BIT_KEYLEN_BITS) / 32)
- #define SIZE_AES_256BIT_KEYLEN_BITS
  Initial value:(256)
- #define SIZE_AES_256BIT_KEYLEN_BYTES
  Initial value:((SIZE_AES_256BIT_KEYLEN_BITS) / 8)
- #define SIZE_AES_256BIT_KEYLEN_WORDS
  Initial value:((SIZE_AES_256BIT_KEYLEN_BITS) / 32)

s_sce_aes_version

ssp_version_t::s_sce_aes_version

Detailed description

SCE/AES implementation of AES API. Version data structure used by error logger macro.

Initialized as

```
s_sce_aes_version=
{
    .api_version_major  = AES_API_VERSION_MAJOR,
    .api_version_minor  = AES_API_VERSION_MINOR,
    .code_version_major = SCE_AES_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_AES_CODE_VERSION_MINOR
}
```

g_aes128cbc_on_sce

aes_api_t::g_aes128cbc_on_sce

Detailed description

AES 128-bit CBC mode implementation

Initialized as

```
g_aes128cbc_on_sce=
{
    .open                              = R_SCE_AES_Open,
    .encrypt                           = R_SCE_AES_128CbcEncrypt,
    .decrypt                           = R_SCE_AES_128CbcDecrypt,
    .close                             = R_SCE_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                         = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                         = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128ctr_on_sce

aes_api_t::g_aes128ctr_on_sce

Detailed description

AES 128-bit CTR mode implementation

Initialized as

```
g_aes128ctr_on_sce=
{
    .open                              = R_SCE_AES_Open,
    .encrypt                           = R_SCE_AES_128CtrEncrypt,
    .decrypt                           = R_SCE_AES_128CtrEncrypt,
    .close                             = R_SCE_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
```

```
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                       = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128ecb_on_sce

aes_api_t::g_aes128ecb_on_sce

Detailed description

AES 128-bit ECB mode implementation

Initialized as

```
g_aes128ecb_on_sce=
{
    .open                            = R_SCE_AES_Open,
    .encrypt                         = R_SCE_AES_128EcbEncrypt,
    .decrypt                         = R_SCE_AES_128EcbDecrypt,
    .close                           = R_SCE_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .createKey                       = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt              = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                       = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128gcm_on_sce

aes_api_t::g_aes128gcm_on_sce

Detailed description

AES 128-bit GCM mode implementation

Initialized as

```
g_aes128gcm_on_sce=
{
    .open                            = R_SCE_AES_128GcmOpen,
    .encrypt                         = R_SCE_AES_128GcmEncrypt,
    .decrypt                         = R_SCE_AES_128GcmDecrypt,
    .getGcmTag                       = R_SCE_AES_128GcmGetGcmTag,
    .setGcmTag                       = R_SCE_AES_128GcmSetGcmTag,
    .close                           = R_SCE_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .zeroPaddingEncrypt              = R_SCE_AES_128GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_128GcmZeroPaddingDecrypt,
    .createKey                       = R_SCE_AES_EImpl_CreateKey,
```

```
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData
}
```

g_aes128xts_on_sce

`aes_api_t::g_aes128xts_on_sce`

Detailed description

AES 128-bit CCM mode implementation

Initialized as

```
g_aes128xts_on_sce=
{
    .open                            = R_SCE_AES_Open,
    .encrypt                         = R_SCE_AES_128XtsEncrypt,
    .decrypt                         = R_SCE_AES_128XtsDecrypt,
    .close                           = R_SCE_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .createKey                       = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt              = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                       = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192cbc_on_sce

`aes_api_t::g_aes192cbc_on_sce`

Detailed description

AES 192-bit CBC mode implementation

Initialized as

```
g_aes192cbc_on_sce=
{
    .open                            = R_SCE_AES_Open,
    .encrypt                         = R_SCE_AES_192CbcEncrypt,
    .decrypt                         = R_SCE_AES_192CbcDecrypt,
    .close                           = R_SCE_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .createKey                       = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt              = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
```

```
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192ctr_on_sce

aes_api_t::g_aes192ctr_on_sce

Detailed description

AES 192-bit CTR mode implementation

Initialized as

```
g_aes192ctr_on_sce=
{
    .open                               = R_SCE_AES_Open,
    .encrypt                            = R_SCE_AES_192CtrEncrypt,
    .decrypt                            = R_SCE_AES_192CtrEncrypt,
    .close                              = R_SCE_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .createKey                          = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                 = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                          = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192ecb_on_sce

aes_api_t::g_aes192ecb_on_sce

Detailed description

AES 192-bit ECB mode implementation

Initialized as

```
g_aes192ecb_on_sce=
{
    .open                               = R_SCE_AES_Open,
    .encrypt                            = R_SCE_AES_192EcbEncrypt,
    .decrypt                            = R_SCE_AES_192EcbDecrypt,
    .close                              = R_SCE_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .createKey                          = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                 = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                          = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192gcm_on_sce

aes_api_t::g_aes192gcm_on_sce

Detailed description

AES 192-bit GCM mode implementation

Initialized as

```
g_aes192gcm_on_sce=
{
    .open                               = R_SCE_AES_192GcmOpen,
    .encrypt                            = R_SCE_AES_192GcmEncrypt,
    .decrypt                            = R_SCE_AES_192GcmDecrypt,
    .getGcmTag                          = R_SCE_AES_192GcmGetGcmTag,
    .setGcmTag                          = R_SCE_AES_192GcmSetGcmTag,
    .close                              = R_SCE_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .zeroPaddingEncrypt                 = R_SCE_AES_192GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_192GcmZeroPaddingDecrypt,
    .createKey                          = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
}
```

g_aes256cbc_on_sce

aes_api_t::g_aes256cbc_on_sce

Detailed description

AES 256-bit CBC mode implementation

Initialized as

```
g_aes256cbc_on_sce=
{
    .open                               = R_SCE_AES_Open,
    .encrypt                            = R_SCE_AES_256CbcEncrypt,
    .decrypt                            = R_SCE_AES_256CbcDecrypt,
    .close                              = R_SCE_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .createKey                          = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                 = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                          = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256ctr_on_sce

aes_api_t::g_aes256ctr_on_sce

Detailed description

AES 256-bit CTR mode implementation

Initialized as

```
g_aes256ctr_on_sce=
{
    .open                             = R_SCE_AES_Open,
    .encrypt                          = R_SCE_AES_256CtrEncrypt,
    .decrypt                          = R_SCE_AES_256CtrEncrypt,
    .close                            = R_SCE_AES_Close,
    .versionGet                       = R_SCE_AES_VersionGet,
    .createKey                        = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                     = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt               = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt               = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                        = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                        = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256ecb_on_sce

aes_api_t::g_aes256ecb_on_sce

Detailed description

AES 256-bit ECB mode implementation

Initialized as

```
g_aes256ecb_on_sce=
{
    .open                             = R_SCE_AES_Open,
    .encrypt                          = R_SCE_AES_256EcbEncrypt,
    .decrypt                          = R_SCE_AES_256EcbDecrypt,
    .close                            = R_SCE_AES_Close,
    .versionGet                       = R_SCE_AES_VersionGet,
    .createKey                        = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                     = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt               = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt               = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                        = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                        = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256gcm_on_sce

aes_api_t::g_aes256gcm_on_sce

Detailed description

AES 256-bit GCM mode implementation

Initialized as

```
g_aes256gcm_on_sce=
{
    .open                              = R_SCE_AES_256GcmOpen,
    .encrypt                           = R_SCE_AES_256GcmEncrypt,
    .decrypt                           = R_SCE_AES_256GcmDecrypt,
    .getGcmTag                         = R_SCE_AES_256GcmGetGcmTag,
    .setGcmTag                         = R_SCE_AES_256GcmSetGcmTag,
    .close                             = R_SCE_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .zeroPaddingEncrypt                = R_SCE_AES_256GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_256GcmZeroPaddingDecrypt,
    .createKey                         = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
}
```

g_aes256xts_on_sce

aes_api_t::g_aes256xts_on_sce

Detailed description

AES 256-bit XTS mode implementation

Initialized as

```
g_aes256xts_on_sce=
{
    .open                              = R_SCE_AES_Open,
    .encrypt                           = R_SCE_AES_256XtsEncrypt,
    .decrypt                           = R_SCE_AES_256XtsDecrypt,
    .close                             = R_SCE_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_AES_EImpl_CreateKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                         = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                         = R_SCE_AES_EImpl_GetGcmTag
}
```

R_SCE_AES_Open

```
R_SCE_AES_Open ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const *const p_cfg )
```

Detailed description

AES Initialization

Table 1655:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_AES_VersionGet

```
R_SCE_AES_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1656:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_AES_Close

```
R_SCE_AES_Close ( aes_ctrl_t  *const p_ctrl )
```

Detailed description

AES Close function

Table 1657:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_AES_128CbcEncrypt

```
R_SCE_AES_128CbcEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 128-bit CBC mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1658:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_128CbcDecrypt

```
 R_SCE_AES_128CbcDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1659:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_128CtrEncrypt

```
R_SCE_AES_128CtrEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 128-bit CTR mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1660:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_128EcbEncrypt

```
R_SCE_AES_128EcbEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 128-bit ECB mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1661:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_128EcbDecrypt

```
R_SCE_AES_128EcbDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
 uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 128-bit ECB mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1662:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_128GcmInitialize

```
R_SCE_AES_128GcmInitialize ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv )
```

R_SCE_AES_128GcmTagCompute

```
R_SCE_AES_128GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

r_sce_aes_128gcmtag_compute_and_verify

```
r_sce_aes_128gcmtag_compute_and_verify ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_AES_128GcmOpen

```
R_SCE_AES_128GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_AES_128GcmEncrypt

```
R_SCE_AES_128GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 128-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1663:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_128GcmGetGcmTag

```
R_SCE_AES_128GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
  uint32_t * p_dest )
```

R_SCE_AES_128GcmSetGcmTag

```
R_SCE_AES_128GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
  uint32_t * p_source )
```

R_SCE_AES_128GcmDecrypt

```
R_SCE_AES_128GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1664:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end

Table 1664:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_128GcmZeroPaddingEncrypt

```
 R_SCE_AES_128GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Encrypt num_bytes bytes of input data from buffer p_source using the 128-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1665:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least (num_bytes/16+1)*16 bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: this function is not thread safe.

R_SCE_AES_128GcmZeroPaddingDecrypt

```
 R_SCE_AES_128GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Decrypt num_bytes bytes of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1666:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_bytes bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_128XtsEncrypt

```
R_SCE_AES_128XtsEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 128-bit XTS mode implementation for encrypt interface API Encrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1667:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_128XtsDecrypt

```
R_SCE_AES_128XtsDecrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 128-bit XTS mode implementation for decrypt interface API Decrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1668:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_192CbcEncrypt

```
R_SCE_AES_192CbcEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 192-bit CBC mode implementation for encrypt interface API Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1669:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 24 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_192CbcDecrypt

```
R_SCE_AES_192CbcDecrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 192-bit CBC mode implementation for decrypt interface API Decrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1670:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 24 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_192CtrEncrypt

```
R_SCE_AES_192CtrEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 192-bit CTR mode implementation for encrypt and decrypt interface APIs Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1671:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1671:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 24 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_192EcbEncrypt

```
R_SCE_AES_192EcbEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 192-bit ECB mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1672:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 24 bytes of AES key data and

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_192EcbDecrypt

```
R_SCE_AES_192EcbDecrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
 uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 192-bit ECB mode implementation for decrypt interface API Decrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1673:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 24 bytes of AES key data and

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_192GcmInitialize

```
 R_SCE_AES_192GcmInitialize ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv )
```

R_SCE_AES_192GcmTagCompute

```
 R_SCE_AES_192GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

r_sce_aes_192gcmtag_compute_and_verify

```
 r_sce_aes_192gcmtag_compute_and_verify ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_AES_192GcmOpen

```
 R_SCE_AES_192GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_AES_192GcmEncrypt

```
 R_SCE_AES_192GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1674:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1674:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_192GcmGetGcmTag

```
R_SCE_AES_192GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_dest )
```

R_SCE_AES_192GcmSetGcmTag

```
R_SCE_AES_192GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_source )
```

R_SCE_AES_192GcmDecrypt

```
R_SCE_AES_192GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
 uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1675:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_192GcmZeroPaddingEncrypt

```
 R_SCE_AES_192GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Encrypt num_bytes bytes of input data from buffer p_source using the 128-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1676:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_bytes bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data

NOTE: this function is not thread safe.

R_SCE_AES_192GcmZeroPaddingDecrypt

```
 R_SCE_AES_192GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Decrypt num_bytes bytes of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1677:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_bytes bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_256CbcEncrypt

```
R_SCE_AES_256CbcEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
  uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 256-bit CBC mode implementation for encrypt interface API Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1678:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_AES_256CbcDecrypt

```
R_SCE_AES_256CbcDecrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
  uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 256-bit CBC mode implementation for decrypt interface API Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1679:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_AES_256CtrEncrypt

```
R_SCE_AES_256CtrEncrypt ( aes_ctrl_t *const p_ctrl , const uint32_t * p_key ,
  uint32_t * p_iv , uint32_t num_words , uint32_t * p_source , uint32_t
* p_dest )
```

Detailed description

AES 256-bit CTR mode implementation for encrypt and decrypt interface APIs Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1680:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_AES_256EcbEncrypt

```
 R_SCE_AES_256EcbEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 256-bit ECB mode implementation for encrypt interface API Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1681:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_256EcbDecrypt

```
 R_SCE_AES_256EcbDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1682:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: The contents of p_iv buffer are ignored in ECB chaining mode. NULL value is acceptable.

R_SCE_AES_256GcmInitialize

```
 R_SCE_AES_256GcmInitialize ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv )
```

R_SCE_AES_256GcmTagCompute

```
 R_SCE_AES_256GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

r_sce_aes_256gcmtag_compute_and_verify

```
 r_sce_aes_256gcmtag_compute_and_verify ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_AES_256GcmOpen

```
 R_SCE_AES_256GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_AES_256GcmEncrypt

```
 R_SCE_AES_256GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 256-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1683:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_256GcmGetGcmTag

```
R_SCE_AES_256GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_dest )
```

R_SCE_AES_256GcmSetGcmTag

```
R_SCE_AES_256GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_source )
```

R_SCE_AES_256GcmDecrypt

```
R_SCE_AES_256GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
 uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1684:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_256GcmZeroPaddingEncrypt

```
R_SCE_AES_256GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Encrypt num_bytes bytes of input data from buffer p_source using the 128-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1685:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_bytes bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: this function is not thread safe.

R_SCE_AES_256GcmZeroPaddingDecrypt

```
 R_SCE_AES_256GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Decrypt num_bytes bytes of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_bytes bytes of data.

Table 1686:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_bytes bytes of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

NOTE: For description of memcpy, memcmp and memset functions refer to C Standard library <string.h>

R_SCE_AES_256XtsEncrypt

```
 R_SCE_AES_256XtsEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 256-bit XTS mode implementation for encrypt interface API Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1687:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_256XtsDecrypt

```
 R_SCE_AES_256XtsDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t * p_key ,
  uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,   uint32_t
* p_dest )
```

Detailed description

AES 256-bit XTS mode implementation for decrypt interface API Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1688:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_AES_EImpl_CreateKey

```
R_SCE_AES_EImpl_CreateKey ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_key )
```

Detailed description

SCE/AES implementation of AES API.

Table 1689:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_EncryptFinal

```
R_SCE_AES_EImpl_EncryptFinal ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t input_num_words ,   uint32_t
* p_source ,   uint32_t output_num_words ,   uint32_t * p_dest )
```

Detailed description

Table 1690:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_ZeroPaddingEncrypt

```
R_SCE_AES_EImpl_ZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Table 1691:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_ZeroPaddingDecrypt

```
R_SCE_AES_EImpl_ZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Detailed description

Table 1692:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_SetGcmTag

```
R_SCE_AES_EImpl_SetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_source )
```

Detailed description

Table 1693:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_GetGcmTag

```
R_SCE_AES_EImpl_GetGcmTag ( aes_ctrl_t  *const p_ctrl ,   uint32_t num_words ,
 uint32_t * p_dest )
```

Detailed description

Table 1694:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_AES_EImpl_AddAdditionalAuthenticationData

```
R_SCE_AES_EImpl_AddAdditionalAuthenticationData ( aes_ctrl_t  *const p_ctrl ,
 const uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t
* p_source )
```

Detailed description

Table 1695:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

10.49.8.2 SCE HRK AES

Primitive cryptographic functions.

SCE HRK Encrypted Key AES Cipher implementation functions

Functions

- R_SCE_HRK_AES_Open
- R_SCE_HRK_AES_Close
- R_SCE_HRK_AES_VersionGet
- R_SCE_HRK_AES_128CreateEncryptedKey
- R_SCE_HRK_AES_192CreateEncryptedKey
- R_SCE_HRK_AES_256CreateEncryptedKey
- R_SCE_HRK_AES_128CbcEncrypt
- R_SCE_HRK_AES_128CbcDecrypt
- R_SCE_HRK_AES_128CtrEncrypt
- R_SCE_HRK_AES_128EcbEncrypt
- R_SCE_HRK_AES_128EcbDecrypt
- R_SCE_HRK_AES_128GcmOpen
- R_SCE_HRK_AES_128GcmEncrypt
- R_SCE_HRK_AES_128GcmZeroPaddingEncrypt
- R_SCE_HRK_AES_128GcmGetGcmTag
- R_SCE_HRK_AES_128GcmSetGcmTag
- R_SCE_HRK_AES_128GcmDecrypt
- R_SCE_HRK_AES_128GcmZeroPaddingDecrypt
- R_SCE_HRK_AES_128GcmTagCompute
- R_SCE_HRK_AES_128XtsCreateEncryptedKey
- R_SCE_HRK_AES_128XtsEncrypt
- R_SCE_HRK_AES_128XtsDecrypt
- R_SCE_HRK_AES_192CbcEncrypt
- R_SCE_HRK_AES_192CbcDecrypt
- R_SCE_HRK_AES_192CtrEncrypt
- R_SCE_HRK_AES_192EcbEncrypt
- R_SCE_HRK_AES_192EcbDecrypt
- R_SCE_HRK_AES_192GcmInitialize
- R_SCE_HRK_AES_192GcmTagCompute
- r_sce_hrk_aes_192gcmtag_compute_and_verify

- R_SCE_HRK_AES_192GcmOpen
- R_SCE_HRK_AES_192GcmEncrypt
- R_SCE_HRK_AES_192GcmGetGcmTag
- R_SCE_HRK_AES_192GcmSetGcmTag
- R_SCE_HRK_AES_192GcmDecrypt
- R_SCE_HRK_AES_192GcmZeroPaddingEncrypt
- R_SCE_HRK_AES_192GcmZeroPaddingDecrypt
- R_SCE_HRK_AES_256CbcEncrypt
- R_SCE_HRK_AES_256CbcDecrypt
- R_SCE_HRK_AES_256CtrEncrypt
- R_SCE_HRK_AES_256EcbEncrypt
- R_SCE_HRK_AES_256EcbDecrypt
- R_SCE_HRK_AES_256GcmInitialize
- R_SCE_HRK_AES_256GcmTagCompute
- r_sce_hrk_aes_256gcmtag_compute_and_verify
- R_SCE_HRK_AES_256GcmOpen
- R_SCE_HRK_AES_256GcmEncrypt
- R_SCE_HRK_AES_256GcmGetGcmTag
- R_SCE_HRK_AES_256GcmSetGcmTag
- R_SCE_HRK_AES_256GcmDecrypt
- R_SCE_HRK_AES_256GcmZeroPaddingEncrypt
- R_SCE_HRK_AES_256GcmZeroPaddingDecrypt
- R_SCE_HRK_AES_256XtsCreateEncryptedKey
- R_SCE_HRK_AES_256XtsEncrypt
- R_SCE_HRK_AES_256XtsDecrypt

Variables

- s_sce_hrk_aes_version
- g_aes128cbc_on_sceHrk
- g_aes128ctr_on_sceHrk
- g_aes128ecb_on_sceHrk
- g_aes128gcm_on_sceHrk
- g_aes128xts_on_sceHrk
- g_aes192cbc_on_sceHrk
- g_aes192ctr_on_sceHrk

- g_aes192ecb_on_sceHrk
- g_aes192gcm_on_sceHrk
- g_aes256cbc_on_sceHrk
- g_aes256ctr_on_sceHrk
- g_aes256ecb_on_sceHrk
- g_aes256gcm_on_sceHrk
- g_aes256xts_on_sceHrk

s_sce_hrk_aes_version

`ssp_version_t::s_sce_hrk_aes_version`

Detailed description

Version data structure used by error logger macro.

Initialized as

```
s_sce_hrk_aes_version=
{
    .api_version_major  = AES_API_VERSION_MAJOR,
    .api_version_minor  = AES_API_VERSION_MINOR,
    .code_version_major = SCE_AES_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_AES_CODE_VERSION_MINOR
}
```

g_aes128cbc_on_sceHrk

`aes_api_t::g_aes128cbc_on_sceHrk`

Initialized as

```
g_aes128cbc_on_sceHrk=
{
    .open                              = R_SCE_HRK_AES_Open,
    .encrypt                           = R_SCE_HRK_AES_128CbcEncrypt,
    .decrypt                           = R_SCE_HRK_AES_128CbcDecrypt,
    .close                             = R_SCE_HRK_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_HRK_AES_128CreateEncryptedKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                         = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                         = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128ctr_on_sceHrk

`aes_api_t::g_aes128ctr_on_sceHrk`

Initialized as

```
g_aes128ctr_on_sceHrk=
{
    .open                           = R_SCE_HRK_AES_Open,
    .encrypt                        = R_SCE_HRK_AES_128CtrEncrypt,
    .decrypt                        = R_SCE_HRK_AES_128CtrEncrypt,
    .close                          = R_SCE_HRK_AES_Close,
    .versionGet                     = R_SCE_AES_VersionGet,
    .createKey                      = R_SCE_HRK_AES_128CreateEncryptedKey,
    .encryptFinal                   = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt             = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt             = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                      = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                      = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128ecb_on_sceHrk

`aes_api_t::g_aes128ecb_on_sceHrk`

Detailed description

HRK Supported global structure definitions

Initialized as

```
g_aes128ecb_on_sceHrk=
{
    .open                           = R_SCE_HRK_AES_Open,
    .encrypt                        = R_SCE_HRK_AES_128EcbEncrypt,
    .decrypt                        = R_SCE_HRK_AES_128EcbDecrypt,
    .close                          = R_SCE_HRK_AES_Close,
    .versionGet                     = R_SCE_AES_VersionGet,
    .createKey                      = R_SCE_HRK_AES_128CreateEncryptedKey,
    .encryptFinal                   = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt             = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt             = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                      = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                      = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes128gcm_on_sceHrk

`aes_api_t::g_aes128gcm_on_sceHrk`

Initialized as

```
g_aes128gcm_on_sceHrk=
{
    .open       = R_SCE_HRK_AES_128GcmOpen,
    .createKey  = R_SCE_HRK_AES_128CreateEncryptedKey,
    .encrypt    = R_SCE_HRK_AES_128GcmEncrypt,
```

```
    .decrypt    = R_SCE_HRK_AES_128GcmDecrypt,
    .getGcmTag  = R_SCE_HRK_AES_128GcmGetGcmTag,
    .setGcmTag  = R_SCE_HRK_AES_128GcmSetGcmTag,
    .close      = R_SCE_HRK_AES_Close,
    .versionGet = R_SCE_HRK_AES_VersionGet,
    .zeroPaddingEncrypt = R_SCE_HRK_AES_128GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt = R_SCE_HRK_AES_128GcmZeroPaddingDecrypt
}
```

g_aes128xts_on_sceHrk

aes_api_t::g_aes128xts_on_sceHrk

Initialized as

```
g_aes128xts_on_sceHrk=
{
    .open                           = R_SCE_HRK_AES_Open,
    .encrypt                        = R_SCE_HRK_AES_128XtsEncrypt,
    .decrypt                        = R_SCE_HRK_AES_128XtsDecrypt,
    .close                          = R_SCE_HRK_AES_Close,
    .versionGet                     = R_SCE_AES_VersionGet,
    .createKey                      = R_SCE_HRK_AES_128XtsCreateEncryptedKey,
    .encryptFinal                   = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt             = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt             = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                      = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                      = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192cbc_on_sceHrk

aes_api_t::g_aes192cbc_on_sceHrk

Initialized as

```
g_aes192cbc_on_sceHrk=
{
    .open                           = R_SCE_HRK_AES_Open,
    .encrypt                        = R_SCE_HRK_AES_192CbcEncrypt,
    .decrypt                        = R_SCE_HRK_AES_192CbcDecrypt,
    .close                          = R_SCE_HRK_AES_Close,
    .versionGet                     = R_SCE_AES_VersionGet,
    .createKey                      = R_SCE_HRK_AES_192CreateEncryptedKey,
    .encryptFinal                   = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt             = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt             = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                      = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                      = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192ctr_on_sceHrk

aes_api_t::g_aes192ctr_on_sceHrk

Initialized as

```
g_aes192ctr_on_sceHrk=
{
    .open                              = R_SCE_HRK_AES_Open,
    .encrypt                           = R_SCE_HRK_AES_192CtrEncrypt,
    .decrypt                           = R_SCE_HRK_AES_192CtrEncrypt,
    .close                             = R_SCE_HRK_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_HRK_AES_192CreateEncryptedKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                         = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                         = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192ecb_on_sceHrk

aes_api_t::g_aes192ecb_on_sceHrk

Initialized as

```
g_aes192ecb_on_sceHrk=
{
    .open                              = R_SCE_HRK_AES_Open,
    .encrypt                           = R_SCE_HRK_AES_192EcbEncrypt,
    .decrypt                           = R_SCE_HRK_AES_192EcbDecrypt,
    .close                             = R_SCE_HRK_AES_Close,
    .versionGet                        = R_SCE_AES_VersionGet,
    .createKey                         = R_SCE_HRK_AES_192CreateEncryptedKey,
    .encryptFinal                      = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                         = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                         = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes192gcm_on_sceHrk

aes_api_t::g_aes192gcm_on_sceHrk

Initialized as

```
g_aes192gcm_on_sceHrk=
{
    .open       = R_SCE_HRK_AES_192GcmOpen,
    .createKey  = R_SCE_HRK_AES_192CreateEncryptedKey,
```

```
    .encrypt    = R_SCE_HRK_AES_192GcmEncrypt,
    .decrypt    = R_SCE_HRK_AES_192GcmDecrypt,
    .getGcmTag  = R_SCE_HRK_AES_192GcmGetGcmTag,
    .setGcmTag  = R_SCE_HRK_AES_192GcmSetGcmTag,
    .close      = R_SCE_HRK_AES_Close,
    .versionGet = R_SCE_HRK_AES_VersionGet,
    .zeroPaddingEncrypt = R_SCE_HRK_AES_192GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt = R_SCE_HRK_AES_192GcmZeroPaddingDecrypt
}
```

g_aes256cbc_on_sceHrk

aes_api_t::g_aes256cbc_on_sceHrk

Initialized as

```
g_aes256cbc_on_sceHrk=
{
    .open                            = R_SCE_HRK_AES_Open,
    .encrypt                         = R_SCE_HRK_AES_256CbcEncrypt,
    .decrypt                         = R_SCE_HRK_AES_256CbcDecrypt,
    .close                           = R_SCE_HRK_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .createKey                       = R_SCE_HRK_AES_256CreateEncryptedKey,
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt              = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                       = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256ctr_on_sceHrk

aes_api_t::g_aes256ctr_on_sceHrk

Initialized as

```
g_aes256ctr_on_sceHrk=
{
    .open                            = R_SCE_HRK_AES_Open,
    .encrypt                         = R_SCE_HRK_AES_256CtrEncrypt,
    .decrypt                         = R_SCE_HRK_AES_256CtrEncrypt,
    .close                           = R_SCE_HRK_AES_Close,
    .versionGet                      = R_SCE_AES_VersionGet,
    .createKey                       = R_SCE_HRK_AES_256CreateEncryptedKey,
    .encryptFinal                    = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt              = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt              = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                       = R_SCE_AES_EImpl_SetGcmTag,
```

```
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256ecb_on_sceHrk

aes_api_t::g_aes256ecb_on_sceHrk

Initialized as

```
g_aes256ecb_on_sceHrk=
{
    .open                               = R_SCE_HRK_AES_Open,
    .encrypt                            = R_SCE_HRK_AES_256EcbEncrypt,
    .decrypt                            = R_SCE_HRK_AES_256EcbDecrypt,
    .close                              = R_SCE_HRK_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .createKey                          = R_SCE_HRK_AES_256CreateEncryptedKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                 = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                          = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

g_aes256gcm_on_sceHrk

aes_api_t::g_aes256gcm_on_sceHrk

Initialized as

```
g_aes256gcm_on_sceHrk=
{
    .open       = R_SCE_HRK_AES_256GcmOpen,
    .createKey  = R_SCE_HRK_AES_256CreateEncryptedKey,
    .encrypt    = R_SCE_HRK_AES_256GcmEncrypt,
    .decrypt    = R_SCE_HRK_AES_256GcmDecrypt,
    .getGcmTag  = R_SCE_HRK_AES_256GcmGetGcmTag,
    .setGcmTag  = R_SCE_HRK_AES_256GcmSetGcmTag,
    .close      = R_SCE_HRK_AES_Close,
    .versionGet = R_SCE_HRK_AES_VersionGet,
    .zeroPaddingEncrypt = R_SCE_HRK_AES_256GcmZeroPaddingEncrypt,
    .zeroPaddingDecrypt = R_SCE_HRK_AES_256GcmZeroPaddingDecrypt
}
```

g_aes256xts_on_sceHrk

aes_api_t::g_aes256xts_on_sceHrk

Initialized as

```
g_aes256xts_on_sceHrk=
{
    .open                               = R_SCE_HRK_AES_Open,
    .encrypt                            = R_SCE_HRK_AES_256XtsEncrypt,
```

```
    .decrypt                            = R_SCE_HRK_AES_256XtsDecrypt,
    .close                              = R_SCE_HRK_AES_Close,
    .versionGet                         = R_SCE_AES_VersionGet,
    .createKey                          = R_SCE_HRK_AES_256XtsCreateEncryptedKey,
    .encryptFinal                       = R_SCE_AES_EImpl_EncryptFinal,
    .zeroPaddingEncrypt                 = R_SCE_AES_EImpl_ZeroPaddingEncrypt,
    .zeroPaddingDecrypt                 = R_SCE_AES_EImpl_ZeroPaddingDecrypt,
    .addAdditionalAuthenticationData =
R_SCE_AES_EImpl_AddAdditionalAuthenticationData,
    .setGcmTag                          = R_SCE_AES_EImpl_SetGcmTag,
    .getGcmTag                          = R_SCE_AES_EImpl_GetGcmTag
}
```

R_SCE_HRK_AES_Open

```
 R_SCE_HRK_AES_Open ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

Detailed description

AES Initialization

Table 1696:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_HRK_AES_Close

```
 R_SCE_HRK_AES_Close ( aes_ctrl_t  *const p_ctrl )
```

Detailed description

AES Initialization

Table 1697:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_HRK_AES_VersionGet

```
R_SCE_HRK_AES_VersionGet ( ssp_version_t  *const p_version )
```

R_SCE_HRK_AES_128CreateEncryptedKey

```
R_SCE_HRK_AES_128CreateEncryptedKey ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_key )
```

Brief description

Encrypted Wrapped key creation for AES 128-bit applies for CBC, ECB, CTR, GCM chaining mode implementations. Create a num_words words of wrapped key for AES XTS chaining mode. The result will be written to the output buffer p_key. The p_key array is assumed to have space for at least num_words words of data.

Detailed description

Table 1698:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_INVALID_SIZE	Insufficient buffer size to accommodate created key

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

R_SCE_HRK_AES_192CreateEncryptedKey

```
R_SCE_HRK_AES_192CreateEncryptedKey ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_key )
```

Brief description

Encrypted Wrapped key creation for AES 192-bit applies for CBC, ECB, CTR, GCM chaining mode implementations. Create a num_words words of wrapped key for AES XTS chaining mode. The result will be written to the output buffer p_key. The p_key array is assumed to have space for at least num_words words of data.

Detailed description

Table 1699:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_INVALID_SIZE	Insufficient buffer size to accommodate created key

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

R_SCE_HRK_AES_256CreateEncryptedKey

```
R_SCE_HRK_AES_256CreateEncryptedKey ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_key )
```

Brief description

Encrypted Wrapped key creation for AES 256-bit applies for CBC, ECB, CTR, GCM chaining mode implementations. Create a num_words words of wrapped key for AES XTS chaining mode. The result will be written to the output buffer p_key. The p_key array is assumed to have space for at least num_words words of data.

Detailed description

Table 1700:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_INVALID_SIZE	Insufficient buffer size to accommodate created key

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

R_SCE_HRK_AES_128CbcEncrypt

```
 R_SCE_HRK_AES_128CbcEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit CBC mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1701:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1701:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

R_SCE_HRK_AES_128CbcDecrypt

```
 R_SCE_HRK_AES_128CbcDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit CBC mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using the wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1702:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

R_SCE_HRK_AES_128CtrEncrypt

```
 R_SCE_HRK_AES_128CtrEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit CTR mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using a wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1703:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

R_SCE_HRK_AES_128EcbEncrypt

```
 R_SCE_HRK_AES_128EcbEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit ECB mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1704:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1704:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_128EcbDecrypt

```
 R_SCE_HRK_AES_128EcbDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit ECB mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using the wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1705:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_128GcmOpen

```
 R_SCE_HRK_AES_128GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_HRK_AES_128GcmEncrypt

```
 R_SCE_HRK_AES_128GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 128-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1706:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_128GcmZeroPaddingEncrypt

```
 R_SCE_HRK_AES_128GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_128GcmGetGcmTag

```
 R_SCE_HRK_AES_128GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,
  uint32_t num_words ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_128GcmSetGcmTag

```
 R_SCE_HRK_AES_128GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,
  uint32_t num_words ,   uint32_t * p_source )
```

R_SCE_HRK_AES_128GcmDecrypt

```
 R_SCE_HRK_AES_128GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1707:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_128GcmZeroPaddingDecrypt

```
 R_SCE_HRK_AES_128GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_128GcmTagCompute

```
 R_SCE_HRK_AES_128GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_128XtsCreateEncryptedKey

```
 R_SCE_HRK_AES_128XtsCreateEncryptedKey ( aes_ctrl_t  *const p_ctrl ,
  uint32_t num_words ,   uint32_t * p_key )
```

Detailed description

Encrypted Wrapped key creation for AES 1286-bit XTS mode implementation

Create a num_words words of wrapped key for AES XTS chaining mode. The result will be written to the output buffer p_key. The p_key array is assumed to have space for at least num_words words of data.

Table 1708:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1708:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_INVALID_SIZE	Insufficient buffer size to accommodate created key

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4 and must be less than 128M

R_SCE_HRK_AES_128XtsEncrypt

```
 R_SCE_HRK_AES_128XtsEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit XTS mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1709:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Invalid Key
SSP_ERR_INVALID_SIZE	Invalid size

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4 and must be less than 128M

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128XtsCreateEncryptedKey.

Function steps

- Verify the upper-limit of num_words
- HW_SCE_AES_128XtsEncryptUsingEncryptedKey takes a pointer (InData_Len) whose value is in number of bits

R_SCE_HRK_AES_128XtsDecrypt

```
 R_SCE_HRK_AES_128XtsDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted Key AES 128-bit XTS mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using a wrapped 128-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1710:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Invalid Key
SSP_ERR_INVALID_SIZE	Invalid size

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4 and must be less than 128M

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 128-bit AES key generated using the R_SCE_HRK_AES_128CreateEncrypt-edKey.

Function steps

- Verify the upper-limit of num_words
- HW_SCE_AES_128XtsDecryptUsingEncryptedKey takes a pointer (InData_Len) whose value is in number of bits

R_SCE_HRK_AES_192CbcEncrypt

```
 R_SCE_HRK_AES_192CbcEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 192-bit CBC mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1711:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 192-bit AES key generated using R_SCE_HRK_AES_192CreateEncrypt-edKey.

R_SCE_HRK_AES_192CbcDecrypt

```
 R_SCE_HRK_AES_192CbcDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 192-bit CBC mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using a wrapped 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1712:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 192-bit AES key generated using the R_SCE_HRK_AES_192CreateEncrypt-edKey.

R_SCE_HRK_AES_192CtrEncrypt

```
 R_SCE_HRK_AES_192CtrEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 192-bit CTR mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using a wrapped 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1713:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 192-bit AES key generated using the R_SCE_HRK_AES_192CreateEncrypt-edKey.

R_SCE_HRK_AES_192EcbEncrypt

```
 R_SCE_HRK_AES_192EcbEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 192-bit ECB mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1714:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 192-bit AES key generated using the R_SCE_HRK_AES_192CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_192EcbDecrypt

```
 R_SCE_HRK_AES_192EcbDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 192-bit ECB mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using a wrapped 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1715:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 192-bit AES key generated using the R_SCE_HRK_AES_192CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_192GcmInitialize

```
 R_SCE_HRK_AES_192GcmInitialize ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv )
```

R_SCE_HRK_AES_192GcmTagCompute

```
 R_SCE_HRK_AES_192GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

r_sce_hrk_aes_192gcmtag_compute_and_verify

```
 r_sce_hrk_aes_192gcmtag_compute_and_verify ( aes_ctrl_t  *const p_ctrl ,
  const uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_192GcmOpen

```
 R_SCE_HRK_AES_192GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_HRK_AES_192GcmEncrypt

```
 R_SCE_HRK_AES_192GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 192-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1716:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_192GcmGetGcmTag

```
R_SCE_HRK_AES_192GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_192GcmSetGcmTag

```
R_SCE_HRK_AES_192GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_source )
```

R_SCE_HRK_AES_192GcmDecrypt

```
R_SCE_HRK_AES_192GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
 uint32_t * p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 192-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1717:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_192GcmZeroPaddingEncrypt

```
R_SCE_HRK_AES_192GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_192GcmZeroPaddingDecrypt

```
R_SCE_HRK_AES_192GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_256CbcEncrypt

```
R_SCE_HRK_AES_256CbcEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
 uint32_t * p_dest )
```

Detailed description

Encrypted AES 256-bit CBC mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1718:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256CreateEncrypt-edKey.

R_SCE_HRK_AES_256CbcDecrypt

```
 R_SCE_HRK_AES_256CbcDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted AES 256-bit ECB mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using a wrapped 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1719:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256CreateEncrypt-edKey.

R_SCE_HRK_AES_256CtrEncrypt

```
 R_SCE_HRK_AES_256CtrEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 256-bit CTR mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using a wrapped 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1720:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256CreateEncrypt-edKey.

R_SCE_HRK_AES_256EcbEncrypt

```
 R_SCE_HRK_AES_256EcbEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 256-bit ECB mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using a wrapped 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1721:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_256EcbDecrypt

```
 R_SCE_HRK_AES_256EcbDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 256-bit ECB mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1722:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256CreateEncrypt-edKey.

NOTE: The contents of p_iv buffer are ignored and can be NULL in ECB chaining mode.

R_SCE_HRK_AES_256GcmInitialize

```
 R_SCE_HRK_AES_256GcmInitialize ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv )
```

R_SCE_HRK_AES_256GcmTagCompute

```
 R_SCE_HRK_AES_256GcmTagCompute ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

r_sce_hrk_aes_256gcmtag_compute_and_verify

```
 r_sce_hrk_aes_256gcmtag_compute_and_verify ( aes_ctrl_t  *const p_ctrl ,
  const uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_256GcmOpen

```
 R_SCE_HRK_AES_256GcmOpen ( aes_ctrl_t  *const p_ctrl ,  aes_cfg_t  const
*const p_cfg )
```

R_SCE_HRK_AES_256GcmEncrypt

```
 R_SCE_HRK_AES_256GcmEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypt num_words words of input data from buffer p_source using the 256-bit AES p_key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1723:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_256GcmGetGcmTag

```
R_SCE_HRK_AES_256GcmGetGcmTag ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_256GcmSetGcmTag

```
R_SCE_HRK_AES_256GcmSetGcmTag ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_source )
```

R_SCE_HRK_AES_256GcmDecrypt

```
R_SCE_HRK_AES_256GcmDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
 uint32_t * p_dest )
```

Detailed description

Decrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1724:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Plain text key is passed

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 16 bytes of AES key data and

NOTE: the p_iv buffer must have at least 16 bytes of random data.

R_SCE_HRK_AES_256GcmZeroPaddingEncrypt

```
R_SCE_HRK_AES_256GcmZeroPaddingEncrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_256GcmZeroPaddingDecrypt

```
R_SCE_HRK_AES_256GcmZeroPaddingDecrypt ( aes_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   uint32_t * p_iv ,   uint32_t num_bytes ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_AES_256XtsCreateEncryptedKey

```
R_SCE_HRK_AES_256XtsCreateEncryptedKey ( aes_ctrl_t  *const p_ctrl ,
 uint32_t num_words ,   uint32_t * p_key )
```

Detailed description

Encrypted Wrapped key creation for AES 256-bit XTS mode implementation

Create a num_words words of wrapped key for AES XTS chaining mode. The result will be written to the output buffer p_key. The p_key array is assumed to have space for at least num_words words of data.

Table 1725:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_INVALID_SIZE	Insufficient buffer size to accommodate created key

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4

R_SCE_HRK_AES_256XtsEncrypt

```
 R_SCE_HRK_AES_256XtsEncrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 256-bit XTS mode implementation for encrypt interface API

Encrypt num_words words of input data from buffer p_source using the 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1726:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Invalid Key
SSP_ERR_INVALID_SIZE	Invalid size

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4 and must be less than 128M

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256XtsCreateEncryptedKey.

Function steps

- Verify the upper-limit of num_words
- HW_SCE_AES_256XtsEncryptUsingEncryptedKey takes a pointer (InData_Len) whose value is in number of bits

R_SCE_HRK_AES_256XtsDecrypt

```
 R_SCE_HRK_AES_256XtsDecrypt ( aes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Detailed description

Encrypted key AES 256-bit XTS mode implementation for decrypt interface API

Decrypt num_words words of input data from buffer p_source using a wrapped 256-bit AES key from buffer p_key and initialization vector from buffer p_iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1727:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Invalid Key
SSP_ERR_INVALID_SIZE	Invalid size

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: num_words must be a multiple of 4 and must be less than 128M

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key is a wrapped/encrypted 256-bit AES key generated using the R_SCE_HRK_AES_256XtsCreateEncryptedKey.

Function steps

- Verify the upper-limit of num_words
- HW_SCE_AES_256XtsDecryptUsingEncryptedKey takes a pointer (InData_Len) whose value is in number of bits

10.49.8.3 SCE ARC4

Primitive cryptographic functions.

ARC4 encryption and decryption functions

Functions

- R_SCE_ARC4_Open
- R_SCE_ARC4_Close
- R_SCE_ARC4_Process
- R_SCE_ARC4_VersionGet
- R_SCE_ARC4_KeySet

Defines

- #define SCE_ARC4_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_ARC4_CODE_VERSION_MINOR

  Initial value:`(01)`

R_SCE_ARC4_Open

```
R_SCE_ARC4_Open (  arc4_ctrl_t *const p_ctrl ,  arc4_cfg_t const *const p_cfg )
```

Detailed description

ARC4 Initialization

Table 1728:Return values

Name	Description
SF_CRYPTO_SUCCESS	successful.
SSP_ERR_CRYPTO_SCE_ALREADY_OPEN	ARC4 module is already in open state and is usable for the given p_ctrl parameter.
SSP_ERR_ASSERTION	An input parameter is invalid.

R_SCE_ARC4_Close

```
R_SCE_ARC4_Close (  arc4_ctrl_t *const p_ctrl )
```

Detailed description

ARC4 Finalization

Table 1729:Return values

Name	Description
SF_CRYPTO_SUCCESS	successful

R_SCE_ARC4_Process

```
R_SCE_ARC4_Process (  arc4_ctrl_t *const p_ctrl ,   uint32_t num_bytes ,
 uint8_t * p_source ,   uint8_t * p_dest )
```

Brief description

ARC4 Encrypt or decrypt source data and output result to destination buffer.

Detailed description

Table 1730:Return values

Name	Description
SSP_ERR_CRYPTO_INVALID_STATE	Invalid state, ensure that key data is configured either using the open() or using the keyset() function.
SSP_ERR_CRYPTO_INVALID_SIZE	invalid num_bytes passed. Should be a multiple of 16.
SF_CRYPTO_SUCCESS	successful.
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	An internal error has occurred.
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	Hardware is busy, unable to encrypt at this time.

Encrypt or decrypt input data using the previously configured key data

R_SCE_ARC4_VersionGet

```
R_SCE_ARC4_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1731:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_ARC4_KeySet

```
 R_SCE_ARC4_KeySet (  arc4_ctrl_t *const p_ctrl ,   uint32_t length ,   uint8_t
const * p_key )
```

Brief description

Sets user provided for use with subsequent encryptions.

Detailed description

Table 1732:Return values

Name	Description
SF_CRYPTO_SUCCESS	KeySet successful
SSP_ERR_NOT_OPEN	Module not opened.
SSP_ERR_ASSERTION	One of input parameter is NULL.

10.49.8.4 SCE DSA

Primitive cryptographic functions.

Primitive cryptographic functions (L=2048,N=256) DSA.

DSA signature generation and verification functions

DSA signature generation and verification functions (L=1024,N=160) DSA

DSA signature generation and verification functions (L=2048,N=224) DSA

Functions

- R_SCE_DSA_Open
- R_SCE_DSA_VersionGet
- R_SCE_DSA_Close
- R_SCE_DSA_1024_160SignatureVerify
- R_SCE_DSA_1024_160SignatureGenerate
- R_SCE_DSA_1024_160HashSignatureVerify
- R_SCE_DSA_1024_160HashSignatureGenerate
- R_SCE_DSA_2048_224SignatureVerify
- R_SCE_DSA_2048_224SignatureGenerate
- R_SCE_DSA_2048_224HashSignatureVerify
- R_SCE_DSA_2048_224HashSignatureGenerate
- R_SCE_DSA_2048_256SignatureVerify
- R_SCE_DSA_2048_256SignatureGenerate
- R_SCE_DSA_2048_256HashSignatureVerify
- R_SCE_DSA_2048_256HashSignatureGenerate

Variables

- s_sce_dsa_version
- g_dsa1024_160_on_sce

- g_dsa2048_224_on_sce
- g_dsa2048_256_on_sce

Defines

- #define SCE_DSA_CODE_VERSION_MAJOR
  Initial value:`(01)`
  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_DSA_CODE_VERSION_MINOR
  Initial value:`(00)`
- #define DSA_PARAM_LENGTH_Q
  Initial value:`(160)`
- #define DSA_PARAM_LENGTH_P
  Initial value:`(1024)`
- #define DSA_PARAM_LENGTH_G
  Initial value:`(DSA_PARAM_LENGTH_P)`
- #define DSA_PRIVATE_KEYSIZE
  Initial value:`((DSA_PARAM_LENGTH_Q) / 32)`
- #define DSA_PUBLIC_KEYSIZE
  Initial value:`((DSA_PARAM_LENGTH_P) / 32)`
- #define DSA_DOMAIN_SIZE
  Initial value:`(DSA_PRIVATE_KEYSIZE + 2 * (DSA_PUBLIC_KEYSIZE))`
- #define DSA_SIGNATURE_SIZE
  Initial value:`(2 * (DSA_PRIVATE_KEYSIZE))`

dsa_domain_1024_160_t

dsa_domain_1024_160_t

Detailed description

Variables

- uint32_t q[(160/32)]
  DSA (1024,160) domain parameter Q.
- uint32_t p[(1024/32)]
  DSA (1024,160) domain parameter P.
- uint32_t g[(1024/32)]
  DSA (1024,160) domain parameter G.

dsa_signature_1024_160_t

dsa_signature_1024_160_t

Detailed description

Variables

- uint32_t r[(160/32)]

  DSA (1024,160) signature component R.

- uint32_t s[(160/32)]

  DSA (1024,160) signature component S.

dsa_domain_2048_224_t

dsa_domain_2048_224_t

Detailed description

Variables

- uint32_t q[(224/32)]

  DSA (2048,224) domain parameter Q.

- uint32_t p[(2048/32)]

  DSA (2048,224) domain parameter P.

- uint32_t g[(2048/32)]

  DSA (2048,224) domain parameter G.

dsa_signature_2048_224_t

dsa_signature_2048_224_t

Detailed description

Variables

- uint32_t r[(224/32)]

  DSA (2048,224) signature component R.

- uint32_t s[(224/32)]

  DSA (2048,224) signature component S.

dsa_domain_2048_256_t

dsa_domain_2048_256_t

Detailed description

Variables

- uint32_t q[(256/32)]

  DSA (2048,256) domain parameter Q.

- uint32_t p[(2048/32)]

  DSA (2048,256) domain parameter P.

- uint32_t g[(2048/32)]

  DSA (2048,256) domain parameter G.

dsa_signature_2048_256_t

dsa_signature_2048_256_t

Detailed description

Variables

- uint32_t r[(256/32)]

  DSA (2048,256) signature component R.

- uint32_t s[(256/32)]

  DSA (2048,256) signature component S.

s_sce_dsa_version

`ssp_version_t::s_sce_dsa_version`

Detailed description

Version data structure used by error logger macro.

Initialized as

```
s_sce_dsa_version=
{
    .api_version_major  = DSA_API_VERSION_MAJOR,
    .api_version_minor  = DSA_API_VERSION_MINOR,
    .code_version_major = SCE_DSA_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_DSA_CODE_VERSION_MINOR
}
```

g_dsa1024_160_on_sce

`dsa_api_t::g_dsa1024_160_on_sce`

Detailed description

SCE/DSA implementation of DSA API.

Initialized as

```
g_dsa1024_160_on_sce=
{
    .open       = R_SCE_DSA_Open,
    .sign       = R_SCE_DSA_1024_160SignatureGenerate,
    .verify     = R_SCE_DSA_1024_160SignatureVerify,
    .close      = R_SCE_DSA_Close,
    .versionGet = R_SCE_DSA_VersionGet,
    .hashSign   = R_SCE_DSA_1024_160HashSignatureGenerate,
    .hashVerify = R_SCE_DSA_1024_160HashSignatureVerify
}
```

g_dsa2048_224_on_sce

`dsa_api_t::g_dsa2048_224_on_sce`

Detailed description

SCE/DSA implementation of DSA API.

Initialized as

```
g_dsa2048_224_on_sce=
{
    .open       = R_SCE_DSA_Open,
    .sign       = R_SCE_DSA_2048_224SignatureGenerate,
    .verify     = R_SCE_DSA_2048_224SignatureVerify,
    .close      = R_SCE_DSA_Close,
    .versionGet = R_SCE_DSA_VersionGet,
    .hashSign   = R_SCE_DSA_2048_224HashSignatureGenerate,
    .hashVerify = R_SCE_DSA_2048_224HashSignatureVerify,
}
```

g_dsa2048_256_on_sce

`dsa_api_t::g_dsa2048_256_on_sce`

Detailed description

SCE/DSA implementation of DSA API.

Initialized as

```
g_dsa2048_256_on_sce=
{
    .open       = R_SCE_DSA_Open,
    .sign       = R_SCE_DSA_2048_256SignatureGenerate,
    .verify     = R_SCE_DSA_2048_256SignatureVerify,
    .close      = R_SCE_DSA_Close,
    .versionGet = R_SCE_DSA_VersionGet,
    .hashSign   = R_SCE_DSA_2048_256HashSignatureGenerate,
    .hashVerify = R_SCE_DSA_2048_256HashSignatureVerify
}
```

R_SCE_DSA_Open

```
R_SCE_DSA_Open ( dsa_ctrl_t  *const p_ctrl ,  dsa_cfg_t  const *const p_cfg )
```

Detailed description

DSA Initialization

Table 1733:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL.
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy

Table 1733:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_DSA_VersionGet

```
R_SCE_DSA_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1734:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_DSA_Close

```
R_SCE_DSA_Close ( dsa_ctrl_t  *const p_ctrl )
```

Detailed description

Close DSA driver

Table 1735:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_DSA_1024_160SignatureVerify

```
R_SCE_DSA_1024_160SignatureVerify (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_signature ,   uint32_t
* p_paddedHash )
```

Brief description

Signature verification using (1024-bit,160-bit) DSA public key.

Detailed description

Verify DSA signature data from buffer p_signature of length 2*imaxcnt words using (L=1024,N=160) DSA public key from buffer p_key. The buffer p_paddedHash indicates the message buffer from which the DSA signature should have been generated.

Table 1736:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	Incorrect signature
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the function R_SCE_Open API

NOTE: The p_key buffer must contain 32 words of DSA public key data

NOTE: The p_domain buffer must contain (20+128+128) bytes of data in the format (Q || P || G) where Q is 5 words, P is 32 words and G is 32 words.

R_SCE_DSA_1024_160SignatureGenerate

```
 R_SCE_DSA_1024_160SignatureGenerate (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (1024-bit,160-bit) DSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the (L=1024,N=160)-bit DSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast 2*imaxcnt words of data.

Table 1737:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the function R_SCE_Open API

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The p_key buffer must contain a valid DSA private key data and p_domain should contain DSA domain parameters in the order (Q || P || G) where Q is 5 words, P is 32 words and G is 32 words.

R_SCE_DSA_1024_160HashSignatureVerify

```
 R_SCE_DSA_1024_160HashSignatureVerify ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_signature ,   uint32_t * p_paddedHash )
```

Brief description

Signature verification using (1024-bit,160-bit) DSA public key.

Detailed description

Verify DSA signature data from buffer p_signature of length 2*imaxcnt words using (L=1024,N=160) DSA public key from buffer p_key. The buffer p_paddedHash indicates the message buffer from which the DSA signature should have been generated.

Table 1738:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	Incorrect signature
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the function R_SCE_Open API

NOTE: The p_key buffer must contain 32 words of DSA public key data

NOTE: The p_domain buffer must contain (20+128+128) bytes of data in the format (Q || P || G) where Q is 5 words, P is 32 words and G is 32 words.

R_SCE_DSA_1024_160HashSignatureGenerate

```
 R_SCE_DSA_1024_160HashSignatureGenerate ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (1024-bit,160-bit) DSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the (L=1024,N=160)-bit DSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast 2*imaxcnt words of data.

Table 1739:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the function R_SCE_Open API

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The p_key buffer must contain a valid DSA private key data and p_domain should contain DSA domain parameters in the order (Q || P || G) where Q is 5 words, P is 32 words and G is 32 words.

R_SCE_DSA_2048_224SignatureVerify

```
 R_SCE_DSA_2048_224SignatureVerify (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_signature ,   uint32_t
* p_paddedHash )
```

Brief description

Signature verification using (2048-bit,224-bit) DSA public key.

Detailed description

Verify DSA signature data from buffer p_signature of length num_words using (L=2048,N=224) DSA public key. from buffer p_key. The buffer p_paddedHash indicates the message buffer from which the DSA signature should have been generated.

Table 1740:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open API

NOTE: The p_key buffer must contain 128 bytes of DSA public key data in the format

NOTE: The p_domain buffer must contain (28+256+256) bytes of data in the format (Q || P || G).Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

R_SCE_DSA_2048_224SignatureGenerate

```
 R_SCE_DSA_2048_224SignatureGenerate (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (2048-bit,224-bit) DSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the (L=2048,N=224)-bit DSA private key from buffer key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1741:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open API

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The key buffer must contain 7 words of DSA private key data

NOTE: p_domain must contain valid DSA domain parameters

R_SCE_DSA_2048_224HashSignatureVerify

```
 R_SCE_DSA_2048_224HashSignatureVerify ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_signature ,   uint32_t * p_paddedHash )
```

Brief description

Signature verification using (2048-bit,224-bit) DSA public key.

Detailed description

Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

R_SCE_DSA_2048_224HashSignatureGenerate

```
 R_SCE_DSA_2048_224HashSignatureGenerate ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (2048-bit,224-bit) DSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the (L=2048,N=224)-bit DSA private key from buffer key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1742:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open API

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The key buffer must contain 7 words of DSA private key data

NOTE: p_domain must contain valid DSA domain parameters

R_SCE_DSA_2048_256SignatureVerify

```
 R_SCE_DSA_2048_256SignatureVerify (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_signature ,   uint32_t
* p_paddedHash )
```

Brief description

Signature verification using (2048-bit,256-bit) DSA public key.

Detailed description

Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

Verify DSA signature data from buffer p_signature of length num_words using (L=2048,N=256) DSA public key. from buffer p_key. The buffer p_paddedHash indicates the message buffer from which the DSA signature should have been generated.

Table 1743:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.

Table 1743:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: The p_key buffer must contain 256 bytes of DSA public key data

NOTE: The p_domain buffer must contain (32+256+256) bytes of data in the format (Q || P || G).

R_SCE_DSA_2048_256SignatureGenerate

```
 R_SCE_DSA_2048_256SignatureGenerate (  const uint32_t * p_key ,   const uint32_t
* p_domain ,   uint32_t imaxcnt ,   uint32_t * p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (2048-bit,256-bit) DSA private key.

Detailed description

Generate signature for the buffer p_paddedHash with the given DSA private key p_key for the domain parameters p_domain. The result will be written to the buffer p_dest

Sign imaxcnt words of input data from buffer p_source using the (L=2048,N=256)-bit DSA private key from buffer key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1744:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The key buffer must contain 8 words of DSA private key data

R_SCE_DSA_2048_256HashSignatureVerify

```
 R_SCE_DSA_2048_256HashSignatureVerify ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_signature ,   uint32_t * p_paddedHash )
```

Brief description

Signature verification using (2048-bit,256-bit) DSA public key.

Detailed description

Verify DSA signature from buffer p_signature using the given DSA public key p_key with domain parameters from p_domain for the input message hash p_paddedHash

Verify DSA signature data from buffer p_signature of length num_words using (L=2048,N=256) DSA public key. from buffer p_key. The buffer p_paddedHash indicates the message buffer from which the DSA signature should have been generated.

Table 1745:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: The p_key buffer must contain 256 bytes of DSA public key data

NOTE: The p_domain buffer must contain (32+256+256) bytes of data in the format (Q || P || G).

R_SCE_DSA_2048_256HashSignatureGenerate

```
 R_SCE_DSA_2048_256HashSignatureGenerate ( dsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using (2048-bit,256-bit) DSA private key.

Detailed description

Generate signature for the buffer p_paddedHash with the given DSA private key p_key for the domain parameters p_domain. The result will be written to the buffer p_dest

Sign imaxcnt words of input data from buffer p_source using the (L=2048,N=256)-bit DSA private key from buffer key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1746:Return values

Name	Description
SF_CRYPTO_SUCCESS	Call successful
SSP_ERR_ASSERTION	An input parameter is invalid.

Table 1746:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least 2*imaxcnt words of data.

NOTE: The key buffer must contain 8 words of DSA private key data

10.49.8.5 SCE HASH

Primitive cryptographic functions.

HASH functions

message hasing/digest functions

Functions

- R_SCE_HASH_VersionGet
- R_SCE_SHA1_Open
- R_SCE_SHA1_Close
- R_SCE_SHA1_UpdateHash
- R_SCE_SHA1_HashUpdate
- R_SCE_SHA256_Open
- R_SCE_SHA256_Close
- R_SCE_SHA256_UpdateHash
- R_SCE_SHA256_HashUpdate

Variables

- s_sce_hash_version
- g_sha1_hash_on_sce
- g_sha256_hash_on_sce

Defines

- #define SCE_HASH_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_HASH_CODE_VERSION_MINOR

  Initial value:`(00)`

s_sce_hash_version

`ssp_version_t::s_sce_hash_version`

Detailed description

SCE/AES implementation of HASH API. Version data structure used by error logger macro.

Initialized as

```
s_sce_hash_version=
{
    .api_version_major  = HASH_API_VERSION_MAJOR,
    .api_version_minor  = HASH_API_VERSION_MINOR,
    .code_version_major = SCE_HASH_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_HASH_CODE_VERSION_MINOR
}
```

g_sha1_hash_on_sce

`hash_api_t::g_sha1_hash_on_sce`

Detailed description

SHA1 implementation of HASH API.

Initialized as

```
g_sha1_hash_on_sce=
{
    .open       = R_SCE_SHA1_Open,
    .updateHash = R_SCE_SHA1_UpdateHash,
    .close      = R_SCE_SHA1_Close,
    .versionGet = R_SCE_HASH_VersionGet,
    .hashUpdate = R_SCE_SHA1_HashUpdate
}
```

g_sha256_hash_on_sce

`hash_api_t::g_sha256_hash_on_sce`

Detailed description

SHA256 implementation of HASH API.

Initialized as

```
g_sha256_hash_on_sce=
{
    .open       = R_SCE_SHA256_Open,
    .updateHash = R_SCE_SHA256_UpdateHash,
    .close      = R_SCE_SHA256_Close,
    .versionGet = R_SCE_HASH_VersionGet,
    .hashUpdate = R_SCE_SHA256_HashUpdate
}
```

R_SCE_HASH_VersionGet

```
R_SCE_HASH_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

SCE HASH Get Version

Table 1747:Parameters

Name	Direction	Description
p_version	in	pointer to ssp_version_t structure where the version info will be stored

Table 1748:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_SHA1_Open

```
R_SCE_SHA1_Open ( hash_ctrl_t  *const p_ctrl ,  hash_cfg_t  const *const p_cfg )
```

Detailed description

SCE SHA1 open function using the SCE block

SHA1 HASH Initialization

Table 1749:Return values

Name	Description
SF_CRYPTO_SUCCESS	open successful

R_SCE_SHA1_Close

```
R_SCE_SHA1_Close ( hash_ctrl_t  *const p_ctrl )
```

Detailed description

HASH Initialization

Table 1750:Return values

Name	Description
SF_CRYPTO_SUCCESS	close successful SCE SHA1 Close function

Table 1751:Parameters

Name	Direction	Description
p_ctrl	in	control structure for SCE SHA1

R_SCE_SHA1_UpdateHash

```
R_SCE_SHA1_UpdateHash (  const uint32_t * p_msg ,   uint32_t num_words ,
 uint32_t * p_digest )
```

Detailed description

Compute the SHA1 message digest for the given input message buffer p_msg of length num_words words. The length of the message buffer needs to be a multiple of 64 bytes. Generally the content of the message buffer are the padded value as given by Message||stopbit||zero padding||Message length.

The initial hash value given in buffer p_digest will be used and this buffer will be updated with the computed SHA1 message digest value.

Table 1752:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_SHA1_HashUpdate

```
R_SCE_SHA1_HashUpdate ( hash_ctrl_t  *const p_ctrl ,   const uint32_t * p_msg ,
 uint32_t num_words ,   uint32_t * p_digest )
```

Detailed description

Compute the SHA1 message digest for the given input message buffer p_msg of length num_words words. The length of the message buffer needs to be a multiple of 64 bytes. Generally the content of the message buffer are the padded value as given by Message||stopbit||zero padding||Message length.

The initial hash value given in buffer p_digest will be used and this buffer will be updated with the computed SHA1 message digest value.

Table 1753:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1753:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_SHA256_Open

```
R_SCE_SHA256_Open ( hash_ctrl_t  *const p_ctrl ,  hash_cfg_t  const
*const p_cfg )
```

Detailed description

SCE SHA256 HASH Initialization

Table 1754:Return values

Name	Description
SF_CRYPTO_SUCCESS	successful completion

R_SCE_SHA256_Close

```
R_SCE_SHA256_Close ( hash_ctrl_t  *const p_ctrl )
```

Detailed description

SCE SHA256 Close function

Table 1755:Return values

Name	Description
SF_CRYPTO_SUCCESS	random numbergeneration successful

R_SCE_SHA256_UpdateHash

```
R_SCE_SHA256_UpdateHash (  const uint32_t * p_msg ,   uint32_t num_words ,
 uint32_t * p_digest )
```

Brief description

Update hash value using the given input message from buffer p_source of num_words words, write the result to p_digest

Detailed description

Compute the SHA256 message digest for the given input message buffer p_msg of length num_words words. The length of the message buffer needs to be a multiple of 64 bytes. Generally the contents of the message buffer are the padded value as given by Message||stopbit||zero padding||Message length.

The initial hash value as given in buffer p_digest will be used and this buffer will be updated with the computed SHA256 message digest value.

Table 1756:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

R_SCE_SHA256_HashUpdate

```
 R_SCE_SHA256_HashUpdate ( hash_ctrl_t  *const p_ctrl ,   const uint32_t
* p_msg ,   uint32_t num_words ,   uint32_t * p_digest )
```

Brief description

Update hash value using the given input message from buffer p_source of num_words words, write the result to p_digest

Detailed description

Compute the SHA256 message digest for the given input message buffer p_msg of length num_words words. The length of the message buffer needs to be a multiple of 64 bytes. Generally the contents of the message buffer are the padded value as given by Message||stopbit||zero padding||Message length.

The initial hash value as given in buffer p_digest will be used and this buffer will be updated with the computed SHA256 message digest value.

Table 1757:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

10.49.8.6 SCE_RSA

Primitive cryptographic functions.

RSA 1024-bit implementation for key generation, encryption, decryption, signature generation and signature verification functions for wrapped keys.

RSA 2048-bit implementation for key generation, encryption, decryption, signature generation and signature verification functions for wrapped keys.

RSA 1024-bit implementation for encryption, decryption, signature generation and signature verification functions

RSA 2048-bit implementation for encryption, decryption, signature generation and signature verification functions

Functions

- R_SCE_RSA_Open

- R_SCE_RSA_VersionGet
- R_SCE_RSA_Close
- R_SCE_HRK_RSA_1024PublicKeyEncrypt
- R_SCE_HRK_RSA_1024PublicKeyVerify
- R_SCE_HRK_RSA_1024PrivateKeyDecrypt
- R_SCE_HRK_RSA_1024PrivateKeySign
- R_SCE_HRK_RSA_1024PrivateCrtKeyDecrypt
- R_SCE_HRK_RSA_1024PrivateCrtKeySign
- R_SCE_HRK_RSA_1024KeyCreate
- R_SCE_HRK_RSA_2048PublicKeyEncrypt
- R_SCE_HRK_RSA_2048PublicKeyVerify
- R_SCE_HRK_RSA_2048PrivateKeyDecrypt
- R_SCE_HRK_RSA_2048PrivateKeySign
- R_SCE_HRK_RSA_2048PrivateCrtKeyDecrypt
- R_SCE_HRK_RSA_2048PrivateCrtKeySign
- R_SCE_HRK_RSA_2048KeyCreate
- R_SCE_RSA_1024PublicKeyEncrypt
- R_SCE_RSA_1024PublicKeyVerify
- R_SCE_RSA_1024PrivateKeyDecrypt
- R_SCE_RSA_1024PrivateKeySign
- R_SCE_RSA_1024PrivateCrtKeyDecrypt
- R_SCE_RSA_1024PrivateCrtKeySign
- R_SCE_RSA_1024KeyCreate
- R_SCE_RSA_2048PublicKeyEncrypt
- R_SCE_RSA_2048PublicKeyVerify
- R_SCE_RSA_2048PrivateKeyDecrypt
- R_SCE_RSA_2048PrivateKeySign
- R_SCE_RSA_2048PrivateCrtKeyDecrypt
- R_SCE_RSA_2048PrivateCrtKeySign
- R_SCE_RSA_2048KeyCreate

Variables

- g_rsa1024_on_sce_hrk
- sce_hrk_rsa_table_1024
- g_rsa2048_on_sce_hrk

- sce_hrk_rsa_table_2048
- s_sce_rsa_version
- g_rsa1024_on_sce
- rsa_table_1024
- g_rsa2048_on_sce
- rsa_table_2048

Defines

- #define SCE_RSA_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_RSA_CODE_VERSION_MINOR

  Initial value:`(00)`
- #define HASH_SIZE_BYTES

  Initial value:`((1024 / 8))`
- #define RSA_KEY_LENGTH

  Initial value:`1024U`
- #define INITIAL_STAGE_NUM

  Initial value:`(0)`
- #define RSA_NUM_TRIES

  Initial value:`(5120)`
- #define RSA_KEYLENGTH

  Initial value:`(1024U)`

  RSA Keylength in bits
- #define UINT32_BITS

  Initial value:`(32)`

  uint32_t word size in bits
- #define UINT32_BYTES

  Initial value:`(4)`

  uint32_t word size in bytes
- #define RSA_MODULUS_SIZE

  Initial value:`((RSA_KEYLENGTH)/(UINT32_BITS))`

  rsa modulus data size in words

- #define RSA_PRIVATE_EXPONENT_SIZE

  Initial value:`(RSA_MODULUS_SIZE)`

  rsa private exponent data size in words
- #define RSA_PUBLIC_EXPONENT_SIZE

  Initial value:`(1)`

  rsa public exponent data size in words
- #define RSA_PUBLIC_KEYSIZE

  Initial value:`((RSA_PUBLIC_EXPONENT_SIZE)+(RSA_MODULUS_SIZE))`

  rsa public key size in words
- #define RSA_PRIVATE_KEYSIZE

  Initial value:`((RSA_PRIVATE_EXPONENT_SIZE)+(RSA_MODULUS_SIZE))`

  rsa private key size in words
- #define RSA_PRIVATE_CRT_KEYSIZE

  Initial value:`((5*(RSA_MODULUS_SIZE))/2)`

  RSA private key in CRT format size in words

g_rsa1024_on_sce_hrk

`rsa_api_t::g_rsa1024_on_sce_hrk`

Detailed description

SCE/RSA implementation of RSA API.

Initialized as

```
g_rsa1024_on_sce_hrk=
{
    .open       = R_SCE_RSA_Open,
    .encrypt    = R_SCE_HRK_RSA_1024PublicKeyEncrypt,
    .decrypt    = R_SCE_HRK_RSA_1024PrivateKeyDecrypt,
    .decryptCrt = R_SCE_HRK_RSA_1024PrivateCrtKeyDecrypt,
    .sign       = R_SCE_HRK_RSA_1024PrivateKeySign,
    .signCrt    = R_SCE_HRK_RSA_1024PrivateCrtKeySign,
    .verify     = R_SCE_HRK_RSA_1024PublicKeyVerify,
    .close      = R_SCE_RSA_Close,
    .versionGet = R_SCE_RSA_VersionGet,
    .keyCreate  = R_SCE_HRK_RSA_1024KeyCreate
}
```

sce_hrk_rsa_table_1024

`const sce_hrk_rsa_table_1024`

Initialized as

```
sce_hrk_rsa_table_1024=
{
```

```
    .keylength = RSA_KEY_LENGTH,
    .keygen    = HW_SCE_HRK_RSA_1024KeyGenerate,
    .encrypt   = HW_SCE_RSA_1024PublicEncrypt,
    .decrypt   = HW_SCE_HRK_RSA_1024PrivateKeyDecrypt,
    .decryptCrt= R_SCE_RSA_Dummy_HW_Proc
}
```

g_rsa2048_on_sce_hrk

`rsa_api_t::g_rsa2048_on_sce_hrk`

Detailed description

SCE/RSA implementation of RSA API.

Initialized as

```
g_rsa2048_on_sce_hrk=
{
    .open       = R_SCE_RSA_Open,
    .encrypt    = R_SCE_HRK_RSA_2048PublicKeyEncrypt,
    .decrypt    = R_SCE_HRK_RSA_2048PrivateKeyDecrypt,
    .decryptCrt = R_SCE_HRK_RSA_2048PrivateCrtKeyDecrypt,
    .sign       = R_SCE_HRK_RSA_2048PrivateKeySign,
    .signCrt    = R_SCE_HRK_RSA_2048PrivateCrtKeySign,
    .verify     = R_SCE_HRK_RSA_2048PublicKeyVerify,
    .close      = R_SCE_RSA_Close,
    .versionGet = R_SCE_RSA_VersionGet,
    .keyCreate  = R_SCE_HRK_RSA_2048KeyCreate
}
```

sce_hrk_rsa_table_2048

`const sce_hrk_rsa_table_2048`

s_sce_rsa_version

`ssp_version_t::s_sce_rsa_version`

Detailed description

Version data structure used by error logger macro.

Initialized as

```
s_sce_rsa_version=
{
    .api_version_major  = RSA_API_VERSION_MAJOR,
    .api_version_minor  = RSA_API_VERSION_MINOR,
    .code_version_major = SCE_RSA_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_RSA_CODE_VERSION_MINOR
}
```

g_rsa1024_on_sce

`rsa_api_t::g_rsa1024_on_sce`

Detailed description

SCE/RSA implementation of RSA API.

Initialized as

```
g_rsa1024_on_sce=
{
    .open       = R_SCE_RSA_Open,
    .encrypt    = R_SCE_RSA_1024PublicKeyEncrypt,
    .decrypt    = R_SCE_RSA_1024PrivateKeyDecrypt,
    .decryptCrt = R_SCE_RSA_1024PrivateCrtKeyDecrypt,
    .sign       = R_SCE_RSA_1024PrivateKeySign,
    .signCrt    = R_SCE_RSA_1024PrivateCrtKeySign,
    .verify     = R_SCE_RSA_1024PublicKeyVerify,
    .close      = R_SCE_RSA_Close,
    .versionGet = R_SCE_RSA_VersionGet,
    .keyCreate  = R_SCE_RSA_1024KeyCreate

}
```

rsa_table_1024

```
const rsa_table_1024
```

Initialized as

```
rsa_table_1024=
{
    .keylength = RSA_KEYLENGTH,
    .keygen    = HW_SCE_RSA_1024KeyGenerate
}
```

g_rsa2048_on_sce

```
rsa_api_t::g_rsa2048_on_sce
```

Initialized as

```
g_rsa2048_on_sce=
{
    .open       = R_SCE_RSA_Open,
    .encrypt    = R_SCE_RSA_2048PublicKeyEncrypt,
    .decrypt    = R_SCE_RSA_2048PrivateKeyDecrypt,
    .decryptCrt = R_SCE_RSA_2048PrivateCrtKeyDecrypt,
    .sign       = R_SCE_RSA_2048PrivateKeySign,
    .signCrt    = R_SCE_RSA_2048PrivateCrtKeySign,
    .verify     = R_SCE_RSA_2048PublicKeyVerify,
    .close      = R_SCE_RSA_Close,
    .versionGet = R_SCE_RSA_VersionGet,
    .keyCreate  = R_SCE_RSA_2048KeyCreate
}
```

rsa_table_2048

```
const rsa_table_2048
```

Initialized as

```
rsa_table_2048=
{
```

```
    .keylength = RSA_KEYLENGTH,
    .keygen    = HW_SCE_RSA_2048KeyGenerate
}
```

R_SCE_RSA_Open

```
R_SCE_RSA_Open ( rsa_ctrl_t  *const p_ctrl ,  rsa_cfg_t  const *const p_cfg )
```

Detailed description

RSA Initialization

Table 1758:Return values

Name	Description
SF_CRYPTO_SUCCESS	random number generation successful
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_RSA_VersionGet

```
R_SCE_RSA_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1759:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_RSA_Close

```
R_SCE_RSA_Close ( rsa_ctrl_t  *const p_ctrl )
```

Detailed description

Close RSA driver

Table 1760:Return values

Name	Description
SF_CRYPTO_SUCCESS	random numbergeneration successful

Table 1760:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	SCE resource is busy
SSP_ERR_CRYPTO_SCE_FAIL	SCE internal I/O is not empty

R_SCE_HRK_RSA_1024PublicKeyEncrypt

```
 R_SCE_HRK_RSA_1024PublicKeyEncrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,
  uint32_t * p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_RSA_1024PublicKeyVerify

```
 R_SCE_HRK_RSA_1024PublicKeyVerify ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,   uint32_t
* p_signature ,   uint32_t * p_padded_hash )
```

R_SCE_HRK_RSA_1024PrivateKeyDecrypt

```
 R_SCE_HRK_RSA_1024PrivateKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_RSA_1024PrivateKeySign

```
 R_SCE_HRK_RSA_1024PrivateKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_RSA_1024PrivateCrtKeyDecrypt

```
 R_SCE_HRK_RSA_1024PrivateCrtKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_RSA_1024PrivateCrtKeySign

```
 R_SCE_HRK_RSA_1024PrivateCrtKeySign ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

R_SCE_HRK_RSA_1024KeyCreate

```
 R_SCE_HRK_RSA_1024KeyCreate ( rsa_ctrl_t  *const p_ctrl ,  rsa_key_t
* p_private_key ,  rsa_key_t  * p_public_key )
```

Brief description

Creation of 1024-bit RSA Key pair in with private key in wrapped format.

Detailed description

RSA private key is created based on the key_format specified by p_private_key.
RSA_KEY_FORMAT_WRAPPED_PRIVATE_KEY specifies a wrapped standard format key.

Table 1761:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Key creation was successful.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer may not empty.
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	Resource conflict occurred.
SSP_ERR_CRYPTO_INVALID_SIZE	An input parameter of invalid size.
SSP_ERR_CRYPTO_UNKNOWN	An input parameter is of unknown type.

NOTE: The buffers to hold the keys must be adequate for the key formats requested.

R_SCE_HRK_RSA_2048PublicKeyEncrypt

```
 R_SCE_HRK_RSA_2048PublicKeyEncrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,
  uint32_t * p_source ,   uint32_t * p_dest )
```

Brief description

Encrypt using 2048-bit RSA public key.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 2048-bit RSA public key from buffer p_key and domain parameters from p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1762:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_Open()

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The p_key buffer must contain 260 bytes of RSA public key data in the format (public_exponent || public_modulus), where public_exponent is 1 words of data and public_modulus is 64 words of data

R_SCE_HRK_RSA_2048PublicKeyVerify

```
 R_SCE_HRK_RSA_2048PublicKeyVerify ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,   uint32_t
* p_signature ,   uint32_t * p_padded_hash )
```

Brief description

Signature Verification using 2048-bit RSA public key.

Detailed description

Verify RSA signature data from buffer p_signature of length num_words using 2048-bit RSA public key. The buffer p_padded_hash indicates the message buffer from which the RSA signature should have been generated.

Table 1763:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Normal end, valid signature
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	incorrect signature
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: The p_key buffer must contain 260 bytes of RSA public key data in the format (public_exponent || public_modulus) where public_exponent is 1 word and public_modulus is 64 words.

NOTE: Length of the p_key, p_signature and p_padded_hash must each be equal to num_words. For RSA 2048-bit key, num_words should be equal to 64.

R_SCE_HRK_RSA_2048PrivateKeyDecrypt

```
 R_SCE_HRK_RSA_2048PrivateKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 2048-bit RSA private key in wrapped format.

Detailed description

Decrypt imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1764:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Decrypt operation was successful.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 64 words of RSA private key data followed by 64 words of RSA key modulus data

R_SCE_HRK_RSA_2048PrivateKeySign

```
 R_SCE_HRK_RSA_2048PrivateKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 2048-bit RSA private key in wrapped format.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1765:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Signature generation was successful.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	Resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 64 words of RSA private key data followed by 64 words of RSA key modulus data

R_SCE_HRK_RSA_2048PrivateCrtKeyDecrypt

```
R_SCE_HRK_RSA_2048PrivateCrtKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 2048-bit RSA private key in wrapped CRT format.

Detailed description

Wrapped CRT format is not supported at present. Parameter check is not performed as the function only returns SSP_ERR_CRYPTO_NOT_IMPLEMENTED.

Table 1766:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_HRK_RSA_2048PrivateCrtKeySign

```
R_SCE_HRK_RSA_2048PrivateCrtKeySign ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 2048-bit RSA private key represented in wrapped CRT format.

Detailed description

Wrapped CRT format is not supported at present. Parameter check is not performed as the function only returns SSP_ERR_CRYPTO_NOT_IMPLEMENTED.

Table 1767:Return values

Name	Description
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	This function is not implemented.

R_SCE_HRK_RSA_2048KeyCreate

```
R_SCE_HRK_RSA_2048KeyCreate ( rsa_ctrl_t  *const p_ctrl ,  rsa_key_t
* p_private_key ,  rsa_key_t  * p_public_key )
```

Brief description

Creation of 2048-bit RSA Key pair in with private key in wrapped format.

Detailed description

RSA private key is created based on the key_format specified by p_private_key.
RSA_KEY_FORMAT_WRAPPED_PRIVATE_KEY specifies a wrapped standard format key.

Table 1768:Return values

Name	Description
SSP_ERR_ASSERTION	An input parameter is NULL or of invalid format.
SF_CRYPTO_SUCCESS	Key creation was successful.
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer may not empty.
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	Resource conflict occurred.
SSP_ERR_CRYPTO_INVALID_SIZE	An input parameter of invalid size.
SSP_ERR_CRYPTO_UNKNOWN	An input parameter is of unknown type.

NOTE: The buffers to hold the keys must be adequate for the key formats requested.

R_SCE_RSA_1024PublicKeyEncrypt

```
 R_SCE_RSA_1024PublicKeyEncrypt ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Encryption using 1024-bit RSA public key.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 1024-bit RSA public key from buffer p_key and domain parameters from p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1769:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_Open()

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 132 bytes of RSA public key data in the format (public_exponent || public_modulus), where public_exponent is 1 words of data and public_modulus is 32 words of data

R_SCE_RSA_1024PublicKeyVerify

```
 R_SCE_RSA_1024PublicKeyVerify ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t num_words ,   uint32_t
* p_signature ,   uint32_t * p_padded_hash )
```

Brief description

Signature Verification using 1024-bit RSA public key.

Detailed description

Verify RSA signature data from buffer p_signature of length num_words using 1024-bit RSA public key. The buffer p_padded_hash indicates the message buffer from which the RSA signature should have been generated.

Table 1770:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end, valid signature
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	incorrect signature
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: The p_key buffer must contain 132 bytes of RSA public key data in the format (public_exponent || public_modulus) where public_exponent is 1 word and public_modulus is 32 words.

NOTE: Length of the p_key, p_signature and p_padded_hash must each be equal to num_words. For RSA 1024-bit key, num_words should be equal to 32.

R_SCE_RSA_1024PrivateKeyDecrypt

```
 R_SCE_RSA_1024PrivateKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 1024-bit RSA private key (standard format)

Detailed description

Decrypt imaxcnt words of input data from buffer p_source using the 1024-bit RSA private key from buffer p_key. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1771:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty

Table 1771:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 32 words of RSA private key data followed by 32 words of RSA key modulus data

R_SCE_RSA_1024PrivateKeySign

```
 R_SCE_RSA_1024PrivateKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 1024-bit RSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the 1024-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1772:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 32 words of RSA private key data followed by 32 words of RSA key modulus data

R_SCE_RSA_1024PrivateCrtKeyDecrypt

```
 R_SCE_RSA_1024PrivateCrtKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 1024-bit RSA private key (CRT format)

Detailed description

Decrypt imaxcnt words of input data from buffer p_source using the 1024-bit RSA private key from buffer key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1773:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key pointer to 80-word buffer with 1024-bit RSA key CRT parameters (d mod (q-1) || q || d mod (p-1) || p || q^-1 mod p)

R_SCE_RSA_1024PrivateCrtKeySign

```
 R_SCE_RSA_1024PrivateCrtKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 1024-bit RSA private key represented in CRT format.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the 1024-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1774:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key pointer to 80-word buffer with 1024-bit RSA key CRT parameters (d mod (q-1) || q || d mod (p-1) || p || q^-1 mod p)

R_SCE_RSA_1024KeyCreate

```
 R_SCE_RSA_1024KeyCreate ( rsa_ctrl_t  *const p_ctrl ,  rsa_key_t
* p_private_key ,  rsa_key_t  * p_public_key )
```

R_SCE_RSA_2048PublicKeyEncrypt

```
 R_SCE_RSA_2048PublicKeyEncrypt ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Encrypt using 2048-bit RSA public key.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 2048-bit RSA public key from buffer p_key and domain parameters from p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1775:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_Open()

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 260 bytes of RSA public key data in the format (public_exponent || public_modulus), where public_exponent is 1 words of data and public_modulus is 64 words of data

R_SCE_RSA_2048PublicKeyVerify

```
 R_SCE_RSA_2048PublicKeyVerify ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_signature ,   uint32_t * p_paddedHash )
```

Brief description

Signature Verification using 2048-bit RSA public key.

Detailed description

Verify RSA signature data from buffer p_signature of length num_words using 2048-bit RSA public key. The buffer p_padded_hash indicates the message buffer from which the RSA signature should have been generated.

Table 1776:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end, valid signature
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	incorrect signature

Table 1776:Return values (Continued)

Name	Description
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: The p_key buffer must contain 260 bytes of RSA public key data in the format (public_exponent || public_modulus) where public_exponent is 1 word and public_modulus is 64 words.

NOTE: Length of the p_key, p_signature and p_padded_hash must each be equal to num_words. For RSA 2048-bit key, num_words should be equal to 64.

R_SCE_RSA_2048PrivateKeyDecrypt

```
 R_SCE_RSA_2048PrivateKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 2048-bit RSA private key (standard format)

Detailed description

Decrypt imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1777:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 64 words of RSA private key data followed by 64 words of RSA key modulus data

R_SCE_RSA_2048PrivateKeySign

```
 R_SCE_RSA_2048PrivateKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 2048-bit RSA private key.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1778:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain 64 words of RSA private key data followed by 64 words of RSA key modulus data

R_SCE_RSA_2048PrivateCrtKeyDecrypt

```
 R_SCE_RSA_2048PrivateCrtKeyDecrypt ( rsa_ctrl_t  *const p_ctrl ,   const
uint32_t * p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Decrypt using 2048-bit RSA private key (CRT format)

Detailed description

Decrypt imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1779:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain pointer to 160-word buffer with 2048-bit RSA key CRT parameters (d mod (q-1) || q || d mod (p-1) || p || q^-1 mod p)

R_SCE_RSA_2048PrivateCrtKeySign

```
 R_SCE_RSA_2048PrivateCrtKeySign ( rsa_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   const uint32_t * p_domain ,   uint32_t imaxcnt ,   uint32_t
* p_source ,   uint32_t * p_dest )
```

Brief description

Signature generation using 2048-bit RSA private key represented in CRT format.

Detailed description

Sign imaxcnt words of input data from buffer p_source using the 2048-bit RSA private key from buffer p_key and domain parameters from buffer p_domain. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast imaxcnt words of data.

Table 1780:Return values

Name	Description
SF_CRYPTO_SUCCESS	Normal end
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized by calling the functions R_SCE_Open

NOTE: p_dest must have space to hold at least imaxcnt words of data.

NOTE: The p_key buffer must contain pointer to 160-word buffer with 2048-bit RSA key CRT parameters (d mod (q-1) || q || d mod (p-1) || p || q^-1 mod p)

R_SCE_RSA_2048KeyCreate

```
 R_SCE_RSA_2048KeyCreate ( rsa_ctrl_t  *const p_ctrl ,  rsa_key_t
* p_private_key ,  rsa_key_t  * p_public_key )
```

10.49.8.7 SCE_TDES

Primitive cryptographic functions.

Triple DES encryption and decryption functions

Functions

- R_SCE_TDES_Open
- R_SCE_TDES_VersionGet
- R_SCE_TDES_Close
- R_SCE_TDES_192CbcEncrypt
- R_SCE_TDES_192CbcDecrypt
- R_SCE_TDES_192CtrEncrypt
- R_SCE_TDES_192EcbEncrypt

- R_SCE_TDES_192EcbDecrypt

Variables

- g_tdes192ecb_on_sce
- g_tdes192cbc_on_sce
- g_tdes192ctr_on_sce
- s_sce_tdes_version

Defines

- #define SCE_TDES_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_TDES_CODE_VERSION_MINOR

  Initial value:`(00)`

g_tdes192ecb_on_sce

`tdes_api_t::g_tdes192ecb_on_sce`

Detailed description

SCE/TDES implementation of TDES API.

Initialized as

```
g_tdes192ecb_on_sce=
{
    .open       = R_SCE_TDES_Open,
    .encrypt    = R_SCE_TDES_192EcbEncrypt,
    .decrypt    = R_SCE_TDES_192EcbDecrypt,
    .close      = R_SCE_TDES_Close,
    .versionGet = R_SCE_TDES_VersionGet
}
```

g_tdes192cbc_on_sce

`tdes_api_t::g_tdes192cbc_on_sce`

Initialized as

```
g_tdes192cbc_on_sce=
{
    .open       = R_SCE_TDES_Open,
    .encrypt    = R_SCE_TDES_192CbcEncrypt,
    .decrypt    = R_SCE_TDES_192CbcDecrypt,
    .close      = R_SCE_TDES_Close,
    .versionGet = R_SCE_TDES_VersionGet
}
```

g_tdes192ctr_on_sce

`tdes_api_t::g_tdes192ctr_on_sce`

Initialized as

```
g_tdes192ctr_on_sce=
{
    .open       = R_SCE_TDES_Open,
    .encrypt    = R_SCE_TDES_192CtrEncrypt,
    .decrypt    = R_SCE_TDES_192CtrEncrypt,
    .close      = R_SCE_TDES_Close,
    .versionGet = R_SCE_TDES_VersionGet
}
```

s_sce_tdes_version

`ssp_version_t::s_sce_tdes_version`

Detailed description

Version data structure used by error logger macro.

Initialized as

```
s_sce_tdes_version=
{
    .api_version_major  = TDES_API_VERSION_MAJOR,
    .api_version_minor  = TDES_API_VERSION_MINOR,
    .code_version_major = SCE_TDES_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_TDES_CODE_VERSION_MINOR
}
```

R_SCE_TDES_Open

```
R_SCE_TDES_Open ( tdes_ctrl_t  *const p_ctrl ,  tdes_cfg_t  const *const p_cfg )
```

Detailed description

TDES Initialization

Table 1781:Return values

Name	Description
SF_CRYPTO_SUCCESS	random number generation successful
SF_CRPYTO_ERR_CRYPTO_RESOURCE_CONFLICT	SCE resource is busy
SF_CRYPTO_ERR_CRYPTO_SCEFAIL	SCE internal I/O is not empty

R_SCE_TDES_VersionGet

```
R_SCE_TDES_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1782:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_TDES_Close

```
R_SCE_TDES_Close ( tdes_ctrl_t  *const p_ctrl )
```

Detailed description

Close TDES

Table 1783:Return values

Name	Description
SF_CRYPTO_SUCCESS	random numbergeneration successful
SF_CRPYTO_ERR_CRYPTO_RESOURCE_CONFLICT	SCE resource is busy
SF_CRYPTO_ERR_CRYPTO_SCEFAIL	SCE internal I/O is not empty

R_SCE_TDES_192CbcEncrypt

```
R_SCE_TDES_192CbcEncrypt ( tdes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Brief description

Triple DES Encryption with CBC mode.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 192-bit TDES key from buffer key and initialization vector from buffer iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1784:Return values

Name	Description
PASS	Normal end
FAIL	Internal I/O buffer is not empty
RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized.

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The key buffer must contain 24 bytes of TDES key data and

NOTE: the iv buffer must have 8 bytes of random data.

R_SCE_TDES_192CbcDecrypt

```
 R_SCE_TDES_192CbcDecrypt ( tdes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Brief description

Triple DES Decryption with CBC mode.

Detailed description

Decrypt num_words words of input data from buffer p_source using the 192-bit TDES key from buffer key and initialization vector from buffer iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1785:Return values

Name	Description
PASS	Normal end
FAIL	Internal I/O buffer is not empty
RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized.

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The key buffer must contain 24 bytes of TDES key data and

NOTE: the iv buffer must have 8 bytes of random data.

R_SCE_TDES_192CtrEncrypt

```
 R_SCE_TDES_192CtrEncrypt ( tdes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Brief description

Triple DES Encryption with CTR mode.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 192-bit TDES key from buffer key and initialization vector from buffer iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for at least num_words words of data.

Table 1786:Return values

Name	Description
PASS	Normal end
FAIL	Internal I/O buffer is not empty
RESOURCE_CONFLICT	resource conflict occured
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	API not yet implemented

NOTE: SCE module must have been initialized.

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The key buffer must contain 24 bytes of TDES key data and

NOTE: the iv buffer must have 8 bytes of random data.

R_SCE_TDES_192EcbEncrypt

```
 R_SCE_TDES_192EcbEncrypt ( tdes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Brief description

Triple DES Encryption with ECB mode.

Detailed description

Encrypt num_words words of input data from buffer p_source using the 192-bit TDES key from buffer key and initialization vector from buffer iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Table 1787:Return values

Name	Description
PASS	Normal end
FAIL	Internal I/O buffer is not empty
RESOURCE_CONFLICT	resource conflict occurred

NOTE: SCE module must have been initialized and RNG quality check performed by calling the functions R_SCE_SoftReset(), R_SCE_Initialization1() and R_SCE_Initialization2()

NOTE: Block size for TDES is 64-bits (8 bytes or 2 words). Hence num_words must be a multiple of 2.

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The key buffer must contain 24 bytes of TDES key data.

NOTE: The contents of iv buffer are ignored in ECB chaining mode.

Function steps

- To prevent compiler warning for unused parameter.

R_SCE_TDES_192EcbDecrypt

```
 R_SCE_TDES_192EcbDecrypt ( tdes_ctrl_t  *const p_ctrl ,   const uint32_t
* p_key ,   uint32_t * p_iv ,   uint32_t num_words ,   uint32_t * p_source ,
  uint32_t * p_dest )
```

Brief description

Triple DES Decryption with ECB mode. Decrypt num_words words of input data from buffer p_source using the 192-bit TDES key from buffer key and initialization vector from buffer iv. The result will be written to the output buffer from p_dest. The p_dest array is assumed to have space for atleast num_words words of data.

Detailed description

Table 1788:Return values

Name	Description
PASS	Normal end
FAIL	Internal I/O buffer is not empty
RESOURCE_CONFLICT	resource conflict occured

NOTE: SCE module must have been initialized.

NOTE: p_dest must have space to hold at least num_words words of data.

NOTE: The key buffer must contain 24 bytes of TDES key data and

NOTE: The contents of iv buffer are ignored in ECB chaining mode.

Function steps

- To prevent compiler warning for unused parameter.

10.49.8.8 SCE_TRNG

Primitive cryptographic functions.

random number generation functions

Functions

- R_SCE_TRNG_Open
- R_SCE_TRNG_VersionGet
- R_SCE_TRNG_Read
- R_SCE_TRNG_Close
- s_generate_16byte_random_data

Variables

- g_trng_on_sce
- s_sce_trng_version

Defines

- #define SCE_TRNG_CODE_VERSION_MAJOR

  Initial value:`(01)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
- #define SCE_TRNG_CODE_VERSION_MINOR

  Initial value:`(00)`

g_trng_on_sce

`trng_api_t::g_trng_on_sce`

Detailed description

SCE/TRNG implementation of RNG API.

Initialized as

```
g_trng_on_sce=
{
    .open       = R_SCE_TRNG_Open,
    .read       = R_SCE_TRNG_Read,
    .close      = R_SCE_TRNG_Close,
    .versionGet = R_SCE_TRNG_VersionGet
}
```

s_sce_trng_version

`ssp_version_t::s_sce_trng_version`

Detailed description

Version data structure used by error logger macro.

Initialized as

```
s_sce_trng_version=
{
    .api_version_major  = TRNG_API_VERSION_MAJOR,
    .api_version_minor  = TRNG_API_VERSION_MINOR,
    .code_version_major = SCE_TRNG_CODE_VERSION_MAJOR,
    .code_version_minor = SCE_TRNG_CODE_VERSION_MINOR
}
```

R_SCE_TRNG_Open

```
R_SCE_TRNG_Open ( trng_ctrl_t  *const p_ctrl ,  trng_cfg_t  const *const p_cfg )
```

Detailed description

SCE_TRNG example: extern const r_sce_t g_<interface>_on_sce;

Open the TRNG driver for reading random data from the hardware TRNG module TRNG Initialization

Table 1789:Return values

Name	Description
SF_CRYPTO_SUCCESS	random number generation successful
SF_CRPYTO_ERR_CRYPTO_RESOURCE_CONFLICT	SCE resource is busy
SF_CRYPTO_ERR_CRYPTO_SCEFAIL	SCE internal I/O is not empty

R_SCE_TRNG_VersionGet

```
R_SCE_TRNG_VersionGet ( ssp_version_t  *const p_version )
```

Brief description

Sets driver version based on compile time macros.

Detailed description

Table 1790:Return values

Name	Description
SSP_SUCCESS	Successful close.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

R_SCE_TRNG_Read

```
R_SCE_TRNG_Read ( trng_ctrl_t  *const p_ctrl ,   uint32_t *const p_dest ,
 uint32_t nwords )
```

Detailed description

Generate nwords of random number words (4-bytes each) and store them in p_rngbuf buffer SCE hardware TRNG module will be used for generating the random data.

Table 1791:Return values

Name	Description
SF_CRYPTO_SUCCESS	random number generation successful
SSP_ERR_CRYPTO_RNG_FATAL_ERROR	HW_SCE_RNG_Read failed to generate 128-bit (16-byte) random number in p_ctrl->nattempts number of attempts.
SF_CRPYTO_ERR_CRYPTO_RESOURCE_CONFLICT	SCE resource is busy
SF_CRYPTO_ERR_CRYPTO_SCEFAIL	SCE internal I/O is not empty

R_SCE_TRNG_Close

```
R_SCE_TRNG_Close ( trng_ctrl_t  *const p_ctrl )
```

Detailed description

Close the TRNG interface driver

s_generate_16byte_random_data

```
s_generate_16byte_random_data ( trng_ctrl_t  * p_ctrl )
```

Brief description

Generate 16-bytes of random data using the SCE TRNG hardware block.

Detailed description

This function makes p_ctrl->nattempts to generate 16-bytes of random data using the SCE TRNG block. If successful, the buffer p_ctrl->currbuf[0..15] will be updated with newly generated 16-byte block. This data will be different from previously generated 16-byte block (p_ctrl->prevbuf[0..15]).

Table 1792:Return values

Name	Description
SF_CRYPTO_SUCCESS	successul generation of 16-byte random data
SSP_ERR_CRYPTO_RNG_FATAL_ERROR	fatal error, unable to generate 16-byte random data that differs from the previously generated 16-byte block.

Chapter 11 API Reference: Board Support Package

11.1 Board Support Package

Common BSP includes.

The BSP is responsible for getting the MCU from reset to the user application (i.e. the main function). Before reaching the user application the BSP sets up the stacks, heap, clocks, interrupts, and C runtime environment. The BSP is configurable to allow users to modify the process to meet design requirements.

11.2 Supported MCUs

Supported MCUs in this version of the BSP.

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

The BSP has code specific to a MCU and a board. The code that is specific to a MCU can be shared amongst boards that use the MCU.

11.2.1 Functions

- R_SSP_VersionGet

11.2.2 R_SSP_VersionGet

```
ssp_err_t R_SSP_VersionGet ( ssp_pack_version_t  *const p_version )
```

11.2.2.1 Brief description

Set SSP version based on compile time macros.

11.2.2.2 Detailed description

Table 1793:Parameters

Name	Direction	Description
p_version	out	Memory address to return version information to.

Table 1794:Return values

Name	Description
SSP_SUCCESS	Version information stored.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

11.2.3 S124

Code that is common to S124 MCUs.

Implements functions that are common to S124 MCUs.

11.2.3.1 elc_peripheral_t

Possible peripherals to be linked to event signals

11.2.3.2 elc_event_t

Sources of event signals to be linked to other peripherals or the CPU1 ***NOTE:*** *This list may change based on device. This list is for S124.*

11.2.3.3 Cache Functions

This module implements cache functions.

Typedefs

- bsp_cache_frequency_t

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

bsp_cache_frequency_t

```
typedef uint32_t bsp_cache_frequency_t
```

11.2.3.4 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clock_set_callback

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen. S124 has no PLL.
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1795:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk

11.2.3.5 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   0U  )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  AES  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TRNG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.3.6 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop

- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID

  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION

  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG

  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT

  Initial value:`(x)`
- #define BSP_MSTPCRA

  Initial value:`(0U)`
- #define BSP_MSTPCRB

  Initial value:`(1U)`
- #define BSP_MSTPCRC

  Initial value:`(2U)`
- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1796:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1797:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1798:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1799:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1800:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1801:Return values

Name	Description
SSP_SUCCESS	Module is started

Table 1801:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.3.7 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:`(((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFF8FFFU) | ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 12))`

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Some have masks to make sure reserved bits are set appropriately. S124 has no MPU.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING
}
```

11.2.4 S128

Code that is common to S128 MCUs.

Implements functions that are common to S128 MCUs.

11.2.4.1 Cache Functions

This module implements cache functions.

Typedefs

- bsp_cache_frequency_t

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

bsp_cache_frequency_t

```
typedef uint32_t bsp_cache_frequency_t
```

11.2.4.2 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clock_set_callback

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen. S124 has no PLL.
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1802:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk

11.2.4.3 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( ADC , 0U , 0U )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AES , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   1U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   1U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TRNG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.4.4 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop

- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID

  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION

  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG

  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT

  Initial value:`(x)`
- #define BSP_MSTPCRA

  Initial value:`(0U)`
- #define BSP_MSTPCRB

  Initial value:`(1U)`
- #define BSP_MSTPCRC

  Initial value:`(2U)`
- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1803:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1804:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1805:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1806:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1807:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1808:Return values

Name	Description
SSP_SUCCESS	Module is started

Table 1808:Return values (Continued)

Name	Description
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.4.5 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:
  ```
  (((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFF8FFFU) |
  ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 12))
  ```

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

- #define BSP_ROM_REG_MPU_CONTROL_SETTING

  Initial value:
  ```
  ((0xFFFFFCF0U) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_ENABLE << 8) |
  \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_ENABLE << 9) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_ENABLE) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_ENABLE << 1) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_ENABLE << 2) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_ENABLE << 3))
  ```

  Build up SECMPUAC register based on MPU settings.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Some have masks to make sure reserved bits are set appropriately.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_START  & 0x000FFFCU),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_END   & 0x000FFFFU) | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_START  & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_END    | 0x00000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_END   & 0x407FFFFCU) | 0x40000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_END   & 0x407FFFFCU) | 0x40000003U),
    (uint32_t)BSP_ROM_REG_MPU_CONTROL_SETTING
}
```

11.2.5 S3A3

Code that is common to S3A3 MCUs.

Implements functions that are common to S3A3 MCUs.

11.2.5.1 Cache Functions

This module implements cache functions.

Typedefs

- bsp_cache_frequency_t

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

bsp_cache_frequency_t

```
typedef uint32_t bsp_cache_frequency_t
```

11.2.5.2 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clocks_mem_wait_set
- bsp_clocks_mem_wait_get
- bsp_clock_set_callback

Defines

- #define CGC_SRAM_ZERO_WAIT_CYCLES

  Initial value:`(0U)`

  Specify zero wait states for SRAM.
- #define CGC_SRAM_ONE_WAIT_CYCLES

  Initial value:`(1U)`

  Specify one wait states for SRAM.
- #define CGC_ZERO_WAIT_CYCLES

  Initial value:`(0U)`
- #define CGC_TWO_WAIT_CYCLES

  Initial value:`(1U)`
- #define CGC_MAX_ZERO_WAIT_FREQ

  Initial value:`(32000000U)`
- #define CGC_PRCR_KEY

  Initial value:`(0xA500U)`

- #define CGC_PRCR_UNLOCK

  Initial value:`((CGC_PRCR_KEY) | 0x1U)`
- #define CGC_PRCR_LOCK

  Initial value:`((CGC_PRCR_KEY) | 0x0U)`

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- PLL Source clock is always the main oscillator.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- PLL Source clock is always the main oscillator.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1809:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_mem_wait_set

```
bsp_clocks_mem_wait_set (  uint32_t setting )
```

Brief description

This function sets the value of the MEMWAIT register which controls wait cycles for flash read access.

Detailed description

Table 1810:Parameters

Name	Direction	Description
setting	in	MEMWAIT setting

Table 1811:Return values

Name	Description
none	

bsp_clocks_mem_wait_get

```
bsp_clocks_mem_wait_get (  void  )
```

Brief description

This function gets the value of the MEMWAIT register.

Detailed description

Table 1812:Return values

Name	Description
MEMWAIT	setting

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.

11.2.5.3 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   0U  )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   0U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   1U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( USB , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( WDT , 0U , 0U )
```

11.2.5.4 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID
  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION
  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG
  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT
  Initial value:`(x)`
- #define BSP_MSTPCRA
  Initial value:`(0U)`
- #define BSP_MSTPCRB
  Initial value:`(1U)`
- #define BSP_MSTPCRC
  Initial value:`(2U)`

- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1813:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1814:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid

Table 1814:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1815:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1816:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1817:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1818:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.5.5 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:`(((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFF8FFFU) | ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 12))`

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

- #define BSP_ROM_REG_MPU_CONTROL_SETTING

  Initial value:`((0xFFFFFCF0U) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_ENABLE << 8) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_ENABLE << 9) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_ENABLE) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_ENABLE << 1) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_ENABLE << 2) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_ENABLE << 3) \ )`

  Build up SECMPUAC register based on MPU settings.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Some have masks to make sure reserved bits are set appropriately.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_START  & 0x00FFFFCU),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_END   & 0x00FFFFFU) | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_START  & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_END    | 0x00000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_END   & 0x407FFFFCU) | 0x40000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_END   & 0x407FFFFCU) | 0x40000003U),
    (uint32_t)BSP_ROM_REG_MPU_CONTROL_SETTING
}
```

11.2.6 S3A6

Code that is common to S3A6 MCUs.

Implements functions that are common to S3A6 MCUs.

11.2.6.1 Cache Functions

This module implements cache functions.

Typedefs

- bsp_cache_frequency_t

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

bsp_cache_frequency_t

```
typedef uint32_t bsp_cache_frequency_t
```

11.2.6.2 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clocks_mem_wait_set
- bsp_clocks_mem_wait_get
- bsp_clock_set_callback

Defines

- #define CGC_SRAM_ZERO_WAIT_CYCLES

  Initial value:`(0U)`

  Specify zero wait states for SRAM.
- #define CGC_SRAM_ONE_WAIT_CYCLES

  Initial value:`(1U)`

  Specify one wait states for SRAM.
- #define CGC_ZERO_WAIT_CYCLES

  Initial value:`(0U)`
- #define CGC_TWO_WAIT_CYCLES

  Initial value:`(1U)`

- #define CGC_MAX_ZERO_WAIT_FREQ

  Initial value:`(32000000U)`
- #define CGC_PRCR_KEY

  Initial value:`(0xA500U)`
- #define CGC_PRCR_UNLOCK

  Initial value:`((CGC_PRCR_KEY) | 0x1U)`
- #define CGC_PRCR_LOCK

  Initial value:`((CGC_PRCR_KEY) | 0x0U)`

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- PLL Source clock is always the main oscillator.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- PLL Source clock is always the main oscillator.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1819:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_mem_wait_set

```
bsp_clocks_mem_wait_set (  uint32_t setting )
```

Brief description

This function sets the value of the MEMWAIT register which controls wait cycles for flash read access.

Detailed description

Table 1820:Parameters

Name	Direction	Description
setting	in	MEMWAIT setting

Table 1821:Return values

Name	Description
none	

bsp_clocks_mem_wait_get

```
bsp_clocks_mem_wait_get (  void  )
```

Brief description

This function gets the value of the MEMWAIT register.

Detailed description

Table 1822:Return values

Name	Description
MEMWAIT	setting

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.

11.2.6.3 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   0U  )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.6.4 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- calc_mask
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID
  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION
  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG
  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT
  Initial value:`(x)`
- #define BSP_MSTPCRA
  Initial value:`(0U)`
- #define BSP_MSTPCRB
  Initial value:`(1U)`
- #define BSP_MSTPCRC
  Initial value:`(2U)`

- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1823:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1824:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid

Table 1824:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

calc_mask

```
 calc_mask ( ssp_feature_t  const *const p_feature ,   uint8_t data ,   uint32_t
* mask )
```

Brief description

Calculate the mask that will be used with the mstp base register used by the calling function.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1825:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1826:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid

Table 1826:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1827:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1828:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.6.5 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:
  ```
  (((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFF8FFFU) |
  ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 12))
  ```

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

- #define BSP_ROM_REG_MPU_CONTROL_SETTING

  Initial value:
  ```
  ((0xFFFFFCF0U) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_ENABLE << 8) |
  \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_ENABLE << 9) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_ENABLE) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_ENABLE << 1) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_ENABLE << 2) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_ENABLE << 3) \ )
  ```

  Build up SECMPUAC register based on MPU settings.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Some have masks to make sure reserved bits are set appropriately.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_START      & 0xFFFFFFFCU),
```

```
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_START  & 0x00FFFFFCU),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_END   & 0x00FFFFFFU) | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_START  & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_END    | 0x00000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_END   & 0x407FFFFCU) | 0x40000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_END   & 0x407FFFFCU) | 0x40000003U),
    (uint32_t)BSP_ROM_REG_MPU_CONTROL_SETTING
}
```

11.2.7 S3A7

Code that is common to S3A7 MCUs.

Implements functions that are common to S3A7 MCUs.

11.2.7.1 elc_peripheral_t

Possible peripherals to be linked to event signals

11.2.7.2 elc_event_t

Sources of event signals to be linked to other peripherals or the CPU1 ***NOTE:*** *This list may change based on device. This list is for S3A7.*

11.2.7.3 Cache Functions

This module implements cache functions.

Typedefs

- bsp_cache_frequency_t

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

bsp_cache_frequency_t

```
typedef uint32_t bsp_cache_frequency_t
```

11.2.7.4 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init

- bsp_cpu_clock_get
- bsp_clocks_mem_wait_set
- bsp_clocks_mem_wait_get
- bsp_clock_set_callback

Defines

- #define CGC_SRAM_ZERO_WAIT_CYCLES

  Initial value:`(0U)`

  Specify zero wait states for SRAM.
- #define CGC_SRAM_ONE_WAIT_CYCLES

  Initial value:`(1U)`

  Specify one wait states for SRAM.
- #define CGC_ZERO_WAIT_CYCLES

  Initial value:`(0U)`
- #define CGC_TWO_WAIT_CYCLES

  Initial value:`(1U)`
- #define CGC_MAX_ZERO_WAIT_FREQ

  Initial value:`(32000000U)`
- #define CGC_PRCR_KEY

  Initial value:`(0xA500U)`
- #define CGC_PRCR_UNLOCK

  Initial value:`((CGC_PRCR_KEY) | 0x1U)`
- #define CGC_PRCR_LOCK

  Initial value:`((CGC_PRCR_KEY) | 0x0U)`

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- PLL Source clock is always the main oscillator.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- PLL Source clock is always the main oscillator.

- Wait for PLL clock source to stabilize
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1829:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_mem_wait_set

```
bsp_clocks_mem_wait_set (  uint32_t setting )
```

Brief description

This function sets the value of the MEMWAIT register which controls wait cycles for flash read access.

Detailed description

Table 1830:Parameters

Name	Direction	Description
setting	in	

Table 1831:Return values

Name	Description
none	

bsp_clocks_mem_wait_get

```
bsp_clocks_mem_wait_get (  void  )
```

Brief description

This function gets the value of the MEMWAIT register.

Detailed description

Table 1832:Return values

Name	Description
MEMWAIT	setting

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.
- < No MEMWAIT cycles

11.2.7.5 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   0U  )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_LP  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   0U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.7.6 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID

  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION

  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG

  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT

  Initial value:`(x)`
- #define BSP_MSTPCRA

  Initial value:`(0U)`
- #define BSP_MSTPCRB

  Initial value:`(1U)`
- #define BSP_MSTPCRC

  Initial value:`(2U)`
- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1833:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1834:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1835:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1836:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1837:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1838:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.7.7 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:`(((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFF8FFFU) | ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 12))`

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

- #define BSP_ROM_REG_MPU_CONTROL_SETTING

  Initial value:`((0xFFFFFCFEU) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_ENABLE << 8) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_ENABLE << 9) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_ENABLE))`

  Build up SECMPUAC register based on MPU settings.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Some have masks to make sure reserved bits are set appropriately.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_START      & 0x000FFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_START      & 0x000FFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_START  & 0x000FFFFCU),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_END   & 0x000FFFFFU) | 0x00000003U),
    0xFFFFFFFFU,
    0xFFFFFFFFU,
    0xFFFFFFFFU,
    0xFFFFFFFFU,
    0xFFFFFFFFU,
    0xFFFFFFFFU,
    (uint32_t)BSP_ROM_REG_MPU_CONTROL_SETTING
}
```

11.2.8 S5D5

Code that is common to S5D5 MCUs.

Implements functions that are common to S5D5 MCUs.

11.2.8.1 Cache Functions

This module implements cache functions.

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

11.2.8.2 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clocks_rom_wait_set
- bsp_clocks_rom_wait_get
- bsp_clocks_sram_wait_set
- bsp_clock_set_callback

Defines

- #define CGC_SYS_CLOCK_FREQ_NO_RAM_WAITS

  Initial value:`(60000000U)`

  SRAM requires 1 wait state be inserted at ICLK > 60 MHz. SRAMHS is always no wait state
- #define CGC_SYS_CLOCK_FREQ_ONE_ROM_WAITS

  Initial value:`(40000000U)`

  FLASH requires 1 wait state be inserted when (40 MHz < ICLK ≤ 80 MHz)
- #define CGC_SYS_CLOCK_FREQ_TWO_ROM_WAITS

  Initial value:`(80000000U)`

  FLASH requires 2 wait states be inserted when (80 MHz < ICLK ≤ 120 MHz)
- #define CGC_SRAM_ZERO_WAIT_CYCLES

  Initial value:`(0U)`

  Specify zero wait states for SRAM.
- #define CGC_SRAM_ONE_WAIT_CYCLES

  Initial value:`(1U)`

  Specify one wait states for SRAM.
- #define CGC_SRAM_PRCR_KEY

  Initial value:`(0x78U)`
- #define CGC_SRAM_UNLOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x1U)`
- #define CGC_SRAM_LOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x0U)`

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- Set PLL Source clock.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.

- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.
- Set USB clock divisor.
- Configure BCLK
- Configure SDRAM Clock
- MOCO is default clock out of reset. Enable new clock if chosen.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- Set PLL Source clock.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.
- Set USB clock divisor.
- Configure BCLK
- Configure SDRAM Clock

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1839:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_rom_wait_set

```
bsp_clocks_rom_wait_set (  uint8_t setting )
```

Brief description

This function sets the value of the ROMWT register which is used to specify wait states required when accessing Flash ROM.

Detailed description

Table 1840:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1841:Return values

Name	Description
none	

bsp_clocks_rom_wait_get

```
bsp_clocks_rom_wait_get (  void  )
```

Brief description

This function gets the value of the ROMWT register.

Detailed description

Table 1842:Return values

Name	Description
MEMWAIT	setting

bsp_clocks_sram_wait_set

```
bsp_clocks_sram_wait_set (  uint32_t setting )
```

Brief description

This function sets the RAM wait state settings for both the SRAM0 and SRAM0(ECC) RAM memory.

Detailed description

This function sets the RAM wait state settings for the SRAM0, SRAM0 ECC and SRAM1 RAM memory.

Table 1843:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1844:Return values

Name	Description
none	

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk
- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk

11.2.8.3 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   0U  )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  ADC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  AGT  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  BSC  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CAC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CAN  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   2U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   3U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   4U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   5U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
```

SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DAC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DAC , 0U , 1U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DOC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 1U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 2U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 3U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 4U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 5U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 6U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DMAC , 0U , 7U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DRW , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (DTC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (ELC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (EPTPC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (ETHER , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (ETHER , 0U , 1U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (FCU , 0U , 0U)
SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  GLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
```

SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  JPEG  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE

```
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  PDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
```

SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 2U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 3U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 4U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 5U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 6U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 7U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 8U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SCI , 0U , 9U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SRC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SSI , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SSI , 0U , 1U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SDHIMMC , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (SDHIMMC , 0U , 1U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (TSN , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (USB , 0U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (USB , 1U , 0U)
SSP_HW_LOCK_DEFINE
SSP_HW_LOCK_DEFINE (WDT , 0U , 0U)

11.2.8.4 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID

  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION

  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG

  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT

  Initial value:`(x)`
- #define BSP_MSTPCRA

  Initial value:`(0U)`
- #define BSP_MSTPCRB

  Initial value:`(1U)`
- #define BSP_MSTPCRC

  Initial value:`(2U)`
- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1845:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1846:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Do nothing.

- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1847:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1848:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1849:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1850:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.8.5 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:
  ```
  (((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFFF9FFU) |
  ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 9))
  ```

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

- #define BSP_ROM_REG_MPU_CONTROL_SETTING

  Initial value:
  ```
  ((0xFFFFFCF0U) | \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_ENABLE << 8) |
  \ ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_ENABLE << 9) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_ENABLE) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_ENABLE << 1) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_ENABLE << 2) | \
  ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_ENABLE << 3))
  ```

  Build up SECMPUAC register based on MPU settings.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC0_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_START      & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_PC1_END        | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION0_START  & 0x00FFFFCU),
    ((BSP_CFG_ROM_REG_MPU_REGION0_END   & 0x00FFFFFU) | 0x00000003U),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_START  & 0xFFFFFFFCU),
    ((uint32_t)BSP_CFG_ROM_REG_MPU_REGION1_END    | 0x00000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION2_END   & 0x407FFFFCU) | 0x40000003U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_START & 0x407FFFFCU) | 0x40000000U),
    (((uint32_t)BSP_CFG_ROM_REG_MPU_REGION3_END   & 0x407FFFFCU) | 0x40000003U),
    (uint32_t)BSP_ROM_REG_MPU_CONTROL_SETTING
}
```

11.2.9 S5D9

Code that is common to S5D9 MCUs.

Implements functions that are common to S5D9 MCUs.

11.2.9.1 Cache Functions

This module implements cache functions.

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

11.2.9.2 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clocks_rom_wait_set
- bsp_clocks_rom_wait_get
- bsp_clocks_sram_wait_set
- bsp_clock_set_callback

Defines

- #define CGC_SYS_CLOCK_FREQ_NO_RAM_WAITS
  Initial value:`(60000000U)`
  SRAM requires 1 wait state be inserted at ICLK > 60 MHz. SRAMHS is always no wait state
- #define CGC_SYS_CLOCK_FREQ_ONE_ROM_WAITS
  Initial value:`(40000000U)`
  FLASH requires 1 wait state be inserted when (40 MHz < ICLK ≤ 80 MHz)
- #define CGC_SYS_CLOCK_FREQ_TWO_ROM_WAITS
  Initial value:`(80000000U)`
  FLASH requires 2 wait states be inserted when (80 MHz < ICLK ≤ 120 MHz)
- #define CGC_SRAM_ZERO_WAIT_CYCLES
  Initial value:`(0U)`
  Specify zero wait states for SRAM.
- #define CGC_SRAM_ONE_WAIT_CYCLES
  Initial value:`(1U)`
  Specify one wait states for SRAM.
- #define CGC_SRAM_PRCR_KEY
  Initial value:`(0x78U)`

- #define CGC_SRAM_UNLOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x1U)`
- #define CGC_SRAM_LOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x0U)`

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- Set PLL Source clock.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.
- Set USB clock divisor.
- Configure BCLK
- Configure SDRAM Clock
- MOCO is default clock out of reset. Enable new clock if chosen.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- Set PLL Source clock.
- Wait for PLL clock source to stabilize
- Enable ROM cache
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.
- Set USB clock divisor.
- Configure BCLK
- Configure SDRAM Clock

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1851:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_rom_wait_set

```
bsp_clocks_rom_wait_set (  uint8_t setting )
```

Brief description

This function sets the value of the ROMWT register which is used to specify wait states required when accessing Flash ROM.

Detailed description

Table 1852:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1853:Return values

Name	Description
none	

bsp_clocks_rom_wait_get

```
bsp_clocks_rom_wait_get (  void  )
```

Brief description

This function gets the value of the ROMWT register.

Detailed description

Table 1854:Return values

Name	Description
MEMWAIT	setting

bsp_clocks_sram_wait_set

```
bsp_clocks_sram_wait_set (  uint32_t setting )
```

Brief description

This function sets the RAM wait state settings for both the SRAM0 and SRAM0(ECC) RAM memory.

Detailed description

This function sets the RAM wait state settings for the SRAM0, SRAM0 ECC and SRAM1 RAM memory.

Table 1855:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1856:Return values

Name	Description
none	

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk
- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM

- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk

11.2.9.3 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( ADC , 0U , 0U )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( ADC , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( BSC , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAC , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAN , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAN , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( COMP_HS , 0U , 0U )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DRW  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  EPTPC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ETHER  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ETHER  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  JPEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  PDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  USB  ,   1U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.9.4 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID
  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION
  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG
  Initial value:`((x) << 6)`
- #define BSP_MSTP_BIT
  Initial value:`(x)`

- #define BSP_MSTPCRA

  Initial value:`(0)`
- #define BSP_MSTPCRB

  Initial value:`(1)`
- #define BSP_MSTPCRC

  Initial value:`(2)`
- #define BSP_MSTPCRD

  Initial value:`(3)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1857:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.
stop	in	true to stop module, false to start module

Table 1858:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Do nothing.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1859:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1860:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid

Table 1860:Return values (Continued)

Name	Description
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1861:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1862:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

11.2.10 S7G2

Code that is common to S7G2 MCUs.

Implements functions that are common to S7G2 MCUs.

11.2.10.1 elc_peripheral_t

Possible peripherals to be linked to event signals

11.2.10.2 elc_event_t

Sources of event signals to be linked to other peripherals or the CPU1 ***NOTE:*** *This list may change based on device. This list is for S7G2.*

11.2.10.3 Cache Functions

This module implements cache functions.

bsp_cache_state_t

Cache enum. Passed into cache functions such as R_BSP_CacheOff() and R_BSP_CacheSet.

11.2.10.4 Clock Initialization

Functions in this file configure the system clocks based upon the macros in bsp_clock_cfg.h.

Functions

- bsp_clock_set_prechange
- bsp_clock_set_postchange
- bsp_clock_init
- bsp_cpu_clock_get
- bsp_clocks_rom_wait_set
- bsp_clocks_rom_wait_get
- bsp_clocks_sram_wait_set
- bsp_clocks_hsram_wait_set
- bsp_clock_set_callback

Defines

- #define CGC_SYS_CLOCK_FREQ_NO_RAM_WAITS

  Initial value:`(120000000U)`
- #define CGC_SYS_CLOCK_FREQ_NO_HSRAM_WAITS

  Initial value:`(200000000U)`
- #define CGC_SYS_CLOCK_FREQ_ONE_ROM_WAITS

  Initial value:`(80000000U)`
- #define CGC_SYS_CLOCK_FREQ_TWO_ROM_WAITS

  Initial value:`(160000000U)`
- #define CGC_SRAM_ZERO_WAIT_CYCLES

  Initial value:`(0U)`

  Specify zero wait states for SRAM.

- #define CGC_SRAM_ONE_WAIT_CYCLES

  Initial value:`(1U)`

  Specify one wait states for SRAM.
- #define CGC_SRAM_PRCR_KEY

  Initial value:`(0x78U)`
- #define CGC_SRAM_UNLOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x1U)`
- #define CGC_SRAM_LOCK

  Initial value:`(((CGC_SRAM_PRCR_KEY) << 1) | 0x0U)`

bsp_clock_set_prechange

```
bsp_clock_set_prechange ( bsp_clock_set_callback_args_t  * p_args )
```

Brief description

This function sets the ROM and RAM wait state settings based on the requested clock change to the CGC.

Detailed description

Table 1863:Return values

Name	Description
none.	

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK < 120 MHz
- 1 wait: ICLK >= 120 MHz
- Wait states for High speed RAM (SRAMHS)
- No wait: ICLK <= 200 MHz
- 1 wait: ICLK > 200 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk

bsp_clock_set_postchange

```
bsp_clock_set_postchange ( bsp_clock_set_callback_args_t  * p_args )
```

Brief description

This function sets the ROM and RAM wait state settings based on the system clock which has just been set by the CGC.

Detailed description

Table 1864:Return values

Name	Description
none.	

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 120 MHz
- 1 wait: ICLK > 120 MHz
- Wait states for High speed RAM (SRAMHS)
- No wait: ICLK <= 200 MHz
- 1 wait: ICLK > 200 MHz
- In this case we need to set the wait state AFTER we change iclk

bsp_clock_init

```
bsp_clock_init (  void  )
```

Brief description

Sets up system clocks.

Function steps

- MOCO is default clock out of reset. Enable new clock if chosen.
- Need to start PLL source clock and let it stabilize before starting PLL
- Set PLL Divider.
- Set PLL Multiplier.
- Set PLL Source clock.
- Wait for PLL clock source to stabilize
- MOCO, LOCO, and subclock do not have stabilization flags that can be checked.
- Wait for clock source to stabilize
- Set which clock to use for system clock and divisors for all system clocks.
- Set USB clock divisor.
- Configure BCLK
- Configure SDRAM Clock

bsp_cpu_clock_get

```
bsp_cpu_clock_get (  void  )
```

Brief description

Returns frequency of CPU clock in Hz.

Detailed description

Table 1865:Return values

Name	Description
Frequency	of the CPU in Hertz

bsp_clocks_rom_wait_set

```
bsp_clocks_rom_wait_set (  uint8_t setting )
```

Brief description

This function sets the value of the ROMWT register which is used to specify wait states required when accessing Flash ROM.

Detailed description

Table 1866:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1867:Return values

Name	Description
none	

bsp_clocks_rom_wait_get

```
bsp_clocks_rom_wait_get (  void  )
```

Brief description

This function gets the value of the ROMWT register.

Detailed description

Table 1868:Return values

Name	Description
MEMWAIT	setting

bsp_clocks_sram_wait_set

```
bsp_clocks_sram_wait_set (  uint32_t setting )
```

Brief description

This function sets the RAM wait state settings for both the SRAM0 and SRAM1 RAM memory.

Detailed description

Table 1869:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1870:Return values

Name	Description
none	

bsp_clocks_hsram_wait_set

```
bsp_clocks_hsram_wait_set (  uint32_t setting )
```

Brief description

This function sets the RAM wait state settings for High Speed RAM memory.

Detailed description

Table 1871:Parameters

Name	Direction	Description
setting	in	The number of wait states to be used.

Table 1872:Return values

Name	Description
none	

bsp_clock_set_callback

```
ssp_err_t bsp_clock_set_callback ( bsp_clock_set_callback_args_t  * p_args )
```

Brief description

This function sets the ROM and RAM wait state settings for a requested system clock change (PRE) or a just updated system clock change (POST)

Detailed description

Table 1873:Parameters

Name	Direction	Description
p_args	in	- Pre/Post request and clock frequency information

Table 1874:Return values

Name	Description
return	SSP_SUCCESS

Function steps

- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk
- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.
- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.
- The current frequency must be less than 32 MHz and the mcu must be in high speed mode, before changing wait cycles to 0.
- < No MEMWAIT cycles
- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM
- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk
- Wait states for low speed RAM (SRAM0 and SRAM1)
- No wait: ICLK <= 60 MHz
- 1 wait: ICLK > 60 MHz
- Calculate the Wait states for ROM

- Set the wait state BEFORE we change iclk
- In this case we need to set the wait state AFTER we change iclk

11.2.10.5 Hardware Locks

This file allocates hardware locks used in Atomic Locking.

Functions

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE
- SSP_HW_LOCK_DEFINE

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( ADC , 0U , 0U )
```

Detailed description

Used to allocated hardware locks. Parameters are as follows:

1) IP name (ssp_ip_t enum without the SSP_IP_ prefix).
2) Unit number (used for blocks with variations like USB, not to be confused with ADC unit).
3) Channel number

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( ADC , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( AGT , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( BSC , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAC , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAN , 0U , 0U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( CAN , 0U , 1U )
```

SSP_HW_LOCK_DEFINE

```
SSP_HW_LOCK_DEFINE ( COMP_HS , 0U , 0U )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  COMP_HS  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  CTSU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAAD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DOC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  DMAC  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DRW  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  DTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ELC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  EPTPC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ETHER  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ETHER  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  FCU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GLCDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   9U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  GPT  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   10U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   11U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   12U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   13U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   14U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  ICU  ,   0U  ,   15U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IIC  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IRDA  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  IWDT  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  JPEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  KEY  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LPM  ,   1U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  LVD  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MMF  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  MPU  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  OPS  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  PDC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   2U  )
```

```
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  POEG  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  QSPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SPI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  RTC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCE  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   1U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   2U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   3U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   4U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   5U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   6U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   7U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   8U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SCI  ,   0U  ,   9U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SRC  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   0U  )
SSP_HW_LOCK_DEFINE
```

```
 SSP_HW_LOCK_DEFINE (  SSI  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  SDHIMMC  ,   0U  ,   1U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  TSN  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  USB  ,   0U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  USB  ,   1U  ,   0U  )
```
SSP_HW_LOCK_DEFINE
```
 SSP_HW_LOCK_DEFINE (  WDT  ,   0U  ,   0U  )
```

11.2.10.6 Module Start and Stop

Module start and stop functions are provided to enable or disable peripherals.

Functions

- r_bsp_module_start_stop
- R_BSP_ModuleStop
- R_BSP_ModuleStart

Data structures

- bsp_mstp_bit_t
- bsp_mstp_data_t

Variables

- g_bsp_module_stop
- gp_mstp

Defines

- #define BSP_MSTP_DATA_INVALID
  Initial value:`(0xFFU)`
- #define BSP_MSTP_EXCEPTION
  Initial value:`(1U << 5)`
- #define BSP_MSTP_REG
  Initial value:`((x) << 6)`

- #define BSP_MSTP_BIT

  Initial value:`(x)`
- #define BSP_MSTPCRA

  Initial value:`(0U)`
- #define BSP_MSTPCRB

  Initial value:`(1U)`
- #define BSP_MSTPCRC

  Initial value:`(2U)`
- #define BSP_MSTPCRD

  Initial value:`(3U)`
- #define BSP_COMPILE_TIME_ASSERT

  Initial value:`((void) sizeof(char[1 - 2 * !(e)]))`

  Used to generate a compiler error (divided by 0 error) if the assertion fails. This is used in place of "#error" for expressions that cannot be evaluated by the preprocessor like sizeof().

g_bsp_module_stop

```
const uint8_t g_bsp_module_stop[]
```

Detailed description

Module stop bit definitions for each ssp_ip_t enum value.

gp_mstp

```
volatile uint32_t* const gp_mstp[]
```

Detailed description

Module stop register addresses.

r_bsp_module_start_stop

```
ssp_err_t r_bsp_module_start_stop ( ssp_feature_t  const *const p_feature ,
  bool stop )
```

Brief description

Start module (cancel module stop) or stop module (enter module stop).

Detailed description

NOTE: Some module stop bits are shared between peripherals. Entering module stop is not permitted for these peripherals.

Table 1875:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1875:Parameters (Continued)

Name	Direction	Description
stop	in	true to stop module, false to start module

Table 1876:Return values

Name	Description
SSP_SUCCESS	Module is stopped or started based on stop parameter
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

Function steps

- The g_bsp_module_stop array must have entries for each ssp_ip_t enum value.
- Do nothing.
- Set MSTP bit if stop parameter is true and the bit is not shared with other channels.
- Clear MSTP bit if stop parameter is false.

R_BSP_ModuleStop

```
ssp_err_t R_BSP_ModuleStop ( ssp_feature_t  const *const p_feature )
```

Brief description

Stop module (enter module stop). Stopping a module disables clocks to the peripheral to save power.

Detailed description

NOTE: Some module stop bits are shared between peripherals. Modules with shared module stop bits cannot be stopped to prevent unintentionally stopping related modules.

Table 1877:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1878:Return values

Name	Description
SSP_SUCCESS	Module is stopped
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit, or module stop bit is shared and entering module stop is not supported because it could affect other modules.

R_BSP_ModuleStart

```
ssp_err_t R_BSP_ModuleStart ( ssp_feature_t  const *const p_feature )
```

Brief description

Start module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to be set.

Detailed description

Table 1879:Parameters

Name	Direction	Description
p_feature	in	Pointer to definition of the feature, defined by ssp_feature_t.

Table 1880:Return values

Name	Description
SSP_SUCCESS	Module is started
SSP_ERR_ASSERTION	p_feature::id is invalid
SSP_ERR_INVALID_ARGUMENT	Module has no module stop bit.

bsp_mstp_bit_t

bsp_mstp_bit_t

Detailed description

Definition of BSP module stop bit.

Variables

- uint8_t bit

  Which bit in MSTP register.

- uint8_t reg

  < Special processing required

  Which MSTP register

bsp_mstp_data_t

Detailed description

Definition of BSP module stop bit.

Variables

byte

bit

bit

reg

bit

11.2.10.7 ROM Registers

Defines MCU registers that are in ROM (e.g. OFS) and must be set at compile-time. All registers can be set using bsp_cfg.h.

Variables

- g_bsp_rom_registers

Defines

- #define BSP_ROM_REG_OFS1_SETTING

  Initial value:`(((uint32_t)BSP_CFG_ROM_REG_OFS1 & 0xFFFFF9FFU) | ((uint32_t)BSP_CFG_HOCO_FREQUENCY << 9))`

  OR in the HOCO frequency setting from bsp_clock_cfg.h with the OFS1 setting from bsp_cfg.h.

g_bsp_rom_registers

```
const uint32_t g_bsp_rom_registers[]
```

Detailed description

ROM registers defined here. Secure MPU functionality has been removed from the S7 UM so those registers are now set to 0xFFFFFFFF.

Initialized as

```
g_bsp_rom_registers=
{
    (uint32_t)BSP_CFG_ROM_REG_OFS0,
    (uint32_t)BSP_ROM_REG_OFS1_SETTING,
    0xFFFFFFFF,
    0xFFFFFFFF,
```

```
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
0xFFFFFFFF
}
```

11.3 Common BSP Code

Code common to all BSPs.

Implements functions that are common to all BSPs.

11.3.1 Functions

- R_BSP_VersionGet
- R_SSP_VersionGet
- bsp_init_internal

11.3.2 Defines

- #define BSP_IRQ_DISABLED

  Initial value:`(0xFFU)`

  Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file. Used to signify that an ELC event is not able to be used as an interrupt.
- #define BSP_CODE_VERSION_MAJOR

  Initial value:`(1U)`
- #define BSP_CODE_VERSION_MINOR

  Initial value:`(8U)`
- #define BSP_API_VERSION_MAJOR

  Initial value:`(1U)`
- #define BSP_API_VERSION_MINOR

  Initial value:`(0U)`

- #define SF_CONTEXT_SAVE

  Initial value:

- #define SF_CONTEXT_RESTORE

  Initial value:

- #define SSP_ASSERT_FAIL

  Initial value:

  Function call to insert before returning assertion error.

- #define SSP_ERROR_LOG

  Initial value:

  This function is called before returning an error code. To stop on a runtime error, define ssp_error_log in user code and do required debugging (breakpoints, stack dump, etc) in this function.

- #define SSP_ERROR_RETURN

  Initial value:
  ```
  { \ if ((a)) \ { \ (void) 0; /* Do nothing */ \ } \ else \ { \
  SSP_ERROR_LOG((err), (module), (version)); \ return (err); \ } \ }
  ```

  All SSP error codes are returned using this macro. Calls SSP_ERROR_LOG function if condition "a" is false. Used to identify runtime errors in SSP functions.

- #define SSP_CRITICAL_SECTION_DEFINE

  Initial value:`uint32_t old_mask_level = 0U`

  This check is performed to select suitable ASM API with respect to core This macro defines a variable for saving previous mask value

- #define SSP_CRITICAL_SECTION_ENTER

  Initial value:`old_mask_level = __get_PRIMASK(); \ __set_PRIMASK(1U)`

  This macro defined to get the mask value

- #define SSP_CRITICAL_SECTION_EXIT

  Initial value:`__set_PRIMASK(old_mask_level)`

  This macro defined to restore the old mask value

11.3.3 R_BSP_VersionGet

```
ssp_err_t R_BSP_VersionGet ( ssp_version_t  * p_version )
```

11.3.3.1 Brief description

Set BSP version based on compile time macros.

11.3.3.2 Detailed description

Table 1881:Parameters

Name	Direction	Description
p_version	out	Memory address to return version information to.

Table 1882:Return values

Name	Description
SSP_SUCCESS	Version information stored.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

11.3.4 R_SSP_VersionGet

```
ssp_err_t R_SSP_VersionGet ( ssp_pack_version_t  *const p_version )
```

11.3.4.1 Brief description

Set SSP version based on compile time macros.

11.3.4.2 Detailed description

Table 1883:Parameters

Name	Direction	Description
p_version	out	Memory address to return version information to.

Table 1884:Return values

Name	Description
SSP_SUCCESS	Version information stored.
SSP_ERR_ASSERTION	The parameter p_version is NULL.

11.3.5 bsp_init_internal

```
bsp_init_internal (  void * p_args )
```

11.3.5.1 Brief description

Default initialization function, used only if bsp_init is not defined in the user application.

11.3.6 Common BSP LED Code and Types

Common support for board LEDs.

Contains types and functions that allow for common use of LEDs on boards

11.3.6.1 Data structures

- bsp_leds_t

11.3.6.2 Functions

- R_BSP_LedsGet

11.3.6.3 R_BSP_LedsGet

```
ssp_err_t R_BSP_LedsGet ( bsp_leds_t  * p_leds )
```

Brief description

Return information about the LEDs on the current board.

Detailed description

Structure with LED information.

Table 1885:Parameters

Name	Direction	Description
p_leds	out	Pointer to structure where LED info is stored.

11.3.6.4 bsp_leds_t

bsp_leds_t

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.
Information on how many LEDs and what pins they are on.

Variables

- uint16_t led_count

  The number of LEDs on this board.

- ioport_port_pin_t const * p_leds

  Pointer to an array of IOPORT pins for controlling LEDs.

11.3.7 Compiler Support

The macros in this file are defined based upon which compiler is being used. The macros abstract common section names and gives a common way to place code in a particular section.

Description of macros:

- BSP_SECTION_STACK - Name of section where stack(s) are stored
- BSP_SECTION_HEAP - Name of section where heap(s) are stored
- BSP_SECTION_VECTOR - Name of section where vector table is stored
- BSP_SECTION_ROM_REGISTERS - Name of section where ROM registers are located
- BSP_PLACE_IN_SECTION - Macro for placing code in a particular section
- BSP_DONT_REMOVE - Keyword to tell linker/compiler to not optimize out a variable or function

NOTE: *Currently supported compilers are GCC and IAR*

11.3.8 Software Delay

Common function to implement a software delay.

Implements a software delay function for all BSPs.

11.3.8.1 Functions

- R_BSP_SoftwareDelay
- software_delay_loop

11.3.8.2 R_BSP_SoftwareDelay

```
R_BSP_SoftwareDelay (  uint32_t delay ,   bsp_delay_units_t units )
```

Brief description

Delay the specified duration in units and return.

Detailed description

Table 1886:Parameters

Name	Direction	Description
delay	in	The number of 'units' to delay.
units	in	The 'base' (bsp_delay_units_t) for the units specified. Valid values are: BSP_DELAY_UNITS_SECONDS, BSP_DELAY_UNITS_MILLISECONDS , BSP_DELAY_UNITS_MICROSECONDS. For example: At 1 MHz one cycle takes 1 microsecond (.000001 seconds). At 12 MHz one cycle takes 1/12 microsecond or 83 nanoseconds. Therefore one run through software_delay_loop() takes: ~ (83 * DELAY_LOOP_CYCLES) or 332 ns. A delay of 2 us therefore requires 2000ns/332ns or 6 loops.

The 'theoretical' maximum delay that may be obtained is determined by a full 32 bit loop count and the system clock rate. @240MHz: ((0xFFFFFFFF loops * 4 cycles /loop) / 240000000) = 71 seconds. @32MHz: ((0xFFFFFFFF loops * 4 cycles /loop) / 32000000) = 536 seconds

Note that requests for very large delays will be affected by rounding in the calculations and the actual delay achieved may be slightly less. @32 MHz, for example, a request for 532 seconds will be closer to 536 seconds.

Note also that if the calculations result in a loop_cnt of zero, the software_delay_loop() function is not called at all. In this case the requested delay is too small (nanoseconds) to be carried out by the loop itself, and the overhead associated with executing the code to just get to this point has certainly satisfied the requested delay.

NOTE: This function calls bsp_cpu_clock_get which ultimately calls R_CGC_SystemClockFreqGet and therefore requires that the BSP has already initialized the CGC (which it does as part of the Sysinit). Care should be taken to ensure this remains the case if in the future this function were to be called as part of the BSP initialization.

Table 1887:Return values

Name	Description
None.	

Function steps

- Convert the requested time to microseconds.
- Get the system clock frequency in Hz.
- Get the # of nanoseconds/cycle.
- Only delay if the supplied parameters constitute a delay.

11.3.8.3 software_delay_loop

```
software_delay_loop (  uint32_t loop_cnt )
```

Brief description

This assembly language routine takes roughly 4 cycles per loop. 2 additional cycles occur when the loop exits. The 'naked' attribute indicates that the specified function does not need prologue/epilogue sequences generated by the compiler.

Detailed description

Table 1888:Parameters

Name	Direction	Description
loop_cnt	in	The number of loops to iterate.

Table 1889:Return values

Name	Description
None.	

Function steps

- < 1 cycle
- < 2 cycles
- < 2 cycles
- loop_cnt is used but since it is used in assembly an unused parameter warning can be generated.

11.3.9 Error Checking

This file performs build time error checking where possible.

11.3.9.1 Defines

- #define BSP_STACK_ALIGNMENT

  Initial value:(8)

  Stacks (and heap) must be sized and aligned to an integer multiple of this number.

11.3.10 Module specific feature overrides

This group contains lookup functions that provide MCU specific feature information that is not available in the factory flash.

11.3.10.1 Functions

- R_BSP_FeatureSciGet
- R_BSP_FeatureRspiGet
- R_BSP_FeatureLvdGet
- R_BSP_FeatureAdcGet
- R_BSP_FeatureCtsuGet
- R_BSP_FeatureIoportGet
- R_BSP_FeatureCgcGet
- R_BSP_FeatureCanGet
- R_BSP_FeatureDacGet
- R_BSP_FeatureFlashLpGet
- R_BSP_FeatureSdhiGet
- R_BSP_FeatureSsiGet

11.3.10.2 Data structures

- bsp_feature_sci_t
- bsp_feature_rspi_t
- bsp_feature_lvd_t
- bsp_feature_adc_t
- bsp_feature_can_t
- bsp_feature_dac_t
- bsp_feature_flash_lp

- bsp_feature_flash_hp
- bsp_feature_ctsu_t
- bsp_feature_ioport_t
- bsp_feature_cgc_t
- bsp_feature_sdhi_t
- bsp_feature_ssi_t

11.3.10.3 R_BSP_FeatureSciGet

```
R_BSP_FeatureSciGet ( bsp_feature_sci_t  * p_sci_feature )
```

Detailed description

Get MCU specific SCI features.

Table 1890:Parameters

Name	Direction	Description
p_sci_feature	out	Pointer to structure to store SCI features.

11.3.10.4 R_BSP_FeatureRspiGet

```
R_BSP_FeatureRspiGet ( bsp_feature_rspi_t  * p_rspi_feature )
```

Detailed description

Get MCU specific RSPI features.

Table 1891:Parameters

Name	Direction	Description
p_rspi_feature	out	Pointer to structure to store RSPI features.

11.3.10.5 R_BSP_FeatureLvdGet

```
R_BSP_FeatureLvdGet ( bsp_feature_lvd_t  * p_lvd_feature )
```

Detailed description

Get MCU specific LVD features.

Table 1892:Parameters

Name	Direction	Description
p_lvd_feature	out	Pointer to structure to store LVD features.

11.3.10.6 R_BSP_FeatureAdcGet

```
R_BSP_FeatureAdcGet ( bsp_feature_adc_t  * p_adc_feature )
```

Detailed description

Get MCU specific ADC features

Table 1893:Parameters

Name	Direction	Description
p_adc_feature	out	Pointer to structure to store ADC features.

11.3.10.7 R_BSP_FeatureCtsuGet

```
R_BSP_FeatureCtsuGet ( bsp_feature_ctsu_t  * p_ctsu_feature )
```

Detailed description

Get MCU specific CTSU features

Table 1894:Parameters

Name	Direction	Description
p_ctsu_feature	out	Pointer to structure to store CTSU features.

11.3.10.8 R_BSP_FeatureIoportGet

```
R_BSP_FeatureIoportGet ( bsp_feature_ioport_t  * p_ioport_feature )
```

Detailed description

Get MCU specific I/O port features

Table 1895:Parameters

Name	Direction	Description
p_ioport_feature	out	Pointer to structure to store I/O port features.

11.3.10.9 R_BSP_FeatureCgcGet

```
R_BSP_FeatureCgcGet ( bsp_feature_cgc_t  const ** pp_cgc_feature )
```

Detailed description

Get MCU specific CGC features

Table 1896:Parameters

Name	Direction	Description
pp_cgc_feature	out	Pointer to structure to store pointer to CGC features.

11.3.10.10 R_BSP_FeatureCanGet

```
R_BSP_FeatureCanGet ( bsp_feature_can_t  * p_can_feature )
```

Detailed description

Get MCU specific CAN features

Table 1897:Parameters

Name	Direction	Description
p_can_feature	out	Pointer to structure to store CAN features.

11.3.10.11 R_BSP_FeatureDacGet

```
R_BSP_FeatureDacGet ( bsp_feature_dac_t  * p_dac_feature )
```

Detailed description

Get MCU specific DAC features

Table 1898:Parameters

Name	Direction	Description
p_dac_feature	out	Pointer to structure to store DAC features.

11.3.10.12 R_BSP_FeatureFlashLpGet

```
R_BSP_FeatureFlashLpGet ( bsp_feature_flash_lp  * p_flash_lp_feature )
```

Detailed description

Get MCU specific FLASH LP features

Table 1899:Parameters

Name	Direction	Description
p_flash_lp_feature	out	Pointer to structure to store Flash LP features.

Function steps

- S124 uses 1 macro of 128K and single access for Code Flash. It can therefore access 128K as a single macro and it's Code Flash memory is effectively organized as a single macro of 128K, yielding a total of 128K Code Flash.
- S128 uses 1 macro of 256K and single access for Code Flash. It can therefore access 256K as a single macro and it's Code Flash memory is effectively organized as a single macro of 256K, yielding a total of 256K Code Flash.
- S3A3 uses 2 macros of 256K and double access for Code Flash. It can therefore access 512K as a single macro and it's Code Flash memory is effectively organized as a single macro of 512K, yielding a total of 512K Code Flash.
- S3A3 uses 2 macros of 256K and double access for Code Flash. It can therefore access 512K as a single macro and it's Code Flash memory is effectively organized as a single macro of 512K, yielding a total of 512K Code Flash.
- S3A7 uses 4 macros of 256K and double access for Code Flash. It can therefore access 512K as a single macro and it's Code Flash memory is effectively organized as 2 macros of 512K each, yielding a total of 1MB Code Flash. This generates a macro boundary at 512K which is important for blank checking.

11.3.10.13 R_BSP_FeatureSdhiGet

```
R_BSP_FeatureSdhiGet ( bsp_feature_sdhi_t  * p_sdhi_feature )
```

Detailed description

Get MCU specific SDHI features

Table 1900:Parameters

Name	Direction	Description
p_sdhi_feature	out	Pointer to structure to store SDHI features.

11.3.10.14 R_BSP_FeatureSsiGet

```
R_BSP_FeatureSsiGet ( bsp_feature_ssi_t  * p_ssi_feature )
```

Detailed description

Get MCU specific SSI features

Table 1901:Parameters

Name	Direction	Description
p_ssi_feature	out	Pointer to structure to store SSI features.

11.3.10.15 bsp_feature_sci_t

bsp_feature_sci_t

Detailed description

SCI MCU specific features.

Variables

- uint8_t clock

  Which clock the SCI is connected to.

11.3.10.16 bsp_feature_rspi_t

bsp_feature_rspi_t

Detailed description

RSPI MCU specific features.

Variables

- uint8_t clock

  Which clock the RSPI is connected to.

- uint8_t has_ssl_level_keep

11.3.10.17 bsp_feature_lvd_t

bsp_feature_lvd_t

Detailed description

LVD MCU specific features.

Variables

- uint8_t monitor_1_low_threshold

  Monitor 1 lowest valid voltage threshold.

- uint8_t monitor_1_hi_threshold

  Monitor 1 highest valid voltage threshold.

- uint8_t monitor_2_low_threshold

  Monitor 2 lowest valid voltage threshold.

- uint8_t monitor_2_hi_threshold

  Monitor 2 highest valid voltage threshold.

- uint32_t has_digital_filter

  Whether or not LVD has a digital filter.

- uint32_t negation_delay_clock

  Clock required for LVD signal negation delay after reset.

11.3.10.18 bsp_feature_adc_t

bsp_feature_adc_t

Detailed description

ADC MCU specific features.

Variables

- uint8_t has_sample_hold_reg

  Whether or not sample and hold registers are present.

- uint8_t group_b_sensors_allowed

  Whether or not sensors are allowed on group b.

- uint8_t sensors_exclusive

  Whether or not sensors can be used with other sensors/channels.

- uint8_t tsn_calibration_available

  Identify if the TSN calibration data is available.

- uint8_t tsn_control_available

  Identify if the TSN control register is available.

- int16_t tsn_slope

  TSN slope in micro-volts/°C.
- uint32_t sensor_min_sampling_time

  The minimum sampling time required by the on-chip temperature and voltage sensor in nsec.
- uint32_t clock_source

  The conversion clock used by the ADC peripheral.

11.3.10.19 bsp_feature_can_t

bsp_feature_can_t

Detailed description

CAN MCU specific features.

Variables

- uint8_t mclock_only

  Whether or not MCLK is the only valid clock.
- uint8_t check_pclkb_ratio

  Whether clock:PCLKB must be 2:1.
- uint8_t clock

  Which clock to compare PCLKB to.

11.3.10.20 bsp_feature_dac_t

bsp_feature_dac_t

Detailed description

DAC MCU specific features.

Variables

- uint8_t has_davrefcr

  Whether or not DAC has DAVREFCR register.

11.3.10.21 bsp_feature_flash_lp

bsp_feature_flash_lp

Detailed description

FLASH LP MCU specific features.

Variables

- uint8_t flash_clock_src

  Source clock for Flash (ie. FCLK or ICLK)

- uint8_t flash_cf_macros

  Number of implemented code flash hardware macros.
- uint32_t cf_macro_size

  The size of the implemented Code Flash macro.

11.3.10.22 bsp_feature_flash_hp

bsp_feature_flash_hp

Detailed description

FLASH HP MCU specific features.

Variables

- uint8_t cf_block_size_write

  Code Flash Block size write.

11.3.10.23 bsp_feature_ctsu_t

bsp_feature_ctsu_t

Detailed description

CTSU MCU specific features.

Variables

- uint8_t ctsucr0_mask

  Mask of valid bits in CTSUCR0.
- uint8_t ctsucr1_mask

  Mask of valid bits in CTSUCR1.
- uint8_t ctsumch0_mask

  Mask of valid bits in CTSUMCH0.
- uint8_t ctsumch1_mask

  Mask of valid bits in CTSUMCH1.
- uint8_t ctsuchac_register_count

  Number of CTSUCHAC registers.
- uint8_t ctsuchtrc_register_count

  Number of CTSUCHTRC registers.

11.3.10.24 bsp_feature_ioport_t

bsp_feature_ioport_t

Detailed description

I/O port MCU specific features.

Variables

- uint8_t has_ethernet

  Whether or not MCU has Ethernet port configurations.

- uint8_t has_vbatt_pins

  Whether or not MCU has pins on vbatt domain.

11.3.10.25 bsp_feature_cgc_t

bsp_feature_cgc_t

Detailed description

CGC MCU specific features.

Variables

- uint32_t high_speed_freq_hz

  Frequency above which high speed mode must be used.

- uint32_t middle_speed_max_freq_hz

  Max frequency for middle speed, 0 indicates not available.

- uint32_t low_speed_max_freq_hz

  Max frequency for low speed, 0 indicates not available.

- uint32_t low_voltage_max_freq_hz

  Max frequency for low voltage, 0 indicates not available.

- uint32_t low_speed_pclk_div_min

  Minimum divisor for peripheral clocks when using oscillator stop detect.

- uint32_t low_voltage_pclk_div_min

  Minimum divisor for peripheral clocks when using oscillator stop detect.

- uint32_t hoco_freq_hz

  HOCO frequency.

- uint32_t main_osc_freq_hz

  Main oscillator frequency.

- uint8_t modrv_mask

  Mask for MODRV in MOMCR.

- uint8_t modrv_shift

  Offset of lowest bit of MODRV in MOMCR.

- uint8_t sodrv_mask

  Mask for SODRV in SOMCR.

- uint8_t sodrv_shift

  Offset of lowest bit of SODRV in SOMCR.
- uint8_t pll_div_max

  Maximum PLL divisor.
- uint8_t pll_mul_min

  Minimum PLL multiplier.
- uint8_t pll_mul_max

  Maximum PLL multiplier.
- uint8_t mainclock_drive

  Main clock drive capacity.
- uint32_t iclk_div

  ICLK divisor.
- uint32_t pllccr_type

  0: No PLL, 1: PLLCCR, 2: PLLCCR2
- uint32_t pll_src_configurable

  Whether or not PLL clock source is configurable.
- uint32_t has_subosc_speed

  Whether or not MCU has subosc speed mode.
- uint32_t has_lcd_clock

  Whether or not MCU has LCD clock.
- uint32_t has_sdram_clock

  Whether or not MCU has SDRAM clock.
- uint32_t has_usb_clock_div

  Whether or not MCU has USB clock divisor.
- uint32_t has_pclka

  Whether or not MCU has PCLKA clock.
- uint32_t has_pclkb

  Whether or not MCU has PCLKB clock.
- uint32_t has_pclkc

  Whether or not MCU has PCLKC clock.
- uint32_t has_pclkd

  Whether or not MCU has PCLKD clock.
- uint32_t has_fclk

  Whether or not MCU has FCLK clock.

- uint32_t has_bclk

  Whether or not MCU has BCLK clock.

11.3.10.26 bsp_feature_sdhi_t

bsp_feature_sdhi_t

Detailed description

SDHI MCU specific features.

Variables

- uint8_t has_card_detection

  Whether or not MCU has card detection.

- uint32_t max_clock_frequency

  Maximum clock rate supported by the peripheral.

11.3.10.27 bsp_feature_ssi_t

bsp_feature_ssi_t

Detailed description

SSI MCU specific features.

Variables

- uint8_t fifo_num_stages

  Number of FIFO stages on this MCU.

11.3.11 Grouped Interrupt Support

Support for grouped interrupts. Grouped interrupts occur when multiple interrupt events trigger the same interrupt vector. When this common vector is triggered the activation source must be discovered. The functions in this file allow users to register a callback function for a single interrupt source in an interrupt group.

11.3.11.1 Functions

- R_BSP_GroupIrqWrite
- NMI_Handler

11.3.11.2 R_BSP_GroupIrqWrite

```
ssp_err_t R_BSP_GroupIrqWrite (  bsp_grp_irq_t irq ,   void(*)(bsp_grp_irq_t
irq) p_callback )
```

Brief description

Register a callback function for supported interrupts. If NULL is passed for the callback argument then any previously registered callbacks are unregistered.

Detailed description

Table 1902:Parameters

Name	Direction	Description
irq	in	Interrupt for which to register a callback.
p_callback	in	Pointer to function to call when interrupt occurs.

Table 1903:Return values

Name	Description
SSP_SUCCESS	Callback registered

Function steps

- Check for valid address.
- Callback was NULL. De-register callback.
- Register callback.
- Check for valid address.
- Callback was NULL. De-register callback.
- Register callback.

11.3.11.3 NMI_Handler

```
NMI_Handler (  void  )
```

Brief description

Non-maskable interrupt handler. This exception is defined by the BSP, unlike other system exceptions, because there are many sources that map to the NMI exception.

Function steps

- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.

- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.

- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.

- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.

- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.

- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.
- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.
- Determine what is the cause of this interrupt.
- IWDT underflow/refresh error interrupt is requested.
- Clear IWDT flag.
- WDT underflow/refresh error interrupt is requested.
- Clear WDT flag.
- Voltage monitoring 1 interrupt is requested.
- Clear LVD1 flag.
- Voltage monitoring 2 interrupt is requested.
- Clear LVD2 flag.
- VBATT Monitor interrupt is requested.
- Clear VBATT flag.
- Oscillation stop detection interrupt is requested.
- Clear oscillation stop detect flag.
- NMI pin interrupt is requested.

- Clear NMI pin interrupt flag.
- RAM Parity Error interrupt is requested.
- Clear RAM parity error flag.
- RAM ECC Error interrupt is requested.
- Clear RAM ECC error flag.
- MPU Bus Slave Error interrupt is requested.
- Clear MPU Bus Slave error flag.
- MPU Bus Master Error interrupt is requested.
- Clear MPU Bus Master error flag.
- MPU Stack Error interrupt is requested.
- Clear MPU Stack error flag.

11.3.12 Interrupt Initialization

This module configures certain ELC events so that they can trigger NVIC interrupts. Which events are used as interrupts depends on settings in bsp_irq_cfg.h.

11.3.12.1 Functions

- R_BSP_IrqStatusClear
- bsp_irq_cfg

11.3.12.2 R_BSP_IrqStatusClear

```
R_BSP_IrqStatusClear (  IRQn_Type irq )
```

Brief description

Clear the interrupt status flag (IR) for a given interrupt. When an interrupt is triggered the IR bit is set. If it is not cleared in the ISR then the interrupt will trigger again immediately.

Detailed description

Table 1904:Parameters

Name	Direction	Description
irq	in	Interrupt for which to clear the IR bit. Note that the enums listed for IRQn_Type are only those for the Cortex Processor Exceptions Numbers. In prior releases this list contained all of the interrupts enabled for the specfic MCU but enabled interrupts are now configured using the SSP_VECTOR_DEFINE macro.

NOTE: This does not work for system exceptions where the IRQn_Type value is < 0.

11.3.12.3 bsp_irq_cfg

```
bsp_irq_cfg (  void  )
```

Brief description

Using the vector table information section that has been built by the linker and placed into ROM in the .vector_info. section, this function will initialize the ICU so that configured ELC events will trigger interrupts in the NVIC.

Detailed description

Common macro for SSP header files. There is also a corresponding SSP_FOOTER macro at the end of this file.

11.3.13 Atomic Locking

This module implements atomic locking mechanisms.

11.3.13.1 Functions

- R_BSP_SoftwareLock
- r_bsp_software_lock_cm0plus
- R_BSP_SoftwareUnlock
- R_BSP_HardwareLock
- R_BSP_HardwareUnlock
- bsp_init_hardware_locks
- R_BSP_SoftwareLockInit

11.3.13.2 Data structures

- bsp_lock_t

11.3.13.3 R_BSP_SoftwareLock

```
ssp_err_t R_BSP_SoftwareLock ( bsp_lock_t  * p_lock )
```

Brief description

Attempt to acquire the lock that has been sent in.

Detailed description

Table 1905:Parameters

Name	Direction	Description
p_lock	in	Pointer to the structure which contains the lock to be acquired.

Table 1906:Return values

Name	Description
SSP_SUCCESS	Lock was acquired
SSP_ERR_IN_USE	Lock was not acquired

11.3.13.4 r_bsp_software_lock_cm0plus

```
ssp_err_t r_bsp_software_lock_cm0plus ( bsp_lock_t  * p_lock )
```

Brief description

Attempt to acquire the lock that has been sent in (CM0+ implementation).

Detailed description

Table 1907:Parameters

Name	Direction	Description
p_lock	in	Pointer to the structure which contains the lock to be acquired.

Table 1908:Return values

Name	Description
SSP_SUCCESS	Lock was acquired

Table 1908:Return values (Continued)

Name	Description
SSP_ERR_IN_USE	Lock was not acquired

11.3.13.5 R_BSP_SoftwareUnlock

```
R_BSP_SoftwareUnlock ( bsp_lock_t  * p_lock )
```

Brief description

Release hold on lock.

Detailed description

Table 1909:Parameters

Name	Direction	Description
p_lock	in	Pointer to the structure which contains the lock to unlock.

11.3.13.6 R_BSP_HardwareLock

```
ssp_err_t R_BSP_HardwareLock ( ssp_feature_t  const *const p_feature )
```

Brief description

Attempt to reserve a hardware resource lock.

Detailed description

Table 1910:Parameters

Name	Direction	Description
p_feature	in	Pointer to the module specific feature information.

Table 1911:Return values

Name	Description
SSP_SUCCESS	Lock was acquired
SSP_ERR_IN_USE	Lock was not acquired

11.3.13.7 R_BSP_HardwareUnlock

```
R_BSP_HardwareUnlock ( ssp_feature_t  const *const p_feature )
```

Brief description

Release hold on lock.

Detailed description

Table 1912:Parameters

Name	Direction	Description
p_feature	in	Pointer to the module specific feature information.

11.3.13.8 bsp_init_hardware_locks

```
bsp_init_hardware_locks (  void  )
```

Brief description

Initialize all of the hardware locks to BSP_LOCK_UNLOCKED.

11.3.13.9 R_BSP_SoftwareLockInit

```
R_BSP_SoftwareLockInit ( bsp_lock_t  * p_lock )
```

Brief description

Initialize lock value to be unlocked.

Detailed description

Table 1913:Parameters

Name	Direction	Description
p_lock	in	Pointer to the structure which contains the lock to initialize.

11.3.13.10 bsp_lock_t

bsp_lock_t

Detailed description

Lock structure. Passed into software locking functions such as R_BSP_SoftwareLock and R_BSP_SoftwareUnLock.

Variables

- uint8_t lock

  A uint8_t is used instead of a enum because the size must be 8-bits.

11.3.14 Register Protection

Important registers are write protected. This module provides APIs for configuring the protection of these registers. Reference counters are used to ensure proper operation.

11.3.14.1 Functions

- R_BSP_RegisterProtectEnable
- R_BSP_RegisterProtectDisable
- bsp_register_protect_open

11.3.14.2 R_BSP_RegisterProtectEnable

```
R_BSP_RegisterProtectEnable (  bsp_reg_protect_t regs_to_protect )
```

Brief description

Enable register protection. Registers that are protected cannot be written to. Register protection is enabled by using the Protect Register (PRCR) and the MPC's Write-Protect Register (PWPR).

Detailed description

Table 1914:Parameters

Name	Direction	Description
regs_to_protect	in	Registers which have write protection enabled.

Function steps

- Get/save the current state of interrupts
- Enable protection using PRCR register.
- When writing to the PRCR register the upper 8-bits must be the correct key. Set lower bits to 0 to disable writes.
- Restore the interrupt state

11.3.14.3 R_BSP_RegisterProtectDisable

```
R_BSP_RegisterProtectDisable (  bsp_reg_protect_t regs_to_unprotect )
```

Brief description

Disable register protection. Registers that are protected cannot be written to. Register protection is disabled by using the Protect Register (PRCR) and the MPC's Write-Protect Register (PWPR).

Detailed description

Table 1915:Parameters

Name	Direction	Description
regs_to_unprotect	in	Registers which have write protection disabled.

Function steps

- Get/save the current state of interrupts
- Disable protection using PRCR register.
- When writing to the PRCR register the upper 8-bits must be the correct key. Set lower bits to 0 to disable writes.
- Increment the protect counter
- Restore the interrupt state

11.3.14.4 bsp_register_protect_open

```
bsp_register_protect_open (  void  )
```

Brief description

Initializes variables needed for register protection functionality.

Function steps

- Initialize reference counters to 0.

Chapter 12 API Reference: Structure Index

12.1 Structures

12.1.1 adc_api_t

```
typedef struct{
    ssp_err_t(*  open)(adc_ctrl_t *const p_ctrl, adc_cfg_t const *const p_cfg)
    ssp_err_t(*  scanCfg)(adc_ctrl_t *const p_ctrl, adc_channel_cfg_t const
*const p_channel_cfg)
    ssp_err_t(*  scanStart)(adc_ctrl_t *const p_ctrl)
    ssp_err_t(*  scanStop)(adc_ctrl_t *const p_ctrl)
    ssp_err_t(*  scanStatusGet)(adc_ctrl_t *const p_ctrl)
    ssp_err_t(*  read)(adc_ctrl_t *const p_ctrl, adc_register_t const reg_id,
adc_data_size_t *const p_data)
    ssp_err_t(*  sampleStateCountSet)(adc_ctrl_t *const p_ctrl,
adc_sample_state_t *p_sample)
    ssp_err_t(*  close)(adc_ctrl_t *const p_ctrl)
    ssp_err_t(*  infoGet)(adc_ctrl_t *const p_ctrl, adc_info_t *const p_adc_info)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} adc_api_t
```

12.1.2 adc_callback_args_t

```
typedef struct{
    uint16_t  unit
    adc_cb_event_t  event
    void const  *  p_context
} adc_callback_args_t
```

12.1.2.1 unit

`uint16_t adc_callback_args_t::unit`

Brief description

ADC device in use.

12.1.2.2 event

`adc_cb_event_t::event`

Brief description

ADC callback event.

12.1.2.3 p_context

```
void const* adc_callback_args_t::p_context
```

Brief description

Placeholder for user data.

12.1.3 adc_cfg_t

```
typedef struct{
    uint16_t  unit
    adc_mode_t  mode
    adc_resolution_t  resolution
    adc_alignment_t  alignment
    adc_add_t  add_average_count
    adc_clear_t  clearing
    adc_trigger_t  trigger
    adc_trigger_t  trigger_group_b
    uint8_t  scan_end_ipl
    uint8_t  scan_end_b_ipl
    void(*  p_callback)(adc_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} adc_cfg_t
```

12.1.3.1 unit

```
uint16_t adc_cfg_t::unit
```

Brief description

ADC Unit to be used.

12.1.3.2 mode

```
adc_mode_t::mode
```

Brief description

ADC operation mode.

12.1.3.3 resolution

```
adc_resolution_t::resolution
```

Brief description

ADC resolution 8, 10, or 12-bit.

12.1.3.4 alignment

`adc_alignment_t::alignment`

Brief description

Specify left or right alignment; ignored if addition used.

12.1.3.5 add_average_count

`adc_add_t::add_average_count`

Brief description

Add or average samples.

12.1.3.6 clearing

`adc_clear_t::clearing`

Brief description

Clear after read.

12.1.3.7 trigger

`adc_trigger_t::trigger`

Brief description

Default and Group A trigger source.

12.1.3.8 trigger_group_b

`adc_trigger_t::trigger_group_b`

Brief description

Group B trigger source; valid only for group mode.

12.1.3.9 scan_end_ipl

`uint8_t adc_cfg_t::scan_end_ipl`

Brief description

Scan end interrupt priority.

12.1.3.10 scan_end_b_ipl

`uint8_t adc_cfg_t::scan_end_b_ipl`

Brief description

Scan end group B interrupt priority.

12.1.3.11 p_callback

```
void(* adc_cfg_t::p_callback) (adc_callback_args_t *p_args)
```

Brief description

Callback function; set to NULL for none.

12.1.3.12 p_context

```
void const* adc_cfg_t::p_context
```

Brief description

Placeholder for user data. Passed to the user callback in adc_api_t::adc_callback_args_t.

12.1.3.13 p_extend

```
void const* adc_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.4 adc_channel_cfg_t

```
typedef struct{
    uint32_t  scan_mask
    uint32_t  scan_mask_group_b
    adc_group_a_t  priority_group_a
    uint32_t  add_mask
    uint8_t  sample_hold_mask
    uint8_t  sample_hold_states
} adc_channel_cfg_t
```

12.1.4.1 scan_mask

```
uint32_t adc_channel_cfg_t::scan_mask
```

Detailed description

Channels/bits: bit 0 is ch0; bit 15 is ch15. Use #define ADC_MASK_xxx from r_adc.h.

12.1.4.2 scan_mask_group_b

```
uint32_t adc_channel_cfg_t::scan_mask_group_b
```

Brief description

Valid for group modes. Use #define ADC_MASK_xxx from r_adc.h.

12.1.4.3 priority_group_a

`adc_group_a_t::priority_group_a`

Brief description

Valid for group modes.

12.1.4.4 add_mask

`uint32_t adc_channel_cfg_t::add_mask`

Brief description

Valid if add enabled in Open(). Use #define ADC_MASK_xxx from r_adc.h.

12.1.4.5 sample_hold_mask

`uint8_t adc_channel_cfg_t::sample_hold_mask`

Brief description

Channels/bits 0-2. Use #define ADC_MASK_xxx from r_adc.h.

12.1.4.6 sample_hold_states

`uint8_t adc_channel_cfg_t::sample_hold_states`

Brief description

Number of states to be used for sample and hold. Affects channels 0-2.

12.1.5 adc_info_t

```
typedef struct{
    __I uint16_t *  p_address
    uint32_t  length
    elc_peripheral_t  elc_peripheral
    elc_event_t  elc_event
    uint32_t  calibration_data
    int16_t  slope_microvolts
} adc_info_t
```

12.1.5.1 p_address

`__I uint16_t* adc_info_t::p_address`

Brief description

The address to start reading the data from.

12.1.5.2 length

`uint32_t adc_info_t::length`

Brief description

The total number of bytes to read.

12.1.5.3 elc_peripheral

`elc_peripheral_t::elc_peripheral`

Brief description

Name of the peripheral in the ELC list.

12.1.5.4 elc_event

`elc_event_t::elc_event`

Brief description

Name of the ELC event for the peripheral.

12.1.5.5 calibration_data

`uint32_t adc_info_t::calibration_data`

Detailed description

Temperature sensor calibration data (0xFFFFFFFF if unsupported). Refer to hardware manual for steps on using slope with calibration data to determine temperature

12.1.5.6 slope_microvolts

`int16_t adc_info_t::slope_microvolts`

Brief description

Temperature sensor slope in microvolts/°C.

12.1.6 adc_instance_ctrl_t

```
typedef struct{
    uint16_t  unit
    int16_t  slope_microvolts
    adc_mode_t  mode
    uint8_t  max_resolution
    uint8_t  pga_available
    uint8_t  tsn_ctrl_available
    uint8_t  tsn_calib_available
    R_TSN_Control_Type *  p_tsn_ctrl_regs
    R_TSN_Calibration_Type *  p_tsn_calib_regs
    void const  *  p_context
```

```
    void *  p_reg
    void(*  callback)(adc_callback_args_t *p_args)
    adc_trigger_t  trigger
    uint32_t  opened
    uint32_t  scan_mask
    IRQn_Type  scan_end_irq
    IRQn_Type  scan_end_b_irq
} adc_instance_ctrl_t
```

12.1.6.1 unit

`uint16_t ::unit`

Brief description

ADC Unit in use.

12.1.6.2 slope_microvolts

`int16_t ::slope_microvolts`

Brief description

Temperature sensor slope in microvolts/°C.

12.1.6.3 mode

`adc_mode_t::mode`

Brief description

operational mode

12.1.6.4 max_resolution

`uint8_t ::max_resolution`

Brief description

ADC max resolution: 8, 10, 12, or 14-bit.

12.1.6.5 pga_available

`uint8_t ::pga_available`

Brief description

PGA available or not on MCU.

12.1.6.6 tsn_ctrl_available

`uint8_t ::tsn_ctrl_available`

Brief description

Availability of TSN control register.

12.1.6.7 tsn_calib_available

```
uint8_t ::tsn_calib_available
```

Brief description

Availability of TSn calibration register.

12.1.6.8 p_tsn_ctrl_regs

```
R_TSN_Control_Type* ::p_tsn_ctrl_regs
```

Brief description

Pointer to temperature control register.

12.1.6.9 p_tsn_calib_regs

```
R_TSN_Calibration_Type* ::p_tsn_calib_regs
```

Brief description

Pointer to temperature calibration register.

12.1.6.10 p_context

```
void const* ::p_context
```

Brief description

Placeholder for user data.

12.1.6.11 p_reg

```
void* ::p_reg
```

Brief description

Base register for this unit.

12.1.6.12 callback

```
void(* ::callback) ( *p_args)
```

Brief description

User callback pointer.

12.1.6.13 trigger

```
adc_trigger_t::trigger
```

Brief description

Trigger defined for normal mode.

12.1.6.14 opened

`uint32_t ::opened`

Brief description

Boolean to verify that the Unit has been initialized.

12.1.6.15 scan_mask

`uint32_t ::scan_mask`

Brief description

Scan mask used for Normal scan.

12.1.6.16 scan_end_irq

`IRQn_Type ::scan_end_irq`

Brief description

Scan end IRQ number.

12.1.6.17 scan_end_b_irq

`IRQn_Type ::scan_end_b_irq`

Brief description

Scan end group B IRQ number.

12.1.7 adc_instance_t

```
typedef struct{
    adc_ctrl_t *  p_ctrl
    adc_cfg_t const  *  p_cfg
    adc_channel_cfg_t const  *  p_channel_cfg
    adc_api_t const  *  p_api
} adc_instance_t
```

12.1.7.1 p_ctrl

`adc_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.7.2 p_cfg

`adc_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.7.3 p_channel_cfg

`adc_channel_cfg_t::p_channel_cfg`

Brief description

Pointer to the channel configuration structure for this instance.

12.1.7.4 p_api

`adc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.8 adc_sample_state_t

```
typedef struct{
    adc_sample_state_reg_t  reg_id
    uint8_t  num_states
} adc_sample_state_t
```

12.1.8.1 reg_id

`adc_sample_state_reg_t::reg_id`

Brief description

Sample state register ID.

12.1.8.2 num_states

`uint8_t adc_sample_state_t::num_states`

Brief description

Number of sampling states for conversion. Ch16-20/21 use the same value.

12.1.9 aes_api_t

```
typedef struct{
    uint32_t(*  open)(aes_ctrl_t *const p_ctrl, aes_cfg_t const *const p_cfg)
    uint32_t(*  createKey)(aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t
*p_key)
```

```
    uint32_t(*  encrypt)(aes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  addAdditionalAuthenticationData)(aes_ctrl_t *const p_ctrl, const
uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source)
    uint32_t(*  encryptFinal)(aes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t input_num_words, uint32_t *p_source, uint32_t
output_num_words, uint32_t *p_dest)
    uint32_t(*  decrypt)(aes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t imaxcnt, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  setGcmTag)(aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t
*p_source)
    uint32_t(*  getGcmTag)(aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t
*p_dest)
    uint32_t(*  close)(aes_ctrl_t *const p_ctrl)
    uint32_t(*  zeroPaddingEncrypt)(aes_ctrl_t *const p_ctrl, const uint32_t
*p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  zeroPaddingDecrypt)(aes_ctrl_t *const p_ctrl, const uint32_t
*p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} aes_api_t
```

12.1.10 aes_cfg_t

```
typedef struct{
    crypto_api_t const  *  p_crypto_api
} aes_cfg_t
```

12.1.10.1 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine api

12.1.11 aes_ctrl_t

```
typedef struct{
    crypto_ctrl_t *  p_crypto_ctrl
    crypto_api_t const  *  p_crypto_api
    uint32_t  work_buffer[DRV_AES_CONTEXT_BUFFER_SIZE]
} aes_ctrl_t
```

12.1.11.1 p_crypto_ctrl

`crypto_ctrl_t::p_crypto_ctrl`

Brief description

pointer to crypto engine control structure

12.1.11.2 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine API

12.1.11.3 work_buffer

`uint32_t aes_ctrl_t::work_buffer[DRV_AES_CONTEXT_BUFFER_SIZE]`

Brief description

Examples: AES-GCM mode uses this for storing authentication tag etc.

Detailed description

used for storing context/state of the Cipher

12.1.12 aes_instance_t

```
typedef struct{
    aes_ctrl_t *  p_ctrl
    aes_cfg_t const  *  p_cfg
    aes_api_t const  *  p_api
} aes_instance_t
```

12.1.12.1 p_ctrl

`aes_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.12.2 p_cfg

`aes_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.12.3 p_api

`aes_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.13 agt_instance_ctrl_t

```
typedef struct{
    void(*  p_callback)(timer_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    uint32_t  open
    uint16_t  period
    uint8_t  channel
    IRQn_Type  irq
    timer_mode_t  mode
} agt_instance_ctrl_t
```

12.1.13.1 p_callback

```
void(* ::p_callback) ( *p_args)
```

Detailed description

Callback provided when a timer ISR occurs. NULL indicates no CPU interrupt.

12.1.13.2 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in timer_callback_args_t.

12.1.13.3 p_reg

```
void* ::p_reg
```

Brief description

Base register for this channel.

12.1.13.4 open

```
uint32_t ::open
```

Brief description

Whether or not channel is open.

12.1.13.5 period

```
uint16_t ::period
```

Brief description

Current timer period (counts)

12.1.13.6 channel

`uint8_t ::channel`

Brief description

Channel number.

12.1.13.7 irq

`IRQn_Type ::irq`

Brief description

Counter overflow IRQ number.

12.1.13.8 mode

`timer_mode_t::mode`

Brief description

Timer mode.

12.1.14 bsp_feature_adc_t

```
typedef struct{
    uint8_t  has_sample_hold_reg
    uint8_t  group_b_sensors_allowed
    uint8_t  sensors_exclusive
    uint8_t  tsn_calibration_available
    uint8_t  tsn_control_available
    int16_t  tsn_slope
    uint32_t  sensor_min_sampling_time
    uint32_t  clock_source
} bsp_feature_adc_t
```

12.1.14.1 has_sample_hold_reg

`uint8_t ::has_sample_hold_reg`

Brief description

Whether or not sample and hold registers are present.

12.1.14.2 group_b_sensors_allowed

`uint8_t ::group_b_sensors_allowed`

Brief description

Whether or not sensors are allowed on group b.

12.1.14.3 sensors_exclusive

```
uint8_t ::sensors_exclusive
```

Brief description

Whether or not sensors can be used with other sensors/channels.

12.1.14.4 tsn_calibration_available

```
uint8_t ::tsn_calibration_available
```

Brief description

Identify if the TSN calibration data is available.

12.1.14.5 tsn_control_available

```
uint8_t ::tsn_control_available
```

Brief description

Identify if the TSN control register is available.

12.1.14.6 tsn_slope

```
int16_t ::tsn_slope
```

Brief description

TSN slope in micro-volts/°C.

12.1.14.7 sensor_min_sampling_time

```
uint32_t ::sensor_min_sampling_time
```

Brief description

The minimum sampling time required by the on-chip temperature and voltage sensor in nsec.

12.1.14.8 clock_source

```
uint32_t ::clock_source
```

Brief description

The conversion clock used by the ADC peripheral.

12.1.15 bsp_feature_can_t

```
typedef struct{
    uint8_t  mclock_only
    uint8_t  check_pclkb_ratio
```

```
    uint8_t  clock
} bsp_feature_can_t
```

12.1.15.1 mclock_only

`uint8_t ::mclock_only`

Brief description

Whether or not MCLK is the only valid clock.

12.1.15.2 check_pclkb_ratio

`uint8_t ::check_pclkb_ratio`

Brief description

Whether clock:PCLKB must be 2:1.

12.1.15.3 clock

`uint8_t ::clock`

Brief description

Which clock to compare PCLKB to.

12.1.16 bsp_feature_cgc_t

```
typedef struct{
    uint32_t  high_speed_freq_hz
    uint32_t  middle_speed_max_freq_hz
    uint32_t  low_speed_max_freq_hz
    uint32_t  low_voltage_max_freq_hz
    uint32_t  low_speed_pclk_div_min
    uint32_t  low_voltage_pclk_div_min
    uint32_t  hoco_freq_hz
    uint32_t  main_osc_freq_hz
    uint8_t  modrv_mask
    uint8_t  modrv_shift
    uint8_t  sodrv_mask
    uint8_t  sodrv_shift
    uint8_t  pll_div_max
    uint8_t  pll_mul_min
    uint8_t  pll_mul_max
    uint8_t  mainclock_drive
    uint32_t  iclk_div
    uint32_t  pllccr_type
    uint32_t  pll_src_configurable
    uint32_t  has_subosc_speed
    uint32_t  has_lcd_clock
```

```
    uint32_t  has_sdram_clock
    uint32_t  has_usb_clock_div
    uint32_t  has_pclka
    uint32_t  has_pclkb
    uint32_t  has_pclkc
    uint32_t  has_pclkd
    uint32_t  has_fclk
    uint32_t  has_bclk
} bsp_feature_cgc_t
```

12.1.16.1 high_speed_freq_hz

`uint32_t ::high_speed_freq_hz`

Brief description

Frequency above which high speed mode must be used.

12.1.16.2 middle_speed_max_freq_hz

`uint32_t ::middle_speed_max_freq_hz`

Brief description

Max frequency for middle speed, 0 indicates not available.

12.1.16.3 low_speed_max_freq_hz

`uint32_t ::low_speed_max_freq_hz`

Brief description

Max frequency for low speed, 0 indicates not available.

12.1.16.4 low_voltage_max_freq_hz

`uint32_t ::low_voltage_max_freq_hz`

Brief description

Max frequency for low voltage, 0 indicates not available.

12.1.16.5 low_speed_pclk_div_min

`uint32_t ::low_speed_pclk_div_min`

Brief description

Minimum divisor for peripheral clocks when using oscillator stop detect.

12.1.16.6 low_voltage_pclk_div_min

`uint32_t ::low_voltage_pclk_div_min`

Brief description

Minimum divisor for peripheral clocks when using oscillator stop detect.

12.1.16.7 hoco_freq_hz

```
uint32_t ::hoco_freq_hz
```

Brief description

HOCO frequency.

12.1.16.8 main_osc_freq_hz

```
uint32_t ::main_osc_freq_hz
```

Brief description

Main oscillator frequency.

12.1.16.9 modrv_mask

```
uint8_t ::modrv_mask
```

Brief description

Mask for MODRV in MOMCR.

12.1.16.10 modrv_shift

```
uint8_t ::modrv_shift
```

Brief description

Offset of lowest bit of MODRV in MOMCR.

12.1.16.11 sodrv_mask

```
uint8_t ::sodrv_mask
```

Brief description

Mask for SODRV in SOMCR.

12.1.16.12 sodrv_shift

```
uint8_t ::sodrv_shift
```

Brief description

Offset of lowest bit of SODRV in SOMCR.

12.1.16.13 pll_div_max

```
uint8_t ::pll_div_max
```

Brief description

Maximum PLL divisor.

12.1.16.14 pll_mul_min

```
uint8_t ::pll_mul_min
```

Brief description

Minimum PLL multiplier.

12.1.16.15 pll_mul_max

```
uint8_t ::pll_mul_max
```

Brief description

Maximum PLL multiplier.

12.1.16.16 mainclock_drive

```
uint8_t ::mainclock_drive
```

Brief description

Main clock drive capacity.

12.1.16.17 iclk_div

```
uint32_t ::iclk_div
```

Brief description

ICLK divisor.

12.1.16.18 pllccr_type

```
uint32_t ::pllccr_type
```

Brief description

0: No PLL, 1: PLLCCR, 2: PLLCCR2

12.1.16.19 pll_src_configurable

```
uint32_t ::pll_src_configurable
```

Brief description

Whether or not PLL clock source is configurable.

12.1.16.20 has_subosc_speed

```
uint32_t ::has_subosc_speed
```

Brief description

Whether or not MCU has subosc speed mode.

12.1.16.21 has_lcd_clock

```
uint32_t ::has_lcd_clock
```

Brief description

Whether or not MCU has LCD clock.

12.1.16.22 has_sdram_clock

```
uint32_t ::has_sdram_clock
```

Brief description

Whether or not MCU has SDRAM clock.

12.1.16.23 has_usb_clock_div

```
uint32_t ::has_usb_clock_div
```

Brief description

Whether or not MCU has USB clock divisor.

12.1.16.24 has_pclka

```
uint32_t ::has_pclka
```

Brief description

Whether or not MCU has PCLKA clock.

12.1.16.25 has_pclkb

```
uint32_t ::has_pclkb
```

Brief description

Whether or not MCU has PCLKB clock.

12.1.16.26 has_pclkc

```
uint32_t ::has_pclkc
```

Brief description

Whether or not MCU has PCLKC clock.

12.1.16.27 has_pclkd

```
uint32_t ::has_pclkd
```

Brief description

Whether or not MCU has PCLKD clock.

12.1.16.28 has_fclk

```
uint32_t ::has_fclk
```

Brief description

Whether or not MCU has FCLK clock.

12.1.16.29 has_bclk

```
uint32_t ::has_bclk
```

Brief description

Whether or not MCU has BCLK clock.

12.1.17 bsp_feature_ctsu_t

```
typedef struct{
    uint8_t  ctsucr0_mask
    uint8_t  ctsucr1_mask
    uint8_t  ctsumch0_mask
    uint8_t  ctsumch1_mask
    uint8_t  ctsuchac_register_count
    uint8_t  ctsuchtrc_register_count
} bsp_feature_ctsu_t
```

12.1.17.1 ctsucr0_mask

```
uint8_t ::ctsucr0_mask
```

Brief description

Mask of valid bits in CTSUCR0.

12.1.17.2 ctsucr1_mask

```
uint8_t ::ctsucr1_mask
```

Brief description

Mask of valid bits in CTSUCR1.

12.1.17.3 ctsumch0_mask

```
uint8_t ::ctsumch0_mask
```

Brief description

Mask of valid bits in CTSUMCH0.

12.1.17.4 ctsumch1_mask

`uint8_t ::ctsumch1_mask`

Brief description

Mask of valid bits in CTSUMCH1.

12.1.17.5 ctsuchac_register_count

`uint8_t ::ctsuchac_register_count`

Brief description

Number of CTSUCHAC registers.

12.1.17.6 ctsuchtrc_register_count

`uint8_t ::ctsuchtrc_register_count`

Brief description

Number of CTSUCHTRC registers.

12.1.18 bsp_feature_dac_t

```
typedef struct{
    uint8_t  has_davrefcr
} bsp_feature_dac_t
```

12.1.18.1 has_davrefcr

`uint8_t ::has_davrefcr`

Brief description

Whether or not DAC has DAVREFCR register.

12.1.19 bsp_feature_flash_hp

```
typedef struct{
    uint8_t  cf_block_size_write
} bsp_feature_flash_hp
```

12.1.19.1 cf_block_size_write

`uint8_t bsp_feature_flash_hp::cf_block_size_write`

Brief description

Code Flash Block size write.

12.1.20 bsp_feature_flash_lp

```
typedef struct{
    uint8_t  flash_clock_src
    uint8_t  flash_cf_macros
    uint32_t  cf_macro_size
} bsp_feature_flash_lp
```

12.1.20.1 flash_clock_src

```
uint8_t bsp_feature_flash_lp::flash_clock_src
```

Brief description

Source clock for Flash (ie. FCLK or ICLK)

12.1.20.2 flash_cf_macros

```
uint8_t bsp_feature_flash_lp::flash_cf_macros
```

Brief description

Number of implemented code flash hardware macros.

12.1.20.3 cf_macro_size

```
uint32_t bsp_feature_flash_lp::cf_macro_size
```

Brief description

The size of the implemented Code Flash macro.

12.1.21 bsp_feature_ioport_t

```
typedef struct{
    uint8_t  has_ethernet
    uint8_t  has_vbatt_pins
} bsp_feature_ioport_t
```

12.1.21.1 has_ethernet

```
uint8_t ::has_ethernet
```

Brief description

Whether or not MCU has Ethernet port configurations.

12.1.21.2 has_vbatt_pins

```
uint8_t ::has_vbatt_pins
```

Brief description

Whether or not MCU has pins on vbatt domain.

12.1.22 bsp_feature_lvd_t

```
typedef struct{
    uint8_t  monitor_1_low_threshold
    uint8_t  monitor_1_hi_threshold
    uint8_t  monitor_2_low_threshold
    uint8_t  monitor_2_hi_threshold
    uint32_t  has_digital_filter
    uint32_t  negation_delay_clock
} bsp_feature_lvd_t
```

12.1.22.1 monitor_1_low_threshold

```
uint8_t ::monitor_1_low_threshold
```

Brief description

Monitor 1 lowest valid voltage threshold.

12.1.22.2 monitor_1_hi_threshold

```
uint8_t ::monitor_1_hi_threshold
```

Brief description

Monitor 1 highest valid voltage threshold.

12.1.22.3 monitor_2_low_threshold

```
uint8_t ::monitor_2_low_threshold
```

Brief description

Monitor 2 lowest valid voltage threshold.

12.1.22.4 monitor_2_hi_threshold

```
uint8_t ::monitor_2_hi_threshold
```

Brief description

Monitor 2 highest valid voltage threshold.

12.1.22.5 has_digital_filter

```
uint32_t ::has_digital_filter
```

Brief description

Whether or not LVD has a digital filter.

12.1.22.6 negation_delay_clock

```
uint32_t ::negation_delay_clock
```

Brief description

Clock required for LVD signal negation delay after reset.

12.1.23 bsp_feature_rspi_t

```
typedef struct{
    uint8_t  clock
    uint8_t  has_ssl_level_keep
} bsp_feature_rspi_t
```

12.1.23.1 clock

```
uint8_t ::clock
```

Brief description

Which clock the RSPI is connected to.

12.1.23.2 has_ssl_level_keep

```
uint8_t ::has_ssl_level_keep
```

12.1.24 bsp_feature_sci_t

```
typedef struct{
    uint8_t  clock
} bsp_feature_sci_t
```

12.1.24.1 clock

```
uint8_t ::clock
```

Brief description

Which clock the SCI is connected to.

12.1.25 bsp_feature_sdhi_t

```
typedef struct{
    uint8_t  has_card_detection
    uint32_t  max_clock_frequency
} bsp_feature_sdhi_t
```

12.1.25.1 has_card_detection

`uint8_t ::has_card_detection`

Brief description

Whether or not MCU has card detection.

12.1.25.2 max_clock_frequency

`uint32_t ::max_clock_frequency`

Brief description

Maximum clock rate supported by the peripheral.

12.1.26 bsp_feature_ssi_t

```
typedef struct{
    uint8_t  fifo_num_stages
} bsp_feature_ssi_t
```

12.1.26.1 fifo_num_stages

`uint8_t ::fifo_num_stages`

Brief description

Number of FIFO stages on this MCU.

12.1.27 bsp_leds_t

```
typedef struct{
    uint16_t  led_count
    ioport_port_pin_t const  *  p_leds
} bsp_leds_t
```

12.1.27.1 led_count

`uint16_t ::led_count`

Brief description

The number of LEDs on this board.

12.1.27.2 p_leds

`ioport_port_pin_t::p_leds`

Brief description

Pointer to an array of IOPORT pins for controlling LEDs.

12.1.28 bsp_lock_t

```
typedef struct{
    uint8_t  lock
} bsp_lock_t
```

12.1.28.1 lock

`uint8_t ::lock`

Brief description

A uint8_t is used instead of a enum because the size must be 8-bits.

12.1.29 bsp_mstp_bit_t

```
typedef struct{
    uint8_t  bit
    uint8_t  reg
} bsp_mstp_bit_t
```

12.1.29.1 bit

`uint8_t ::bit`

Brief description

Which bit in MSTP register.

12.1.29.2 reg

`uint8_t ::reg`

Brief description

< Special processing required

Detailed description

Which MSTP register

12.1.30 bsp_mstp_data_t

```
typedef struct{
    uint8_t  byte
    bsp_mstp_bit_t  bit
    uint8_t  bit
    uint8_t  reg
    struct{}                bit
} bsp_mstp_data_t
```

12.1.30.1 byte

`uint8_t ::byte`

Brief description

Byte access for comparison.

12.1.30.2 bit

`::bit`

12.1.30.3 bit

`uint8_t ::bit`

Brief description

Which bit in MSTP register.

12.1.30.4 reg

`uint8_t ::reg`

Brief description

< Special processing required

Detailed description

Which MSTP register

12.1.30.5 bit

`See source code for the definition of this member.`

12.1.31 cac_api_t

```
typedef struct{
    ssp_err_t(*  open)(cac_ctrl_t *const p_ctrl, cac_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(cac_ctrl_t *const p_ctrl, uint8_t *const p_status,
```

```
uint16_t *const p_counter)
    ssp_err_t(*  close)(cac_ctrl_t *const p_ctrl)
    ssp_err_t(*  stopMeasurement)(cac_ctrl_t *const p_ctrl)
    ssp_err_t(*  startMeasurement)(cac_ctrl_t *const p_ctrl)
    ssp_err_t(*  reset)(cac_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} cac_api_t
```

12.1.32 cac_callback_args_t

```
typedef struct{
    cac_event_t  event
    void const  *  p_context
} cac_callback_args_t
```

12.1.32.1 event

`cac_event_t::event`

Brief description

The event can be used to identify what caused the callback (cac ready or error).

12.1.32.2 p_context

`void const* cac_callback_args_t::p_context`

Brief description

Placeholder for user data. Set in open function in cac_cfg_t.

12.1.33 cac_cfg_t

```
typedef struct{
    cac_ref_clock_config_t  cac_ref_clock
    cac_meas_clock_config_t  cac_meas_clock
    uint16_t  cac_upper_limit
    uint16_t  cac_lower_limit
    bool  mei_interrupt_enabled
    bool  ovf_interrupt_enabled
    bool  ferr_interrupt_enabled
    bool  continuous_mode
    uint8_t  frequency_error_ipl
    uint8_t  measurement_end_ipl
    uint8_t  overflow_ipl
    void(*  p_callback)(cac_callback_args_t *p_args)
    void const  *  p_extend
    void const  *  p_context
} cac_cfg_t
```

12.1.33.1 cac_ref_clock

`cac_ref_clock_config_t::cac_ref_clock`

Brief description

reference clock specific settings

12.1.33.2 cac_meas_clock

`cac_meas_clock_config_t::cac_meas_clock`

Brief description

measurement clock specific settings

12.1.33.3 cac_upper_limit

`uint16_t cac_cfg_t::cac_upper_limit`

Brief description

the upper limit counter threshold

12.1.33.4 cac_lower_limit

`uint16_t cac_cfg_t::cac_lower_limit`

Brief description

the lower limit counter threshold

12.1.33.5 mei_interrupt_enabled

`bool cac_cfg_t::mei_interrupt_enabled`

Brief description

True if Measurement Complete interrupt is enabled.

12.1.33.6 ovf_interrupt_enabled

`bool cac_cfg_t::ovf_interrupt_enabled`

Brief description

True if Overflow interrupt is enabled.

12.1.33.7 ferr_interrupt_enabled

`bool cac_cfg_t::ferr_interrupt_enabled`

Brief description

True if Frequency Error interrupt is enabled.

12.1.33.8 continuous_mode

```
bool cac_cfg_t::continuous_mode
```

Brief description

True if measurement continuously restarts after completing.

12.1.33.9 frequency_error_ipl

```
uint8_t cac_cfg_t::frequency_error_ipl
```

Brief description

Frequency error interrupt priority.

12.1.33.10 measurement_end_ipl

```
uint8_t cac_cfg_t::measurement_end_ipl
```

Brief description

Measurement end interrupt priority.

12.1.33.11 overflow_ipl

```
uint8_t cac_cfg_t::overflow_ipl
```

Brief description

Overflow interrupt priority.

12.1.33.12 p_callback

```
void(* cac_cfg_t::p_callback) (cac_callback_args_t *p_args)
```

Brief description

Callback provided when a CAC interrupt ISR occurs.

12.1.33.13 p_extend

```
void const* cac_cfg_t::p_extend
```

Brief description

CAC hardware dependent configuration */.

12.1.33.14 p_context

```
void const* cac_cfg_t::p_context
```

Brief description

Placeholder for user data. Passed to user callback in cac_callback_args_t.

12.1.34 cac_instance_ctrl_t

```
typedef struct{
    void *  p_reg
    void(*  p_callback)(cac_callback_args_t *cb_data)
    void const  *  p_context
    IRQn_Type  frequency_error_irq
    IRQn_Type  measurement_end_irq
    IRQn_Type  overflow_irq
    uint32_t  cac_api_open
    bool  cac_continous_mode
    bsp_lock_t  cac_lock
    cac_clock_source_t  measurement_clock
    cac_clock_source_t  reference_clock
    void const  *  p_extend
} cac_instance_ctrl_t
```

12.1.34.1 p_reg

```
void* ::p_reg
```

Brief description

Pointer to register base address.

12.1.34.2 p_callback

```
void(* ::p_callback) ( *cb_data)
```

Brief description

Called from the ISR.

12.1.34.3 p_context

```
void const* ::p_context
```

Brief description

Passed to the callback.

12.1.34.4 frequency_error_irq

```
IRQn_Type ::frequency_error_irq
```

Brief description

Frequency error IRQ number.

12.1.34.5 measurement_end_irq

```
IRQn_Type ::measurement_end_irq
```

Brief description

Measurement end IRQ number.

12.1.34.6 overflow_irq

`IRQn_Type ::overflow_irq`

Brief description

Overflow IRQ number.

12.1.34.7 cac_api_open

`uint32_t ::cac_api_open`

Brief description

Set to "CAC" once API has been successfully opened.

12.1.34.8 cac_continous_mode

`bool ::cac_continous_mode`

Brief description

Set as a result of the Open() call.

12.1.34.9 cac_lock

`bsp_lock_t::cac_lock`

Brief description

CAC commands software lock.

12.1.34.10 measurement_clock

`cac_clock_source_t::measurement_clock`

Brief description

Clock specified in Open() as the measurement clock.

12.1.34.11 reference_clock

`cac_clock_source_t::reference_clock`

Brief description

Clock specified in Open() as the reference clock.

12.1.34.12 p_extend

`void const* ::p_extend`

12.1.35 cac_instance_t

```
typedef struct{
    cac_ctrl_t *  p_ctrl
    cac_cfg_t const  *  p_cfg
    cac_api_t const  *  p_api
} cac_instance_t
```

12.1.35.1 p_ctrl

cac_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.35.2 p_cfg

cac_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.35.3 p_api

cac_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.36 cac_meas_clock_config_t

```
typedef struct{
    cac_meas_divider_t  divider
    cac_clock_source_t  clock
} cac_meas_clock_config_t
```

12.1.36.1 divider

cac_meas_divider_t::divider

Brief description

Divider specification for the Measurement clock.

12.1.36.2 clock

cac_clock_source_t::clock

Brief description

Clock source for the Measurement clock.

12.1.37 cac_ref_clock_config_t

```
typedef struct{
    cac_ref_divider_t  divider
    cac_clock_source_t  clock
    cac_ref_digfilter_t  digfilter
    cac_ref_edge_t  edge
} cac_ref_clock_config_t
```

12.1.37.1 divider

`cac_ref_divider_t::divider`

Brief description

Divider specification for the Reference clock.

12.1.37.2 clock

`cac_clock_source_t::clock`

Brief description

Clock source for the Reference clock.

12.1.37.3 digfilter

`cac_ref_digfilter_t::digfilter`

Brief description

Digital filter selection for the CACREF ext clock.

12.1.37.4 edge

`cac_ref_edge_t::edge`

Brief description

Edge detection for the Reference clock.

12.1.38 can_api_t

```
typedef struct{
    ssp_err_t(*  open)(can_ctrl_t *const p_ctrl, can_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(can_ctrl_t *const p_ctrl, uint32_t mailbox, can_frame_t
*const p_frame)
    ssp_err_t(*  write)(can_ctrl_t *const p_ctrl, uint32_t mailbox, can_frame_t
*const p_frame)
```

```
    ssp_err_t(*  close)(can_ctrl_t *const p_ctrl)
    ssp_err_t(*  control)(can_ctrl_t *const p_ctrl, can_command_t const command,
void *p_data)
    ssp_err_t(*  infoGet)(can_ctrl_t *const p_ctrl, can_info_t *const p_info)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} can_api_t
```

12.1.39 can_bit_timing_cfg_t

```
typedef struct{
    uint32_t  baud_rate_prescaler
    can_time_segment1_t  time_segment_1
    can_time_segment2_t  time_segment_2
    can_sync_jump_width_t  synchronization_jump_width
} can_bit_timing_cfg_t
```

12.1.39.1 baud_rate_prescaler

`uint32_t can_bit_timing_cfg_t::baud_rate_prescaler`

Brief description

Baud rate prescaler. Valid values: 1 - 1024.

12.1.39.2 time_segment_1

`can_time_segment1_t::time_segment_1`

Brief description

Time segment 1 control.

12.1.39.3 time_segment_2

`can_time_segment2_t::time_segment_2`

Brief description

Time segment 2 control.

12.1.39.4 synchronization_jump_width

`can_sync_jump_width_t::synchronization_jump_width`

Brief description

Synchronization jump width.

12.1.40 can_callback_args_t

```
typedef struct{
    uint32_t  channel
    can_event_t  event
    uint32_t  mailbox
    void const  *  p_context
} can_callback_args_t
```

12.1.40.1 channel

`uint32_t can_callback_args_t::channel`

Brief description

Device channel number.

12.1.40.2 event

`can_event_t::event`

Brief description

Event code.

12.1.40.3 mailbox

`uint32_t can_callback_args_t::mailbox`

Brief description

Mailbox number of interrupt source.

12.1.40.4 p_context

`void const* can_callback_args_t::p_context`

Brief description

Context provided to user during callback.

12.1.41 can_cfg_t

```
typedef struct{
    uint32_t  channel
    can_bit_timing_cfg_t *  p_bit_timing
    can_id_mode_t  id_mode
    uint32_t  mailbox_count
    can_mailbox_t *  p_mailbox
    can_message_mode_t  message_mode
    void(*  p_callback)(can_callback_args_t *p_args)
    void const  *  p_context
```

```
    void const  *  p_extend
    uint8_t  error_ipl
    uint8_t  mailbox_rx_ipl
    uint8_t  mailbox_tx_ipl
} can_cfg_t
```

12.1.41.1 channel

`uint32_t can_cfg_t::channel`

Brief description

CAN channel.

12.1.41.2 p_bit_timing

`can_bit_timing_cfg_t::p_bit_timing`

Brief description

CAN bit timing.

12.1.41.3 id_mode

`can_id_mode_t::id_mode`

Brief description

Standard or Extended ID mode.

12.1.41.4 mailbox_count

`uint32_t can_cfg_t::mailbox_count`

Brief description

Number of mailboxes.

12.1.41.5 p_mailbox

`can_mailbox_t::p_mailbox`

Brief description

Pointer to mailboxes.

12.1.41.6 message_mode

`can_message_mode_t::message_mode`

Brief description

Overwrite message or overrun.

12.1.41.7 p_callback

```
void(* can_cfg_t::p_callback) (can_callback_args_t *p_args)
```

Brief description

Pointer to callback function.

12.1.41.8 p_context

```
void const* can_cfg_t::p_context
```

Brief description

User defined callback context.

12.1.41.9 p_extend

```
void const* can_cfg_t::p_extend
```

Brief description

CAN hardware dependent configuration.

12.1.41.10 error_ipl

```
uint8_t can_cfg_t::error_ipl
```

Brief description

Error interrupt priority.

12.1.41.11 mailbox_rx_ipl

```
uint8_t can_cfg_t::mailbox_rx_ipl
```

Brief description

Receive interrupt priority.

12.1.41.12 mailbox_tx_ipl

```
uint8_t can_cfg_t::mailbox_tx_ipl
```

Brief description

Transmit interrupt priority.

12.1.42 can_error_t

```
typedef struct{
    uint32_t  error
    struct can_error_t::st_error_b  error_b
} can_error_t
```

12.1.42.1 error

```
uint32_t can_error_t::error
```

12.1.42.2 error_b

```
can_error_t::st_error_b::error_b
```

12.1.43 can_extended_cfg_t

```
typedef struct{
    can_clock_source_t  clock_source
    uint32_t *  p_mailbox_mask
} can_extended_cfg_t
```

12.1.43.1 clock_source

```
can_clock_source_t::clock_source
```

Brief description

Source of the CAN clock.

12.1.43.2 p_mailbox_mask

```
uint32_t* ::p_mailbox_mask
```

Brief description

Mailbox mask, one for every 4 mailboxes.

12.1.44 can_frame_t

```
typedef struct{
    can_id_t  id
    uint8_t  data_length_code
    uint8_t  data[8]
    can_frame_type_t  type
} can_frame_t
```

12.1.44.1 id

```
can_id_t::id
```

Brief description

CAN id.

12.1.44.2 data_length_code

`uint8_t can_frame_t::data_length_code`

Brief description

CAN Data Length code, number of bytes in the message.

12.1.44.3 data

`uint8_t can_frame_t::data[8]`

Brief description

CAN data, up to 8 bytes.

12.1.44.4 type

`can_frame_type_t::type`

Brief description

Frame type, data or remote frame.

12.1.45 can_info_t

```
typedef struct{
    can_mode_t  operation_mode
    can_status_t  status
    uint32_t  bit_rate
    uint8_t  error_count_transmit
    uint8_t  error_count_receive
    can_error_t  error_code
} can_info_t
```

12.1.45.1 operation_mode

`can_mode_t::operation_mode`

Brief description

Can operation mode.

12.1.45.2 status

`can_status_t::status`

Brief description

CAN status.

12.1.45.3 bit_rate

`uint32_t can_info_t::bit_rate`

Brief description

CAN bit rate.

12.1.45.4 error_count_transmit

`uint8_t can_info_t::error_count_transmit`

Brief description

Transmit error count.

12.1.45.5 error_count_receive

`uint8_t can_info_t::error_count_receive`

Brief description

Receive error count.

12.1.45.6 error_code

`can_error_t::error_code`

Brief description

Error code, cleared after reading.

12.1.46 can_instance_ctrl_t

```
typedef struct{
    uint32_t  channel
    uint32_t  open
    can_mode_t  operation_mode
    can_id_mode_t  id_mode
    uint32_t  mailbox_count
    can_mailbox_t *  p_mailbox
    can_message_mode_t  message_mode
    can_clock_source_t  clock_source
    void(*  p_callback)(can_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    IRQn_Type  error_irq
    IRQn_Type  mailbox_rx_irq
    IRQn_Type  mailbox_tx_irq
} can_instance_ctrl_t
```

12.1.46.1 channel

`uint32_t ::channel`

Brief description

Channel number.

Detailed description

Parameters to control CAN peripheral device

12.1.46.2 open

`uint32_t ::open`

Brief description

Open status of channel.

12.1.46.3 operation_mode

`can_mode_t::operation_mode`

Brief description

Can operation mode.

12.1.46.4 id_mode

`can_id_mode_t::id_mode`

Brief description

Standard or Extended ID mode.

12.1.46.5 mailbox_count

`uint32_t ::mailbox_count`

Brief description

Number of mailboxes.

12.1.46.6 p_mailbox

`can_mailbox_t::p_mailbox`

Brief description

Pointer to mailboxes.

12.1.46.7 message_mode

`can_message_mode_t::message_mode`

Brief description

Overwrite message or overrun.

12.1.46.8 clock_source

`can_clock_source_t::clock_source`

Brief description

Clock source. CANMCLK or PCLKB.

12.1.46.9 p_callback

`void(* ::p_callback) ( *p_args)`

Brief description

Pointer to callback function.

Detailed description

Parameters to process CAN Event

12.1.46.10 p_context

`void const* ::p_context`

Brief description

Pointer to the higher level device context.

12.1.46.11 p_reg

`void* ::p_reg`

Brief description

Pointer to register base address.

12.1.46.12 error_irq

`IRQn_Type ::error_irq`

Brief description

Error IRQ number.

12.1.46.13 mailbox_rx_irq

`IRQn_Type ::mailbox_rx_irq`

Brief description

Receive mailbox IRQ number.

12.1.46.14 mailbox_tx_irq

IRQn_Type ::mailbox_tx_irq

Brief description

Transmit mailbox IRQ number.

12.1.47 can_instance_t

```
typedef struct{
    can_ctrl_t *  p_ctrl
    can_cfg_t const  *  p_cfg
    can_api_t const  *  p_api
} can_instance_t
```

12.1.47.1 p_ctrl

can_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.47.2 p_cfg

can_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.47.3 p_api

can_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.48 can_mailbox_t

```
typedef struct{
    can_id_t  mailbox_id
    can_mailbox_send_receive_t  mailbox_type
    can_frame_type_t  frame_type
} can_mailbox_t
```

12.1.48.1 mailbox_id

can_id_t::mailbox_id

Brief description

Mailbox ID.

12.1.48.2 mailbox_type

`can_mailbox_send_receive_t::mailbox_type`

Brief description

Receive or Transmit mailbox type.

12.1.48.3 frame_type

`can_frame_type_t::frame_type`

Brief description

Frame type for receive mailbox.

12.1.49 can_status_t

```
typedef struct{
    uint32_t  status
    struct can_status_t::st_status_b  status_b
} can_status_t
```

12.1.49.1 status

`uint32_t can_status_t::status`

12.1.49.2 status_b

`can_status_t::st_status_b::status_b`

12.1.50 cgc_api_t

```
typedef struct{
    ssp_err_t(*  init)(void)
    ssp_err_t(*  clocksCfg)(cgc_clocks_cfg_t const *const p_clock_cfg)
    ssp_err_t(*  clockStart)(cgc_clock_t clock_source, cgc_clock_cfg_t
*p_clock_cfg)
    ssp_err_t(*  clockStop)(cgc_clock_t clock_source)
    ssp_err_t(*  systemClockSet)(cgc_clock_t clock_source,
cgc_system_clock_cfg_t const *const p_clock_cfg)
    ssp_err_t(*  systemClockGet)(cgc_clock_t *p_clock_source,
cgc_system_clock_cfg_t *p_set_clock_cfg)
    ssp_err_t(*  systemClockFreqGet)(cgc_system_clocks_t clock, uint32_t
*p_freq_hz)
```

```
    ssp_err_t(*  clockCheck)(cgc_clock_t clock_source)
    ssp_err_t(*  oscStopDetect)(void(*p_callback)(cgc_callback_args_t *p_args),
bool enable)
    ssp_err_t(*  oscStopStatusClear)(void)
    ssp_err_t(*  busClockOutCfg)(cgc_bclockout_dividers_t divider)
    ssp_err_t(*  busClockOutEnable)(void)
    ssp_err_t(*  busClockOutDisable)(void)
    ssp_err_t(*  clockOutCfg)(cgc_clock_t clock, cgc_clockout_dividers_t divider)
    ssp_err_t(*  clockOutEnable)(void)
    ssp_err_t(*  clockOutDisable)(void)
    ssp_err_t(*  lcdClockCfg)(cgc_clock_t clock)
    ssp_err_t(*  lcdClockEnable)(void)
    ssp_err_t(*  lcdClockDisable)(void)
    ssp_err_t(*  sdramClockOutEnable)(void)
    ssp_err_t(*  sdramClockOutDisable)(void)
    ssp_err_t(*  usbClockCfg)(cgc_usb_clock_div_t divider)
    ssp_err_t(*  systickUpdate)(uint32_t period_count,
cgc_systick_period_units_t units)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} cgc_api_t
```

12.1.51 cgc_callback_args_t

```
typedef struct{
    cgc_event_t  event
    void const  *  p_context
} cgc_callback_args_t
```

12.1.51.1 event

`cgc_event_t::event`

Brief description

The event can be used to identify what caused the callback.

12.1.51.2 p_context

`void const* cgc_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.52 cgc_clock_cfg_t

```
typedef struct{
    cgc_clock_t  source_clock
    cgc_pll_div_t  divider
```

```
    float  multiplier
} cgc_clock_cfg_t
```

12.1.52.1 source_clock

`cgc_clock_t::source_clock`

Brief description

PLL source clock (S7G2 only).

12.1.52.2 divider

`cgc_pll_div_t::divider`

Brief description

PLL divider.

12.1.52.3 multiplier

`float cgc_clock_cfg_t::multiplier`

Brief description

PLL multiplier.

12.1.53 cgc_clocks_cfg_t

```
typedef struct{
    cgc_clock_t  system_clock
    cgc_clock_cfg_t  pll_cfg
    cgc_system_clock_cfg_t  sys_cfg
    cgc_clock_change_t  loco_state
    cgc_clock_change_t  moco_state
    cgc_clock_change_t  hoco_state
    cgc_clock_change_t  subosc_state
    cgc_clock_change_t  mainosc_state
    cgc_clock_change_t  pll_state
} cgc_clocks_cfg_t
```

12.1.53.1 system_clock

`cgc_clock_t::system_clock`

Brief description

System clock source enumeration.

12.1.53.2 pll_cfg

`cgc_clock_cfg_t::pll_cfg`

Brief description

PLL configuration structure.

12.1.53.3 sys_cfg

`cgc_system_clock_cfg_t::sys_cfg`

Brief description

Clock dividers structure.

12.1.53.4 loco_state

`cgc_clock_change_t::loco_state`

Brief description

State of LOCO.

12.1.53.5 moco_state

`cgc_clock_change_t::moco_state`

Brief description

State of MOCO.

12.1.53.6 hoco_state

`cgc_clock_change_t::hoco_state`

Brief description

State of HOCO.

12.1.53.7 subosc_state

`cgc_clock_change_t::subosc_state`

Brief description

State of Sub-oscillator.

12.1.53.8 mainosc_state

`cgc_clock_change_t::mainosc_state`

Brief description

State of Main oscillator.

12.1.53.9 pll_state

cgc_clock_change_t::pll_state

Brief description

State of PLL.

12.1.54 cgc_instance_t

```
typedef struct{
    cgc_clock_cfg_t const  *  p_cfg
    cgc_api_t const  *  p_api
} cgc_instance_t
```

12.1.54.1 p_cfg

cgc_clock_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.54.2 p_api

cgc_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.55 cgc_system_clock_cfg_t

```
typedef struct{
    cgc_sys_clock_div_t  pclka_div
    cgc_sys_clock_div_t  pclkb_div
    cgc_sys_clock_div_t  pclkc_div
    cgc_sys_clock_div_t  pclkd_div
    cgc_sys_clock_div_t  bclk_div
    cgc_sys_clock_div_t  fclk_div
    cgc_sys_clock_div_t  iclk_div
} cgc_system_clock_cfg_t
```

12.1.55.1 pclka_div

cgc_sys_clock_div_t::pclka_div

Brief description

Divider value for PCLKA.

12.1.55.2 pclkb_div

`cgc_sys_clock_div_t::pclkb_div`

Brief description

Divider value for PCLKB.

12.1.55.3 pclkc_div

`cgc_sys_clock_div_t::pclkc_div`

Brief description

Divider value for PCLKC.

12.1.55.4 pclkd_div

`cgc_sys_clock_div_t::pclkd_div`

Brief description

Divider value for PCLKD.

12.1.55.5 bclk_div

`cgc_sys_clock_div_t::bclk_div`

Brief description

Divider value for BCLK.

12.1.55.6 fclk_div

`cgc_sys_clock_div_t::fclk_div`

Brief description

Divider value for FCLK.

12.1.55.7 iclk_div

`cgc_sys_clock_div_t::iclk_div`

Brief description

Divider value for ICLK.

12.1.56 crc_api_t

```
typedef struct{
    ssp_err_t(*  open)(crc_ctrl_t *const p_ctrl, crc_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(crc_ctrl_t *const p_ctrl)
    ssp_err_t(*  crcResultGet)(crc_ctrl_t *const p_ctrl, uint32_t *crc_result)
```

```
    ssp_err_t(*  snoopEnable)(crc_ctrl_t *const p_ctrl, uint32_t crc_seed)
    ssp_err_t(*  snoopDisable)(crc_ctrl_t *const p_ctrl)
    ssp_err_t(*  snoopCfg)(crc_ctrl_t *const p_ctrl, crc_snoop_cfg_t *const
p_snoop_cfg)
    ssp_err_t(*  calculate)(crc_ctrl_t *const p_ctrl, void *p_input_buffer,
uint32_t num_bytes, uint32_t crc_seed, uint32_t *p_crc_result)
    ssp_err_t(*  versionGet)(ssp_version_t *version)
} crc_api_t
```

12.1.57 crc_cfg_t

```
typedef struct{
    crc_polynomial_t  polynomial
    crc_bit_order_t  bit_order
    void const  *  p_extend
} crc_cfg_t
```

12.1.57.1 polynomial

`crc_polynomial_t::polynomial`

Brief description

CRC Generating Polynomial Switching. (GPS)

12.1.57.2 bit_order

`crc_bit_order_t::bit_order`

Brief description

CRC Calculation Switching (LMS)

12.1.57.3 p_extend

`void const* crc_cfg_t::p_extend`

Brief description

CRC Hardware Dependent Configuration.

12.1.58 crc_instance_ctrl_t

```
typedef struct{
    R_CRC_Type *  p_reg
    uint32_t  open
    crc_polynomial_t  polynomial
    crc_bit_order_t  bit_order
} crc_instance_ctrl_t
```

12.1.58.1 p_reg

`R_CRC_Type* ::p_reg`

Brief description

Pointer to register base address.

12.1.58.2 open

`uint32_t ::open`

Brief description

Whether or not channel is open.

12.1.58.3 polynomial

`crc_polynomial_t::polynomial`

Brief description

CRC Generating Polynomial Switching (GPS).

12.1.58.4 bit_order

`crc_bit_order_t::bit_order`

Brief description

CRC Calculation Switching (LMS).

12.1.59 crc_instance_t

```
typedef struct{
    crc_ctrl_t *  p_ctrl
    crc_cfg_t const  *  p_cfg
    crc_api_t const  *  p_api
} crc_instance_t
```

12.1.59.1 p_ctrl

`crc_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.59.2 p_cfg

`crc_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.59.3 p_api

`crc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.60 crc_snoop_cfg_t

```
typedef struct{
    uint32_t  snoop_channel
    crc_snoop_direction_t  snoop_direction
} crc_snoop_cfg_t
```

12.1.60.1 snoop_channel

`uint32_t crc_snoop_cfg_t::snoop_channel`

Brief description

Register Snoop Address (CRCSA)

12.1.60.2 snoop_direction

`crc_snoop_direction_t::snoop_direction`

Brief description

Snoop-On-Write/Read Switch (CRCSWR)

12.1.61 crypto_api_t

```
typedef struct{
    uint32_t(*  open)(crypto_ctrl_t *const p_ctrl, crypto_cfg_t const *const
p_cfg)
    uint32_t(*  close)(crypto_ctrl_t *const p_ctrl)
    uint32_t(*  interfaceGet)(crypto_interface_get_param_t *const
interface_info, void *const p_interface)
    uint32_t(*  statusGet)(uint32_t *p_status)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} crypto_api_t
```

12.1.62 crypto_cfg_t

```
typedef struct{
    void(*  p_sce_long_plg_start_callback)(void)
```

```
    void(*  p_sce_long_plg_end_callback)(void)
    crypto_word_endian_t  endian_flag
} crypto_cfg_t
```

12.1.62.1 p_sce_long_plg_start_callback

```
void(* crypto_cfg_t::p_sce_long_plg_start_callback) (void)
```

12.1.62.2 p_sce_long_plg_end_callback

```
void(* crypto_cfg_t::p_sce_long_plg_end_callback) (void)
```

Detailed description

Callback provided when a ISR occurs. Set to NULL for no CPU interrupt.

12.1.62.3 endian_flag

```
crypto_word_endian_t::endian_flag
```

Detailed description

Endian flag, indicates word endianness for the uint32_t[] array inputs used in Crypto APIs

12.1.63 crypto_ctrl_t

```
typedef struct{
    uint32_t  state
    uint32_t  cb_data
    void(*  p_sce_long_plg_start_callback)(void)
    void(*  p_sce_long_plg_end_callback)(void)
} crypto_ctrl_t
```

12.1.63.1 state

```
uint32_t crypto_ctrl_t::state
```

Brief description

indicates state of the SCE/SCE-Lite driver e.g whether it is initialized

12.1.63.2 cb_data

```
uint32_t crypto_ctrl_t::cb_data
```

12.1.63.3 p_sce_long_plg_start_callback

```
void(* crypto_ctrl_t::p_sce_long_plg_start_callback) (void)
```

12.1.63.4 p_sce_long_plg_end_callback

```
void(* crypto_ctrl_t::p_sce_long_plg_end_callback) (void)
```

12.1.64 crypto_instance_t

```
typedef struct{
    crypto_ctrl_t *  p_ctrl
    crypto_cfg_t const  *  p_cfg
    crypto_api_t const  *  p_api
} crypto_instance_t
```

12.1.64.1 p_ctrl

```
crypto_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.64.2 p_cfg

```
crypto_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.64.3 p_api

```
crypto_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.65 crypto_interface_get_param_t

```
typedef struct{
    crypto_algorithm_type_t  algorithm_type
    crypto_key_type_t  key_type
    crypto_key_size_t  key_size
    crypto_chaining_mode_t  chaining_mode
    crypto_hash_type_t  hash_type
} crypto_interface_get_param_t
```

12.1.65.1 algorithm_type

```
crypto_algorithm_type_t::algorithm_type
```

12.1.65.2 key_type

crypto_key_type_t::key_type

12.1.65.3 key_size

crypto_key_size_t::key_size

12.1.65.4 chaining_mode

crypto_chaining_mode_t::chaining_mode

12.1.65.5 hash_type

crypto_hash_type_t::hash_type

12.1.66 ctsu_api_t

```
typedef struct{
    ssp_err_t(*  open)(ctsu_ctrl_t *p_ctrl, ctsu_cfg_t *p_cfg)
    ssp_err_t(*  close)(ctsu_ctrl_t *p_ctrl, ctsu_close_option_t opts)
    ssp_err_t(*  scan)(ctsu_ctrl_t *p_ctrl)
    ssp_err_t(*  update)(ctsu_ctrl_t *p_ctrl)
    ssp_err_t(*  read)(ctsu_ctrl_t *p_ctrl, void *p_dest, ctsu_read_t opts, const
ctsu_channel_pair_t *channels, const uint16_t count)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} ctsu_api_t
```

12.1.67 ctsu_callback_args_t

```
typedef struct{
    ctsu_event_t  event
    void const  *  p_context
} ctsu_callback_args_t
```

12.1.67.1 event

ctsu_event_t::event

Brief description

CTSU callback event.

12.1.67.2 p_context

void const* ctsu_callback_args_t::p_context

Brief description

Placeholder for user data.

12.1.68 ctsu_cfg_t

```
typedef struct{
    transfer_instance_t const  *const  p_lower_lvl_transfer_read
    transfer_instance_t const  *const  p_lower_lvl_transfer_write
    ctsu_hw_cfg_t *  p_ctsu_hw_cfg
    ctsu_functions_t *  p_ctsu_functions
    void(*  p_callback)(ctsu_callback_args_t *p_args)
    void *  p_context
    ctsu_process_option_t  ctsu_soft_option
    ctsu_close_option_t  ctsu_close_option
    uint8_t  write_ipl
    uint8_t  read_ipl
    uint8_t  end_ipl
} ctsu_cfg_t
```

12.1.68.1 p_lower_lvl_transfer_read

`transfer_instance_t::p_lower_lvl_transfer_read`

Brief description

Pointer to the Transfer instance to read results.

12.1.68.2 p_lower_lvl_transfer_write

`transfer_instance_t::p_lower_lvl_transfer_write`

Brief description

Pointer to the Transfer instance to write cfg.

12.1.68.3 p_ctsu_hw_cfg

`ctsu_hw_cfg_t::p_ctsu_hw_cfg`

Brief description

Pointer to a CTSU configuration.

12.1.68.4 p_ctsu_functions

`ctsu_functions_t::p_ctsu_functions`

Brief description

Pointer to a place holder for custom data functions.

12.1.68.5 p_callback

```
void(* ctsu_cfg_t::p_callback) (ctsu_callback_args_t *p_args)
```

Brief description

Callback to the function to use when scan is complete.

12.1.68.6 p_context

```
void* ctsu_cfg_t::p_context
```

Brief description

Pointer to data that should be passed to update_complete notification.

12.1.68.7 ctsu_soft_option

```
ctsu_process_option_t::ctsu_soft_option
```

Brief description

Software options to use when performing Open and Process.

12.1.68.8 ctsu_close_option

```
ctsu_close_option_t::ctsu_close_option
```

Brief description

Software options to use when closing touch operation.

12.1.68.9 write_ipl

```
uint8_t ctsu_cfg_t::write_ipl
```

Brief description

Write interrupt priority.

12.1.68.10 read_ipl

```
uint8_t ctsu_cfg_t::read_ipl
```

Brief description

Read interrupt priority.

12.1.68.11 end_ipl

```
uint8_t ctsu_cfg_t::end_ipl
```

Brief description

End interrupt priority.

12.1.69 ctsu_channel_data_mutual_t

```
typedef struct{
    uint16_t  sen_cnt_1
    uint16_t  ref_cnt_1
    uint16_t  sen_cnt_2
    uint16_t  ref_cnt_2
} ctsu_channel_data_mutual_t
```

12.1.69.1 sen_cnt_1

```
uint16_t ctsu_channel_data_mutual_t::sen_cnt_1
```

Brief description

Raw Sensor count primary reading.

12.1.69.2 ref_cnt_1

```
uint16_t ctsu_channel_data_mutual_t::ref_cnt_1
```

Brief description

Raw reference ICO count primary reading.

12.1.69.3 sen_cnt_2

```
uint16_t ctsu_channel_data_mutual_t::sen_cnt_2
```

Brief description

Raw sensor ICO count secondary reading.

12.1.69.4 ref_cnt_2

```
uint16_t ctsu_channel_data_mutual_t::ref_cnt_2
```

Brief description

Raw reference ICO count secondary reading.

12.1.70 ctsu_channel_data_self_t

```
typedef struct{
    uint16_t  sensor_count
    uint16_t  reference_count
} ctsu_channel_data_self_t
```

12.1.70.1 sensor_count

```
uint16_t ctsu_channel_data_self_t::sensor_count
```

Brief description

Raw sensor count.

12.1.70.2 reference_count

`uint16_t ctsu_channel_data_self_t::reference_count`

Brief description

Raw reference count.

12.1.71 ctsu_channel_pair_t

```
typedef struct{
    int8_t  rx
    int8_t  tx
} ctsu_channel_pair_t
```

12.1.71.1 rx

`int8_t ctsu_channel_pair_t::rx`

Brief description

Denotes the primary channel.

12.1.71.2 tx

`int8_t ctsu_channel_pair_t::tx`

Brief description

Denotes the secondary channel (used only for mutual capacitance mode)

12.1.72 ctsu_channel_setting_t

```
typedef struct{
    uint16_t  ctsussc
    uint16_t  ctsuso0
    uint16_t  ctsuso1
} ctsu_channel_setting_t
```

12.1.72.1 ctsussc

`volatile uint16_t ctsu_channel_setting_t::ctsussc`

Brief description

Holds value for the CTSUSSC register.

12.1.72.2 ctsuso0

```
volatile uint16_t ctsu_channel_setting_t::ctsuso0
```

Brief description

Holds value for the CTSUSO0 register.

12.1.72.3 ctsuso1

```
volatile uint16_t ctsu_channel_setting_t::ctsuso1
```

Brief description

Holds value for the CTSUSO1 register.

12.1.73 ctsu_functions_t

```
typedef struct{
    int32_t(*  preFilter)(void *p_args)
    int32_t(*  filter)(volatile uint16_t *output, volatile uint16_t *input)
    int32_t(*  postFilter)(void *p_args)
    int32_t(*  ctsuDecode)(void *p_args)
    int32_t(*  otDriftComp)(void *p_args)
    int32_t(*  rtDriftComp)(void *p_args)
    int32_t(*  otAutoTune)(void *p_args)
    int32_t(*  rtAutoTune)(void *p_args)
} ctsu_functions_t
```

12.1.73.1 preFilter

```
int32_t(* ctsu_functions_t::preFilter) (void *p_args)
```

Brief description

Used for calculating raw data SNR before data gets filtered.

12.1.73.2 filter

```
int32_t(* ctsu_functions_t::filter) (volatile uint16_t *output, volatile uint16_t
*input)
```

Brief description

Weighted averaging using CTSU_CFG_FILTER_DEPTH.

12.1.73.3 postFilter

```
int32_t(* ctsu_functions_t::postFilter) (void *p_args)
```

Brief description

Processing the filter results.

12.1.73.4 ctsuDecode

```
int32_t(* ctsu_functions_t::ctsuDecode) (void *p_args)
```

Brief description

Algorithm to decide if channel is touched or not.

12.1.73.5 otDriftComp

```
int32_t(* ctsu_functions_t::otDriftComp) (void *p_args)
```

Brief description

Algorithm to perform manipulations to baseline, envelope, and thresholds on initialization.

12.1.73.6 rtDriftComp

```
int32_t(* ctsu_functions_t::rtDriftComp) (void *p_args)
```

Brief description

Algorithm to perform run time manipulations to baseline, envelope, and thresholds.

12.1.73.7 otAutoTune

```
int32_t(* ctsu_functions_t::otAutoTune) (void *p_args)
```

Brief description

Function called once on initialization of system.

12.1.73.8 rtAutoTune

```
int32_t(* ctsu_functions_t::rtAutoTune) (void *p_args)
```

Brief description

Function called to auto tune sensor when system is running.

12.1.74 ctsu_hw_cfg_t

```
typedef struct{
    R_CTSU_Type  ctsu_settings
    ctsu_channel_setting_t *  write_settings
    uint16_t *  threshold
    uint16_t *  hysteresis
    uint16_t *  baseline
    void *  raw_result
    void *  filter_output
    void *  binary_result
    ctsu_channel_pair_t *  excluded
    int8_t  num_excluded
```

```
    const uint16_t *  series_resistance
} ctsu_hw_cfg_t
```

12.1.74.1 ctsu_settings

`R_CTSU_Type ctsu_hw_cfg_t::ctsu_settings`

Brief description

User defined SFR settings for CR0, CR1, SDPRS, SST, CHACn, CHTRCn, DCLKC.

12.1.74.2 write_settings

`ctsu_channel_setting_t::write_settings`

Brief description

User defined initial settings for thresholds for each active channel . Threshold is difference between runtime baseline and filtered output of sensor count.

Detailed description

User defined settings for SSC, SO0, SO1 for each active channel.

12.1.74.3 threshold

`uint16_t* ctsu_hw_cfg_t::threshold`

12.1.74.4 hysteresis

`uint16_t* ctsu_hw_cfg_t::hysteresis`

Brief description

User defined settings for tolerance in count values.

12.1.74.5 baseline

`uint16_t* ctsu_hw_cfg_t::baseline`

Brief description

A baseline of the expected count for each active channel when not touched.

12.1.74.6 raw_result

`void* ctsu_hw_cfg_t::raw_result`

Brief description

A pointer to a buffer which will hold raw results of CTSU measurement.

12.1.74.7 filter_output

`void* ctsu_hw_cfg_t::filter_output`

Brief description

A pointer to a buffer which will hold output after filtering raw results.

12.1.74.8 binary_result

`void* ctsu_hw_cfg_t::binary_result`

Brief description

A pointer to a location where binary data can be stored.

12.1.74.9 excluded

`ctsu_channel_pair_t::excluded`

Brief description

A pointer to an array which contains a list of channel pairs which need to be ignored in ascending order of rx and then tx.

12.1.74.10 num_excluded

`int8_t ctsu_hw_cfg_t::num_excluded`

Brief description

Number of elements in the array excluded.

12.1.74.11 series_resistance

`const uint16_t* ctsu_hw_cfg_t::series_resistance`

Brief description

Resistance of the channel (to determine RC constant when tuning).

12.1.75 ctsu_instance_ctrl_t

```
typedef struct{
    transfer_api_t const  *  p_api_transfer
    transfer_ctrl_t *  p_lowerl_lvl_transfer_read_ctrl
    transfer_ctrl_t *  p_lowerl_lvl_transfer_write_ctrl
    bool  ctsu_opened
    uint8_t  ctsu_unit
    ctsu_hw_cfg_t *  p_ctsu_hw_cfg
    ctsu_process_option_t  ctsu_open_option
    ctsu_process_option_t  ctsu_update_option
    ctsu_close_option_t  ctsu_close_option
    ctsu_action_t  ctsu_process_state
```

```
    void(*  p_callback)(ctsu_callback_args_t *p_args)
    void *  p_context
    R_CTSU_Type *  p_reg
} ctsu_instance_ctrl_t
```

12.1.75.1 p_api_transfer

`transfer_api_t::p_api_transfer`

Brief description

Pointer to lower level Transfer driver function pointers.

12.1.75.2 p_lowerl_lvl_transfer_read_ctrl

`transfer_ctrl_t::p_lowerl_lvl_transfer_read_ctrl`

Brief description

Pointer to the Transfer Read control.

12.1.75.3 p_lowerl_lvl_transfer_write_ctrl

`transfer_ctrl_t::p_lowerl_lvl_transfer_write_ctrl`

Brief description

Pointer to the Transfer Write control.

12.1.75.4 ctsu_opened

`bool ::ctsu_opened`

Brief description

Store initialization state.

12.1.75.5 ctsu_unit

`uint8_t ::ctsu_unit`

Brief description

CTSU Unit in use.

12.1.75.6 p_ctsu_hw_cfg

`ctsu_hw_cfg_t::p_ctsu_hw_cfg`

Brief description

Pointer to a CTSU configuration.

12.1.75.7 ctsu_open_option

`ctsu_process_option_t::ctsu_open_option`

Brief description

Software options to use when performing Open and Process.

12.1.75.8 ctsu_update_option

`ctsu_process_option_t::ctsu_update_option`

Brief description

Software options to use when performing parameter Update process.

12.1.75.9 ctsu_close_option

`ctsu_close_option_t::ctsu_close_option`

Brief description

Software options to use when closing touch operation.

12.1.75.10 ctsu_process_state

`ctsu_action_t::ctsu_process_state`

Brief description

Variable to observe CTSU processing state machine operation.

12.1.75.11 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback to the function to use when updating dependent parameters is complete.

12.1.75.12 p_context

```
void* ::p_context
```

Brief description

Pointer to data that should be passed to update_complete notification.

12.1.75.13 p_reg

```
R_CTSU_Type* ::p_reg
```

Brief description

Pointer to base register address.

12.1.76 ctsu_instance_t

```
typedef struct{
    ctsu_ctrl_t *  p_ctrl
    ctsu_cfg_t *  p_cfg
    ctsu_api_t const  *  p_api
} ctsu_instance_t
```

12.1.76.1 p_ctrl

ctsu_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.76.2 p_cfg

ctsu_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.76.3 p_api

ctsu_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.77 dac8_extended_cfg_t

```
typedef struct{
    bool  enable_charge_pump
    dac8_mode_t  dac_mode
} dac8_extended_cfg_t
```

12.1.77.1 enable_charge_pump

bool ::enable_charge_pump

Brief description

Enable DAC charge pump.

12.1.77.2 dac_mode

dac8_mode_t::dac_mode

Brief description

DAC mode.

12.1.78 dac8_instance_ctrl_t

```
typedef struct{
    void *  p_reg
    uint8_t  channel
    uint8_t  channel_started
    uint32_t  channel_opened
    dac_data_format_t  data_format
} dac8_instance_ctrl_t
```

12.1.78.1 p_reg

`void* ::p_reg`

Brief description

Pointer to DAC base register.

12.1.78.2 channel

`uint8_t ::channel`

Brief description

ID associated with this DAC channel.

12.1.78.3 channel_started

`uint8_t ::channel_started`

Brief description

DAC operation on channel started.

12.1.78.4 channel_opened

`uint32_t ::channel_opened`

Brief description

DAC channel open.

12.1.78.5 data_format

`dac_data_format_t::data_format`

Brief description

DAC data format.

12.1.79 dac_api_t

```
typedef struct{
    ssp_err_t(*  open)(dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(dac_ctrl_t *p_ctrl)
    ssp_err_t(*  write)(dac_ctrl_t *p_ctrl, dac_size_t value)
    ssp_err_t(*  start)(dac_ctrl_t *p_ctrl)
    ssp_err_t(*  stop)(dac_ctrl_t *p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} dac_api_t
```

12.1.80 dac_cfg_t

```
typedef struct{
    uint8_t  channel
    bool  ad_da_synchronized
    dac_data_format_t  data_format
    bool  output_amplifier_enabled
    void const  *  p_extend
} dac_cfg_t
```

12.1.80.1 channel

`uint8_t dac_cfg_t::channel`

Brief description

ID associated with this DAC channel.

12.1.80.2 ad_da_synchronized

`bool dac_cfg_t::ad_da_synchronized`

Brief description

AD/DA synchronization.

12.1.80.3 data_format

`dac_data_format_t::data_format`

Brief description

Data format.

12.1.80.4 output_amplifier_enabled

`bool dac_cfg_t::output_amplifier_enabled`

Brief description

Output amplifier enable.

12.1.80.5 p_extend

```
void const* dac_cfg_t::p_extend
```

12.1.81 dac_instance_ctrl_t

```
typedef struct{
    void *  p_reg
    uint8_t  channel
    uint8_t  channel_started
    uint8_t  channel_opened
} dac_instance_ctrl_t
```

12.1.81.1 p_reg

```
void* ::p_reg
```

Brief description

Pointer to DAC base register.

12.1.81.2 channel

```
uint8_t ::channel
```

Brief description

ID associated with this DAC channel.

12.1.81.3 channel_started

```
uint8_t ::channel_started
```

Brief description

DAC operation on channel started.

12.1.81.4 channel_opened

```
uint8_t ::channel_opened
```

Brief description

DAC channel open.

12.1.82 dac_instance_t

```
typedef struct{
    dac_ctrl_t *  p_ctrl
```

```
    dac_cfg_t const  *  p_cfg
    dac_api_t const  *  p_api
} dac_instance_t
```

12.1.82.1 p_ctrl

dac_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.82.2 p_cfg

dac_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.82.3 p_api

dac_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.83 display_api_t

```
typedef struct{
    ssp_err_t(*  open)(display_ctrl_t *const p_ctrl, display_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(display_ctrl_t *const p_ctrl)
    ssp_err_t(*  start)(display_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(display_ctrl_t *const p_ctrl)
    ssp_err_t(*  layerChange)(display_ctrl_t const *const p_ctrl,
display_runtime_cfg_t const *const p_cfg, display_frame_layer_t frame)
    ssp_err_t(*  correction)(display_ctrl_t const *const p_ctrl,
display_correction_t const *const p_param)
    ssp_err_t(*  clut)(display_ctrl_t const *const p_ctrl, display_clut_cfg_t
const *const p_clut_cfg, display_frame_layer_t frame)
    ssp_err_t(*  statusGet)(display_ctrl_t const *const p_ctrl, display_status_t
*const p_status)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} display_api_t
```

12.1.84 display_brightness_t

```
typedef struct{
    bool  enable
    uint16_t  r
    uint16_t  g
    uint16_t  b
} display_brightness_t
```

12.1.84.1 enable

```
bool display_brightness_t::enable
```

Brief description

Brightness Correction On/Off.

12.1.84.2 r

```
uint16_t display_brightness_t::r
```

Brief description

Brightness (DC) adjustment for R channel.

12.1.84.3 g

```
uint16_t display_brightness_t::g
```

Brief description

Brightness (DC) adjustment for G channel.

12.1.84.4 b

```
uint16_t display_brightness_t::b
```

Brief description

Brightness (DC) adjustment for B channel.

12.1.85 display_callback_args_t

```
typedef struct{
    display_event_t  event
    void const  *  p_context
} display_callback_args_t
```

12.1.85.1 event

```
display_event_t::event
```

Brief description

Event code.

12.1.85.2 p_context

```
void const* display_callback_args_t::p_context
```

Brief description

Context provided to user during callback.

12.1.86 display_cfg_t

```
typedef struct{
    display_input_cfg_t  input[DISPLAY_FRAME_LAYER_2+1]
    display_output_cfg_t  output
    display_layer_t  layer[DISPLAY_FRAME_LAYER_2+1]
    uint8_t  line_detect_ipl
    uint8_t  underflow_1_ipl
    uint8_t  underflow_2_ipl
    void(*  p_callback)(display_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} display_cfg_t
```

12.1.86.1 input

```
display_input_cfg_t::input
```

Brief description

Graphics input frame setting.

Detailed description

Generic configuration for display devices

12.1.86.2 output

```
display_output_cfg_t::output
```

Brief description

Graphics output frame setting.

12.1.86.3 layer

```
display_layer_t::layer
```

Brief description

Graphics layer blend setting.

12.1.86.4 line_detect_ipl

```
uint8_t display_cfg_t::line_detect_ipl
```

Brief description

Line detect interrupt priority.

12.1.86.5 underflow_1_ipl

```
uint8_t display_cfg_t::underflow_1_ipl
```

Brief description

Underflow 1 interrupt priority.

12.1.86.6 underflow_2_ipl

```
uint8_t display_cfg_t::underflow_2_ipl
```

Brief description

Underflow 1 interrupt priority.

12.1.86.7 p_callback

```
void(* display_cfg_t::p_callback) (display_callback_args_t *p_args)
```

Brief description

Pointer to callback function.

Detailed description

Configuration for display event processing

12.1.86.8 p_context

```
void const* display_cfg_t::p_context
```

Brief description

User defined context passed into callback function.

12.1.86.9 p_extend

```
void const* display_cfg_t::p_extend
```

Brief description

Display hardware dependent configuration.

Detailed description

Pointer to display peripheral specific configuration

12.1.87 display_clut_cfg_t

```
typedef struct{
    uint32_t *  p_base
    uint16_t  start
    uint16_t  size
} display_clut_cfg_t
```

12.1.87.1 p_base

```
uint32_t* display_clut_cfg_t::p_base
```

Brief description

Pointer to CLUT source data.

12.1.87.2 start

```
uint16_t display_clut_cfg_t::start
```

Brief description

Beginning of CLUT entry to be updated.

12.1.87.3 size

```
uint16_t display_clut_cfg_t::size
```

Brief description

Size of CLUT entry to be updated.

12.1.88 display_clut_t

```
typedef struct{
    uint32_t  color_num
    const uint32_t *  p_clut
} display_clut_t
```

12.1.88.1 color_num

```
uint32_t display_clut_t::color_num
```

Brief description

The number of colors in CLUT.

12.1.88.2 p_clut

```
const uint32_t* display_clut_t::p_clut
```

Brief description

Address of the area storing the CLUT data (in ARGB8888 format)

12.1.89 display_color_t

```
typedef struct{
    uint32_t  argb
    uint8_t  b
    uint8_t  g
    uint8_t  r
    uint8_t  a
    struct{}                byte
    union{}                 union{}
} display_color_t
```

12.1.89.1 argb

```
uint32_t display_color_t::argb
```

12.1.89.2 b

```
uint8_t display_color_t::b
```

Brief description

blue

12.1.89.3 g

```
uint8_t display_color_t::g
```

Brief description

green

12.1.89.4 r

```
uint8_t display_color_t::r
```

Brief description

red

12.1.89.5 a

```
uint8_t display_color_t::a
```

Brief description

a

12.1.89.6 byte

See source code for the definition of this member.

12.1.89.7 union{}

See source code for the definition of this member.

12.1.90 display_contrast_t

```
typedef struct{
    bool  enable
    uint8_t  r
    uint8_t  g
    uint8_t  b
} display_contrast_t
```

12.1.90.1 enable

bool display_contrast_t::enable

Brief description

Contrast Correction On/Off.

12.1.90.2 r

uint8_t display_contrast_t::r

Brief description

Contrast (gain) adjustment for R channel.

12.1.90.3 g

uint8_t display_contrast_t::g

Brief description

Contrast (gain) adjustment for G channel.

12.1.90.4 b

uint8_t display_contrast_t::b

Brief description

Contrast (gain) adjustment for B channel.

12.1.91 display_coordinate_t

```
typedef struct{
    int16_t  x
    int16_t  y
} display_coordinate_t
```

12.1.91.1 x

```
int16_t display_coordinate_t::x
```

Brief description

Coordinate X, this allows to set signed value.

12.1.91.2 y

```
int16_t display_coordinate_t::y
```

Brief description

Coordinate Y, this allows to set signed value.

12.1.92 display_correction_t

```
typedef struct{
    display_brightness_t  brightness
    display_contrast_t  contrast
} display_correction_t
```

12.1.92.1 brightness

```
display_brightness_t::brightness
```

Brief description

Brightness.

12.1.92.2 contrast

```
display_contrast_t::contrast
```

Brief description

Contrast.

12.1.93 display_gamma_correction_t

```
typedef struct{
    gamma_correction_t  r
    gamma_correction_t  g
```

```
    gamma_correction_t  b
} display_gamma_correction_t
```

12.1.93.1 r

`gamma_correction_t::r`

Brief description

Gamma correction for R channel.

12.1.93.2 g

`gamma_correction_t::g`

Brief description

Gamma correction for G channel.

12.1.93.3 b

`gamma_correction_t::b`

Brief description

Gamma correction for B channel.

12.1.94 display_input_cfg_t

```
typedef struct{
    uint32_t *  p_base
    uint16_t  hsize
    uint16_t  vsize
    uint32_t  hstride
    display_in_format_t  format
    bool  line_descending_enable
    bool  lines_repeat_enable
    uint16_t  lines_repeat_times
} display_input_cfg_t
```

12.1.94.1 p_base

`uint32_t* display_input_cfg_t::p_base`

Brief description

Base address to the frame buffer.

12.1.94.2 hsize

`uint16_t display_input_cfg_t::hsize`

Brief description

Horizontal pixel size in a line.

12.1.94.3 vsize

```
uint16_t display_input_cfg_t::vsize
```

Brief description

Vertical pixel size in a frame.

12.1.94.4 hstride

```
uint32_t display_input_cfg_t::hstride
```

Brief description

Memory stride (bytes) in a line.

12.1.94.5 format

```
display_in_format_t::format
```

Brief description

Input format setting.

12.1.94.6 line_descending_enable

```
bool display_input_cfg_t::line_descending_enable
```

Brief description

Line descending enable.

12.1.94.7 lines_repeat_enable

```
bool display_input_cfg_t::lines_repeat_enable
```

Brief description

Line repeat enable.

12.1.94.8 lines_repeat_times

```
uint16_t display_input_cfg_t::lines_repeat_times
```

Brief description

Expected number of line repeating.

12.1.95 display_instance_t

```
typedef struct{
    display_ctrl_t *  p_ctrl
    display_cfg_t const  *  p_cfg
    display_api_t const  *  p_api
} display_instance_t
```

12.1.95.1 p_ctrl

display_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.95.2 p_cfg

display_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.95.3 p_api

display_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.96 display_layer_t

```
typedef struct{
    display_coordinate_t  coordinate
    display_color_t  bg_color
    display_fade_control_t  fade_control
    uint8_t  fade_speed
} display_layer_t
```

12.1.96.1 coordinate

display_coordinate_t::coordinate

Brief description

Blending location (starting point of image)

12.1.96.2 bg_color

`display_color_t::bg_color`

Brief description

Color outside region.

12.1.96.3 fade_control

`display_fade_control_t::fade_control`

Brief description

Layer fade-in/out control on/off.

12.1.96.4 fade_speed

`uint8_t display_layer_t::fade_speed`

Brief description

Layer fade-in/out frame rate.

12.1.97 display_output_cfg_t

```
typedef struct{
    display_timing_t  htiming
    display_timing_t  vtiming
    display_out_format_t  format
    display_endian_t  endian
    display_color_order_t  color_order
    display_signal_polarity_t  data_enable_polarity
    display_sync_edge_t  sync_edge
    display_color_t  bg_color
    display_brightness_t  brightness
    display_contrast_t  contrast
    display_gamma_correction_t *  p_gamma_correction
    bool  dithering_on
} display_output_cfg_t
```

12.1.97.1 htiming

`display_timing_t::htiming`

Brief description

Horizontal display cycle setting.

12.1.97.2 vtiming

`display_timing_t::vtiming`

Brief description

Vertical display cycle setting.

12.1.97.3 format

`display_out_format_t::format`

Brief description

Output format setting.

12.1.97.4 endian

`display_endian_t::endian`

Brief description

Bit order of output data.

12.1.97.5 color_order

`display_color_order_t::color_order`

Brief description

Color order in pixel.

12.1.97.6 data_enable_polarity

`display_signal_polarity_t::data_enable_polarity`

Brief description

Data Enable signal polarity.

12.1.97.7 sync_edge

`display_sync_edge_t::sync_edge`

Brief description

Signal sync edge selection.

12.1.97.8 bg_color

`display_color_t::bg_color`

Brief description

Background color.

12.1.97.9 brightness

`display_brightness_t::brightness`

Brief description

Brightness setting.

12.1.97.10 contrast

`display_contrast_t::contrast`

Brief description

Contrast setting.

12.1.97.11 p_gamma_correction

`display_gamma_correction_t::p_gamma_correction`

Brief description

Pointer to gamma correction setting.

12.1.97.12 dithering_on

`bool display_output_cfg_t::dithering_on`

Brief description

Dithering on/off.

12.1.98 display_runtime_cfg_t

```
typedef struct{
    display_input_cfg_t  input
    display_layer_t  layer
} display_runtime_cfg_t
```

12.1.98.1 input

`display_input_cfg_t::input`

Brief description

Graphics input frame setting.

Detailed description

Generic configuration for display devices

12.1.98.2 layer

`display_layer_t::layer`

Brief description

Graphics layer alpha blending setting.

12.1.99 display_status_t

```
typedef struct{
    display_state_t  state
    display_fade_status_t  fade_status[DISPLAY_FRAME_LAYER_2+1]
} display_status_t
```

12.1.99.1 state

`display_state_t::state`

Brief description

Status of GLCD module.

12.1.99.2 fade_status

`display_fade_status_t::fade_status`

Brief description

Status of fade-in/fade-out status.

12.1.100 display_timing_t

```
typedef struct{
    uint16_t  total_cyc
    uint16_t  display_cyc
    uint16_t  back_porch
    uint16_t  sync_width
    display_signal_polarity_t  sync_polarity
} display_timing_t
```

12.1.100.1 total_cyc

`uint16_t display_timing_t::total_cyc`

Brief description

Total cycles in one line or total lines in one frame.

12.1.100.2 display_cyc

`uint16_t display_timing_t::display_cyc`

Brief description

Active video cycles or lines.

12.1.100.3 back_porch

`uint16_t display_timing_t::back_porch`

Brief description

Back poach cycles or lines.

12.1.100.4 sync_width

`uint16_t display_timing_t::sync_width`

Brief description

Sync signal asserting width.

12.1.100.5 sync_polarity

`display_signal_polarity_t::sync_polarity`

Brief description

Sync signal polarity.

12.1.101 dmac_instance_ctrl_t

```
typedef struct{
    uint32_t  id
    elc_event_t  trigger
    IRQn_Type  irq
    uint8_t  channel
    void(*  p_callback)(transfer_callback_args_t *cb_data)
    void const  *  p_context
    void *  p_reg
} dmac_instance_ctrl_t
```

12.1.101.1 id

`uint32_t ::id`

Brief description

Driver ID.

12.1.101.2 trigger

`elc_event_t::trigger`

Brief description

Transfer activation event. Matches event returned by infoGet.

12.1.101.3 irq

`IRQn_Type ::irq`

Brief description

Transfer activation IRQ.

12.1.101.4 channel

`uint8_t ::channel`

Brief description

Channel number.

12.1.101.5 p_callback

`void(* ::p_callback) ( *cb_data)`

Detailed description

Callback for transfer end interrupt.

12.1.101.6 p_context

`void const* ::p_context`

Detailed description

Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

12.1.101.7 p_reg

`void* ::p_reg`

Detailed description

Pointer to base register.

12.1.102 doc_api_t

```
typedef struct{
    ssp_err_t(*  open)(doc_ctrl_t *const p_ctrl, doc_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(doc_ctrl_t *const p_ctrl)
    ssp_err_t(*  statusGet)(doc_ctrl_t *const p_ctrl, doc_status_t *p_status)
    ssp_err_t(*  statusClear)(doc_ctrl_t *const p_ctrl)
    ssp_err_t(*  write)(doc_ctrl_t *const p_ctrl, doc_data_t *const p_data)
    ssp_err_t(*  inputRegisterWrite)(doc_ctrl_t *const p_ctrl, doc_size_t data)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} doc_api_t
```

12.1.103 doc_callback_args_t

```
typedef struct{
    doc_event_t  event
    void const  *  p_context
} doc_callback_args_t
```

12.1.103.1 event

`doc_event_t::event`

Brief description

The event is used to identify what caused the callback.

12.1.103.2 p_context

`void const* doc_callback_args_t::p_context`

Detailed description

Placeholder for user data. Set in open function in doc_cfg_t.

12.1.104 doc_cfg_t

```
typedef struct{
    doc_event_t  event
    uint8_t  irq_ipl
    void(*  p_callback)(doc_callback_args_t *p_args)
    void const  *  p_context
} doc_cfg_t
```

12.1.104.1 event

`doc_event_t::event`

Brief description

Select enumerated value from doc_event_t.

12.1.104.2 irq_ipl

`uint8_t doc_cfg_t::irq_ipl`

Brief description

DOC interrupt priority.

12.1.104.3 p_callback

`void(* doc_cfg_t::p_callback) (doc_callback_args_t *p_args)`

Detailed description

Callback provided when a DOC ISR occurs.

12.1.104.4 p_context

```
void const* doc_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in doc_callback_args_t.

12.1.105 doc_data_t

```
typedef struct{
    doc_size_t  dodir
    doc_size_t  dodsr
} doc_data_t
```

12.1.105.1 dodir

```
doc_size_t::dodir
```

Brief description

Value to be written to the DOC DODIR.

12.1.105.2 dodsr

```
doc_size_t::dodsr
```

Brief description

Value to be written to the DOC DODSR.

12.1.106 doc_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    void(*  p_callback)(doc_callback_args_t *p_args)
    doc_event_t  event
    void const  *  p_context
    void *  p_reg
} doc_instance_ctrl_t
```

12.1.106.1 open

```
uint32_t ::open
```

Brief description

Used by driver to check if the control structure is valid.

12.1.106.2 p_callback

```
void(* ::p_callback) ( *p_args)
```

Detailed description

Callback provided when a DOC ISR occurs. NULL indicates no CPU interrupt.

12.1.106.3 event

```
doc_event_t::event
```

Brief description

The event DOC is configured for. Passed in ISR callback.

12.1.106.4 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in doc_callback_args_t.

12.1.106.5 p_reg

```
void* ::p_reg
```

Brief description

Base register.

12.1.107 doc_instance_t

```
typedef struct{
    doc_ctrl_t *  p_ctrl
    doc_cfg_t const  *  p_cfg
    doc_api_t const  *  p_api
} doc_instance_t
```

12.1.107.1 p_ctrl

```
doc_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.107.2 p_cfg

```
doc_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.107.3 p_api

`doc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.108 dsa_api_t

```
typedef struct{
    uint32_t(*  open)(dsa_ctrl_t *const p_ctrl, dsa_cfg_t const *const p_cfg)
    uint32_t(*  verify)(const uint32_t *p_key, const uint32_t *p_domain, uint32_t
num_words, uint32_t *p_signature, uint32_t *p_paddedHash)
    uint32_t(*  hashVerify)(dsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t
*p_paddedHash)
    uint32_t(*  sign)(const uint32_t *p_key, const uint32_t *p_domain, uint32_t
num_words, uint32_t *p_paddedHash, uint32_t *p_dest)
    uint32_t(*  hashSign)(dsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_paddedHash, uint32_t *p_dest)
    uint32_t(*  close)(dsa_ctrl_t *const p_ctrl)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} dsa_api_t
```

12.1.109 dsa_cfg_t

```
typedef struct{
    crypto_api_t const  *  p_crypto_api
} dsa_cfg_t
```

12.1.109.1 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine api

12.1.110 dsa_ctrl_t

```
typedef struct{
    crypto_ctrl_t *  p_crypto_ctrl
    crypto_api_t const  *  p_crypto_api
} dsa_ctrl_t
```

12.1.110.1 p_crypto_ctrl

`crypto_ctrl_t::p_crypto_ctrl`

Brief description

pointer to crypto engine control structure

12.1.110.2 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine API

12.1.111 dsa_domain_1024_160_t

```
typedef struct{
    uint32_t  q[(160/32)]
    uint32_t  p[(1024/32)]
    uint32_t  g[(1024/32)]
} dsa_domain_1024_160_t
```

12.1.111.1 q

`uint32_t ::q[(160/32)]`

Brief description

DSA (1024,160) domain parameter Q.

12.1.111.2 p

`uint32_t ::p[(1024/32)]`

Brief description

DSA (1024,160) domain parameter P.

12.1.111.3 g

`uint32_t ::g[(1024/32)]`

Brief description

DSA (1024,160) domain parameter G.

12.1.112 dsa_domain_2048_224_t

```
typedef struct{
    uint32_t  q[(224/32)]
    uint32_t  p[(2048/32)]
```

```
    uint32_t  g[(2048/32)]
} dsa_domain_2048_224_t
```

12.1.112.1 q

`uint32_t ::q[(224/32)]`

Brief description

DSA (2048,224) domain parameter Q.

12.1.112.2 p

`uint32_t ::p[(2048/32)]`

Brief description

DSA (2048,224) domain parameter P.

12.1.112.3 g

`uint32_t ::g[(2048/32)]`

Brief description

DSA (2048,224) domain parameter G.

12.1.113 dsa_domain_2048_256_t

```
typedef struct{
    uint32_t  q[(256/32)]
    uint32_t  p[(2048/32)]
    uint32_t  g[(2048/32)]
} dsa_domain_2048_256_t
```

12.1.113.1 q

`uint32_t ::q[(256/32)]`

Brief description

DSA (2048,256) domain parameter Q.

12.1.113.2 p

`uint32_t ::p[(2048/32)]`

Brief description

DSA (2048,256) domain parameter P.

12.1.113.3 g

```
uint32_t ::g[(2048/32)]
```

Brief description

DSA (2048,256) domain parameter G.

12.1.114 dsa_instance_t

```
typedef struct{
    dsa_ctrl_t *  p_ctrl
    dsa_cfg_t const  *  p_cfg
    dsa_api_t const  *  p_api
} dsa_instance_t
```

12.1.114.1 p_ctrl

```
dsa_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.114.2 p_cfg

```
dsa_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.114.3 p_api

```
dsa_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.115 dsa_signature_1024_160_t

```
typedef struct{
    uint32_t  r[(160/32)]
    uint32_t  s[(160/32)]
} dsa_signature_1024_160_t
```

12.1.115.1 r

```
uint32_t ::r[(160/32)]
```

Brief description

DSA (1024,160) signature component R.

12.1.115.2 s

```
uint32_t ::s[(160/32)]
```

Brief description

DSA (1024,160) signature component S.

12.1.116 dsa_signature_2048_224_t

```
typedef struct{
    uint32_t  r[(224/32)]
    uint32_t  s[(224/32)]
} dsa_signature_2048_224_t
```

12.1.116.1 r

```
uint32_t ::r[(224/32)]
```

Brief description

DSA (2048,224) signature component R.

12.1.116.2 s

```
uint32_t ::s[(224/32)]
```

Brief description

DSA (2048,224) signature component S.

12.1.117 dsa_signature_2048_256_t

```
typedef struct{
    uint32_t  r[(256/32)]
    uint32_t  s[(256/32)]
} dsa_signature_2048_256_t
```

12.1.117.1 r

```
uint32_t ::r[(256/32)]
```

Brief description

DSA (2048,256) signature component R.

12.1.117.2 s

```
uint32_t ::s[(256/32)]
```

Brief description

DSA (2048,256) signature component S.

12.1.118 dtc_instance_ctrl_t

```
typedef struct{
    uint32_t  id
    elc_event_t  trigger
    IRQn_Type  irq
    void(*  p_callback)(transfer_callback_args_t *cb_data)
    void const  *  p_context
} dtc_instance_ctrl_t
```

12.1.118.1 id

`uint32_t ::id`

Brief description

Driver ID.

12.1.118.2 trigger

`elc_event_t::trigger`

Brief description

Transfer activation event. Matches event returned by infoGet.

12.1.118.3 irq

`IRQn_Type ::irq`

Brief description

Transfer activation IRQ, does not apply to all HAL drivers.

12.1.118.4 p_callback

`void(* ::p_callback) ( *cb_data)`

Detailed description

Callback for transfer end interrupt used for ELC software trigger.

12.1.118.5 p_context

`void const* ::p_context`

Detailed description

Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

12.1.119 dtc_reg_t

```
typedef struct{
    uint32_t  __pad0__
    uint8_t  MRB
    uint8_t  __pad0__
    uint8_t  DM
    uint8_t  DTS
    uint8_t  DISEL
    uint8_t  CHNS
    uint8_t  CHNE
    struct{}                MRB_b
    uint8_t  MRA
    uint8_t  SM
    uint8_t  SZ
    uint8_t  MD
    struct{}                MRA_b
    struct{}                struct{}
    void *volatile  SAR
    void *volatile  DAR
    uint16_t  CRB
    uint16_t  CRA
    uint8_t  CRAL
    uint8_t  CRAH
    struct{}                CRA_b
    struct{}                struct{}
} dtc_reg_t
```

12.1.119.1 __pad0__

```
uint32_t ::__pad0__
```

12.1.119.2 MRB

```
uint8_t ::MRB
```

Detailed description

Mode Register B

12.1.119.3 __pad0__

```
uint8_t ::__pad0__
```

12.1.119.4 DM

```
uint8_t ::DM
```

Brief description

Transfer Destination Address mode.

12.1.119.5 DTS

`uint8_t ::DTS`

Brief description

DTC Transfer Mode Select.

12.1.119.6 DISEL

`uint8_t ::DISEL`

Brief description

DTC Interrupt Select.

12.1.119.7 CHNS

`uint8_t ::CHNS`

Brief description

DTC Chain Transfer Select.

12.1.119.8 CHNE

`uint8_t ::CHNE`

Brief description

DTC CHain Transfer Enable.

12.1.119.9 MRB_b

`See source code for the definition of this member.`

Detailed description

- MRB bits */

12.1.119.10 MRA

`uint8_t ::MRA`

Detailed description

Mode Register A

12.1.119.11 SM

`uint8_t ::SM`

Brief description

Transfer Source Address mode.

12.1.119.12 SZ

`uint8_t ::SZ`

Brief description

DTC Data Transfer Size.

12.1.119.13 MD

`uint8_t ::MD`

Brief description

DTC Transfer Mode Select.

12.1.119.14 MRA_b

`See source code for the definition of this member.`

Detailed description

- MRA bits */

12.1.119.15 struct{}

`See source code for the definition of this member.`

Detailed description

- Mode registers */

12.1.119.16 SAR

`void* volatile ::SAR`

Brief description

Source address register.

12.1.119.17 DAR

`void* volatile ::DAR`

Detailed description

Destination address register

12.1.119.18 CRB

`volatile uint16_t ::CRB`

Detailed description

Transfer count register B

12.1.119.19 CRA

`uint16_t ::CRA`

Detailed description

Transfer count register A

12.1.119.20 CRAL

`uint8_t ::CRAL`

Brief description

Transfer counter A lower register.

12.1.119.21 CRAH

`uint8_t ::CRAH`

Brief description

Transfer counter B upper register.

12.1.119.22 CRA_b

`See source code for the definition of this member.`

Detailed description

- bits */

12.1.119.23 struct{}

`See source code for the definition of this member.`

Detailed description

- Transfer count registers */

12.1.120 elc_api_t

```
typedef struct{
    ssp_err_t(*  init)(elc_cfg_t const *const p_cfg)
    ssp_err_t(*  softwareEventGenerate)(elc_software_event_t event_num)
    ssp_err_t(*  linkSet)(elc_peripheral_t peripheral, elc_event_t signal)
    ssp_err_t(*  linkBreak)(elc_peripheral_t peripheral)
    ssp_err_t(*  enable)(void)
    ssp_err_t(*  disable)(void)
```

```
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} elc_api_t
```

12.1.121 elc_cfg_t

```
typedef struct{
    bool  autostart
    uint32_t  link_count
    elc_link_t const  *  link_list
} elc_cfg_t
```

12.1.121.1 autostart

`bool elc_cfg_t::autostart`

Brief description

Start operation and enable interrupts during open().

12.1.121.2 link_count

`uint32_t elc_cfg_t::link_count`

Brief description

Number of event links.

12.1.121.3 link_list

`elc_link_t::link_list`

Brief description

Event links.

12.1.122 elc_instance_t

```
typedef struct{
    elc_cfg_t const  *  p_cfg
    elc_api_t const  *  p_api
} elc_instance_t
```

12.1.122.1 p_cfg

`elc_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.122.2 p_api

`elc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.123 elc_link_t

```
typedef struct{
    elc_peripheral_t  peripheral
    elc_event_t  event
} elc_link_t
```

12.1.123.1 peripheral

`elc_peripheral_t::peripheral`

Brief description

Peripheral to receive the signal.

12.1.123.2 event

`elc_event_t::event`

Brief description

Signal that gets sent to the Peripheral.

12.1.124 external_irq_api_t

```
typedef struct{
    ssp_err_t(*  open)(external_irq_ctrl_t *const p_ctrl, external_irq_cfg_t
const *const p_cfg)
    ssp_err_t(*  enable)(external_irq_ctrl_t *const p_ctrl)
    ssp_err_t(*  disable)(external_irq_ctrl_t *const p_ctrl)
    ssp_err_t(*  triggerSet)(external_irq_ctrl_t *const p_ctrl,
external_irq_trigger_t const trigger)
    ssp_err_t(*  filterEnable)(external_irq_ctrl_t *const p_ctrl)
    ssp_err_t(*  filterDisable)(external_irq_ctrl_t *const p_ctrl)
    ssp_err_t(*  close)(external_irq_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} external_irq_api_t
```

12.1.125 external_irq_callback_args_t

```
typedef struct{
    void const  *  p_context
```

```
    uint32_t  channel
} external_irq_callback_args_t
```

12.1.125.1 p_context

```
void const* external_irq_callback_args_t::p_context
```

Detailed description

Placeholder for user data. Set in open function in external_irq_cfg_t.

12.1.125.2 channel

```
uint32_t external_irq_callback_args_t::channel
```

Brief description

The physical hardware channel that caused the interrupt.

12.1.126 external_irq_cfg_t

```
typedef struct{
    uint8_t  channel
    uint8_t  irq_ipl
    external_irq_trigger_t  trigger
    external_irq_pclk_div_t  pclk_div
    bool  autostart
    bool  filter_enable
    void(*  p_callback)(external_irq_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} external_irq_cfg_t
```

12.1.126.1 channel

```
uint8_t external_irq_cfg_t::channel
```

Brief description

Hardware channel used.

12.1.126.2 irq_ipl

```
uint8_t external_irq_cfg_t::irq_ipl
```

Brief description

Interrupt priority.

12.1.126.3 trigger

`external_irq_trigger_t::trigger`

Brief description

Trigger setting.

12.1.126.4 pclk_div

`external_irq_pclk_div_t::pclk_div`

Brief description

Digital filter clock divisor setting.

12.1.126.5 autostart

`bool external_irq_cfg_t::autostart`

Brief description

Start operation and enable interrupts during open().

12.1.126.6 filter_enable

`bool external_irq_cfg_t::filter_enable`

Brief description

Digital filter enable/disable setting.

12.1.126.7 p_callback

```
void(* external_irq_cfg_t::p_callback) (external_irq_callback_args_t *p_args)
```

Detailed description

Callback provided external input trigger occurs.

12.1.126.8 p_context

`void const* external_irq_cfg_t::p_context`

Detailed description

Placeholder for user data. Passed to the user callback in external_irq_callback_args_t.

12.1.126.9 p_extend

`void const* external_irq_cfg_t::p_extend`

Brief description

External IRQ hardware dependent configuration.

12.1.127 external_irq_instance_t

```
typedef struct{
    external_irq_ctrl_t *  p_ctrl
    external_irq_cfg_t const  *  p_cfg
    external_irq_api_t const  *  p_api
} external_irq_instance_t
```

12.1.127.1 p_ctrl

external_irq_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.127.2 p_cfg

external_irq_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.127.3 p_api

external_irq_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.128 flash_api_t

```
typedef struct{
    ssp_err_t(*  open)(flash_ctrl_t *const p_ctrl, flash_cfg_t const *const
p_cfg)
    ssp_err_t(*  write)(flash_ctrl_t *const p_ctrl, uint32_t const src_address,
uint32_t const flash_address, uint32_t const num_bytes)
    ssp_err_t(*  read)(flash_ctrl_t *const p_ctrl, uint8_t *const p_dest_address,
uint32_t const flash_address, uint32_t const num_bytes)
    ssp_err_t(*  erase)(flash_ctrl_t *const p_ctrl, uint32_t const address,
uint32_t const num_blocks)
    ssp_err_t(*  blankCheck)(flash_ctrl_t *const p_ctrl, uint32_t const address,
uint32_t const num_bytes, flash_result_t *const p_blank_check_result)
    ssp_err_t(*  infoGet)(flash_ctrl_t *const p_ctrl, flash_info_t *const p_info)
    ssp_err_t(*  close)(flash_ctrl_t *const p_ctrl)
    ssp_err_t(*  statusGet)(flash_ctrl_t *const p_ctrl)
    ssp_err_t(*  accessWindowSet)(flash_ctrl_t *const p_ctrl, uint32_t const
start_addr, uint32_t const end_addr)
    ssp_err_t(*  accessWindowClear)(flash_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  reset)(flash_ctrl_t *const p_ctrl)
    ssp_err_t(*  updateFlashClockFreq)(flash_ctrl_t *const p_ctrl)
    ssp_err_t(*  startupAreaSelect)(flash_ctrl_t *const p_ctrl,
flash_startup_area_swap_t swap_type, bool is_temporary)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} flash_api_t
```

12.1.129 flash_callback_args_t

```
typedef struct{
    flash_event_t  event
    void const  *  p_context
} flash_callback_args_t
```

12.1.129.1 event

`flash_event_t::event`

Brief description

Event can be used to identify what caused the callback (flash ready or error).

12.1.129.2 p_context

`void const* flash_callback_args_t::p_context`

Brief description

Placeholder for user data. Set in open function in::flash_cfg_t.

12.1.130 flash_cfg_t

```
typedef struct{
    bool  data_flash_bgo
    void(*  p_callback)(flash_callback_args_t *p_args)
    void const  *  p_extend
    void const  *  p_context
    uint8_t  irq_ipl
    uint8_t  err_irq_ipl
} flash_cfg_t
```

12.1.130.1 data_flash_bgo

`bool flash_cfg_t::data_flash_bgo`

Brief description

True if BGO (Background Operation) is enabled for Data Flash.

12.1.130.2 p_callback

```
void(* flash_cfg_t::p_callback) (flash_callback_args_t *p_args)
```

Brief description

Callback provided when a Flash interrupt ISR occurs.

12.1.130.3 p_extend

```
void const* flash_cfg_t::p_extend
```

Brief description

FLASH hardware dependent configuration.

12.1.130.4 p_context

```
void const* flash_cfg_t::p_context
```

Brief description

Placeholder for user data. Passed to user callback in flash_callback_args_t.

12.1.130.5 irq_ipl

```
uint8_t flash_cfg_t::irq_ipl
```

Brief description

Flash ready interrupt priority.

12.1.130.6 err_irq_ipl

```
uint8_t flash_cfg_t::err_irq_ipl
```

Brief description

Flash error interrupt priority (unused in r_flash_lp)

12.1.131 flash_fmi_block_info_t

```
typedef struct{
    uint32_t  block_section_st_addr
    uint32_t  block_section_end_addr
    uint32_t  block_size
    uint32_t  block_size_write
} flash_fmi_block_info_t
```

12.1.131.1 block_section_st_addr

```
uint32_t flash_fmi_block_info_t::block_section_st_addr
```

Brief description

starting address for this block section (blocks of this size)

12.1.131.2 block_section_end_addr

`uint32_t flash_fmi_block_info_t::block_section_end_addr`

Brief description

ending address for this block section (blocks of this size)

12.1.131.3 block_size

`uint32_t flash_fmi_block_info_t::block_size`

Brief description

Flash erase block size.

12.1.131.4 block_size_write

`uint32_t flash_fmi_block_info_t::block_size_write`

Brief description

Flash write block size.

12.1.132 flash_fmi_regions_t

```
typedef struct{
    uint32_t  num_regions
    flash_fmi_block_info_t const  *  p_block_array
} flash_fmi_regions_t
```

12.1.132.1 num_regions

`uint32_t flash_fmi_regions_t::num_regions`

Brief description

Length of block info array.

12.1.132.2 p_block_array

`flash_fmi_block_info_t::p_block_array`

Brief description

Block info array base address.

12.1.133 flash_hp_instance_ctrl_t

```
typedef struct{
    uint32_t  opened
    R_FACI_Type *  p_reg
    void(*  p_callback)(flash_callback_args_t *p_args)
    bsp_cache_state_t  cache_state
    IRQn_Type  irq
    IRQn_Type  err_irq
} flash_hp_instance_ctrl_t
```

12.1.133.1 opened

`uint32_t ::opened`

Brief description

To check whether api has been opened or not.

12.1.133.2 p_reg

`R_FACI_Type* ::p_reg`

Brief description

Base address of flash registers.

12.1.133.3 p_callback

`void(* ::p_callback) ( *p_args)`

12.1.133.4 cache_state

`bsp_cache_state_t::cache_state`

Brief description

User Callback function.

Detailed description

Used to disable and then restore Flash Cache while API is open.

12.1.133.5 irq

`IRQn_Type ::irq`

Brief description

Flash ready interrupt number.

12.1.133.6 err_irq

`IRQn_Type ::err_irq`

Brief description

Flash error interrupt number.

12.1.134 flash_info_t

```
typedef struct{
    flash_fmi_regions_t  code_flash
    flash_fmi_regions_t  data_flash
} flash_info_t
```

12.1.134.1 code_flash

`flash_fmi_regions_t::code_flash`

Brief description

Information about the code flash regions.

12.1.134.2 data_flash

`flash_fmi_regions_t::data_flash`

Brief description

Information about the code flash regions.

12.1.135 flash_instance_t

```
typedef struct{
    flash_ctrl_t *  p_ctrl
    flash_cfg_t const  *  p_cfg
    flash_api_t const  *  p_api
} flash_instance_t
```

12.1.135.1 p_ctrl

`flash_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.135.2 p_cfg

`flash_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.135.3 p_api

`flash_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.136 flash_lp_instance_ctrl_t

```
typedef struct{
    uint32_t  opened
    void *  p_reg
    void(*  p_callback)(flash_callback_args_t *p_args)
    bsp_cache_state_t  cache_state
    IRQn_Type  irq
} flash_lp_instance_ctrl_t
```

12.1.136.1 opened

`uint32_t ::opened`

Brief description

To check whether api has been opened or not.

12.1.136.2 p_reg

`void* ::p_reg`

Brief description

Base address of flash registers.

12.1.136.3 p_callback

`void(* ::p_callback) ( *p_args)`

12.1.136.4 cache_state

`bsp_cache_state_t::cache_state`

Brief description

Used to disable and then restore Flash Cache while API is open.

12.1.136.5 irq

`IRQn_Type ::irq`

Brief description

Flash ready interrupt number.

12.1.137 fmi_api_t

```
typedef struct{
    ssp_err_t(*  init)(void)
    ssp_err_t(*  productInfoGet)(fmi_product_info_t **pp_product_info)
    ssp_err_t(*  uniqueIdGet)(fmi_unique_id_t *p_unique_id)
    ssp_err_t(*  productFeatureGet)(ssp_feature_t const *const p_feature,
fmi_feature_info_t *const p_info)
    ssp_err_t(*  eventInfoGet)(ssp_feature_t const *const p_feature, ssp_signal_t
signal, fmi_event_info_t *const p_info)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} fmi_api_t
```

12.1.138 fmi_event_info_t

```
typedef struct{
    IRQn_Type  irq
    elc_event_t  event
} fmi_event_info_t
```

12.1.138.1 irq

```
IRQn_Type fmi_event_info_t::irq
```

12.1.138.2 event

```
elc_event_t::event
```

12.1.139 fmi_feature_info_t

```
typedef struct{
    void *  ptr
    uint32_t  channel_count
    uint32_t  variant_data
    uint32_t  extended_data_count
    uint32_t  version_major
    uint32_t  version_minor
    struct{}                struct{}
    void *  ptr_extended_data
} fmi_feature_info_t
```

12.1.139.1 ptr

```
void* fmi_feature_info_t::ptr
```

12.1.139.2 channel_count

```
uint32_t fmi_feature_info_t::channel_count
```

12.1.139.3 variant_data

```
uint32_t fmi_feature_info_t::variant_data
```

12.1.139.4 extended_data_count

```
uint32_t fmi_feature_info_t::extended_data_count
```

12.1.139.5 version_major

```
uint32_t fmi_feature_info_t::version_major
```

12.1.139.6 version_minor

```
uint32_t fmi_feature_info_t::version_minor
```

12.1.139.7 struct{}

```
See source code for the definition of this member.
```

12.1.139.8 ptr_extended_data

```
void* fmi_feature_info_t::ptr_extended_data
```

12.1.140 fmi_header_t

```
typedef struct{
    uint32_t  contents
    uint32_t  variant
    uint32_t  count
    uint32_t  minor
    uint32_t  major
} fmi_header_t
```

12.1.140.1 contents

```
uint32_t fmi_header_t::contents
```

12.1.140.2 variant

```
uint32_t fmi_header_t::variant
```

12.1.140.3 count

```
uint32_t fmi_header_t::count
```

12.1.140.4 minor

```
uint32_t fmi_header_t::minor
```

12.1.140.5 major

```
uint32_t fmi_header_t::major
```

12.1.141 fmi_instance_t

```
typedef struct{
    fmi_api_t const  *  p_api
} fmi_instance_t
```

12.1.141.1 p_api

```
fmi_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.142 fmi_product_info_t

```
typedef struct{
    fmi_header_t  header
    uint8_t  unique_id[16]
    uint8_t  product_name[16]
    uint8_t  product_marking[16]
    uint32_t  mask_revision
    uint32_t  pin_count
    uint32_t  pkg_type
    uint32_t  temp_range
    uint32_t  quality_code
    uint32_t  reserved
    struct{}                struct{}
    uint32_t  max_freq
    uint32_t  reserved1
```

```
    struct{}                struct{}
} fmi_product_info_t
```

12.1.142.1 header

`fmi_header_t::header`

12.1.142.2 unique_id

`uint8_t fmi_product_info_t::unique_id[16]`

Brief description

DEPRECATED, use uniqueIdGet instead.

12.1.142.3 product_name

`uint8_t fmi_product_info_t::product_name[16]`

12.1.142.4 product_marking

`uint8_t fmi_product_info_t::product_marking[16]`

12.1.142.5 mask_revision

`uint32_t fmi_product_info_t::mask_revision`

12.1.142.6 pin_count

`uint32_t fmi_product_info_t::pin_count`

12.1.142.7 pkg_type

`uint32_t fmi_product_info_t::pkg_type`

12.1.142.8 temp_range

`uint32_t fmi_product_info_t::temp_range`

12.1.142.9 quality_code

`uint32_t fmi_product_info_t::quality_code`

12.1.142.10 reserved

`uint32_t fmi_product_info_t::reserved`

12.1.142.11 struct{}

See source code for the definition of this member.

12.1.142.12 max_freq

uint32_t fmi_product_info_t::max_freq

12.1.142.13 reserved1

uint32_t fmi_product_info_t::reserved1

12.1.142.14 struct{}

See source code for the definition of this member.

12.1.143 fmi_unique_id_t

```
typedef struct{
    uint32_t  unique_id[4]
} fmi_unique_id_t
```

12.1.143.1 unique_id

uint32_t fmi_unique_id_t::unique_id[4]

12.1.144 gamma_correction_t

```
typedef struct{
    bool  enable
    uint16_t  gain[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]
    uint16_t  threshold[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]
} gamma_correction_t
```

12.1.144.1 enable

bool gamma_correction_t::enable

Brief description

Gamma Correction On/Off.

12.1.144.2 gain

uint16_t gamma_correction_t::gain[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]

Brief description

Gain adjustment.

12.1.144.3 threshold

```
uint16_t gamma_correction_t::threshold[DISPLAY_GAMMA_CURVE_ELEMENT_NUM]
```

Brief description

Start threshold.

12.1.145 glcd_cfg_t

```
typedef struct{
    glcd_tcon_pin_t  tcon_hsync
    glcd_tcon_pin_t  tcon_vsync
    glcd_tcon_pin_t  tcon_de
    glcd_correction_proc_order_t  correction_proc_order
    glcd_clk_src_t  clksrc
    glcd_panel_clk_div_t  clock_div_ratio
    glcd_dithering_mode_t  dithering_mode
    glcd_dithering_pattern_t  dithering_pattern_A
    glcd_dithering_pattern_t  dithering_pattern_B
    glcd_dithering_pattern_t  dithering_pattern_C
    glcd_dithering_pattern_t  dithering_pattern_D
} glcd_cfg_t
```

12.1.145.1 tcon_hsync

```
glcd_tcon_pin_t::tcon_hsync
```

Brief description

GLCD TCON output pin select.

12.1.145.2 tcon_vsync

```
glcd_tcon_pin_t::tcon_vsync
```

Brief description

GLCD TCON output pin select.

12.1.145.3 tcon_de

```
glcd_tcon_pin_t::tcon_de
```

Brief description

GLCD TCON output pin select.

12.1.145.4 correction_proc_order

`glcd_correction_proc_order_t::correction_proc_order`

Brief description

Correction control route select.

12.1.145.5 clksrc

`glcd_clk_src_t::clksrc`

Brief description

Clock Source selection.

12.1.145.6 clock_div_ratio

`glcd_panel_clk_div_t::clock_div_ratio`

Brief description

Clock divide ratio for dot clock.

12.1.145.7 dithering_mode

`glcd_dithering_mode_t::dithering_mode`

Brief description

Dithering mode.

12.1.145.8 dithering_pattern_A

`glcd_dithering_pattern_t::dithering_pattern_A`

Brief description

Dithering pattern A.

12.1.145.9 dithering_pattern_B

`glcd_dithering_pattern_t::dithering_pattern_B`

Brief description

Dithering pattern B.

12.1.145.10 dithering_pattern_C

`glcd_dithering_pattern_t::dithering_pattern_C`

Brief description

Dithering pattern C.

12.1.145.11 dithering_pattern_D

`glcd_dithering_pattern_t::dithering_pattern_D`

Brief description

Dithering pattern D.

12.1.146 glcd_ctrl_t

```
typedef struct{
    display_coordinate_t  back_porch
    uint16_t  hsize
    uint16_t  vsize
    bsp_lock_t  resource_lock
    void *  p_context
} glcd_ctrl_t
```

12.1.146.1 back_porch

`display_coordinate_t::back_porch`

Brief description

Zero coordinate for graphics plane(Bach porch End)

12.1.146.2 hsize

`uint16_t ::hsize`

Brief description

Horizontal pixel size in a line.

12.1.146.3 vsize

`uint16_t ::vsize`

Brief description

Vertical pixel size in a frame.

12.1.146.4 resource_lock

`bsp_lock_t::resource_lock`

Brief description

Resource lock.

12.1.146.5 p_context

`void* ::p_context`

Detailed description

Pointer to the function level device context (e.g. display_ctrl_t type data)

12.1.147 glcd_instance_ctrl_t

```
typedef struct{
    display_state_t  state
    void(*  p_callback)(display_callback_args_t *p_args)
    void const  *  p_context
    R_GLCDC_Type *  p_reg
} glcd_instance_ctrl_t
```

12.1.147.1 state

`display_state_t::state`

Brief description

Status of GLCD module.

12.1.147.2 p_callback

`void(* ::p_callback) ( *p_args)`

Brief description

Pointer to callback function.

12.1.147.3 p_context

`void const* ::p_context`

Brief description

Pointer to the higher level device context.

12.1.147.4 p_reg

`R_GLCDC_Type* ::p_reg`

Brief description

Base register address.

12.1.148 gpt_input_capture_extend_t

```
typedef struct{
    gpt_input_capture_signal_t  signal
    gpt_input_capture_signal_filter_t  signal_filter
    gpt_input_capture_clock_divider_t  clock_divider
    input_capture_signal_level_t  enable_level
```

```
    gpt_input_capture_signal_filter_t  enable_filter
} gpt_input_capture_extend_t
```

12.1.148.1 signal

`gpt_input_capture_signal_t::signal`

Brief description

One of gpt_input_capture_signal_t.

12.1.148.2 signal_filter

`gpt_input_capture_signal_filter_t::signal_filter`

Brief description

One of gpt_input_capture_signal_filter_t.

12.1.148.3 clock_divider

`gpt_input_capture_clock_divider_t::clock_divider`

Brief description

One of gpt_input_capture_clock_divider_t.

12.1.148.4 enable_level

`input_capture_signal_level_t::enable_level`

Detailed description

The unused GTIOCx pin can be used as an enable signal to enable captures. If the Input Capture Signal Pin is GTIOCA, then the enable pin is GTIOCB. The enable level is set here if used.

12.1.148.5 enable_filter

`gpt_input_capture_signal_filter_t::enable_filter`

Brief description

One of gpt_input_capture_signal_filter_t.

12.1.149 gpt_input_capture_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    uint8_t  channel
    input_capture_mode_t  mode
    input_capture_repetition_t  repetition
    uint32_t  capture_count
    uint32_t  overflows_last
```

```
    uint32_t  overflows_current
    void(*  p_callback)(input_capture_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    IRQn_Type  capture_irq
    IRQn_Type  overflow_irq
    input_capture_variant_t  variant
} gpt_input_capture_instance_ctrl_t
```

12.1.149.1 open

`uint32_t ::open`

Brief description

Whether or not channel is open.

12.1.149.2 channel

`uint8_t ::channel`

Brief description

The channel in use.

12.1.149.3 mode

`input_capture_mode_t::mode`

Brief description

The mode of measurement being performed.

12.1.149.4 repetition

`input_capture_repetition_t::repetition`

Brief description

One-shot or periodic measurement.

12.1.149.5 capture_count

`uint32_t ::capture_count`

Brief description

The value of the timer captured at the time of interrupt.

12.1.149.6 overflows_last

`uint32_t ::overflows_last`

Brief description

Overflow count that occurred during last measurement.

12.1.149.7 overflows_current

```
uint32_t ::overflows_current
```

Brief description

Running count of overflows in current measurement.

12.1.149.8 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Pointer to user callback.

12.1.149.9 p_context

```
void const* ::p_context
```

Brief description

Pointer to user's context data, to be passed to the callback function.

12.1.149.10 p_reg

```
void* ::p_reg
```

Brief description

GPT base register for this channel.

12.1.149.11 capture_irq

```
IRQn_Type ::capture_irq
```

Brief description

Capture IRQ number.

12.1.149.12 overflow_irq

```
IRQn_Type ::overflow_irq
```

Brief description

Overflow IRQ number.

12.1.149.13 variant

```
input_capture_variant_t::variant
```

Brief description

Timer variant.

12.1.150 gpt_instance_ctrl_t

```
typedef struct{
    void(*  p_callback)(timer_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    uint32_t  open
    uint8_t  channel
    bool  one_shot
    IRQn_Type  irq
    timer_variant_t  variant
} gpt_instance_ctrl_t
```

12.1.150.1 p_callback

`void(* ::p_callback) ( *p_args)`

Detailed description

Callback provided when a timer ISR occurs. NULL indicates no CPU interrupt.

12.1.150.2 p_context

`void const* ::p_context`

Detailed description

Placeholder for user data. Passed to the user callback in timer_callback_args_t.

12.1.150.3 p_reg

`void* ::p_reg`

Brief description

Base register for this channel.

12.1.150.4 open

`uint32_t ::open`

Brief description

Whether or not channel is open.

12.1.150.5 channel

`uint8_t ::channel`

Brief description

Channel number.

12.1.150.6 one_shot

`bool ::one_shot`

Brief description

Whether or not timer is in one shot mode.

12.1.150.7 irq

`IRQn_Type ::irq`

Brief description

Counter overflow IRQ number.

12.1.150.8 variant

`timer_variant_t::variant`

Brief description

Timer variant.

12.1.151 gpt_output_pin_t

```
typedef struct{
    bool  output_enabled
    gpt_pin_level_t  stop_level
} gpt_output_pin_t
```

12.1.151.1 output_enabled

`bool ::output_enabled`

Brief description

Set to true to enable output, false to disable output.

12.1.151.2 stop_level

`gpt_pin_level_t::stop_level`

Brief description

Select a stop level from gpt_pin_level_t.

12.1.152 hash_api_t

```
typedef struct{
    uint32_t(*  open)(hash_ctrl_t *const p_ctrl, hash_cfg_t const *const p_cfg)
    uint32_t(*  updateHash)(const uint32_t *p_source, uint32_t num_words,
uint32_t *p_dest)
    uint32_t(*  hashUpdate)(hash_ctrl_t *const p_ctrl, const uint32_t *p_source,
uint32_t num_words, uint32_t *p_dest)
    uint32_t(*  close)(hash_ctrl_t *const p_ctrl)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} hash_api_t
```

12.1.153 hash_cfg_t

```
typedef struct{
    crypto_ctrl_t *  p_crypto_ctrl
    crypto_api_t const  *  p_crypto_api
} hash_cfg_t
```

12.1.153.1 p_crypto_ctrl

`crypto_ctrl_t::p_crypto_ctrl`

Brief description

pointer to crypto engine control structure

12.1.153.2 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine API structure

12.1.154 hash_ctrl_t

```
typedef struct{
    uint32_t  msgbuf[HASH_MESSAGE_BLOCK_SIZE_WORDS]
    uint32_t  hash[HASH_MAX_DIGEST_SIZE_WORDS]
    uint64_t  length
} hash_ctrl_t
```

12.1.154.1 msgbuf

`uint32_t hash_ctrl_t::msgbuf[HASH_MESSAGE_BLOCK_SIZE_WORDS]`

Brief description

message buffer to be hashed

12.1.154.2 hash

`uint32_t hash_ctrl_t::hash[HASH_MAX_DIGEST_SIZE_WORDS]`

Brief description

current hash value

12.1.154.3 length

`uint64_t hash_ctrl_t::length`

Brief description

64-bit message length (number of bits)

12.1.155 hash_instance_t

```
typedef struct{
    hash_ctrl_t *  p_ctrl
    hash_cfg_t const  *  p_cfg
    hash_api_t const  *  p_api
} hash_instance_t
```

12.1.155.1 p_ctrl

`hash_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.155.2 p_cfg

`hash_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.155.3 p_api

`hash_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.156 i2c_api_master_t

```
typedef struct{
    ssp_err_t(*  open)(i2c_ctrl_t *const p_ctrl, i2c_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(i2c_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  read)(i2c_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t
const bytes, bool const restart)
    ssp_err_t(*  write)(i2c_ctrl_t *const p_ctrl, uint8_t *const p_src, uint32_t
const bytes, bool const restart)
    ssp_err_t(*  reset)(i2c_ctrl_t *const p_ctrl)
    ssp_err_t(*  slaveAddressSet)(i2c_ctrl_t *const p_ctrl, uint16_t const slave,
i2c_addr_mode_t const addr_mode)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} i2c_api_master_t
```

12.1.157 i2c_api_slave_t

```
typedef struct{
    ssp_err_t(*  open)(i2c_ctrl_t *const p_ctrl, i2c_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(i2c_ctrl_t *const p_ctrl)
    ssp_err_t(*  masterWriteSlaveRead)(i2c_ctrl_t *const p_ctrl, uint8_t *const
p_dest, uint32_t const bytes)
    ssp_err_t(*  masterReadSlaveWrite)(i2c_ctrl_t *const p_ctrl, uint8_t *const
p_src, uint32_t const bytes)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} i2c_api_slave_t
```

12.1.158 i2c_callback_args_t

```
typedef struct{
    void const  *const  p_context
    uint32_t const  bytes
    i2c_event_t const  event
} i2c_callback_args_t
```

12.1.158.1 p_context

`void const* const i2c_callback_args_t::p_context`

Brief description

Pointer to user-provided context.

12.1.158.2 bytes

`uint32_t const i2c_callback_args_t::bytes`

Brief description

Number of received/transmitted bytes in buff.

12.1.158.3 event

`i2c_event_t::event`

Brief description

Event code.

12.1.159 i2c_cfg_t

```
typedef struct{
    uint8_t  channel
    i2c_rate_t  rate
    uint16_t  slave
    i2c_addr_mode_t  addr_mode
    uint16_t  sda_delay
    uint8_t  rxi_ipl
    uint8_t  txi_ipl
    uint8_t  tei_ipl
    uint8_t  eri_ipl
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    void(*  p_callback)(i2c_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} i2c_cfg_t
```

12.1.159.1 channel

`uint8_t i2c_cfg_t::channel`

Brief description

Identifier recognizable by implementation.

Detailed description

Generic configuration

12.1.159.2 rate

`i2c_rate_t::rate`

Brief description

Device's maximum clock rate from enum i2c_rate_t.

12.1.159.3 slave

`uint16_t i2c_cfg_t::slave`

Brief description

The address of the slave device.

12.1.159.4 addr_mode

`i2c_addr_mode_t::addr_mode`

Brief description

Indicates how slave fields should be interpreted.

12.1.159.5 sda_delay

`uint16_t i2c_cfg_t::sda_delay`

Brief description

The SDA output delay.

12.1.159.6 rxi_ipl

`uint8_t i2c_cfg_t::rxi_ipl`

Brief description

Receive interrupt priority.

12.1.159.7 txi_ipl

`uint8_t i2c_cfg_t::txi_ipl`

Brief description

Transmit interrupt priority.

12.1.159.8 tei_ipl

`uint8_t i2c_cfg_t::tei_ipl`

Brief description

Transmit end interrupt priority.

12.1.159.9 eri_ipl

`uint8_t i2c_cfg_t::eri_ipl`

Brief description

Error interrupt priority.

12.1.159.10 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Brief description

DTC instance for I2C transmit.Set to NULL if unused.

Detailed description

DTC/DMA support

12.1.159.11 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Brief description

DTC instance for I2C receive. Set to NULL if unused.

12.1.159.12 p_callback

`void(* i2c_cfg_t::p_callback) (i2c_callback_args_t *p_args)`

Brief description

Pointer to callback function.

Detailed description

Parameters to control software behavior

12.1.159.13 p_context

`void const* i2c_cfg_t::p_context`

Brief description

Pointer to the user-provided context.

12.1.159.14 p_extend

`void const* i2c_cfg_t::p_extend`

Brief description

Any configuration data needed by the hardware.

Detailed description

Implementation-specific configuration

12.1.160 i2c_master_instance_t

```
typedef struct{
    i2c_ctrl_t *  p_ctrl
    i2c_cfg_t const  *  p_cfg
    i2c_api_master_t const  *  p_api
} i2c_master_instance_t
```

12.1.160.1 p_ctrl

`i2c_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.160.2 p_cfg

`i2c_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.160.3 p_api

`i2c_api_master_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.161 i2c_slave_instance_t

```
typedef struct{
    i2c_ctrl_t *  p_ctrl
    i2c_cfg_t const  *  p_cfg
    i2c_api_slave_t const  *  p_api
} i2c_slave_instance_t
```

12.1.161.1 p_ctrl

`i2c_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.161.2 p_cfg

`i2c_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.161.3 p_api

`i2c_api_slave_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.162 i2s_api_t

```
typedef struct{
    ssp_err_t(*  open)(i2s_ctrl_t *const p_ctrl, i2s_cfg_t const *const p_cfg)
    ssp_err_t(*  stop)(i2s_ctrl_t *const p_ctrl, i2s_dir_t const dir)
    ssp_err_t(*  mute)(i2s_ctrl_t *const p_ctrl, i2s_mute_t const mute_enable)
    ssp_err_t(*  write)(i2s_ctrl_t *const p_ctrl, uint8_t const *const p_src,
uint16_t const bytes)
    ssp_err_t(*  read)(i2s_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint16_t
const bytes)
    ssp_err_t(*  writeRead)(i2s_ctrl_t *const p_ctrl, uint8_t const *const p_src,
uint8_t *const p_dest, uint16_t const bytes)
    ssp_err_t(*  infoGet)(i2s_ctrl_t *const p_ctrl, i2s_info_t *const p_info)
    ssp_err_t(*  close)(i2s_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} i2s_api_t
```

12.1.163 i2s_callback_args_t

```
typedef struct{
    void const  *  p_context
    i2s_event_t  event
} i2s_callback_args_t
```

12.1.163.1 p_context

`void const* i2s_callback_args_t::p_context`

Detailed description

Placeholder for user data. Set in open function in i2s_cfg_t.

12.1.163.2 event

`i2s_event_t::event`

Brief description

The event can be used to identify what caused the callback (overflow or error).

12.1.164 i2s_cfg_t

```
typedef struct{
    uint8_t  channel
    i2s_pcm_width_t  pcm_width
    i2s_word_length_t  word_length
    i2s_ws_continue_t  ws_continue
    uint32_t  sampling_freq_hz
    uint32_t  audio_clk_freq_hz
```

```
    timer_instance_t const  *  p_timer
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    void(*  p_callback)(i2s_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
    uint8_t  rxi_ipl
    uint8_t  txi_ipl
    uint8_t  idle_err_ipl
} i2s_cfg_t
```

12.1.164.1 channel

`uint8_t i2s_cfg_t::channel`

Detailed description

Select a channel corresponding to the channel number of the hardware.

12.1.164.2 pcm_width

`i2s_pcm_width_t::pcm_width`

Brief description

Audio PCM data width.

12.1.164.3 word_length

`i2s_word_length_t::word_length`

Brief description

Audio word length, bits must be >= pcm_width bits.

12.1.164.4 ws_continue

`i2s_ws_continue_t::ws_continue`

Brief description

Whether to continue WS transmission during idle state.

12.1.164.5 sampling_freq_hz

`uint32_t i2s_cfg_t::sampling_freq_hz`

Brief description

Sampling frequency in Hertz.

12.1.164.6 audio_clk_freq_hz

```
uint32_t i2s_cfg_t::audio_clk_freq_hz
```

Detailed description

Audio clock frequency in Hertz. Must be a multiple between 1 and 128 of (16 * sampling_freq_hz * (word_length <enum_value> + 1)

12.1.164.7 p_timer

```
timer_instance_t::p_timer
```

Detailed description

To generate audio clock with GPT, link a timer instance here. Set to NULL if unused.

12.1.164.8 p_transfer_tx

```
transfer_instance_t::p_transfer_tx
```

Detailed description

To use DTC during write, link a DTC instance here. Set to NULL if unused.

12.1.164.9 p_transfer_rx

```
transfer_instance_t::p_transfer_rx
```

Detailed description

To use DTC during read, link a DTC instance here. Set to NULL if unused.

12.1.164.10 p_callback

```
void(* i2s_cfg_t::p_callback) (i2s_callback_args_t *p_args)
```

Detailed description

Callback provided when an I2S ISR occurs. Set to NULL for no CPU interrupt.

12.1.164.11 p_context

```
void const* i2s_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in i2s_callback_args_t.

12.1.164.12 p_extend

```
void const* i2s_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.164.13 rxi_ipl

`uint8_t i2s_cfg_t::rxi_ipl`

Brief description

Receive interrupt priority.

12.1.164.14 txi_ipl

`uint8_t i2s_cfg_t::txi_ipl`

Brief description

Transmit interrupt priority.

12.1.164.15 idle_err_ipl

`uint8_t i2s_cfg_t::idle_err_ipl`

Brief description

Idle/Error interrupt priority.

12.1.165 i2s_info_t

```
typedef struct{
    i2s_status_t  status
    uint32_t  sampling_freq_hz
} i2s_info_t
```

12.1.165.1 status

`i2s_status_t::status`

12.1.165.2 sampling_freq_hz

`uint32_t i2s_info_t::sampling_freq_hz`

Brief description

Sampling frequency in Hertz.

12.1.166 i2s_instance_t

```
typedef struct{
    i2s_ctrl_t *  p_ctrl
    i2s_cfg_t const  *  p_cfg
    i2s_api_t const  *  p_api
} i2s_instance_t
```

12.1.166.1 p_ctrl

i2s_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.166.2 p_cfg

i2s_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.166.3 p_api

i2s_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.167 i2s_on_ssi_cfg_t

```
typedef struct{
    ssi_audio_clock_t  audio_clock
} i2s_on_ssi_cfg_t
```

12.1.167.1 audio_clock

ssi_audio_clock_t::audio_clock

Brief description

Audio clock source, default is SSI_AUDIO_CLOCK_EXTERNAL.

12.1.168 icu_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    R_ICU_Type *  p_reg
    void(*  p_callback)(external_irq_callback_args_t *p_args)
    void const  *  p_context
    IRQn_Type  irq
    uint8_t  channel
} icu_instance_ctrl_t
```

12.1.168.1 open

`uint32_t ::open`

Brief description

Used to determine if channel control block is in use.

12.1.168.2 p_reg

`R_ICU_Type* ::p_reg`

Brief description

Pointer to register base address.

12.1.168.3 p_callback

`void(* ::p_callback) ( *p_args)`

Detailed description

Callback provided when a external IRQ ISR occurs. Set to NULL for no CPU interrupt.

12.1.168.4 p_context

`void const* ::p_context`

Detailed description

Placeholder for user data. Passed to the user callback in external_irq_callback_args_t.

12.1.168.5 irq

`IRQn_Type ::irq`

Brief description

NVIC interrupt number.

12.1.168.6 channel

`uint8_t ::channel`

Brief description

Channel.

12.1.169 in_addr

```
typedef struct{
    unsigned long  s_addr
    ULONG  s_addr
} in_addr
```

12.1.169.1 s_addr

```
ULONG in_addr::s_addr
```

Brief description

load with inet_aton()

Detailed description

Load with inet_aton()

12.1.169.2 s_addr

```
ULONG in_addr::s_addr
```

12.1.170 input_capture_api_t

```
typedef struct{
    ssp_err_t(*  open)(input_capture_ctrl_t *const p_ctrl, input_capture_cfg_t
const *const p_cfg)
    ssp_err_t(*  disable)(input_capture_ctrl_t const *const p_ctrl)
    ssp_err_t(*  enable)(input_capture_ctrl_t const *const p_ctrl)
    ssp_err_t(*  infoGet)(input_capture_ctrl_t const *const p_ctrl,
input_capture_info_t *const p_info)
    ssp_err_t(*  lastCaptureGet)(input_capture_ctrl_t const *const p_ctrl,
input_capture_capture_t *const p_counter)
    ssp_err_t(*  close)(input_capture_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} input_capture_api_t
```

12.1.171 input_capture_callback_args_t

```
typedef struct{
    uint8_t  channel
    input_capture_event_t  event
    uint32_t  counter
    uint32_t  overflows
    void const  *  p_context
} input_capture_callback_args_t
```

12.1.171.1 channel

```
uint8_t input_capture_callback_args_t::channel
```

Brief description

The channel being used.

12.1.171.2 event

`input_capture_event_t::event`

Brief description

The event that caused the interrupt and callback.

12.1.171.3 counter

`uint32_t input_capture_callback_args_t::counter`

Brief description

The value of the timer captured at the time of interrupt.

12.1.171.4 overflows

`uint32_t input_capture_callback_args_t::overflows`

Brief description

The number of counter overflows that occurred during this measurement.

12.1.171.5 p_context

`void const* input_capture_callback_args_t::p_context`

Brief description

Placeholder for user data, set in p_context.

12.1.172 input_capture_capture_t

```
typedef struct{
    uint32_t  counter
    uint32_t  overflows
} input_capture_capture_t
```

12.1.172.1 counter

`uint32_t input_capture_capture_t::counter`

Brief description

The value of the timer captured at the time of interrupt.

12.1.172.2 overflows

`uint32_t input_capture_capture_t::overflows`

Brief description

The number of counter overflows that occurred during this measurement.

12.1.173 input_capture_cfg_t

```
typedef struct{
    uint8_t  channel
    uint8_t  capture_irq_ipl
    uint8_t  overflow_irq_ipl
    input_capture_mode_t  mode
    input_capture_signal_edge_t  edge
    input_capture_repetition_t  repetition
    bool  autostart
    void const  *  p_extend
    void(*  p_callback)(input_capture_callback_args_t *p_args)
    void const  *  p_context
} input_capture_cfg_t
```

12.1.173.1 channel

`uint8_t input_capture_cfg_t::channel`

Brief description

The channel in use.

12.1.173.2 capture_irq_ipl

`uint8_t input_capture_cfg_t::capture_irq_ipl`

Brief description

Capture interrupt priority.

12.1.173.3 overflow_irq_ipl

`uint8_t input_capture_cfg_t::overflow_irq_ipl`

Brief description

Overflow interrupt priority.

12.1.173.4 mode

`input_capture_mode_t::mode`

Brief description

The mode of measurement to be performed.

12.1.173.5 edge

`input_capture_signal_edge_t::edge`

Brief description

The triggering edge to start a measurement (rise or fall).

12.1.173.6 repetition

`input_capture_repetition_t::repetition`

Brief description

One-shot or periodic measurement.

12.1.173.7 autostart

`bool input_capture_cfg_t::autostart`

Brief description

Specifies whether interrupts are enabled or not after open.

12.1.173.8 p_extend

`void const* input_capture_cfg_t::p_extend`

Detailed description

REQUIRED. Pointer to peripheral-specific extension parameters. See gpt_input_capture_extend_t for GPT.

12.1.173.9 p_callback

`void(* input_capture_cfg_t::p_callback) (input_capture_callback_args_t *p_args)`

Detailed description

Pointer to user's callback function, or NULL if no interrupt desired.

12.1.173.10 p_context

`void const* input_capture_cfg_t::p_context`

Brief description

Pointer to user's context data, to be passed to the callback.

12.1.174 input_capture_info_t

```
typedef struct{
    input_capture_status_t  status
    input_capture_variant_t  variant
} input_capture_info_t
```

12.1.174.1 status

`input_capture_status_t::status`

Brief description

Whether or not a capture is in progress.

12.1.174.2 variant

`input_capture_variant_t::variant`

Brief description

Whether timer is 16 or 32 bits.

12.1.175 input_capture_instance_t

```
typedef struct{
    input_capture_ctrl_t *  p_ctrl
    input_capture_cfg_t const  *  p_cfg
    input_capture_api_t const  *  p_api
} input_capture_instance_t
```

12.1.175.1 p_ctrl

`input_capture_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.175.2 p_cfg

`input_capture_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.175.3 p_api

`input_capture_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.176 ioport_api_t

```
typedef struct{
    ssp_err_t(*  init)(const ioport_cfg_t *p_cfg)
    ssp_err_t(*  pinsCfg)(const ioport_cfg_t *p_cfg)
    ssp_err_t(*  pinCfg)(ioport_port_pin_t pin, uint32_t cfg)
    ssp_err_t(*  pinDirectionSet)(ioport_port_pin_t pin, ioport_direction_t
direction)
```

```
    ssp_err_t(*  pinEventInputRead)(ioport_port_pin_t pin, ioport_level_t
*p_pin_event)
    ssp_err_t(*  pinEventOutputWrite)(ioport_port_pin_t pin, ioport_level_t
pin_value)
    ssp_err_t(*  pinEthernetModeCfg)(ioport_ethernet_channel_t channel,
ioport_ethernet_mode_t mode)
    ssp_err_t(*  pinRead)(ioport_port_pin_t pin, ioport_level_t *p_pin_value)
    ssp_err_t(*  pinWrite)(ioport_port_pin_t pin, ioport_level_t level)
    ssp_err_t(*  portDirectionSet)(ioport_port_t port, ioport_size_t
direction_values, ioport_size_t mask)
    ssp_err_t(*  portEventInputRead)(ioport_port_t port, ioport_size_t
*p_event_data)
    ssp_err_t(*  portEventOutputWrite)(ioport_port_t port, ioport_size_t
event_data, ioport_size_t mask_value)
    ssp_err_t(*  portRead)(ioport_port_t port, ioport_size_t *p_port_value)
    ssp_err_t(*  portWrite)(ioport_port_t port, ioport_size_t value,
ioport_size_t mask)
    ssp_err_t(*  versionGet)(ssp_version_t *p_data)
} ioport_api_t
```

12.1.177 ioport_cfg_t

```
typedef struct{
    uint16_t  number_of_pins
    ioport_pin_cfg_t const  *  p_pin_cfg_data
} ioport_cfg_t
```

12.1.177.1 number_of_pins

`uint16_t ioport_cfg_t::number_of_pins`

Brief description

Number of pins for which there is configuration data.

12.1.177.2 p_pin_cfg_data

`ioport_pin_cfg_t::p_pin_cfg_data`

Brief description

Pin configuration data.

12.1.178 ioport_instance_t

```
typedef struct{
    ioport_cfg_t const  *  p_cfg
    ioport_api_t const  *  p_api
} ioport_instance_t
```

12.1.178.1 p_cfg

`ioport_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.178.2 p_api

`ioport_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.179 ioport_pin_cfg_t

```
typedef struct{
    uint32_t  pin_cfg
    ioport_port_pin_t  pin
} ioport_pin_cfg_t
```

12.1.179.1 pin_cfg

`uint32_t ioport_pin_cfg_t::pin_cfg`

Brief description

Pin PFS configuration - Use ioport_cfg_options_t parameters to configure.

12.1.179.2 pin

`ioport_port_pin_t::pin`

Brief description

Pin identifier.

12.1.180 iwdt_instance_ctrl_t

```
typedef struct{
    bool  wdt_open
    void const  *  p_context
    R_IWDT_Type *  p_reg
    void(*  p_callback)(wdt_callback_args_t *p_args)
} iwdt_instance_ctrl_t
```

12.1.180.1 wdt_open

`bool ::wdt_open`

Detailed description

Indicates whether the open() API has been successfully called.

12.1.180.2 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

12.1.180.3 p_reg

```
R_IWDT_Type* ::p_reg
```

Brief description

Pointer to register base address.

12.1.180.4 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback provided when a WDT NMI ISR occurs.

12.1.181 jpeg_decode_api_t

```
typedef struct{
    ssp_err_t(*  open)(jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_cfg_t const
*const p_cfg)
    ssp_err_t(*  outputBufferSet)(jpeg_decode_ctrl_t *const p_ctrl, void
*p_buffer, uint32_t buffer_size)
    ssp_err_t(*  horizontalStrideSet)(jpeg_decode_ctrl_t *const p_ctrl, uint32_t
horizontal_stride)
    ssp_err_t(*  imageSubsampleSet)(jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_subsample_t horizontal_subsample, jpeg_decode_subsample_t
vertical_subsample)
    ssp_err_t(*  inputBufferSet)(jpeg_decode_ctrl_t *const p_ctrl, void
*p_buffer, uint32_t buffer_size)
    ssp_err_t(*  linesDecodedGet)(jpeg_decode_ctrl_t *const p_ctrl, uint32_t
*const p_lines)
    ssp_err_t(*  imageSizeGet)(jpeg_decode_ctrl_t *const p_ctrl, uint16_t
*p_horizontal_size, uint16_t *p_vertical_size)
    ssp_err_t(*  statusGet)(jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_status_t *const p_status)
    ssp_err_t(*  close)(jpeg_decode_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
    ssp_err_t(*  pixelFormatGet)(jpeg_decode_ctrl_t *const p_ctrl,
```

```
jpeg_decode_color_space_t *const p_color_space)
} jpeg_decode_api_t
```

12.1.182 jpeg_decode_callback_args_t

```
typedef struct{
    jpeg_decode_status_t  status
    void const  *  p_context
} jpeg_decode_callback_args_t
```

12.1.182.1 status

`jpeg_decode_status_t::status`

Brief description

JPEG status.

12.1.182.2 p_context

`void const* jpeg_decode_callback_args_t::p_context`

Brief description

Pointer to user-provided context.

12.1.183 jpeg_decode_cfg_t

```
typedef struct{
    jpeg_decode_color_space_t  color_space
    jpeg_decode_data_format_t  input_data_format
    jpeg_decode_data_format_t  output_data_format
    jpeg_decode_pixel_format_t  pixel_format
    uint8_t  alpha_value
    uint8_t  jdti_ipl
    uint8_t  jedi_ipl
    void(*  p_callback)(jpeg_decode_callback_args_t *p_args)
    void const  *  p_context
} jpeg_decode_cfg_t
```

12.1.183.1 color_space

`jpeg_decode_color_space_t::color_space`

Brief description

Color space.

12.1.183.2 input_data_format

`jpeg_decode_data_format_t::input_data_format`

Brief description

Input data stream byte order.

12.1.183.3 output_data_format

`jpeg_decode_data_format_t::output_data_format`

Brief description

Output data stream byte order.

12.1.183.4 pixel_format

`jpeg_decode_pixel_format_t::pixel_format`

Brief description

Pixel format.

12.1.183.5 alpha_value

`uint8_t jpeg_decode_cfg_t::alpha_value`

Brief description

Alpha value to be applied to decoded pixel data. Only valid for ARGB888 format.

12.1.183.6 jdti_ipl

`uint8_t jpeg_decode_cfg_t::jdti_ipl`

Brief description

Data transfer interrupt priority.

12.1.183.7 jedi_ipl

`uint8_t jpeg_decode_cfg_t::jedi_ipl`

Brief description

Decompression interrupt priority.

12.1.183.8 p_callback

`void(* jpeg_decode_cfg_t::p_callback) (jpeg_decode_callback_args_t *p_args)`

Brief description

User-supplied callback functions.

12.1.183.9 p_context

```
void const* jpeg_decode_cfg_t::p_context
```

Brief description

Placeholder for user data. Passed to user callback in jpeg_decode_callback_args_t.

12.1.184 jpeg_decode_instance_ctrl_t

```
typedef struct{
    jpeg_decode_status_t  status
    ssp_err_t  error_code
    void(*  p_callback)(jpeg_decode_callback_args_t *p_args)
    void const  *  p_extend
    void const  *  p_context
    R_JPEG_Type *  p_reg
    jpeg_decode_pixel_format_t  pixel_format
    uint32_t  horizontal_stride
    uint32_t  outbuffer_size
    uint16_t  total_lines_decoded
    jpeg_decode_subsample_t  horizontal_subsample
} jpeg_decode_instance_ctrl_t
```

12.1.184.1 status

```
jpeg_decode_status_t::status
```

Brief description

JPEG Codec module status.

12.1.184.2 error_code

```
::error_code
```

Brief description

JPEG Codec error code (if any).

12.1.184.3 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

User-supplied callback functions.

12.1.184.4 p_extend

```
void const* ::p_extend
```

Brief description

JPEG Codec hardware dependent configuration */.

12.1.184.5 p_context

```
void const* ::p_context
```

Brief description

Placeholder for user data. Passed to user callback in jpeg_decode_callback_args_t.

12.1.184.6 p_reg

```
R_JPEG_Type* ::p_reg
```

Brief description

Pointer to register base address.

12.1.184.7 pixel_format

```
jpeg_decode_pixel_format_t::pixel_format
```

Brief description

Pixel format.

12.1.184.8 horizontal_stride

```
uint32_t ::horizontal_stride
```

Brief description

Horizontal Stride settings.

12.1.184.9 outbuffer_size

```
uint32_t ::outbuffer_size
```

Brief description

out buffer size

12.1.184.10 total_lines_decoded

```
uint16_t ::total_lines_decoded
```

Brief description

Track the number of lines decoded so far.

12.1.184.11 horizontal_subsample

```
jpeg_decode_subsample_t::horizontal_subsample
```

Brief description

Horizontal sub-sample setting.

12.1.185 jpeg_decode_instance_t

```
typedef struct{
    jpeg_decode_ctrl_t *  p_ctrl
    jpeg_decode_cfg_t const  *  p_cfg
    jpeg_decode_api_t const  *  p_api
} jpeg_decode_instance_t
```

12.1.185.1 p_ctrl

`jpeg_decode_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.185.2 p_cfg

`jpeg_decode_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.185.3 p_api

`jpeg_decode_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.186 keymatrix_api_t

```
typedef struct{
    ssp_err_t(*  open)(keymatrix_ctrl_t *const p_ctrl, keymatrix_cfg_t const
*const p_cfg)
    ssp_err_t(*  enable)(keymatrix_ctrl_t *const p_ctrl)
    ssp_err_t(*  disable)(keymatrix_ctrl_t *const p_ctrl)
    ssp_err_t(*  triggerSet)(keymatrix_ctrl_t *const p_ctrl, keymatrix_trigger_t
const trigger)
    ssp_err_t(*  close)(keymatrix_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} keymatrix_api_t
```

12.1.187 keymatrix_callback_args_t

```
typedef struct{
    void const  *  p_context
    keymatrix_channels_t  channels
} keymatrix_callback_args_t
```

12.1.187.1 p_context

`void const* keymatrix_callback_args_t::p_context`

Brief description

Holder for user data. Set in open function in keymatrix_cfg_t.

12.1.187.2 channels

`keymatrix_channels_t::channels`

Detailed description

Bit vector representing the physical hardware channel(s) that caused the interrupt. The bit vector is used for compatibility with matrix designs where more than one input will be active at once. ***NOTE:*** *Not all HAL drivers support matrix mode. See r_kint.h for details.*

12.1.188 keymatrix_cfg_t

```
typedef struct{
    keymatrix_channels_t  channels
    keymatrix_trigger_t  trigger
    bool  autostart
    void(*  p_callback)(keymatrix_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
    uint8_t  irq_ipl
} keymatrix_cfg_t
```

12.1.188.1 channels

`keymatrix_channels_t::channels`

Brief description

Key Input channel(s). Bit mask of channels to open.

12.1.188.2 trigger

`keymatrix_trigger_t::trigger`

Brief description

Key Input trigger setting.

12.1.188.3 autostart

```
bool keymatrix_cfg_t::autostart
```

Brief description

Start operation and enable interrupts during open().

12.1.188.4 p_callback

```
void(* keymatrix_cfg_t::p_callback) (keymatrix_callback_args_t *p_args)
```

Brief description

Callback for key interrupt ISR.

12.1.188.5 p_context

```
void const* keymatrix_cfg_t::p_context
```

Brief description

Holder for user data. Passed to callback in keymatrix_user_cb_data_t.

12.1.188.6 p_extend

```
void const* keymatrix_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.188.7 irq_ipl

```
uint8_t keymatrix_cfg_t::irq_ipl
```

Brief description

Interrupt priority level.

12.1.189 keymatrix_instance_t

```
typedef struct{
    keymatrix_ctrl_t *  p_ctrl
    keymatrix_cfg_t const  *  p_cfg
    keymatrix_api_t const  *  p_api
} keymatrix_instance_t
```

12.1.189.1 p_ctrl

```
keymatrix_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.189.2 p_cfg

`keymatrix_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.189.3 p_api

`keymatrix_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.190 kint_instance_ctrl_t

```
typedef struct{
    R_KINT_Type *  p_reg
    keymatrix_channels_t  channels
    IRQn_Type  irq
} kint_instance_ctrl_t
```

12.1.190.1 p_reg

`R_KINT_Type* ::p_reg`

Brief description

Pointer to register base address.

12.1.190.2 channels

`keymatrix_channels_t::channels`

Brief description

Channel bitmask.

12.1.190.3 irq

`IRQn_Type ::irq`

Brief description

Interrupt priority number.

12.1.191 lpm_api_t

```
typedef struct{
    ssp_err_t(*  init)(lpm_cfg_t const *const p_cfg)
    ssp_err_t(*  mstpcrSet)(uint32_t mstpcra_value, uint32_t mstpcrb_value,
uint32_t mstpcrc_value, uint32_t mstpcrd_value)
    ssp_err_t(*  mstpcrGet)(uint32_t *mstpcra_value, uint32_t *mstpcrb_value,
uint32_t *mstpcrc_value, uint32_t *mstpcrd_value)
    ssp_err_t(*  moduleStop)(lpm_mstp_t module)
    ssp_err_t(*  moduleStart)(lpm_mstp_t module)
    ssp_err_t(*  operatingPowerModeSet)(lpm_operating_power_t power_mode,
lpm_subosc_t subosc)
    ssp_err_t(*  snoozeEnable)(lpm_snooze_rxd0_t rdx0_mode, lpm_snooze_dtc_t
dtc_mode, lpm_snooze_request_t requests, lpm_snooze_end_t triggers)
    ssp_err_t(*  snoozeDisable)(void)
    ssp_err_t(*  lowPowerCfg)(lpm_low_power_mode_t power_mode,
lpm_output_port_enable_t output_port_enable, lpm_power_supply_t power_supply,
lpm_io_port_t io_port_state)
    ssp_err_t(*  wupenSet)(uint32_t wupen_value)
    ssp_err_t(*  wupenGet)(uint32_t *wupen_value)
    ssp_err_t(*  deepStandbyCancelRequestEnable)(lpm_deep_standby_t pin_signal,
lpm_cancel_request_edge_t rising_falling)
    ssp_err_t(*  deepStandbyCancelRequestDisable)(lpm_deep_standby_t pin_signal)
    ssp_err_t(*  lowPowerModeEnter)(void)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} lpm_api_t
```

12.1.192 lpm_cfg_t

```
typedef struct{
    lpm_operating_power_t  operating_power
    lpm_subosc_t  sub_oscillator
    lpm_code_flash_t  code_flash
} lpm_cfg_t
```

12.1.192.1 operating_power

`lpm_operating_power_t::operating_power`

Brief description

Operating power mode.

12.1.192.2 sub_oscillator

`lpm_subosc_t::sub_oscillator`

Brief description

Sub oscillator.

12.1.192.3 code_flash

lpm_code_flash_t::code_flash

Brief description

Enable the code flash.

12.1.193 lpm_instance_t

```
typedef struct{
    lpm_cfg_t const  *  p_cfg
    lpm_api_t const  *  p_api
} lpm_instance_t
```

12.1.193.1 p_cfg

lpm_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.193.2 p_api

lpm_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.194 lpmv2_api_t

```
typedef struct{
    ssp_err_t(*  init)(void)
    ssp_err_t(*  lowPowerCfg)(lpmv2_cfg_t const *const p_cfg)
    ssp_err_t(*  lowPowerModeEnter)(void)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} lpmv2_api_t
```

12.1.195 lpmv2_cfg_t

```
typedef struct{
    lpmv2_low_power_mode_t  low_power_mode
    void const  *  p_extend
} lpmv2_cfg_t
```

12.1.195.1 low_power_mode

`lpmv2_low_power_mode_t::low_power_mode`

Detailed description

Low Power Mode

12.1.195.2 p_extend

`void const* lpmv2_cfg_t::p_extend`

Detailed description

MCU Specific configuration

12.1.196 lpmv2_instance_t

```
typedef struct{
    lpmv2_cfg_t const  *const  p_cfg
    lpmv2_api_t const  *const  p_api
} lpmv2_instance_t
```

12.1.196.1 p_cfg

`lpmv2_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.196.2 p_api

`lpmv2_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.197 lpmv2_mcu_cfg_t

```
typedef struct{
    lpmv2_standby_wake_source_bits_t  standby_wake_sources
    lpmv2_snooze_request_t  snooze_request_source
    lpmv2_snooze_end_bits_t  snooze_end_sources
    lpmv2_snooze_dtc_t  dtc_state_in_snooze
    lpmv2_output_port_enable_t  output_port_enable
    lpmv2_io_port_t  io_port_state
    lpmv2_power_supply_t  power_supply_state
    lpmv2_deep_standby_cancel_source_bits_t  deep_standby_cancel_source
    lpmv2_deep_standby_cancel_edge_bits_t  deep_standby_cancel_edge
} lpmv2_mcu_cfg_t
```

12.1.197.1 standby_wake_sources

`lpmv2_standby_wake_source_bits_t::standby_wake_sources`

Detailed description

Bitwise list of sources to wake from standby

12.1.197.2 snooze_request_source

`lpmv2_snooze_request_t::snooze_request_source`

Detailed description

Snooze request source

12.1.197.3 snooze_end_sources

`lpmv2_snooze_end_bits_t::snooze_end_sources`

Detailed description

Bitwise list of snooze end sources

12.1.197.4 dtc_state_in_snooze

`lpmv2_snooze_dtc_t::dtc_state_in_snooze`

Detailed description

State of DTC in snooze mode, enabled or disabled

12.1.197.5 output_port_enable

`lpmv2_output_port_enable_t::output_port_enable`

Detailed description

Output port enabled/disabled in standby and deep standby

12.1.197.6 io_port_state

`lpmv2_io_port_t::io_port_state`

Detailed description

IO port state in deep standby (maintained or reset)

12.1.197.7 power_supply_state

`lpmv2_power_supply_t::power_supply_state`

Detailed description

Internal power supply state in standby and deep standby (deepcut)

12.1.197.8 deep_standby_cancel_source

`lpmv2_deep_standby_cancel_source_bits_t::deep_standby_cancel_source`

Detailed description

Sources that can trigger exit from deep standby

12.1.197.9 deep_standby_cancel_edge

`lpmv2_deep_standby_cancel_edge_bits_t::deep_standby_cancel_edge`

Detailed description

Signal edges for the sources that can trigger exit from deep standby

12.1.198 lvd_api_t

```
typedef struct{
    ssp_err_t(*  open)(lvd_ctrl_t *const p_ctrl, lvd_cfg_t const *const p_cfg)
    ssp_err_t(*  statusGet)(lvd_ctrl_t *const p_ctrl, lvd_status_t *p_lvd_status)
    ssp_err_t(*  statusClear)(lvd_ctrl_t *const p_ctrl)
    ssp_err_t(*  close)(lvd_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} lvd_api_t
```

12.1.199 lvd_callback_args_t

```
typedef struct{
    uint32_t  monitor_number
    lvd_status_t  status
    void const  *  p_context
} lvd_callback_args_t
```

12.1.199.1 monitor_number

`uint32_t lvd_callback_args_t::monitor_number`

Brief description

Monitor number.

12.1.199.2 status

`lvd_status_t::status`

Brief description

Status of monitor.

12.1.199.3 p_context

```
void const* lvd_callback_args_t::p_context
```

Brief description

Placeholder for user data.

12.1.200 lvd_cfg_t

```
typedef struct{
    const uint32_t  monitor_number
    lvd_threshold_t  voltage_threshold
    lvd_response_t  detection_response
    lvd_voltage_slope_t  voltage_slope
    uint8_t  monitor_ipl
    void(*  p_callback)(lvd_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} lvd_cfg_t
```

12.1.200.1 monitor_number

```
const uint32_t lvd_cfg_t::monitor_number
```

Detailed description

Monitor number, 1, 2, ...

12.1.200.2 voltage_threshold

```
lvd_threshold_t::voltage_threshold
```

Detailed description

Threshold for out of range voltage detection

12.1.200.3 detection_response

```
lvd_response_t::detection_response
```

Detailed description

Response on detecting a threshold crossing

12.1.200.4 voltage_slope

```
lvd_voltage_slope_t::voltage_slope
```

Detailed description

Rising or falling voltage is to be detected

12.1.200.5 monitor_ipl

`uint8_t lvd_cfg_t::monitor_ipl`

Detailed description

Interrupt priority level.

12.1.200.6 p_callback

`void(* lvd_cfg_t::p_callback) (lvd_callback_args_t *p_args)`

Detailed description

User function to be called from interrupt

12.1.200.7 p_context

`void const* lvd_cfg_t::p_context`

Detailed description

Placeholder for user data. Passed to the user callback in

12.1.200.8 p_extend

`void const* lvd_cfg_t::p_extend`

Detailed description

Extension parameter for hardware specific settings

12.1.201 lvd_extend_t

```
typedef struct{
    lvd_negation_delay_t  negation_delay
    lvd_sample_clock_t  sample_clock_divisor
} lvd_extend_t
```

12.1.201.1 negation_delay

`lvd_negation_delay_t::negation_delay`

Detailed description

Negation of LVD signal follows reset or voltage in range

12.1.201.2 sample_clock_divisor

`lvd_sample_clock_t::sample_clock_divisor`

Detailed description

Sample clock divider, use LVD_SAMPLE_CLOCK_DISABLED to disable digital filtering

12.1.202 lvd_instance_ctrl_t

```
typedef struct{
    uint32_t  monitor_number
    R_SYSTEM_Type *  p_reg
    void(*  p_callback)(lvd_callback_args_t *p_args)
    lvd_callback_args_t  lvd_callback_args
    uint32_t  opened
} lvd_instance_ctrl_t
```

12.1.202.1 monitor_number

```
uint32_t ::monitor_number
```

Detailed description

Monitor number, 1, 2, ...

12.1.202.2 p_reg

```
R_SYSTEM_Type* ::p_reg
```

Brief description

Pointer to LVD register base address.

12.1.202.3 p_callback

```
void(* ::p_callback) ( *p_args)
```

12.1.202.4 lvd_callback_args

```
lvd_callback_args_t::lvd_callback_args
```

12.1.202.5 opened

```
uint32_t ::opened
```

12.1.203 lvd_instance_t

```
typedef struct{
    lvd_ctrl_t *  p_ctrl
    lvd_cfg_t const  *  p_cfg
    lvd_api_t const  *  p_api
} lvd_instance_t
```

12.1.203.1 p_ctrl

`lvd_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.203.2 p_cfg

`lvd_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this interface instance.

12.1.203.3 p_api

`lvd_api_t::p_api`

Brief description

Pointer to the API structure for this interface instance.

12.1.204 lvd_status_t

```
typedef struct{
    lvd_threshold_crossing_t  crossing_detected
    lvd_current_state_t  current_state
} lvd_status_t
```

12.1.204.1 crossing_detected

`lvd_threshold_crossing_t::crossing_detected`

Detailed description

Threshold crossing detection (latched)

12.1.204.2 current_state

`lvd_current_state_t::current_state`

Detailed description

Instantaneous status of monitored voltage (above or below threshold)

12.1.205 pdc_api_t

```
typedef struct{
    ssp_err_t(*  open)(pdc_ctrl_t *const p_ctrl, pdc_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(pdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  captureStart)(pdc_ctrl_t *const p_ctrl, uint8_t *const p_buffer)
```

```
    ssp_err_t(*  stateGet)(pdc_ctrl_t *const p_ctrl, pdc_state_t *p_state)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_data)
} pdc_api_t
```

12.1.206 pdc_callback_args_t

```
typedef struct{
    pdc_event_t  event
    uint8_t *  p_buffer
    void const  *  p_context
} pdc_callback_args_t
```

12.1.206.1 event

`pdc_event_t::event`

Brief description

Event causing the callback.

12.1.206.2 p_buffer

`uint8_t* pdc_callback_args_t::p_buffer`

Brief description

Pointer to buffer containing the captured data.

12.1.206.3 p_context

`void const* pdc_callback_args_t::p_context`

Brief description

Placeholder for user data. Set in open function in pdc_cfg_t.

12.1.207 pdc_cfg_t

```
typedef struct{
    uint16_t  x_capture_start_pixel
    uint16_t  x_capture_pixels
    uint16_t  y_capture_start_pixel
    uint16_t  y_capture_pixels
    pdc_clock_division_t  clock_division
    pdc_endian_t  endian
    pdc_hsync_polarity_t  hsync_polarity
    pdc_vsync_polarity_t  vsync_polarity
    uint8_t *  p_buffer
    uint8_t  bytes_per_pixel
    uint8_t  frame_end_ipl
```

```
    uint8_t  irq_ipl
    transfer_instance_t const  *const  p_lower_lvl_transfer
    void(*  p_callback)(pdc_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} pdc_cfg_t
```

12.1.207.1 x_capture_start_pixel

`uint16_t pdc_cfg_t::x_capture_start_pixel`

Brief description

Horizontal position to start capture.

12.1.207.2 x_capture_pixels

`uint16_t pdc_cfg_t::x_capture_pixels`

Brief description

Number of horizontal pixels to capture.

12.1.207.3 y_capture_start_pixel

`uint16_t pdc_cfg_t::y_capture_start_pixel`

Brief description

Vertical position to start capture.

12.1.207.4 y_capture_pixels

`uint16_t pdc_cfg_t::y_capture_pixels`

Brief description

Number of vertical lines/pixels to capture.

12.1.207.5 clock_division

`pdc_clock_division_t::clock_division`

Brief description

Clock divider.

12.1.207.6 endian

`pdc_endian_t::endian`

Brief description

Endian of capture data.

12.1.207.7 hsync_polarity

`pdc_hsync_polarity_t::hsync_polarity`

Brief description

Polarity of HSYNC input.

12.1.207.8 vsync_polarity

`pdc_vsync_polarity_t::vsync_polarity`

Brief description

Polarity of VSYNC input.

12.1.207.9 p_buffer

`uint8_t* pdc_cfg_t::p_buffer`

Brief description

Pointer to buffer to write image into.

12.1.207.10 bytes_per_pixel

`uint8_t pdc_cfg_t::bytes_per_pixel`

Brief description

Number of bytes per pixel.

12.1.207.11 frame_end_ipl

`uint8_t pdc_cfg_t::frame_end_ipl`

Brief description

Frame end interrupt priority.

12.1.207.12 irq_ipl

`uint8_t pdc_cfg_t::irq_ipl`

Brief description

PDC interrupt priority.

12.1.207.13 p_lower_lvl_transfer

`transfer_instance_t::p_lower_lvl_transfer`

Brief description

Pointer to the transfer instance the PDC should use.

12.1.207.14 p_callback

```
void(* pdc_cfg_t::p_callback) (pdc_callback_args_t *p_args)
```

Brief description

Callback provided when a PDC transfer ISR occurs.

12.1.207.15 p_context

```
void const* pdc_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in pdc_callback_args_t.

12.1.207.16 p_extend

```
void const* pdc_cfg_t::p_extend
```

12.1.208 pdc_instance_ctrl_t

```
typedef struct{
    R_PDC_Type *  p_reg
    uint32_t  open
    uint8_t  bytes_per_pixel
    uint16_t  x_resolution_pixels
    uint16_t  y_resolution_pixels
    uint16_t  x_capture_start_pixel
    uint16_t  x_capture_pixels
    uint16_t  y_capture_start_pixel
    uint16_t  y_capture_pixels
    pdc_endian_t  endian
    pdc_hsync_polarity_t  hsync_polarity
    pdc_vsync_polarity_t  vsync_polarity
    uint8_t *  p_current_buffer
    bool  transfer_in_progress
    transfer_instance_t const  *  p_lower_lvl_transfer
    transfer_info_t  info_transfer
    void const  *  p_context
    void(*  p_callback)(pdc_callback_args_t *p_args)
    IRQn_Type  frame_end_irq
    IRQn_Type  irq
} pdc_instance_ctrl_t
```

12.1.208.1 p_reg

```
R_PDC_Type* ::p_reg
```

Brief description

Pointer to PDC base address.

12.1.208.2 open

`uint32_t ::open`

Detailed description

Indicates whether or not the driver is open called.

12.1.208.3 bytes_per_pixel

`uint8_t ::bytes_per_pixel`

Brief description

Number of bytes per pixel.

12.1.208.4 x_resolution_pixels

`uint16_t ::x_resolution_pixels`

Brief description

Total number of horizontal pixels input to PDC.

12.1.208.5 y_resolution_pixels

`uint16_t ::y_resolution_pixels`

Brief description

Total number of lines input to the PDC.

12.1.208.6 x_capture_start_pixel

`uint16_t ::x_capture_start_pixel`

Brief description

Horizontal position to start capture.

12.1.208.7 x_capture_pixels

`uint16_t ::x_capture_pixels`

Brief description

Number of horizontal pixels to capture.

12.1.208.8 y_capture_start_pixel

`uint16_t ::y_capture_start_pixel`

Brief description

Vertical position to start capture.

12.1.208.9 y_capture_pixels

`uint16_t ::y_capture_pixels`

Brief description

Number of vertical lines/pixels to capture.

12.1.208.10 endian

`pdc_endian_t::endian`

Brief description

Endian of capture data.

12.1.208.11 hsync_polarity

`pdc_hsync_polarity_t::hsync_polarity`

Brief description

Polarity of HSYNC input.

12.1.208.12 vsync_polarity

`pdc_vsync_polarity_t::vsync_polarity`

Brief description

Polarity of VSYNC input.

12.1.208.13 p_current_buffer

`uint8_t* ::p_current_buffer`

Brief description

Pointer to buffer currently in use.

12.1.208.14 transfer_in_progress

`bool ::transfer_in_progress`

Brief description

Indicates if a PDC transfer is already in progress.

12.1.208.15 p_lower_lvl_transfer

`transfer_instance_t::p_lower_lvl_transfer`

Brief description

Pointer to the transfer instance the PDC should use.

12.1.208.16 info_transfer

```
transfer_info_t::info_transfer
```

Brief description

Transfer info structure for low level Transfer interface.

12.1.208.17 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in pdc_callback_args_t.

12.1.208.18 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback provided when a PDC transfer ISR occurs.

12.1.208.19 frame_end_irq

```
IRQn_Type ::frame_end_irq
```

Brief description

Frame end interrupt number.

12.1.208.20 irq

```
IRQn_Type ::irq
```

Brief description

PDC interrupt number.

12.1.209 pdc_instance_t

```
typedef struct{
    pdc_ctrl_t *  p_ctrl
    pdc_cfg_t const  *  p_cfg
    pdc_api_t const  *  p_api
} pdc_instance_t
```

12.1.209.1 p_ctrl

pdc_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.209.2 p_cfg

pdc_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.209.3 p_api

pdc_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.210 pdc_state_t

```
typedef struct{
    pdc_vsync_state_t  vsync
    pdc_hsync_state_t  hsync
} pdc_state_t
```

12.1.210.1 vsync

pdc_vsync_state_t::vsync

Brief description

VSYNC signal state.

12.1.210.2 hsync

pdc_hsync_state_t::hsync

Brief description

HSYNC signal state.

12.1.211 qspi_api_t

```
typedef struct{
    ssp_err_t(*  open)(qspi_ctrl_t *p_ctrl, qspi_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(qspi_ctrl_t *p_ctrl)
    ssp_err_t(*  read)(qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint8_t
```

```
*p_memory_address, uint32_t byte_count)
    ssp_err_t(*  pageProgram)(qspi_ctrl_t *p_ctrl, uint8_t *p_device_address,
uint8_t *p_memory_address, uint32_t byte_count)
    ssp_err_t(*  erase)(qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint32_t
byte_count)
    ssp_err_t(*  infoGet)(qspi_ctrl_t *const p_ctrl, qspi_info_t *const p_info)
    ssp_err_t(*  sectorErase)(qspi_ctrl_t *p_ctrl, uint8_t *p_device_address)
    ssp_err_t(*  statusGet)(qspi_ctrl_t *p_ctrl, bool *p_write_in_progress)
    ssp_err_t(*  bankSelect)(uint32_t bank)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} qspi_api_t
```

12.1.212 qspi_cfg_t

```
typedef struct{
    void *  p_extend
} qspi_cfg_t
```

12.1.212.1 p_extend

`void* qspi_cfg_t::p_extend`

Brief description

place holder for future development

12.1.213 qspi_info_t

```
typedef struct{
    uint32_t  total_size_bytes
    uint32_t  min_program_size_bytes
    uint32_t *  p_erase_sizes_bytes
    uint8_t  num_erase_sizes
} qspi_info_t
```

12.1.213.1 total_size_bytes

`uint32_t qspi_info_t::total_size_bytes`

Brief description

Size of this QSPI in bytes.

12.1.213.2 min_program_size_bytes

`uint32_t qspi_info_t::min_program_size_bytes`

Brief description

Minimum program size in bytes.

12.1.213.3 p_erase_sizes_bytes

```
uint32_t* qspi_info_t::p_erase_sizes_bytes
```

Brief description

Array of available erase sizes. Each entry is in bytes.

12.1.213.4 num_erase_sizes

```
uint8_t qspi_info_t::num_erase_sizes
```

Brief description

Number of available erase sizes.

12.1.214 qspi_instance_ctrl_t

```
typedef struct{
    R_QSPI_Type *  p_reg
    uint32_t  max_eraseable_size
    uint32_t  num_address_bytes
    uint32_t  spi_mode
    uint32_t  page_size
    uint8_t  manufacturer_id
    uint8_t  memory_type
    uint8_t  memory_capacity
    bool  xip_mode
} qspi_instance_ctrl_t
```

12.1.214.1 p_reg

```
R_QSPI_Type* ::p_reg
```

Brief description

Pointer to QSPI base register.

12.1.214.2 max_eraseable_size

```
uint32_t ::max_eraseable_size
```

Brief description

Largest eraseable sector size in kbytes. Used to determine buffer size for.

12.1.214.3 num_address_bytes

```
uint32_t ::num_address_bytes
```

Brief description

Number of bytes used to represent the address.

12.1.214.4 spi_mode

```
uint32_t ::spi_mode
```

Brief description

SPI mode - 0 = Extended, 1 = Dual, 2 = Quad.

12.1.214.5 page_size

```
uint32_t ::page_size
```

Brief description

Number of bytes in a programmable page.

12.1.214.6 manufacturer_id

```
uint8_t ::manufacturer_id
```

Brief description

Manufacturer ID.

12.1.214.7 memory_type

```
uint8_t ::memory_type
```

Brief description

Memory type.

12.1.214.8 memory_capacity

```
uint8_t ::memory_capacity
```

Brief description

Memory capacity (in MByte)

12.1.214.9 xip_mode

```
bool ::xip_mode
```

Brief description

0 = run in read mode, 1 = run in XIP mode

12.1.215 qspi_instance_t

```
typedef struct{
    qspi_ctrl_t *  p_ctrl
    qspi_cfg_t const  *  p_cfg
```

```
    qspi_api_t const  *  p_api
} qspi_instance_t
```

12.1.215.1 p_ctrl

qspi_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.215.2 p_cfg

qspi_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.215.3 p_api

qspi_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.216 recalculated_param_t

```
typedef struct{
    uint16_t  hpix_size
    uint16_t  vpix_size
    int16_t  hpix_offset
    int16_t  vpix_offset
    uint32_t  hread_size
    uint32_t  base_address
} recalculated_param_t
```

12.1.216.1 hpix_size

uint16_t ::hpix_size

12.1.216.2 vpix_size

uint16_t ::vpix_size

12.1.216.3 hpix_offset

int16_t ::hpix_offset

12.1.216.4 vpix_offset

```
int16_t ::vpix_offset
```

12.1.216.5 hread_size

```
uint32_t ::hread_size
```

12.1.216.6 base_address

```
uint32_t ::base_address
```

12.1.217 riic_instance_ctrl_t

```
typedef struct{
    i2c_cfg_t  info
    uint32_t  open
    void *  p_reg
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  tei_irq
    IRQn_Type  eri_irq
    uint8_t *  p_buff
    uint32_t  total
    uint32_t  remain
    uint32_t  loaded
    uint8_t  addr_low
    uint8_t  addr_high
    uint8_t  addr_total
    uint8_t  addr_remain
    uint8_t  addr_loaded
    bool  read
    bool  restart
    bool  err
    bool  restarted
    bool  transaction_completed
    bool  dummy_read_completed
    bool  activation_on_rxi
    bool  activation_on_txi
    bool  address_restarted
} riic_instance_ctrl_t
```

12.1.217.1 info

```
i2c_cfg_t::info
```

Brief description

Information describing I2C device.

12.1.217.2 open

`uint32_t ::open`

Brief description

Flag to determine if the device is open.

12.1.217.3 p_reg

`void* ::p_reg`

Brief description

Base register for this channel.

12.1.217.4 rxi_irq

`IRQn_Type ::rxi_irq`

Brief description

Receive IRQ number.

12.1.217.5 txi_irq

`IRQn_Type ::txi_irq`

Brief description

Transmit IRQ number.

12.1.217.6 tei_irq

`IRQn_Type ::tei_irq`

Brief description

Transmit end IRQ number.

12.1.217.7 eri_irq

`IRQn_Type ::eri_irq`

Brief description

Error IRQ number.

12.1.217.8 p_buff

`uint8_t* ::p_buff`

Detailed description

Holds the data associated with the transfer

12.1.217.9 total

`uint32_t ::total`

Detailed description

Holds the total number of data bytes to transfer

12.1.217.10 remain

`uint32_t ::remain`

Detailed description

Tracks the remaining data bytes to transfer

12.1.217.11 loaded

`uint32_t ::loaded`

Detailed description

Tracks the number of data bytes written to the register

12.1.217.12 addr_low

`uint8_t ::addr_low`

Detailed description

Holds the last address byte to issue

12.1.217.13 addr_high

`uint8_t ::addr_high`

Detailed description

Holds the first address byte to issue in 10-bit mode

12.1.217.14 addr_total

`uint8_t ::addr_total`

Detailed description

Holds the total number of address bytes to transfer

12.1.217.15 addr_remain

`uint8_t ::addr_remain`

Detailed description

Tracks the remaining address bytes to transfer

12.1.217.16 addr_loaded

```
uint8_t ::addr_loaded
```

Detailed description

Tracks the number of address bytes written to the register

12.1.217.17 read

```
volatile bool ::read
```

Detailed description

Holds the direction of the data byte transfer

12.1.217.18 restart

```
volatile bool ::restart
```

Detailed description

Holds whether or not the restart should be issued when done

12.1.217.19 err

```
volatile bool ::err
```

Detailed description

Tracks whether or not an error occurred during processing

12.1.217.20 restarted

```
volatile bool ::restarted
```

Detailed description

Tracks whether or not a restart was issued during the previous transfer

12.1.217.21 transaction_completed

```
volatile bool ::transaction_completed
```

Detailed description

Tracks if the transaction started earlier was completed

12.1.217.22 dummy_read_completed

```
volatile bool ::dummy_read_completed
```

Detailed description

Tracks whether the dummy read is performed

12.1.217.23 activation_on_rxi

```
volatile bool ::activation_on_rxi
```

Detailed description

< Tracks whether the transfer is activated on RXI interrupt

12.1.217.24 activation_on_txi

```
volatile bool ::activation_on_txi
```

Detailed description

< Tracks whether the transfer is activated on TXI interrupt

12.1.217.25 address_restarted

```
volatile bool ::address_restarted
```

Detailed description

< Tracks whether the restart condition is send on 10 bit read

12.1.218 riic_slave_instance_ctrl_t

```
typedef struct{
    i2c_cfg_t  info
    uint32_t  open
    void *  p_reg
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  eri_irq
    uint8_t *  p_buff
    uint32_t  total
    uint32_t  remain
    uint32_t  loaded
    uint32_t  transaction_count
    bool  read
    bool  err
    bool  slave_busy
    bool  do_dummy_read
    bool  start_interrupt_enabled
    bool  restarted
} riic_slave_instance_ctrl_t
```

12.1.218.1 info

`i2c_cfg_t::info`

Brief description

Information describing I2C device.

12.1.218.2 open

`uint32_t ::open`

Brief description

Flag to determine if the device is open.

12.1.218.3 p_reg

`void* ::p_reg`

Brief description

Base register for this channel.

12.1.218.4 rxi_irq

`IRQn_Type ::rxi_irq`

Brief description

Receive IRQ number.

12.1.218.5 txi_irq

`IRQn_Type ::txi_irq`

Brief description

Transmit IRQ number.

12.1.218.6 eri_irq

`IRQn_Type ::eri_irq`

Brief description

Error IRQ number.

12.1.218.7 p_buff

`uint8_t* ::p_buff`

Detailed description

Holds the data associated with the transfer

12.1.218.8 total

```
uint32_t ::total
```

Detailed description

Holds the total number of data bytes to transfer

12.1.218.9 remain

```
uint32_t ::remain
```

Detailed description

Tracks the remaining data bytes to transfer

12.1.218.10 loaded

```
uint32_t ::loaded
```

Detailed description

Tracks the number of data bytes written to the register

12.1.218.11 transaction_count

```
uint32_t ::transaction_count
```

Detailed description

Tracks the actual number of transactions

12.1.218.12 read

```
volatile bool ::read
```

Detailed description

Holds the direction of the data byte transfer

12.1.218.13 err

```
volatile bool ::err
```

Detailed description

Tracks whether or not an error occurred during processing

12.1.218.14 slave_busy

```
volatile bool ::slave_busy
```

Detailed description

Tracks if the slave is busy performing a transaction

12.1.218.15 do_dummy_read

```
volatile bool ::do_dummy_read
```

12.1.218.16 start_interrupt_enabled

```
volatile bool ::start_interrupt_enabled
```

Detailed description

<< Tracks whether a dummy read is issued on the first RX < Tracks whether the start interrupt is enabled

12.1.218.17 restarted

```
volatile bool ::restarted
```

12.1.219 rsa_api_t

```
typedef struct{
    uint32_t(*  open)(rsa_ctrl_t *const p_ctrl, rsa_cfg_t const *const p_cfg)
    uint32_t(*  encrypt)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  decrypt)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  decryptCrt)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key,
const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t
*p_dest)
    uint32_t(*  verify)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t
*p_padded_hash)
    uint32_t(*  sign)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t *p_dest)
    uint32_t(*  signCrt)(rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const
uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t *p_dest)
    uint32_t(*  close)(rsa_ctrl_t *const p_ctrl)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
    uint32_t(*  keyCreate)(rsa_ctrl_t *const p_ctrl, rsa_key_t *p_private_key,
rsa_key_t *p_public_key)
} rsa_api_t
```

12.1.220 rsa_cfg_t

```
typedef struct{
    crypto_api_t const  *  p_crypto_api
} rsa_cfg_t
```

12.1.220.1 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine api

12.1.221 rsa_ctrl_t

```
typedef struct{
    crypto_ctrl_t *  p_crypto_ctrl
    crypto_api_t const  *  p_crypto_api
    uint32_t  stage_num
} rsa_ctrl_t
```

12.1.221.1 p_crypto_ctrl

`crypto_ctrl_t::p_crypto_ctrl`

Brief description

pointer to crypto engine control structure

12.1.221.2 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto engine API

12.1.221.3 stage_num

`uint32_t rsa_ctrl_t::stage_num`

Brief description

processing stage

12.1.222 rsa_instance_t

```
typedef struct{
    rsa_ctrl_t *  p_ctrl
    rsa_cfg_t const  *  p_cfg
    rsa_api_t const  *  p_api
} rsa_instance_t
```

12.1.222.1 p_ctrl

`rsa_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.222.2 p_cfg

`rsa_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.222.3 p_api

`rsa_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.223 rsa_key_t

```
typedef struct{
    rsa_key_format_t  key_format
    uint32_t  length
    uint8_t *  p_data
} rsa_key_t
```

12.1.223.1 key_format

`rsa_key_format_t::key_format`

Brief description

Indicates if the key is in plain-text format or encrypted using device unique key.

12.1.223.2 length

`uint32_t rsa_key_t::length`

Brief description

Length in bytes of the p_data buffer.

12.1.223.3 p_data

`uint8_t* rsa_key_t::p_data`

Brief description

Buffer where the private key is stored.

12.1.224 rspi_access_delay_t

```
typedef struct{
    rspi_next_access_delay_count_t  rspi_next_access_delay_count
    rspi_next_access_delay_state_t  rspi_next_access_delay_state
} rspi_access_delay_t
```

12.1.224.1 rspi_next_access_delay_count

rspi_next_access_delay_count_t::rspi_next_access_delay_count

12.1.224.2 rspi_next_access_delay_state

rspi_next_access_delay_state_t::rspi_next_access_delay_state

12.1.225 rspi_clock_delay_t

```
typedef struct{
    rspi_clock_delay_count_t  rspi_clock_delay_count
    rspi_clock_delay_state_t  rspi_clock_delay_state
} rspi_clock_delay_t
```

12.1.225.1 rspi_clock_delay_count

rspi_clock_delay_count_t::rspi_clock_delay_count

12.1.225.2 rspi_clock_delay_state

rspi_clock_delay_state_t::rspi_clock_delay_state

12.1.226 rspi_instance_ctrl_t

```
typedef struct{
    uint8_t  channel
    uint8_t  current_slave
    uint32_t  channel_opened
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    void(*  p_callback)(spi_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  tei_irq
    IRQn_Type  eri_irq
```

```
    void *  p_src
    void *  p_dest
    uint32_t  tx_count
    uint32_t  rx_count
    uint32_t  xfr_length
    uint8_t  bytes_per_transfer
    bool  do_tx
    bool  using_dtc
    transfer_addr_mode_t  tx_dtc_addr_mode
    transfer_addr_mode_t  rx_dtc_addr_mode
    spi_operation_t  transfer_mode
    uint32_t  rx_data
    bsp_lock_t  resource_lock_tx_rx
} rspi_instance_ctrl_t
```

12.1.226.1 channel

`uint8_t ::channel`

Brief description

Channel number to be used.

12.1.226.2 current_slave

`uint8_t ::current_slave`

Brief description

Number of the currently assigned slave.

12.1.226.3 channel_opened

`uint32_t ::channel_opened`

Brief description

Internal flag to indicate the peripheral was initialized.

12.1.226.4 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Brief description

To use SPI DTC/DMA write transfer.

12.1.226.5 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Brief description

To use SPI DTC/DMA read transfer.

12.1.226.6 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Pointer to user callback function.

12.1.226.7 p_context

```
void const* ::p_context
```

Brief description

Pointer to the higher level device context.

12.1.226.8 p_reg

```
void* ::p_reg
```

Brief description

Base register for this channel.

12.1.226.9 rxi_irq

```
IRQn_Type ::rxi_irq
```

Brief description

Receive IRQ number.

12.1.226.10 txi_irq

```
IRQn_Type ::txi_irq
```

Brief description

Transmit IRQ number.

12.1.226.11 tei_irq

```
IRQn_Type ::tei_irq
```

Brief description

Transmit end IRQ number.

12.1.226.12 eri_irq

```
IRQn_Type ::eri_irq
```

Brief description

Error IRQ number.

12.1.226.13 p_src

```
void* ::p_src
```

12.1.226.14 p_dest

```
void* ::p_dest
```

12.1.226.15 tx_count

```
uint32_t ::tx_count
```

12.1.226.16 rx_count

```
uint32_t ::rx_count
```

12.1.226.17 xfr_length

```
uint32_t ::xfr_length
```

12.1.226.18 bytes_per_transfer

```
uint8_t ::bytes_per_transfer
```

12.1.226.19 do_tx

```
bool ::do_tx
```

12.1.226.20 using_dtc

```
bool ::using_dtc
```

12.1.226.21 tx_dtc_addr_mode

```
transfer_addr_mode_t::tx_dtc_addr_mode
```

12.1.226.22 rx_dtc_addr_mode

```
transfer_addr_mode_t::rx_dtc_addr_mode
```

12.1.226.23 transfer_mode

```
spi_operation_t::transfer_mode
```

12.1.226.24 rx_data

```
uint32_t ::rx_data
```

12.1.226.25 resource_lock_tx_rx

```
bsp_lock_t::resource_lock_tx_rx
```

Detailed description

Resource lock for transmission/reception

12.1.227 rspi_loopback_t

```
typedef struct{
    rspi_loopback1_t  rspi_loopback1
    rspi_loopback2_t  rspi_loopback2
} rspi_loopback_t
```

12.1.227.1 rspi_loopback1

```
rspi_loopback1_t::rspi_loopback1
```

12.1.227.2 rspi_loopback2

```
rspi_loopback2_t::rspi_loopback2
```

12.1.228 rspi_mosi_idle_t

```
typedef struct{
    rspi_mosi_idle_fixed_val_t  rspi_mosi_idle_fixed_val
    rspi_mosi_idle_val_fixing_t  rspi_mosi_idle_val_fixing
} rspi_mosi_idle_t
```

12.1.228.1 rspi_mosi_idle_fixed_val

```
rspi_mosi_idle_fixed_val_t::rspi_mosi_idle_fixed_val
```

12.1.228.2 rspi_mosi_idle_val_fixing

```
rspi_mosi_idle_val_fixing_t::rspi_mosi_idle_val_fixing
```

12.1.229 rspi_parity_t

```
typedef struct{
    rspi_parity_state_t  rspi_parity
```

```
    rspi_parity_mode_t  rspi_parity_mode
} rspi_parity_t
```

12.1.229.1 rspi_parity

rspi_parity_state_t::rspi_parity

12.1.229.2 rspi_parity_mode

rspi_parity_mode_t::rspi_parity_mode

12.1.230 rspi_ssl_negation_delay_t

```
typedef struct{
    rspi_ssl_negation_delay_count_t  rspi_ssl_neg_delay_count
    rspi_ssl_negation_delay_state_t  rspi_ssl_neg_delay_state
} rspi_ssl_negation_delay_t
```

12.1.230.1 rspi_ssl_neg_delay_count

rspi_ssl_negation_delay_count_t::rspi_ssl_neg_delay_count

12.1.230.2 rspi_ssl_neg_delay_state

rspi_ssl_negation_delay_state_t::rspi_ssl_neg_delay_state

12.1.231 rspi_ssl_polarity_t

```
typedef struct{
    rspi_sslp_t  rspi_ssl2
    rspi_sslp_t  rspi_ssl3
    rspi_sslp_t  rspi_ssl0
    rspi_sslp_t  rspi_ssl1
} rspi_ssl_polarity_t
```

12.1.231.1 rspi_ssl2

rspi_sslp_t::rspi_ssl2

12.1.231.2 rspi_ssl3

rspi_sslp_t::rspi_ssl3

12.1.231.3 rspi_ssl0

rspi_sslp_t::rspi_ssl0

12.1.231.4 rspi_ssl1

rspi_sslp_t::rspi_ssl1

12.1.232 rtc_alarm_time_t

```
typedef struct{
    rtc_time_t  time
    bool  sec_match
    bool  min_match
    bool  hour_match
    bool  mday_match
    bool  mon_match
    bool  year_match
    bool  dayofweek_match
} rtc_alarm_time_t
```

12.1.232.1 time

rtc_time_t::time

Brief description

Time structure.

12.1.232.2 sec_match

bool rtc_alarm_time_t::sec_match

Brief description

Enable the alarm based on a match of the seconds field.

12.1.232.3 min_match

bool rtc_alarm_time_t::min_match

Brief description

Enable the alarm based on a match of the minutes field.

12.1.232.4 hour_match

bool rtc_alarm_time_t::hour_match

Brief description

Enable the alarm based on a match of the hours field.

12.1.232.5 mday_match

`bool rtc_alarm_time_t::mday_match`

Brief description

Enable the alarm based on a match of the days field.

12.1.232.6 mon_match

`bool rtc_alarm_time_t::mon_match`

Brief description

Enable the alarm based on a match of the months field.

12.1.232.7 year_match

`bool rtc_alarm_time_t::year_match`

Brief description

Enable the alarm based on a match of the years field.

12.1.232.8 dayofweek_match

`bool rtc_alarm_time_t::dayofweek_match`

Brief description

Enable the alarm based on a match of the dayofweek field.

12.1.233 rtc_api_t

```
typedef struct{
    ssp_err_t(*  open)(rtc_ctrl_t *const p_ctrl, rtc_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(rtc_ctrl_t *const p_ctrl)
    ssp_err_t(*  calendarTimeSet)(rtc_ctrl_t *const p_ctrl, rtc_time_t *p_time,
bool clock_start)
    ssp_err_t(*  calendarTimeGet)(rtc_ctrl_t *const p_ctrl, rtc_time_t *p_time)
    ssp_err_t(*  calendarAlarmSet)(rtc_ctrl_t *const p_ctrl, rtc_alarm_time_t
*p_alarm, bool irq_enable_flag)
    ssp_err_t(*  calendarAlarmGet)(rtc_ctrl_t *const p_ctrl, rtc_alarm_time_t
*p_alarm)
    ssp_err_t(*  calendarCounterStart)(rtc_ctrl_t *const p_ctrl)
    ssp_err_t(*  calendarCounterStop)(rtc_ctrl_t *const p_ctrl)
    ssp_err_t(*  irqEnable)(rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
    ssp_err_t(*  irqDisable)(rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
    ssp_err_t(*  periodicIrqRateSet)(rtc_ctrl_t *const p_ctrl,
rtc_periodic_irq_select_t rate)
    ssp_err_t(*  infoGet)(rtc_ctrl_t *p_ctrl, rtc_info_t *p_rtc_info)
```

```
    ssp_err_t(*  versionGet)(ssp_version_t *version)
} rtc_api_t
```

12.1.234 rtc_callback_args_t

```
typedef struct{
    rtc_event_t  event
    void const  *  p_context
} rtc_callback_args_t
```

12.1.234.1 event

`rtc_event_t::event`

Brief description

The event can be used to identify what caused the callback (compare match or error).

12.1.234.2 p_context

`void const* rtc_callback_args_t::p_context`

Brief description

Placeholder for user data. Set in r_timer_t::open function in timer_cfg_t.

12.1.235 rtc_cfg_t

```
typedef struct{
    rtc_clock_source_t  clock_source
    uint32_t  error_adjustment_value
    rtc_error_adjustment_t  error_adjustment_type
    uint8_t  alarm_ipl
    uint8_t  periodic_ipl
    uint8_t  carry_ipl
    void(*  p_callback)(rtc_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} rtc_cfg_t
```

12.1.235.1 clock_source

`rtc_clock_source_t::clock_source`

Brief description

Clock source for the RTC block.

12.1.235.2 error_adjustment_value

```
uint32_t rtc_cfg_t::error_adjustment_value
```

Brief description

Value of the prescaler for error adjustment.

12.1.235.3 error_adjustment_type

```
rtc_error_adjustment_t::error_adjustment_type
```

Brief description

How the prescaler value is applied.

12.1.235.4 alarm_ipl

```
uint8_t rtc_cfg_t::alarm_ipl
```

Brief description

Alarm interrupt priority.

12.1.235.5 periodic_ipl

```
uint8_t rtc_cfg_t::periodic_ipl
```

Brief description

Periodic interrupt priority.

12.1.235.6 carry_ipl

```
uint8_t rtc_cfg_t::carry_ipl
```

Brief description

Carry interrupt priority.

12.1.235.7 p_callback

```
void(* rtc_cfg_t::p_callback) (rtc_callback_args_t *p_args)
```

Brief description

Called from the ISR.

12.1.235.8 p_context

```
void const* rtc_cfg_t::p_context
```

Brief description

Passed to the callback.

12.1.235.9 p_extend

```
void const* rtc_cfg_t::p_extend
```

Brief description

RTC hardware dependant configuration.

12.1.236 rtc_info_t

```
typedef struct{
    rtc_clock_source_t  clock_source
} rtc_info_t
```

12.1.236.1 clock_source

```
rtc_clock_source_t::clock_source
```

Brief description

Clock source for the RTC block.

12.1.237 rtc_instance_ctrl_t

```
typedef struct{
    void *  p_reg
    void(*  p_callback)(rtc_callback_args_t *cb_data)
    void const  *  p_context
    uint32_t  open
    IRQn_Type  alarm_irq
    IRQn_Type  periodic_irq
    IRQn_Type  carry_irq
    rtc_clock_source_t  clock_source
} rtc_instance_ctrl_t
```

12.1.237.1 p_reg

```
void* ::p_reg
```

Brief description

Pointer to register base address.

12.1.237.2 p_callback

```
void(* ::p_callback) ( *cb_data)
```

Brief description

Called from the ISR.

12.1.237.3 p_context

`void const* ::p_context`

Brief description

Passed to the callback.

12.1.237.4 open

`uint32_t ::open`

Brief description

Whether or not driver is open.

12.1.237.5 alarm_irq

`IRQn_Type ::alarm_irq`

Brief description

Alarm IRQ number.

12.1.237.6 periodic_irq

`IRQn_Type ::periodic_irq`

Brief description

Periodic IRQ number.

12.1.237.7 carry_irq

`IRQn_Type ::carry_irq`

Brief description

Carry IRQ number.

12.1.237.8 clock_source

`rtc_clock_source_t::clock_source`

Brief description

Clock source for the RTC block.

12.1.238 rtc_instance_t

```
typedef struct{
    rtc_ctrl_t *  p_ctrl
    rtc_cfg_t const  *  p_cfg
```

```
    rtc_api_t const  *  p_api
} rtc_instance_t
```

12.1.238.1 p_ctrl

`rtc_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.238.2 p_cfg

`rtc_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.238.3 p_api

`rtc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.239 sb_ble_prf_ias_set_alert_t

```
typedef struct{
    sf_ble_prf_ias_svc_code_t  svc_code
    sf_ble_prf_ias_alert_type_t  lvl
} sb_ble_prf_ias_set_alert_t
```

12.1.239.1 svc_code

`sf_ble_prf_ias_svc_code_t::svc_code`

Brief description

SVC code.

12.1.239.2 lvl

`sf_ble_prf_ias_alert_type_t::lvl`

Brief description

Alert level.

12.1.240 sci_i2c_extended_cfg

```
typedef struct{
    bool  bitrate_modulation
} sci_i2c_extended_cfg
```

12.1.240.1 bitrate_modulation

```
bool sci_i2c_extended_cfg::bitrate_modulation
```

Detailed description

Bitrate Modulation Function enable or disable

12.1.241 sci_i2c_instance_ctrl_t

```
typedef struct{
    i2c_cfg_t  info
    uint32_t  open
    void *  p_reg
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  tei_irq
    uint8_t *  p_buff
    uint32_t  total
    uint32_t  remain
    uint32_t  loaded
    uint8_t  addr_low
    uint8_t  addr_high
    uint8_t  addr_total
    uint8_t  addr_remain
    uint8_t  addr_loaded
    bool  read
    bool  restart
    bool  err
    bool  restarted
    bool  transaction_completed
    bool  do_dummy_read
    bool  activation_on_rxi
    bool  activation_on_txi
} sci_i2c_instance_ctrl_t
```

12.1.241.1 info

```
i2c_cfg_t::info
```

Brief description

Information describing I2C device.

12.1.241.2 open

```
uint32_t ::open
```

Brief description

Flag to determine if the device is open.

12.1.241.3 p_reg

```
void* ::p_reg
```

Brief description

Base register for this channel.

12.1.241.4 p_transfer_tx

```
transfer_instance_t::p_transfer_tx
```

Brief description

DTC instance for I2C transmit.Set to NULL if unused.

Detailed description

DTC/DMA support

12.1.241.5 p_transfer_rx

```
transfer_instance_t::p_transfer_rx
```

Brief description

DTC instance for I2C receive. Set to NULL if unused.

12.1.241.6 rxi_irq

```
IRQn_Type ::rxi_irq
```

Brief description

Receive IRQ number.

12.1.241.7 txi_irq

```
IRQn_Type ::txi_irq
```

Brief description

Transmit IRQ number.

12.1.241.8 tei_irq

`IRQn_Type ::tei_irq`

Brief description

Transmit end IRQ number.

12.1.241.9 p_buff

`uint8_t* ::p_buff`

Detailed description

Holds the data associated with the transfer

12.1.241.10 total

`uint32_t ::total`

Detailed description

Holds the total number of data bytes to transfer

12.1.241.11 remain

`uint32_t ::remain`

Detailed description

Tracks the remaining data bytes to transfer

12.1.241.12 loaded

`uint32_t ::loaded`

Detailed description

Tracks the number of data bytes written to the register

12.1.241.13 addr_low

`uint8_t ::addr_low`

Detailed description

Holds the last address byte to issue

12.1.241.14 addr_high

`uint8_t ::addr_high`

Detailed description

Holds the first address byte to issue in 10-bit mode

12.1.241.15 addr_total

```
uint8_t ::addr_total
```

Detailed description

Holds the total number of address bytes to transfer

12.1.241.16 addr_remain

```
uint8_t ::addr_remain
```

Detailed description

Tracks the remaining address bytes to transfer

12.1.241.17 addr_loaded

```
uint8_t ::addr_loaded
```

Detailed description

Tracks the number of address bytes written to the register

12.1.241.18 read

```
volatile bool ::read
```

Detailed description

Holds the direction of the data byte transfer

12.1.241.19 restart

```
volatile bool ::restart
```

Detailed description

Holds whether or not the restart should be issued when done

12.1.241.20 err

```
volatile bool ::err
```

Detailed description

Tracks whether or not an error occurred during processing

12.1.241.21 restarted

```
volatile bool ::restarted
```

Detailed description

Tracks whether or not a restart was issued during the previous transfer

12.1.241.22 transaction_completed

```
volatile bool ::transaction_completed
```

Detailed description

Tracks if the transaction started earlier was completed

12.1.241.23 do_dummy_read

```
volatile bool ::do_dummy_read
```

12.1.241.24 activation_on_rxi

```
volatile bool ::activation_on_rxi
```

Detailed description

<< Tracks whether a dummy read is issued on the first RX < Tracks whether the transfer is activated on RXI interrupt

12.1.241.25 activation_on_txi

```
volatile bool ::activation_on_txi
```

Detailed description

< Tracks whether the transfer is activated on TXI interrupt

12.1.242 sci_spi_extended_cfg

```
typedef struct{
    bool  bitrate_modulation
} sci_spi_extended_cfg
```

12.1.242.1 bitrate_modulation

```
bool sci_spi_extended_cfg::bitrate_modulation
```

Detailed description

Bitrate Modulation Function enable or disable

12.1.243 sci_spi_instance_ctrl_t

```
typedef struct{
    uint8_t  channel
    uint32_t  channel_opened
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    void(*  p_callback)(spi_callback_args_t *p_args)
    void const  *  p_context
```

```
    void *  p_reg
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  tei_irq
    IRQn_Type  eri_irq
    void *  p_src
    void *  p_dest
    uint16_t  tx_count
    uint16_t  rx_count
    uint32_t  xfr_length
    uint8_t  bytes_per_transfer
    bool  do_rx_now
    bool  do_tx
    spi_operation_t  transfer_mode
    uint8_t  rx_data
    bsp_lock_t  resource_lock_tx_rx
} sci_spi_instance_ctrl_t
```

12.1.243.1 channel

`uint8_t ::channel`

Brief description

Channel number to be used.

12.1.243.2 channel_opened

`uint32_t ::channel_opened`

Brief description

Internal flag to indicate the peripheral was initialized.

12.1.243.3 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Brief description

To use SPI DTC/DMA write transfer.

12.1.243.4 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Brief description

To use SPI DTC/DMA read transfer.

12.1.243.5 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Pointer to user callback function.

12.1.243.6 p_context

```
void const* ::p_context
```

Brief description

Pointer to the higher level device context.

12.1.243.7 p_reg

```
void* ::p_reg
```

Brief description

Base register for this channel.

12.1.243.8 rxi_irq

```
IRQn_Type ::rxi_irq
```

Brief description

Receive IRQ number.

12.1.243.9 txi_irq

```
IRQn_Type ::txi_irq
```

Brief description

Transmit IRQ number.

12.1.243.10 tei_irq

```
IRQn_Type ::tei_irq
```

Brief description

Transmit end IRQ number.

12.1.243.11 eri_irq

```
IRQn_Type ::eri_irq
```

Brief description

Error IRQ number.

12.1.243.12 p_src

```
void* ::p_src
```

12.1.243.13 p_dest

```
void* ::p_dest
```

12.1.243.14 tx_count

```
uint16_t ::tx_count
```

12.1.243.15 rx_count

```
uint16_t ::rx_count
```

12.1.243.16 xfr_length

```
uint32_t ::xfr_length
```

12.1.243.17 bytes_per_transfer

```
uint8_t ::bytes_per_transfer
```

12.1.243.18 do_rx_now

```
bool ::do_rx_now
```

12.1.243.19 do_tx

```
bool ::do_tx
```

12.1.243.20 transfer_mode

```
spi_operation_t::transfer_mode
```

12.1.243.21 rx_data

```
uint8_t ::rx_data
```

12.1.243.22 resource_lock_tx_rx

```
bsp_lock_t::resource_lock_tx_rx
```

Detailed description

Resource lock for transmission/reception

12.1.244 sci_uart_instance_ctrl_t

```
typedef struct{
    uint8_t  channel
    uint8_t  fifo_depth
    uint8_t  rx_transfer_in_progress
    uint8_t  data_bytes
    uint8_t  bitrate_modulation
    uint32_t  open
    transfer_instance_t const  *  p_transfer_rx
    transfer_instance_t const  *  p_transfer_tx
    uint8_t const  *  p_tx_src
    uint32_t  tx_src_bytes
    IRQn_Type  rxi_irq
    IRQn_Type  txi_irq
    IRQn_Type  tei_irq
    IRQn_Type  eri_irq
    void(*  p_callback)(uart_callback_args_t *p_args)
    void const  *  p_context
    void *  p_reg
    void(*  p_extpin_ctrl)(uint32_t channel, uint32_t level)
} sci_uart_instance_ctrl_t
```

12.1.244.1 channel

`uint8_t ::channel`

Brief description

Channel number.

12.1.244.2 fifo_depth

`uint8_t ::fifo_depth`

Brief description

FIFO depth of the UART channel.

12.1.244.3 rx_transfer_in_progress

`uint8_t ::rx_transfer_in_progress`

Brief description

Set to 1 if a receive transfer is in progress, 0 otherwise.

12.1.244.4 data_bytes

`uint8_t ::data_bytes`

Brief description

1 byte for 7 or 8 bit data, 2 bytes for 9 bit data

12.1.244.5 bitrate_modulation

```
uint8_t ::bitrate_modulation
```

Brief description

1 if bit rate modulation is enabled, 0 otherwise

12.1.244.6 open

```
uint32_t ::open
```

Brief description

Used to determine if the channel is configured.

12.1.244.7 p_transfer_rx

```
transfer_instance_t::p_transfer_rx
```

Detailed description

Optional transfer instance used to send or receive multiple bytes without interrupts.

12.1.244.8 p_transfer_tx

```
transfer_instance_t::p_transfer_tx
```

Detailed description

Optional transfer instance used to send or receive multiple bytes without interrupts.

12.1.244.9 p_tx_src

```
uint8_t const* ::p_tx_src
```

Detailed description

Source buffer pointer used to fill hardware FIFO from transmit ISR.

12.1.244.10 tx_src_bytes

```
uint32_t ::tx_src_bytes
```

Detailed description

Size of source buffer pointer used to fill hardware FIFO from transmit ISR.

12.1.244.11 rxi_irq

```
IRQn_Type ::rxi_irq
```

Brief description

Receive IRQ number.

12.1.244.12 txi_irq

```
IRQn_Type ::txi_irq
```

Brief description

Transmit IRQ number.

12.1.244.13 tei_irq

```
IRQn_Type ::tei_irq
```

Brief description

Transmit end IRQ number.

12.1.244.14 eri_irq

```
IRQn_Type ::eri_irq
```

Brief description

Error IRQ number.

12.1.244.15 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Pointer to callback function.

12.1.244.16 p_context

```
void const* ::p_context
```

Brief description

Pointer to user interrupt context data.

12.1.244.17 p_reg

```
void* ::p_reg
```

Brief description

Base register for this channel.

12.1.244.18 p_extpin_ctrl

```
void(* ::p_extpin_ctrl) (uint32_t channel, uint32_t level)
```

Brief description

External pin control.

12.1.245 sdhi_event_t

```
typedef struct{
    uint32_t  word
    struct sdhi_event_t::s_sdhi_event_type  bit
} sdhi_event_t
```

12.1.245.1 word

```
uint32_t ::word
```

12.1.245.2 bit

```
sdhi_event_t::s_sdhi_event_type::bit
```

12.1.246 sdhi_int_status_t

```
typedef struct{
    uint32_t  info2_mask
} sdhi_int_status_t
```

12.1.246.1 info2_mask

```
uint32_t ::info2_mask
```

12.1.247 sdhi_sdio_event_t

```
typedef struct{
    uint32_t  word
    struct sdhi_sdio_event_t::s_sdhi_sdio_event_type  bit
} sdhi_sdio_event_t
```

12.1.247.1 word

```
uint32_t ::word
```

12.1.247.2 bit

```
sdhi_sdio_event_t::s_sdhi_sdio_event_type::bit
```

12.1.248 sdmmc_api_t

```
typedef struct{
    ssp_err_t(*  open)(sdmmc_ctrl_t *const p_ctrl, sdmmc_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(sdmmc_ctrl_t *const p_ctrl)
    ssp_err_t(*  read)(sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const start_sector, uint32_t const sector_count)
    ssp_err_t(*  write)(sdmmc_ctrl_t *const p_ctrl, uint8_t const *const
p_source, uint32_t const start_sector, uint32_t const sector_count)
    ssp_err_t(*  control)(sdmmc_ctrl_t *const p_ctrl, ssp_command_t const
command, void *p_data)
    ssp_err_t(*  readIo)(sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_data,
uint32_t const function, uint32_t const address)
    ssp_err_t(*  writeIo)(sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_data,
uint32_t const function, uint32_t const address, sdmmc_io_write_mode_t const
read_after_write)
    ssp_err_t(*  readIoExt)(sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const function, uint32_t const address, uint32_t *const count,
sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t address_mode)
    ssp_err_t(*  writeIoExt)(sdmmc_ctrl_t *const p_ctrl, uint8_t const *const
p_source, uint32_t const function, uint32_t const address, uint32_t const count,
sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t address_mode)
    ssp_err_t(*  IoIntEnable)(sdmmc_ctrl_t *const p_ctrl, bool enable)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
    ssp_err_t(*  infoGet)(sdmmc_ctrl_t *const p_ctrl, sdmmc_info_t *const p_info)
    ssp_err_t(*  erase)(sdmmc_ctrl_t *const p_ctrl, uint32_t const start_sector,
uint32_t const sector_count)
} sdmmc_api_t
```

12.1.249 sdmmc_callback_args_t

```
typedef struct{
    sdmmc_event_t  event
    void const  *  p_context
} sdmmc_callback_args_t
```

12.1.249.1 event

`sdmmc_event_t::event`

Brief description

The event can be used to identify what caused the callback.

12.1.249.2 p_context

`void const* sdmmc_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.250 sdmmc_cfg_t

```
typedef struct{
    sdmmc_hw_t  hw
    transfer_instance_t const  *  p_lower_lvl_transfer
    void(*  p_callback)(sdmmc_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
    uint8_t  access_ipl
    uint8_t  sdio_ipl
    uint8_t  card_ipl
    uint8_t  dma_req_ipl
} sdmmc_cfg_t
```

12.1.250.1 hw

`sdmmc_hw_t::hw`

Brief description

Channel, media type, bus width defined by the hardware.

12.1.250.2 p_lower_lvl_transfer

`transfer_instance_t::p_lower_lvl_transfer`

Brief description

Transfer instance used to move data with DMA or DTC.

12.1.250.3 p_callback

```
void(* sdmmc_cfg_t::p_callback) (sdmmc_callback_args_t *p_args)
```

Brief description

Pointer to callback function.

12.1.250.4 p_context

```
void const* sdmmc_cfg_t::p_context
```

Brief description

User defined context passed into callback function.

12.1.250.5 p_extend

```
void const* sdmmc_cfg_t::p_extend
```

Brief description

SD/MMC hardware dependent configuration.

12.1.250.6 access_ipl

```
uint8_t sdmmc_cfg_t::access_ipl
```

Brief description

Access interrupt priority.

12.1.250.7 sdio_ipl

```
uint8_t sdmmc_cfg_t::sdio_ipl
```

Brief description

SDIO interrupt priority.

12.1.250.8 card_ipl

```
uint8_t sdmmc_cfg_t::card_ipl
```

Brief description

Card interrupt priority.

12.1.250.9 dma_req_ipl

```
uint8_t sdmmc_cfg_t::dma_req_ipl
```

Brief description

DMA request interrupt priority.

12.1.251 sdmmc_extended_cfg_t

```
typedef struct{
    uint32_t  block_size
    sdmmc_card_detect_t  card_detect
} sdmmc_extended_cfg_t
```

12.1.251.1 block_size

```
uint32_t ::block_size
```

Detailed description

Block size in bytes. Block size must be 512 bytes for SD cards and eMMC devices. Block size can be 1-512 bytes for SDIO.

12.1.251.2 card_detect

`sdmmc_card_detect_t::card_detect`

Detailed description

Whether or not card detection is used.

12.1.252 sdmmc_hw_t

```
typedef struct{
    uint8_t  channel
    sdmmc_media_type_t  media_type
    sdmmc_bus_width_t  bus_width
} sdmmc_hw_t
```

12.1.252.1 channel

`uint8_t sdmmc_hw_t::channel`

Brief description

Channel of SD/MMC host interface.

12.1.252.2 media_type

`sdmmc_media_type_t::media_type`

Brief description

Embedded or pluggable card.

12.1.252.3 bus_width

`sdmmc_bus_width_t::bus_width`

Brief description

Device bus width is 1, 4 or 8 bits wide.

12.1.253 sdmmc_info_t

```
typedef struct{
    sdmmc_card_type_t  card_type
    bool  ready
    bool  hc
    bool  sdio
    bool  write_protected
```

```
    bool  transfer_in_progress
    uint8_t  csd_version
    uint8_t  device_type
    sdmmc_bus_width_t  bus_width
    uint8_t  hs_timing
    uint32_t  sdhi_rca
    uint32_t  max_clock_rate
    uint32_t  clock_rate
    uint32_t  sector_size
    uint32_t  sector_count
    uint32_t  erase_sector_count
} sdmmc_info_t
```

12.1.253.1 card_type

`sdmmc_card_type_t::card_type`

Brief description

SD or eMMC.

12.1.253.2 ready

`bool sdmmc_info_t::ready`

Detailed description

False if card was removed (only applies if MCU supports card detection and SDnCD pin is connected).

True otherwise.

If ready is false, the driver must be closed, then reopened with a card inserted.

12.1.253.3 hc

`bool sdmmc_info_t::hc`

Brief description

true = Card is High Capacity card

12.1.253.4 sdio

`bool sdmmc_info_t::sdio`

Brief description

true = SDIO present

12.1.253.5 write_protected

`bool sdmmc_info_t::write_protected`

Brief description

true = Card is write protected

12.1.253.6 transfer_in_progress

```
bool sdmmc_info_t::transfer_in_progress
```

Brief description

true = Card is busy

12.1.253.7 csd_version

```
uint8_t sdmmc_info_t::csd_version
```

Brief description

CSD version.

12.1.253.8 device_type

```
uint8_t sdmmc_info_t::device_type
```

Brief description

Speed and data rate (eMMC)

12.1.253.9 bus_width

```
sdmmc_bus_width_t::bus_width
```

Brief description

Current media bus width.

12.1.253.10 hs_timing

```
uint8_t sdmmc_info_t::hs_timing
```

Brief description

High Speed status.

12.1.253.11 sdhi_rca

```
uint32_t sdmmc_info_t::sdhi_rca
```

Brief description

Relative Card Address.

12.1.253.12 max_clock_rate

```
uint32_t sdmmc_info_t::max_clock_rate
```

Brief description

Maximum clock rate for media card.

12.1.253.13 clock_rate

`uint32_t sdmmc_info_t::clock_rate`

Brief description

Current clock rate.

12.1.253.14 sector_size

`uint32_t sdmmc_info_t::sector_size`

Brief description

Sector size.

12.1.253.15 sector_count

`uint32_t sdmmc_info_t::sector_count`

Brief description

Sector count.

12.1.253.16 erase_sector_count

`uint32_t sdmmc_info_t::erase_sector_count`

Brief description

Minimum erasable unit (in 512 byte sectors)

12.1.254 sdmmc_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    sdmmc_hw_t  hw
    transfer_instance_t const  *  p_lower_lvl_transfer
    sdmmc_info_t  status
    bool  transfer_in_progress
    void(*  p_callback)(sdmmc_callback_args_t *p_args)
    void const  *  p_context
    R_SDHI0_Type *  p_reg
    sdhi_event_t  sdhi_event
    IRQn_Type  transfer_irq
    sdmmc_transfer_dir_t  transfer_dir
    uint8_t *  p_transfer_data
    uint32_t  transfer_blocks_total
    uint32_t  transfer_block_current
    uint32_t  transfer_block_size
```

```
    uint32_t  aligned_buff[SDMMC_MAX_BLOCK_SIZE/sizeof(uint32_t)]
} sdmmc_instance_ctrl_t
```

12.1.254.1 open

```
uint32_t ::open
```

12.1.254.2 hw

```
sdmmc_hw_t::hw
```

12.1.254.3 p_lower_lvl_transfer

```
transfer_instance_t::p_lower_lvl_transfer
```

12.1.254.4 status

```
sdmmc_info_t::status
```

12.1.254.5 transfer_in_progress

```
bool ::transfer_in_progress
```

12.1.254.6 p_callback

```
void(* ::p_callback) ( *p_args)
```

12.1.254.7 p_context

```
void const* ::p_context
```

12.1.254.8 p_reg

```
R_SDHI0_Type* ::p_reg
```

12.1.254.9 sdhi_event

```
sdhi_event_t::sdhi_event
```

12.1.254.10 transfer_irq

```
IRQn_Type ::transfer_irq
```

12.1.254.11 transfer_dir

sdmmc_transfer_dir_t::transfer_dir

12.1.254.12 p_transfer_data

uint8_t* ::p_transfer_data

12.1.254.13 transfer_blocks_total

uint32_t ::transfer_blocks_total

12.1.254.14 transfer_block_current

uint32_t ::transfer_block_current

12.1.254.15 transfer_block_size

uint32_t ::transfer_block_size

12.1.254.16 aligned_buff

uint32_t ::aligned_buff[SDMMC_MAX_BLOCK_SIZE/sizeof(uint32_t)]

12.1.255 sdmmc_instance_t

```
typedef struct{
    sdmmc_ctrl_t *  p_ctrl
    sdmmc_cfg_t const  *  p_cfg
    sdmmc_api_t const  *  p_api
} sdmmc_instance_t
```

12.1.255.1 p_ctrl

sdmmc_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.255.2 p_cfg

sdmmc_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.255.3 p_api

sdmmc_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.256 sdmmc_io_mode_t

```
typedef struct{
    sdmmc_io_command_t  command
    sdmmc_io_transfer_mode_t  transfer_mode
    sdmmc_io_address_mode_t  address_mode
    sdmmc_io_write_mode_t  write_mode
} sdmmc_io_mode_t
```

12.1.256.1 command

sdmmc_io_command_t::command

12.1.256.2 transfer_mode

sdmmc_io_transfer_mode_t::transfer_mode

12.1.256.3 address_mode

sdmmc_io_address_mode_t::address_mode

12.1.256.4 write_mode

sdmmc_io_write_mode_t::write_mode

12.1.257 sf_adc_periodic_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_adc_periodic_ctrl_t *const p_ctrl,
sf_adc_periodic_cfg_t const *const p_cfg)
    ssp_err_t(*  start)(sf_adc_periodic_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(sf_adc_periodic_ctrl_t *const p_ctrl)
    ssp_err_t(*  close)(sf_adc_periodic_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_adc_periodic_api_t
```

12.1.258 sf_adc_periodic_callback_args_t

```
typedef struct{
    sf_adc_periodic_event_t  event
    uint32_t  buffer_index
    void const  *  p_context
    adc_data_size_t *  p_data_buffer
    uint32_t  num_new_samples
} sf_adc_periodic_callback_args_t
```

12.1.258.1 event

`sf_adc_periodic_event_t::event`

Brief description

Periodic ADC callback event.

12.1.258.2 buffer_index

`uint32_t sf_adc_periodic_callback_args_t::buffer_index`

Brief description

Index to the buffer where the new data is stored.

12.1.258.3 p_context

`void const* sf_adc_periodic_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.258.4 p_data_buffer

`adc_data_size_t::p_data_buffer`

Brief description

Pointer to the buffer that will store the samples.

12.1.258.5 num_new_samples

`uint32_t sf_adc_periodic_callback_args_t::num_new_samples`

Brief description

Number of new samples in the data buffer.

12.1.259 sf_adc_periodic_cfg_t

```
typedef struct{
    adc_instance_t const  *const  p_lower_lvl_adc
    timer_instance_t const  *const  p_lower_lvl_timer
    transfer_instance_t const  *const  p_lower_lvl_transfer
    adc_data_size_t *  p_data_buffer
    uint32_t  data_buffer_length
    uint32_t  sample_count
    elc_event_t  scan_trigger
    void(*  p_callback)(sf_adc_periodic_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} sf_adc_periodic_cfg_t
```

12.1.259.1 p_lower_lvl_adc

`adc_instance_t::p_lower_lvl_adc`

Brief description

Pointer to the ADC instance.

12.1.259.2 p_lower_lvl_timer

`timer_instance_t::p_lower_lvl_timer`

Brief description

Pointer to the Timer instance.

12.1.259.3 p_lower_lvl_transfer

`transfer_instance_t::p_lower_lvl_transfer`

Brief description

Pointer to the Transfer instance.

12.1.259.4 p_data_buffer

`adc_data_size_t::p_data_buffer`

Brief description

Pointer to the buffer that will store the samples.

12.1.259.5 data_buffer_length

`uint32_t sf_adc_periodic_cfg_t::data_buffer_length`

Brief description

Length of the data buffer that will store the samples.

12.1.259.6 sample_count

```
uint32_t sf_adc_periodic_cfg_t::sample_count
```

Brief description

Samples per channel to be buffered before notifying the app.

12.1.259.7 scan_trigger

```
elc_event_t::scan_trigger
```

Brief description

The hardware trigger that starts the ADC scan.

12.1.259.8 p_callback

```
void(* sf_adc_periodic_cfg_t::p_callback) (sf_adc_periodic_callback_args_t
*p_args)
```

Brief description

Callback function.

12.1.259.9 p_context

```
void const* sf_adc_periodic_cfg_t::p_context
```

Brief description

Placeholder for user data.

12.1.259.10 p_extend

```
void const* sf_adc_periodic_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.260 sf_adc_periodic_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    TX_MUTEX  mutex
    adc_instance_t const  *  p_lower_lvl_adc
    timer_instance_t const  *  p_lower_lvl_timer
    transfer_instance_t const  *  p_lower_lvl_transfer
    void const  *volatile  p_src_transfer
    adc_data_size_t *  p_data_buffer
```

```
    uint32_t  data_buffer_length
    uint32_t  data_buffer_index
    uint32_t  sample_count
    uint32_t  dtc_transfer_length
    void(*  p_callback)(sf_adc_periodic_callback_args_t *p_args)
    void const  *  p_context
} sf_adc_periodic_instance_ctrl_t
```

12.1.260.1 open

`uint32_t ::open`

Brief description

Used by driver to check if pointer to control block is valid.

12.1.260.2 mutex

`TX_MUTEX::mutex`

Brief description

Mutex used to protect access to lower level driver hardware registers.

12.1.260.3 p_lower_lvl_adc

`adc_instance_t::p_lower_lvl_adc`

Brief description

Pointer to the ADC instance.

12.1.260.4 p_lower_lvl_timer

`timer_instance_t::p_lower_lvl_timer`

Brief description

Pointer to the Timer instance.

12.1.260.5 p_lower_lvl_transfer

`transfer_instance_t::p_lower_lvl_transfer`

Brief description

Pointer to the Transfer instance.

12.1.260.6 p_src_transfer

`void const* volatile ::p_src_transfer`

Brief description

Source pointer for the low level transfer method.

12.1.260.7 p_data_buffer

`adc_data_size_t::p_data_buffer`

Brief description

Pointer to the buffer that will store the samples.

12.1.260.8 data_buffer_length

`uint32_t ::data_buffer_length`

Brief description

Length of the data buffer that will store the samples.

12.1.260.9 data_buffer_index

`uint32_t ::data_buffer_index`

Brief description

Index of the data buffer where data is to be written to next.

12.1.260.10 sample_count

`uint32_t ::sample_count`

Brief description

Samples per channel to be buffered before notifying the app.

12.1.260.11 dtc_transfer_length

`uint32_t ::dtc_transfer_length`

Brief description

Total Length of DTC transfer for requested number of samples.

12.1.260.12 p_callback

`void(* ::p_callback) ( *p_args)`

Brief description

Callback function.

12.1.260.13 p_context

`void const* ::p_context`

Brief description

Placeholder for user data.

12.1.261 sf_adc_periodic_instance_t

```
typedef struct{
    sf_adc_periodic_ctrl_t *  p_ctrl
    sf_adc_periodic_cfg_t const  *  p_cfg
    sf_adc_periodic_api_t const  *  p_api
} sf_adc_periodic_instance_t
```

12.1.261.1 p_ctrl

`sf_adc_periodic_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.261.2 p_cfg

`sf_adc_periodic_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.261.3 p_api

`sf_adc_periodic_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.262 sf_audio_playback_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_audio_playback_ctrl_t *const p_ctrl,
sf_audio_playback_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(sf_audio_playback_ctrl_t *const p_ctrl)
    ssp_err_t(*  start)(sf_audio_playback_ctrl_t *const p_ctrl,
sf_audio_playback_data_t *const p_data, UINT const timeout)
    ssp_err_t(*  pause)(sf_audio_playback_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(sf_audio_playback_ctrl_t *const p_ctrl)
    ssp_err_t(*  resume)(sf_audio_playback_ctrl_t *const p_ctrl)
    ssp_err_t(*  volumeSet)(sf_audio_playback_ctrl_t *const p_ctrl, uint8_t const
volume)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_audio_playback_api_t
```

12.1.263 sf_audio_playback_cfg_t

```
typedef struct{
    void(*  p_callback)(sf_message_callback_args_t *p_args)
    sf_audio_playback_common_ctrl_t *  p_common_ctrl
    sf_audio_playback_common_cfg_t const  *  p_common_cfg
    uint8_t  class_instance
} sf_audio_playback_cfg_t
```

12.1.263.1 p_callback

```
void(* sf_audio_playback_cfg_t::p_callback) (sf_message_callback_args_t *p_args)
```

Detailed description

Callback called when playback of a buffer passed to start is complete. Set to NULL for no callback.

12.1.263.2 p_common_ctrl

```
sf_audio_playback_common_ctrl_t::p_common_ctrl
```

Detailed description

Pointer to the hardware control block used by this stream.

12.1.263.3 p_common_cfg

```
sf_audio_playback_common_cfg_t::p_common_cfg
```

Detailed description

Pointer to common configurations shared by all streams using the same hardware.

12.1.263.4 class_instance

```
uint8_t sf_audio_playback_cfg_t::class_instance
```

Brief description

Class instance used to identify the stream to the messaging framework.

12.1.264 sf_audio_playback_common_cfg_t

```
typedef struct{
    UINT  priority
    sf_audio_playback_hw_instance_t const  *  p_lower_lvl_hw
    sf_message_instance_t const  *  p_message
    TX_QUEUE *  p_queue
    void const  *  p_extend
} sf_audio_playback_common_cfg_t
```

12.1.264.1 priority

`UINT sf_audio_playback_common_cfg_t::priority`

Brief description

Priority of the audio playback thread.

12.1.264.2 p_lower_lvl_hw

`sf_audio_playback_hw_instance_t::p_lower_lvl_hw`

Brief description

Hardware instance.

12.1.264.3 p_message

`sf_message_instance_t::p_message`

Detailed description

Pointer to messaging framework instance used to post audio messages.

12.1.264.4 p_queue

`TX_QUEUE::p_queue`

Detailed description

Pointer to the messaging framework queue specified for this audio stream. Must be subscribed to the SF_MESSAGE_EVENT_CLASS_AUDIO event class.

12.1.264.5 p_extend

`void const* sf_audio_playback_common_cfg_t::p_extend`

Detailed description

Implementation specific extension configuration.

12.1.265 sf_audio_playback_common_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    void const  *  p_next_buffer
    uint32_t  next_length
    sf_message_instance_t const  *  p_message
    TX_QUEUE *  p_queue
    sf_audio_playback_hw_instance_t const  *  p_lower_lvl_hw
    sf_audio_playback_instance_ctrl_t
*  p_stream[SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS]
    TX_THREAD  thread
    TX_EVENT_FLAGS_GROUP  flags
```

```
    sf_audio_playback_data_type_t  data_type
    uint8_t  volume
    uint8_t  buffer_index
    uint8_t  stack[SF_AUDIO_PLAYBACK_STACK_SIZE]
    int16_t  samples[2][SF_AUDIO_PLAYBACK_CFG_BUFFER_SIZE_BYTES/sizeof(int16_t)]
    bool  playing
} sf_audio_playback_common_instance_ctrl_t
```

12.1.265.1 open

`uint32_t ::open`

Brief description

Used to determine if driver is initialized.

12.1.265.2 p_next_buffer

`void const* ::p_next_buffer`

Brief description

Pointer to next buffer (to be played when the current buffer completes).

12.1.265.3 next_length

`uint32_t ::next_length`

Brief description

Length of next buffer (to be played when the current buffer completes).

12.1.265.4 p_message

`sf_message_instance_t::p_message`

Brief description

Pointer to message control block.

12.1.265.5 p_queue

`TX_QUEUE::p_queue`

Brief description

Queue subscribed to SF_MESSAGE_EVENT_CLASS_AUDIO events.

12.1.265.6 p_lower_lvl_hw

`sf_audio_playback_hw_instance_t::p_lower_lvl_hw`

Brief description

Hardware API's used.

12.1.265.7 p_stream

`* ::p_stream[SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS]`

Brief description

Stream specific data.

12.1.265.8 thread

`TX_THREAD::thread`

Brief description

Main audio thread.

12.1.265.9 flags

`TX_EVENT_FLAGS_GROUP::flags`

Brief description

Event flags used to end wait in audio thread.

12.1.265.10 data_type

`sf_audio_playback_data_type_t::data_type`

Brief description

Sample format required by the hardware.

12.1.265.11 volume

`uint8_t ::volume`

Brief description

Volume from 0 (muted) to 255 (maximum, default on open).

12.1.265.12 buffer_index

`uint8_t ::buffer_index`

Brief description

Which ping pong buffer to use.

12.1.265.13 stack

`uint8_t ::stack[SF_AUDIO_PLAYBACK_STACK_SIZE]`

Brief description

Stack for audio thread.

12.1.265.14 samples

`int16_t ::samples[2][SF_AUDIO_PLAYBACK_CFG_BUFFER_SIZE_BYTES/sizeof(int16_t)]`

Detailed description

Ping pong buffers, used to store converted data during transfer.

12.1.265.15 playing

`bool ::playing`

Brief description

State of audio instance (currently playing if true)

12.1.266 sf_audio_playback_data_t

```
typedef struct{
    sf_message_header_t  header
    sf_audio_playback_data_type_t  type
    uint32_t  size_bytes
    void const  *  p_data
    UINT  loop_timeout
    bool  stream_end
} sf_audio_playback_data_t
```

12.1.266.1 header

`sf_message_header_t::header`

Brief description

Required common members of messaging framework payloads.

12.1.266.2 type

`sf_audio_playback_data_type_t::type`

Brief description

Data type. Must be uncompressed.

12.1.266.3 size_bytes

`uint32_t sf_audio_playback_data_t::size_bytes`

Brief description

Size of data in bytes.

12.1.266.4 p_data

```
void const* sf_audio_playback_data_t::p_data
```

Brief description

Pointer to data. Data start address must be 4-byte aligned.

12.1.266.5 loop_timeout

```
UINT sf_audio_playback_data_t::loop_timeout
```

Detailed description

ThreadX timeout, select TX_NO_WAIT to play the entire buffer once, TX_WAIT_FOREVER to loop until SF_AUDIO_PLAYBACK_Pause is called from another thread, or any timeout value from (0x00000001 through 0xFFFFFFFE) in ThreadX tick counts to loop until the tick counts expire.

12.1.266.6 stream_end

```
bool sf_audio_playback_data_t::stream_end
```

Detailed description

This releases ownership of the stream and allows other threads to post data. Set to true if not more data will be sent as a part of this logical bitstream. Set to false if more packets are being prepared.

12.1.267 sf_audio_playback_data_type_t

```
typedef struct{
    uint8_t  scale_bits_max
    bool  is_signed
} sf_audio_playback_data_type_t
```

12.1.267.1 scale_bits_max

```
uint8_t sf_audio_playback_data_type_t::scale_bits_max
```

Brief description

Maximum data resolution in bits.

12.1.267.2 is_signed

```
bool sf_audio_playback_data_type_t::is_signed
```

Brief description

Set to 1 for signed samples, or 0 for unsigned samples.

12.1.268 sf_audio_playback_hw_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_audio_playback_hw_ctrl_t *const p_ctrl,
sf_audio_playback_hw_cfg_t const *const p_cfg)
    ssp_err_t(*  start)(sf_audio_playback_hw_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(sf_audio_playback_hw_ctrl_t *const p_ctrl)
    ssp_err_t(*  play)(sf_audio_playback_hw_ctrl_t *const p_ctrl, int16_t const
*const p_buffer, uint32_t length)
    ssp_err_t(*  dataTypeGet)(sf_audio_playback_hw_ctrl_t *const p_ctrl,
sf_audio_playback_data_type_t *const p_data_type)
    ssp_err_t(*  close)(sf_audio_playback_hw_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_audio_playback_hw_api_t
```

12.1.269 sf_audio_playback_hw_callback_args_t

```
typedef struct{
    void *  p_context
} sf_audio_playback_hw_callback_args_t
```

12.1.269.1 p_context

```
void* sf_audio_playback_hw_callback_args_t::p_context
```

Detailed description

Placeholder for user data. Set in open function in sf_audio_playback_hw_cfg_t.

12.1.270 sf_audio_playback_hw_cfg_t

```
typedef struct{
    void(*  p_callback)(sf_audio_playback_hw_callback_args_t *p_args)
    void *  p_context
    void const  *  p_extend
} sf_audio_playback_hw_cfg_t
```

12.1.270.1 p_callback

```
void(* sf_audio_playback_hw_cfg_t::p_callback)
(sf_audio_playback_hw_callback_args_t *p_args)
```

Detailed description

Callback called when play is complete. Set to NULL for no callback.

12.1.270.2 p_context

```
void* sf_audio_playback_hw_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

12.1.270.3 p_extend

```
void const* sf_audio_playback_hw_cfg_t::p_extend
```

Brief description

Hardware dependent configuration.

12.1.271 sf_audio_playback_hw_dac_cfg_t

```
typedef struct{
    dac_instance_t const  *  p_lower_lvl_dac
    timer_instance_t const  *  p_lower_lvl_timer
    transfer_instance_t const  *  p_lower_lvl_transfer
} sf_audio_playback_hw_dac_cfg_t
```

12.1.271.1 p_lower_lvl_dac

```
dac_instance_t::p_lower_lvl_dac
```

Brief description

DAC API used to access DAC hardware.

12.1.271.2 p_lower_lvl_timer

```
timer_instance_t::p_lower_lvl_timer
```

Brief description

Timer API used to generate sampling frequency.

12.1.271.3 p_lower_lvl_transfer

```
transfer_instance_t::p_lower_lvl_transfer
```

Brief description

Transfer API used to transfer data each sampling frequency.

12.1.272 sf_audio_playback_hw_dac_instance_ctrl_t

```
typedef struct{
    void(*  p_callback)(sf_audio_playback_hw_callback_args_t *p_args)
    void *  p_context
```

```
    dac_instance_t const  *  p_lower_lvl_dac
    timer_instance_t const  *  p_lower_lvl_timer
    transfer_instance_t const  *  p_lower_lvl_transfer
} sf_audio_playback_hw_dac_instance_ctrl_t
```

12.1.272.1 p_callback

```
void(* sf_::p_callback) ( *p_args)
```

Detailed description

Callback called when play is complete.

12.1.272.2 p_context

```
void* sf_::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

12.1.272.3 p_lower_lvl_dac

```
dac_instance_t::p_lower_lvl_dac
```

Brief description

DAC API used to access DAC hardware.

12.1.272.4 p_lower_lvl_timer

```
timer_instance_t::p_lower_lvl_timer
```

Brief description

Timer API used to generate sampling frequency.

12.1.272.5 p_lower_lvl_transfer

```
transfer_instance_t::p_lower_lvl_transfer
```

Brief description

Transfer API used to transfer data each sampling frequency.

12.1.273 sf_audio_playback_hw_i2s_cfg_t

```
typedef struct{
    i2s_instance_t const  *  p_lower_lvl_i2s
} sf_audio_playback_hw_i2s_cfg_t
```

12.1.273.1 p_lower_lvl_i2s

`i2s_instance_t::p_lower_lvl_i2s`

Brief description

I2S API used to access I2S hardware.

12.1.274 sf_audio_playback_hw_i2s_instance_ctrl_t

```
typedef struct{
    void(*  p_callback)(sf_audio_playback_hw_callback_args_t *p_args)
    void *  p_context
    i2s_instance_t const  *  p_lower_lvl_i2s
} sf_audio_playback_hw_i2s_instance_ctrl_t
```

12.1.274.1 p_callback

`void(* sf_::p_callback) ( *p_args)`

Detailed description

Callback called when play is complete.

12.1.274.2 p_context

`void* sf_::p_context`

Detailed description

Placeholder for user data. Passed to the user callback in sf_audio_playback_hw_callback_args_t.

12.1.274.3 p_lower_lvl_i2s

`i2s_instance_t::p_lower_lvl_i2s`

Brief description

I2S API used to access I2S hardware.

12.1.275 sf_audio_playback_hw_instance_t

```
typedef struct{
    sf_audio_playback_hw_ctrl_t *  p_ctrl
    sf_audio_playback_hw_cfg_t const  *  p_cfg
    sf_audio_playback_hw_api_t const  *  p_api
} sf_audio_playback_hw_instance_t
```

12.1.275.1 p_ctrl

`sf_audio_playback_hw_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.275.2 p_cfg

`sf_audio_playback_hw_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.275.3 p_api

`sf_audio_playback_hw_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.276 sf_audio_playback_instance_t

```
typedef struct{
    sf_audio_playback_ctrl_t *  p_ctrl
    sf_audio_playback_cfg_t const  *  p_cfg
    sf_audio_playback_api_t const  *  p_api
} sf_audio_playback_instance_t
```

12.1.276.1 p_ctrl

`sf_audio_playback_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.276.2 p_cfg

`sf_audio_playback_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.276.3 p_api

`sf_audio_playback_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.277 sf_audio_record_adc_hw_cfg_t

```
typedef struct{
    sf_adc_periodic_instance_t const  *  p_lower_lvl_adc_periodic
} sf_audio_record_adc_hw_cfg_t
```

12.1.277.1 p_lower_lvl_adc_periodic

sf_adc_periodic_instance_t::p_lower_lvl_adc_periodic

12.1.278 sf_audio_record_adc_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    TX_MUTEX  mutex
    void *  p_capture_data_buffer
    uint32_t  sample_count
    void(*  p_callback)(sf_audio_record_callback_args_t *p_args)
    void const  *  p_context
    sf_adc_periodic_instance_t const  *  p_lower_lvl_adc_periodic
} sf_audio_record_adc_instance_ctrl_t
```

12.1.278.1 open

uint32_t ::open

Detailed description

Used by driver to check if pointer to control block is valid

12.1.278.2 mutex

TX_MUTEX::mutex

Detailed description

Mutex used to protect access to lower level driver hardware registers

12.1.278.3 p_capture_data_buffer

void* ::p_capture_data_buffer

Detailed description

Pointer to the buffer that will store the samples

12.1.278.4 sample_count

uint32_t ::sample_count

Detailed description

Samples per channel to be buffered before notifying the app

12.1.278.5 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback function.

12.1.278.6 p_context

```
void const* ::p_context
```

Brief description

Placeholder for user data.

12.1.278.7 p_lower_lvl_adc_periodic

```
sf_adc_periodic_instance_t::p_lower_lvl_adc_periodic
```

Brief description

Lower level ADC periodic instance.

12.1.279 sf_audio_record_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_audio_record_ctrl_t *const p_ctrl,
sf_audio_record_cfg_t const *const p_cfg)
    ssp_err_t(*  start)(sf_audio_record_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(sf_audio_record_ctrl_t *const p_ctrl)
    ssp_err_t(*  infoGet)(sf_audio_record_ctrl_t *const p_ctrl,
sf_audio_record_info_t *p_info)
    ssp_err_t(*  close)(sf_audio_record_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_audio_record_api_t
```

12.1.280 sf_audio_record_callback_args_t

```
typedef struct{
    sf_audio_record_event_t  event
    uint32_t  buffer_index
    void const  *  p_context
} sf_audio_record_callback_args_t
```

12.1.280.1 event

`sf_audio_record_event_t::event`

Brief description

Audio callback event.

12.1.280.2 buffer_index

`uint32_t sf_audio_record_callback_args_t::buffer_index`

Brief description

Index to the buffer where the new data is stored.

12.1.280.3 p_context

`void const* sf_audio_record_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.281 sf_audio_record_cfg_t

```
typedef struct{
    sf_audio_record_data_size_t  capture_data_size
    uint32_t  sampling_rate_hz
    void *  p_capture_data_buffer
    uint32_t  capture_data_buffer_size
    uint32_t  sample_count
    void(*  p_callback)(sf_audio_record_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} sf_audio_record_cfg_t
```

12.1.281.1 capture_data_size

`sf_audio_record_data_size_t::capture_data_size`

Brief description

Size of data in the sample 8 or 16 bit.

12.1.281.2 sampling_rate_hz

`uint32_t sf_audio_record_cfg_t::sampling_rate_hz`

Brief description

Sampling rate for audio capture.

12.1.281.3 p_capture_data_buffer

```
void* sf_audio_record_cfg_t::p_capture_data_buffer
```

Detailed description

Pointer to the buffer that will store the samples

12.1.281.4 capture_data_buffer_size

```
uint32_t sf_audio_record_cfg_t::capture_data_buffer_size
```

Detailed description

total size of buffer configured by user to store samples

12.1.281.5 sample_count

```
uint32_t sf_audio_record_cfg_t::sample_count
```

Detailed description

Samples per channel to be buffered before notifying the user via callback

12.1.281.6 p_callback

```
void(* sf_audio_record_cfg_t::p_callback) (sf_audio_record_callback_args_t
*p_args)
```

Brief description

Callback function.

12.1.281.7 p_context

```
void const* sf_audio_record_cfg_t::p_context
```

Brief description

Placeholder for user data.

12.1.281.8 p_extend

```
void const* sf_audio_record_cfg_t::p_extend
```

Detailed description

Extension parameter for hardware specific settings.

12.1.282 sf_audio_record_info_t

```
typedef struct{
    sf_audio_record_channel_t  channel_info
} sf_audio_record_info_t
```

12.1.282.1 channel_info

`sf_audio_record_channel_t::channel_info`

12.1.283 sf_audio_record_instance_t

```
typedef struct{
    sf_audio_record_ctrl_t *  p_ctrl
    sf_audio_record_cfg_t const  *  p_cfg
    sf_audio_record_api_t const  *  p_api
} sf_audio_record_instance_t
```

12.1.283.1 p_ctrl

`sf_audio_record_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.283.2 p_cfg

`sf_audio_record_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.283.3 p_api

`sf_audio_record_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.284 sf_ble_addr_t

```
typedef struct{
    uint8_t  addr[SF_BLE_ADDR_LEN]
} sf_ble_addr_t
```

12.1.284.1 addr

`uint8_t sf_ble_addr_t::addr[SF_BLE_ADDR_LEN]`

Brief description

6-byte array address value

12.1.285 sf_ble_addr_verify_ind_t

```
typedef struct{
    uint8_t  bd_addr[SF_BLE_ADDR_LEN]
    sf_ble_addr_type_t  addr_type
    uint8_t  accept_addr
} sf_ble_addr_verify_ind_t
```

12.1.285.1 bd_addr

`uint8_t sf_ble_addr_verify_ind_t::bd_addr[SF_BLE_ADDR_LEN]`

Brief description

input parameter specifying bluetooth address of remote device

12.1.285.2 addr_type

`sf_ble_addr_type_t::addr_type`

Brief description

input parameter specifying Address type of remote BLE device

12.1.285.3 accept_addr

`uint8_t sf_ble_addr_verify_ind_t::accept_addr`

Brief description

output parameter: application has to set this parameter in callback if it accepts this address

12.1.286 sf_ble_adv_data_t

```
typedef struct{
    uint8_t  data[SF_BLE_ADV_DATA_LEN]
    uint8_t  adv_data_length
} sf_ble_adv_data_t
```

12.1.286.1 data

`uint8_t sf_ble_adv_data_t::data[SF_BLE_ADV_DATA_LEN]`

Brief description

Advertising data bytes array.

12.1.286.2 adv_data_length

`uint8_t sf_ble_adv_data_t::adv_data_length`

Brief description

Advertising data length.

12.1.287 sf_ble_adv_info_t

```
typedef struct{
    sf_ble_disc_type_t  disc_mode
    sf_ble_conn_type_t  conn_mode
    uint16_t  adv_intv_min
    uint16_t  adv_intv_max
    sf_ble_addr_type_t  own_addr_type
    sf_ble_adv_chnl_map_t  adv_chnl_map
    sf_ble_adv_filt_type_t  adv_filt_policy
    sf_ble_adv_data_t  adv_data
} sf_ble_adv_info_t
```

12.1.287.1 disc_mode

`sf_ble_disc_type_t::disc_mode`

Brief description

Discovery mode.

12.1.287.2 conn_mode

`sf_ble_conn_type_t::conn_mode`

Brief description

Connection mode.

12.1.287.3 adv_intv_min

`uint16_t sf_ble_adv_info_t::adv_intv_min`

Brief description

Minimum interval for advertising.

12.1.287.4 adv_intv_max

`uint16_t sf_ble_adv_info_t::adv_intv_max`

Brief description

Maximum interval for advertising.

12.1.287.5 own_addr_type

`sf_ble_addr_type_t::own_addr_type`

Brief description

Own address type: public=0x00 /random = 0x01.

12.1.287.6 adv_chnl_map

`sf_ble_adv_chnl_map_t::adv_chnl_map`

Brief description

Advertising channel map.

12.1.287.7 adv_filt_policy

`sf_ble_adv_filt_type_t::adv_filt_policy`

Brief description

Advertising filter policy.

12.1.287.8 adv_data

`sf_ble_adv_data_t::adv_data`

Brief description

Advertising data.

12.1.288 sf_ble_anp_ancp_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_anp_ancp_t  control_point_value
} sf_ble_anp_ancp_change_t
```

12.1.288.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.288.2 control_point_value

`sf_ble_anp_ancp_t::control_point_value`

Brief description

Control point value.

12.1.289 sf_ble_anp_ancp_t

```
typedef struct{
    sf_ble_prf_anp_cmd_id_t  command_id
    sf_ble_prf_anp_category_id  category_id
} sf_ble_anp_ancp_t
```

12.1.289.1 command_id

sf_ble_prf_anp_cmd_id_t::command_id

Brief description

Command type.

12.1.289.2 category_id

sf_ble_prf_anp_category_id::category_id

Brief description

Category on which to act.

12.1.290 sf_ble_anp_api_new_alert_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_anp_api_new_alert_t  new_alert
} sf_ble_anp_api_new_alert_ntf_t
```

12.1.291 sf_ble_anp_api_new_alert_t

```
typedef struct{
    sf_ble_prf_anp_category_id  category_id
    uint8_t  alert_num
    uint8_t  text_size
    uint8_t  text[SF_BLE_ANP_ALT_TEXT_MAX]
} sf_ble_anp_api_new_alert_t
```

12.1.292 sf_ble_anp_api_unread_alert_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_anp_api_unread_alert_t  alert
} sf_ble_anp_api_unread_alert_ntf_t
```

12.1.293 sf_ble_anp_api_unread_alert_t

```
typedef struct{
    sf_ble_prf_anp_category_id  category_id
    uint8_t  unread_count
} sf_ble_anp_api_unread_alert_t
```

12.1.294 sf_ble_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_ble_ctrl_t *const p_ctrl, const sf_ble_cfg_t *p_cfg)
    ssp_err_t(*  close)(sf_ble_ctrl_t *const p_ctrl)
    ssp_err_t(*  infoGet)(sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t
*p_handle, sf_ble_info_t *p_ble_info)
    ssp_err_t(*  provisionGet)(sf_ble_ctrl_t *const p_ctrl, sf_ble_provisioning_t
*p_ble_provisioning)
    ssp_err_t(*  provisionSet)(sf_ble_ctrl_t *const p_ctrl, const
sf_ble_provisioning_t *p_ble_provisioning)
    ssp_err_t(*  scan)(sf_ble_ctrl_t *const p_ctrl, sf_ble_scan_t *p_scan,
uint8_t *p_cnt, sf_ble_scan_info_t *p_scan_info)
    ssp_err_t(*  advertisementStart)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_adv_info_t *const p_advt_info)
    ssp_err_t(*  advertisementStop)(sf_ble_ctrl_t *const p_ctrl)
    ssp_err_t(*  whitelistAdd)(sf_ble_ctrl_t *const p_ctrl, const uint8_t
*p_bd_addr)
    ssp_err_t(*  whitelistDel)(sf_ble_ctrl_t *const p_ctrl, const uint8_t
*p_bd_addr)
    ssp_err_t(*  bondingStart)(sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t
*p_handle, const uint8_t *p_bd_addr, sf_ble_bonding_start_t *p_bonding_start)
    ssp_err_t(*  bondingResponse)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, const uint8_t *p_bd_addr,
sf_ble_bonding_response_t *p_bonding_resp)
    ssp_err_t(*  startEncryption)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_sm_enc_info_t const *p_enc_info)
    ssp_err_t(*  connect)(sf_ble_ctrl_t *const p_ctrl, sf_ble_connection_t const
*const p_conn, sf_ble_conn_handle_t *p_handle)
    ssp_err_t(*  listen)(sf_ble_ctrl_t *const p_ctrl)
    ssp_err_t(*  disconnect)(sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t
*p_handle)
    ssp_err_t(*  gattAddCustomProfiles)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_svc_attribute_t *p_svc_attr, uint32_t svc_attr_len,
sf_ble_char_attribute_t *p_char_attr, uint32_t char_attr_len)
    ssp_err_t(*  gattServiceDiscovery)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_service_discovery_req_t const *const
p_sf_ble_svc_dscv_req, sf_ble_service_discovery_rsp_t *const
p_sf_ble_svc_dscv_rsp, uint32_t *const p_rsp_cnt)
    ssp_err_t(*  gattCharDiscovery)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_char_discovery_req_t const *const
```

```
p_sf_ble_char_dscv_req, sf_ble_char_discovery_rsp_t *const
p_sf_ble_char_dscv_rsp, uint32_t *const p_rsp_cnt)
    ssp_err_t(*  gattCharDescDiscovery)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t start_handle, uint16_t end_handle,
sf_ble_char_desc_discovery_rsp_t *const p_sf_ble_chardesc_dscv_rsp, uint32_t
*const p_rsp_cnt)
    ssp_err_t(*  gattCharRead)(sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t
*p_handle, sf_ble_char_read_req_t const *const p_char_read_req,
sf_ble_char_read_rsp_t *const p_char_read_rsp)
    ssp_err_t(*  gattCharWrite)(sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t
*p_handle, sf_ble_char_write_req_t const *const p_char_write_req)
    ssp_err_t(*  gattCharExecuteWrite)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_ble_execute_write_t execute_flag)
    ssp_err_t(*  gattCharWriteLocal)(sf_ble_ctrl_t *const p_ctrl, uint16_t
char_handle, uint16_t data_length, uint8_t *const p_data)
    ssp_err_t(*  gattSendNotify)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
    ssp_err_t(*  gattSendIndicate)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
    ssp_err_t(*  gattWriteResponse)(sf_ble_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, uint16_t handle, sf_ble_attribute_error_code_t
error_code)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_ble_api_t
```

12.1.295 sf_ble_attr_info_t

```
typedef struct{
    uint8_t  len
    uint8_t  data[SF_BLE_ATTM_MAX_VALUE]
} sf_ble_attr_info_t
```

12.1.295.1 len

`uint8_t sf_ble_attr_info_t::len`

Brief description

data length

12.1.295.2 data

`uint8_t sf_ble_attr_info_t::data[SF_BLE_ATTM_MAX_VALUE]`

Brief description

data

12.1.296 sf_ble_bas_battery_lvl_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_bas_battery_lvl_t  batt_lvl
} sf_ble_bas_battery_lvl_ntf_t
```

12.1.296.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.296.2 batt_lvl

`sf_ble_prf_bas_battery_lvl_t::batt_lvl`

Brief description

Battery Level.

12.1.297 sf_ble_blp_meas_info_t

```
typedef struct{
    uint8_t  flag_stable_meas
    uint8_t  flags
    int16_t  press_val1
    int16_t  press_val2
    int16_t  press_val3
    sf_ble_prf_cts_date_time_t  stamp
    int16_t  rate
    uint8_t  id
    uint8_t  reserved
    uint16_t  meas_sts
} sf_ble_blp_meas_info_t
```

12.1.297.1 flag_stable_meas

`uint8_t sf_ble_blp_meas_info_t::flag_stable_meas`

Brief description

Stable or intermediary type of measurements. Set to 0 if intermediate measurement.

12.1.297.2 flags

`uint8_t sf_ble_blp_meas_info_t::flags`

Brief description

flags

12.1.297.3 press_val1

`int16_t sf_ble_blp_meas_info_t::press_val1`

Brief description

blood pressure value - Systolic or cuff pressure. Cuff pressure has to be filled here

12.1.297.4 press_val2

`int16_t sf_ble_blp_meas_info_t::press_val2`

Brief description

blood pressure value - Diastolic or subfield1

12.1.297.5 press_val3

`int16_t sf_ble_blp_meas_info_t::press_val3`

Brief description

blood pressure value - MAP or subfield2

12.1.297.6 stamp

`sf_ble_prf_cts_date_time_t::stamp`

Brief description

time stamp

12.1.297.7 rate

`int16_t sf_ble_blp_meas_info_t::rate`

Brief description

pulse rate

12.1.297.8 id

`uint8_t sf_ble_blp_meas_info_t::id`

Brief description

user ID

12.1.297.9 reserved

`uint8_t sf_ble_blp_meas_info_t::reserved`

Brief description

Reserved.

12.1.297.10 meas_sts

`uint16_t sf_ble_blp_meas_info_t::meas_sts`

Brief description

measurement status

12.1.298 sf_ble_blp_meas_recv_data_t

```
typedef struct{
    uint16_t  conhdl
    sf_ble_onbp_char_t  char_code
    sf_ble_blp_meas_info_t  meas_info
} sf_ble_blp_meas_recv_data_t
```

12.1.298.1 conhdl

`uint16_t sf_ble_blp_meas_recv_data_t::conhdl`

Brief description

Connection handle.

12.1.298.2 char_code

`sf_ble_onbp_char_t::char_code`

Brief description

Characteristic code.

12.1.298.3 meas_info

`sf_ble_blp_meas_info_t::meas_info`

Brief description

BLP measurement data.

12.1.299 sf_ble_bonding_req_ind_t

```
typedef struct{
    sf_ble_addr_t  bd_addr
    uint8_t  index
    uint8_t  auth_req
    uint8_t  io_cap
    uint8_t  oob_data_flg
    uint8_t  max_enc_size
```

```
    uint8_t  ikey_dist
    uint8_t  rkey_dist
} sf_ble_bonding_req_ind_t
```

12.1.299.1 bd_addr

`sf_ble_addr_t::bd_addr`

Brief description

BLE address.

12.1.299.2 index

`uint8_t sf_ble_bonding_req_ind_t::index`

Brief description

Connection index.

12.1.299.3 auth_req

`uint8_t sf_ble_bonding_req_ind_t::auth_req`

Brief description

Authentication request type.

12.1.299.4 io_cap

`uint8_t sf_ble_bonding_req_ind_t::io_cap`

Brief description

IO capability.

12.1.299.5 oob_data_flg

`uint8_t sf_ble_bonding_req_ind_t::oob_data_flg`

Brief description

Indicating if OOB data is present.

12.1.299.6 max_enc_size

`uint8_t sf_ble_bonding_req_ind_t::max_enc_size`

Brief description

Maximum encryption key size.

12.1.299.7 ikey_dist

`uint8_t sf_ble_bonding_req_ind_t::ikey_dist`

Brief description

Type of key distributed by the initiator.

12.1.299.8 rkey_dist

`uint8_t sf_ble_bonding_req_ind_t::rkey_dist`

Brief description

Type of key distributed by the responder.

12.1.300 sf_ble_bonding_response_t

```
typedef struct{
    uint8_t  accept
    sf_ble_iocap_t  io_cap
    uint8_t  max_key_size
    sf_ble_key_dist_t  ikeys
    sf_ble_key_dist_t  rkeys
} sf_ble_bonding_response_t
```

12.1.300.1 accept

`uint8_t sf_ble_bonding_response_t::accept`

Brief description

accept or reject bonding

12.1.300.2 io_cap

`sf_ble_iocap_t::io_cap`

Brief description

IO capabilities.

12.1.300.3 max_key_size

`uint8_t sf_ble_bonding_response_t::max_key_size`

Brief description

Max key size.

12.1.300.4 ikeys

`sf_ble_key_dist_t::ikeys`

Brief description

Initiator key distribution.

12.1.300.5 rkeys

`sf_ble_key_dist_t::rkeys`

Brief description

Responder key distribution.

12.1.301 sf_ble_bonding_start_t

```
typedef struct{
    sf_ble_iocap_t  iocap
    uint8_t  key_size
    sf_ble_key_dist_t  ikey_dist
    sf_ble_key_dist_t  rkey_dist
} sf_ble_bonding_start_t
```

12.1.301.1 iocap

`sf_ble_iocap_t::iocap`

Brief description

IO capabilities.

12.1.301.2 key_size

`uint8_t sf_ble_bonding_start_t::key_size`

Brief description

Encryption key size.

12.1.301.3 ikey_dist

`sf_ble_key_dist_t::ikey_dist`

Brief description

Initiator key distribution.

12.1.301.4 rkey_dist

`sf_ble_key_dist_t::rkey_dist`

Brief description

Responder key distribution.

12.1.302 sf_ble_cfg_t

```
typedef struct{
    uint8_t  bd_addr[SF_BLE_ADDR_LEN]
    sf_ble_addr_type_t  own_addr_type
    uint8_t  max_slaves
    uint8_t  update_bd_addr
    uint16_t  scan_interval
    uint16_t  scan_window
    uint16_t  disc_time
    uint16_t  con_interval
    uint16_t  slave_latency
    uint16_t  sup_timeout
    void const  *  p_extend
} sf_ble_cfg_t
```

12.1.302.1 bd_addr

`uint8_t sf_ble_cfg_t::bd_addr[SF_BLE_ADDR_LEN]`

Brief description

BLE address.

12.1.302.2 own_addr_type

`sf_ble_addr_type_t::own_addr_type`

Brief description

self address type

12.1.302.3 max_slaves

`uint8_t sf_ble_cfg_t::max_slaves`

Brief description

Maximum slaves allowed to be connected.

12.1.302.4 update_bd_addr

`uint8_t sf_ble_cfg_t::update_bd_addr`

Brief description

Set this to true to update bluetooth address during SF_BLE_Open.

12.1.302.5 scan_interval

`uint16_t sf_ble_cfg_t::scan_interval`

Brief description

BLE scan interval for receiving advertisement.

12.1.302.6 scan_window

`uint16_t sf_ble_cfg_t::scan_window`

Brief description

Period of time during which advertising data is received at the scan interval.

12.1.302.7 disc_time

`uint16_t sf_ble_cfg_t::disc_time`

Brief description

Duration for which the device remain discoverable.

12.1.302.8 con_interval

`uint16_t sf_ble_cfg_t::con_interval`

Brief description

Interval for transmitting and receiving data periodically after connection establishment.

12.1.302.9 slave_latency

`uint16_t sf_ble_cfg_t::slave_latency`

Brief description

Period of time during which data is transmitted and received at the connection interval.

12.1.302.10 sup_timeout

`uint16_t sf_ble_cfg_t::sup_timeout`

Brief description

Link loss time-out.

12.1.302.11 p_extend

`void const* sf_ble_cfg_t::p_extend`

Brief description

Instance specific configuration.

12.1.303 sf_ble_char_attribute_t

```
typedef struct{
    sf_ble_svc_attribute_t *  p_service
```

```
    sf_ble_uuid_t  attr_uuid
    uint16_t  attr_declare_handle
    uint16_t  attr_declare_type
    uint16_t  attr_value_handle
    sf_ble_char_attr_permissions_t  attr_perm
    sf_ble_char_property_t  attr_properties
    uint8_t *  p_attr_value
    uint8_t  attr_value_len
} sf_ble_char_attribute_t
```

12.1.303.1 p_service

sf_ble_svc_attribute_t::p_service

Brief description

Service to which this characteristics belongs.

12.1.303.2 attr_uuid

sf_ble_uuid_t::attr_uuid

Brief description

UUID of characteristics value.

12.1.303.3 attr_declare_handle

uint16_t sf_ble_char_attribute_t::attr_declare_handle

Brief description

Characteristics handle.

12.1.303.4 attr_declare_type

uint16_t sf_ble_char_attribute_t::attr_declare_type

Brief description

Characteristics declare type SF_BLE_GATT_CHAR_DECLARE.

12.1.303.5 attr_value_handle

uint16_t sf_ble_char_attribute_t::attr_value_handle

Brief description

Characteristics value handle.

12.1.303.6 attr_perm

sf_ble_char_attr_permissions_t::attr_perm

Brief description

Characteristics permission.

12.1.303.7 attr_properties

`sf_ble_char_property_t::attr_properties`

Brief description

Characteristics properties.

12.1.303.8 p_attr_value

`uint8_t* sf_ble_char_attribute_t::p_attr_value`

Brief description

Characteristics value data.

12.1.303.9 attr_value_len

`uint8_t sf_ble_char_attribute_t::attr_value_len`

Brief description

Characteristics value data length.

12.1.304 sf_ble_char_desc_discovery_rsp_t

```
typedef struct{
    uint16_t  desc_handle
    sf_ble_uuid_t  uuid
} sf_ble_char_desc_discovery_rsp_t
```

12.1.304.1 desc_handle

`uint16_t sf_ble_char_desc_discovery_rsp_t::desc_handle`

Brief description

Characteristic descriptor handle.

12.1.304.2 uuid

`sf_ble_uuid_t::uuid`

Brief description

Characteristic descriptor UUID.

12.1.305 sf_ble_char_discovery_req_t

```
typedef struct{
    sf_ble_uuid_t  uuid
    uint16_t  start_handle
    uint16_t  end_handle
    sf_ble_char_discovery_t  discovery_type
} sf_ble_char_discovery_req_t
```

12.1.305.1 uuid

`sf_ble_uuid_t::uuid`

Brief description

Characteristic UUID.

12.1.305.2 start_handle

`uint16_t sf_ble_char_discovery_req_t::start_handle`

Brief description

Discovery start handle.

12.1.305.3 end_handle

`uint16_t sf_ble_char_discovery_req_t::end_handle`

Brief description

Discovery end handle.

12.1.305.4 discovery_type

`sf_ble_char_discovery_t::discovery_type`

Brief description

Characteristic discovery type.

12.1.306 sf_ble_char_discovery_rsp_t

```
typedef struct{
    uint16_t  char_handle
    sf_ble_uuid_t  uuid
    uint16_t  value_handle
    sf_ble_char_property_t  property
} sf_ble_char_discovery_rsp_t
```

12.1.306.1 char_handle

```
uint16_t sf_ble_char_discovery_rsp_t::char_handle
```

Brief description

Characteristic handle.

12.1.306.2 uuid

```
sf_ble_uuid_t::uuid
```

Brief description

Characteristic UUID.

12.1.306.3 value_handle

```
uint16_t sf_ble_char_discovery_rsp_t::value_handle
```

Brief description

Characteristic value handle.

12.1.306.4 property

```
sf_ble_char_property_t::property
```

Brief description

Characteristic property.

12.1.307 sf_ble_char_multiple_read_req_t

```
typedef struct{
    uint16_t  handles[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
    uint8_t  expected_result_size[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
} sf_ble_char_multiple_read_req_t
```

12.1.307.1 handles

```
uint16_t
sf_ble_char_multiple_read_req_t::handles[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
```

Brief description

Characteristic handles for multiple read.

12.1.307.2 expected_result_size

```
uint8_t
sf_ble_char_multiple_read_req_t::expected_result_size[SF_BLE_MAX_MULTI_CHAR_READ
_CNT]
```

Brief description

Expected result length in bytes.

12.1.308 sf_ble_char_multiple_read_rsp_t

```
typedef struct{
    uint8_t *  p_data[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
    uint16_t  data_len[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
} sf_ble_char_multiple_read_rsp_t
```

12.1.308.1 p_data

```
uint8_t* sf_ble_char_multiple_read_rsp_t::p_data[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
```

Brief description

Pointer to Characteristic value.

12.1.308.2 data_len

```
uint16_t
sf_ble_char_multiple_read_rsp_t::data_len[SF_BLE_MAX_MULTI_CHAR_READ_CNT]
```

Brief description

Characteristic value length.

12.1.309 sf_ble_char_read_by_handle_rsp_t

```
typedef struct{
    uint8_t *  p_data
    uint16_t  data_len
} sf_ble_char_read_by_handle_rsp_t
```

12.1.309.1 p_data

```
uint8_t* sf_ble_char_read_by_handle_rsp_t::p_data
```

Brief description

Pointer to Characteristic value.

12.1.309.2 data_len

```
uint16_t sf_ble_char_read_by_handle_rsp_t::data_len
```

Brief description

Characteristic value length.

12.1.310 sf_ble_char_read_by_uuid_rsp_t

```
typedef struct{
    uint16_t  handle[SF_BLE_MAX_CHAR_UUID_READ_CNT]
    uint8_t *  p_data[SF_BLE_MAX_CHAR_UUID_READ_CNT]
    uint16_t  data_len[SF_BLE_MAX_CHAR_UUID_READ_CNT]
} sf_ble_char_read_by_uuid_rsp_t
```

12.1.310.1 handle

```
uint16_t sf_ble_char_read_by_uuid_rsp_t::handle[SF_BLE_MAX_CHAR_UUID_READ_CNT]
```

Brief description

Characteristic handles.

12.1.310.2 p_data

```
uint8_t* sf_ble_char_read_by_uuid_rsp_t::p_data[SF_BLE_MAX_CHAR_UUID_READ_CNT]
```

Brief description

Pointer to Characteristic value.

12.1.310.3 data_len

```
uint16_t sf_ble_char_read_by_uuid_rsp_t::data_len[SF_BLE_MAX_CHAR_UUID_READ_CNT]
```

Brief description

Characteristic value length.

12.1.311 sf_ble_char_read_req_t

```
typedef struct{
    sf_ble_char_read_t  char_read_type
    uint16_t  handle
    sf_ble_uuid_t  uuid
    uint16_t  offset
    sf_ble_char_multiple_read_req_t *  p_mul_read_req
    uint16_t  handles_cnt
} sf_ble_char_read_req_t
```

12.1.311.1 char_read_type

```
sf_ble_char_read_t::char_read_type
```

Brief description

Characteristic read type.

12.1.311.2 handle

`uint16_t sf_ble_char_read_req_t::handle`

Brief description

Characteristic value or descriptor handle.

12.1.311.3 uuid

`sf_ble_uuid_t::uuid`

Brief description

Characteristic UUID.

12.1.311.4 offset

`uint16_t sf_ble_char_read_req_t::offset`

Brief description

Offset for long Characteristic value.

12.1.311.5 p_mul_read_req

`sf_ble_char_multiple_read_req_t::p_mul_read_req`

Brief description

Pointer to multiple read Characteristic request.

12.1.311.6 handles_cnt

`uint16_t sf_ble_char_read_req_t::handles_cnt`

Brief description

Characteristic handles count for multiple read.

12.1.312 sf_ble_char_read_rsp_t

```
typedef struct{
    sf_ble_char_read_by_handle_rsp_t *  p_char_read_by_handle_rsp
    sf_ble_char_read_by_uuid_rsp_t *  p_char_read_by_uuid_rsp
    sf_ble_char_multiple_read_rsp_t *  p_char_multiple_read_rsp
} sf_ble_char_read_rsp_t
```

12.1.312.1 p_char_read_by_handle_rsp

`sf_ble_char_read_by_handle_rsp_t::p_char_read_by_handle_rsp`

Brief description

Response for read Characteristic by handle.

12.1.312.2 p_char_read_by_uuid_rsp

`sf_ble_char_read_by_uuid_rsp_t::p_char_read_by_uuid_rsp`

Brief description

Response for read Characteristic by UUID.

12.1.312.3 p_char_multiple_read_rsp

`sf_ble_char_multiple_read_rsp_t::p_char_multiple_read_rsp`

Brief description

Response for read multiple Characteristics.

12.1.313 sf_ble_char_write_req_t

```
typedef struct{
    sf_ble_char_write_t  char_write_type
    uint16_t  handle
    uint8_t *  p_data
    uint16_t  offset
    uint16_t  data_length
    sf_ble_execute_write_t  auto_execute
} sf_ble_char_write_req_t
```

12.1.313.1 char_write_type

`sf_ble_char_write_t::char_write_type`

Brief description

Characteristic write type.

12.1.313.2 handle

`uint16_t sf_ble_char_write_req_t::handle`

Brief description

Characteristic value or descriptor handle.

12.1.313.3 p_data

`uint8_t* sf_ble_char_write_req_t::p_data`

Brief description

Pointer to data.

12.1.313.4 offset

`uint16_t sf_ble_char_write_req_t::offset`

Brief description

Offset for long Characteristic value.

12.1.313.5 data_length

`uint16_t sf_ble_char_write_req_t::data_length`

Brief description

Data length.

12.1.313.6 auto_execute

`sf_ble_execute_write_t::auto_execute`

Brief description

Automatic execute write flag.

12.1.314 sf_ble_chipset_info_t

```
typedef struct{
    uint32_t  version
    uint8_t  bd_addr[SF_BLE_ADDR_LEN]
} sf_ble_chipset_info_t
```

12.1.314.1 version

`uint32_t sf_ble_chipset_info_t::version`

Brief description

chipset version

12.1.314.2 bd_addr

`uint8_t sf_ble_chipset_info_t::bd_addr[SF_BLE_ADDR_LEN]`

Brief description

BLE address.

12.1.315 sf_ble_connect_info_t

```
typedef struct{
    uint8_t  status
    uint8_t  reserved
    sf_ble_conn_handle_t  conhdl
```

```
    uint8_t  peer_addr_type
    sf_ble_addr_t  peer_addr
    uint8_t  reserved2
    uint16_t  con_interval
    uint16_t  con_latency
    uint16_t  sup_to
    uint8_t  clk_accuracy
    uint8_t  reserved3
} sf_ble_connect_info_t
```

12.1.315.1 status

`uint8_t sf_ble_connect_info_t::status`

Brief description

Confirmation status.

12.1.315.2 reserved

`uint8_t sf_ble_connect_info_t::reserved`

Brief description

Reserved.

12.1.315.3 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.315.4 peer_addr_type

`uint8_t sf_ble_connect_info_t::peer_addr_type`

Brief description

Peer address type.

12.1.315.5 peer_addr

`sf_ble_addr_t::peer_addr`

Brief description

Peer BT address.

12.1.315.6 reserved2

`uint8_t sf_ble_connect_info_t::reserved2`

Brief description

Reserved.

12.1.315.7 con_interval

`uint16_t sf_ble_connect_info_t::con_interval`

Brief description

Connection interval.

12.1.315.8 con_latency

`uint16_t sf_ble_connect_info_t::con_latency`

Brief description

Connection latency.

12.1.315.9 sup_to

`uint16_t sf_ble_connect_info_t::sup_to`

Brief description

Link supervision time-out.

12.1.315.10 clk_accuracy

`uint8_t sf_ble_connect_info_t::clk_accuracy`

Brief description

Clock accuracy.

12.1.315.11 reserved3

`uint8_t sf_ble_connect_info_t::reserved3`

Brief description

Reserved.

12.1.316 sf_ble_connection_t

```
typedef struct{
    sf_ble_init_filt_type_t  init_filt_type
    sf_ble_addr_type_t  addr_type
    uint8_t  bd_addr[SF_BLE_ADDR_LEN]
} sf_ble_connection_t
```

12.1.316.1 init_filt_type

`sf_ble_init_filt_type_t::init_filt_type`

Brief description

Connection filter type.

12.1.316.2 addr_type

`sf_ble_addr_type_t::addr_type`

Brief description

BLE address type.

12.1.316.3 bd_addr

`uint8_t sf_ble_connection_t::bd_addr[SF_BLE_ADDR_LEN]`

Brief description

BLE address.

12.1.317 sf_ble_ctrl_t

```
typedef struct{
    void *  p_driver_handle
} sf_ble_ctrl_t
```

12.1.317.1 p_driver_handle

`void* sf_ble_ctrl_t::p_driver_handle`

Brief description

Storage for information needed for each BLE device driver in the system.

12.1.318 sf_ble_cts_curr_time_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_cts_curr_time_t  current_time
} sf_ble_cts_curr_time_ntf_t
```

12.1.318.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.318.2 current_time

`sf_ble_prf_cts_curr_time_t::current_time`

Brief description

heart rate measurement data

12.1.319 sf_ble_cts_local_time_t

```
typedef struct{
    int8_t  time_zone
    uint8_t  dst_offset
} sf_ble_cts_local_time_t
```

12.1.319.1 time_zone

`int8_t sf_ble_cts_local_time_t::time_zone`

Brief description

Time Zone.

12.1.319.2 dst_offset

`uint8_t sf_ble_cts_local_time_t::dst_offset`

Brief description

DST Offset.

12.1.320 sf_ble_cts_ref_time_t

```
typedef struct{
    uint8_t  time_source
    uint8_t  accuracy
    uint8_t  days_since_update
    uint8_t  hours_since_update
} sf_ble_cts_ref_time_t
```

12.1.320.1 time_source

`uint8_t sf_ble_cts_ref_time_t::time_source`

Brief description

Source of time.

12.1.320.2 accuracy

`uint8_t sf_ble_cts_ref_time_t::accuracy`

Brief description

Estimated accuracy of time compared to original time source.

12.1.320.3 days_since_update

`uint8_t sf_ble_cts_ref_time_t::days_since_update`

Brief description

Days that passed since time was updated successfully from time source.

12.1.320.4 hours_since_update

`uint8_t sf_ble_cts_ref_time_t::hours_since_update`

Brief description

Time that passed since time was updated successfully from time source.

12.1.321 sf_ble_disconnect_t

```
typedef struct{
    sf_ble_disconnect_reason_t  reason
    uint8_t  status
    sf_ble_conn_handle_t  conhdl
} sf_ble_disconnect_t
```

12.1.321.1 reason

`sf_ble_disconnect_reason_t::reason`

Brief description

Disconnection reason.

12.1.321.2 status

`uint8_t sf_ble_disconnect_t::status`

Brief description

Disconnect status if initiated by local host.

12.1.321.3 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle of the remote device.

12.1.322 sf_ble_event_info_t

```
typedef struct{
    void *  p_data
    uint16_t  event
} sf_ble_event_info_t
```

12.1.322.1 p_data

`void* sf_ble_event_info_t::p_data`

Brief description

Data for BLE event.

12.1.322.2 event

`uint16_t sf_ble_event_info_t::event`

Brief description

Event type.

12.1.323 sf_ble_gatt_attr_event_t

```
typedef struct{
    uint16_t  attr_handle
    uint16_t  offset
    uint8_t const  *  p_value
    uint16_t  length
    ssp_err_t  response
} sf_ble_gatt_attr_event_t
```

12.1.323.1 attr_handle

`uint16_t sf_ble_gatt_attr_event_t::attr_handle`

Brief description

Attribute handle for which event is generated.

12.1.323.2 offset

`uint16_t sf_ble_gatt_attr_event_t::offset`

Brief description

The offset at which the client is willing to write.

12.1.323.3 p_value

```
uint8_t const* sf_ble_gatt_attr_event_t::p_value
```

Brief description

The value at which the client is willing to write.

12.1.323.4 length

```
uint16_t sf_ble_gatt_attr_event_t::length
```

Brief description

The amount of bytes that the client is willing to write as from this offset.

12.1.323.5 response

```
ssp_err_t sf_ble_gatt_attr_event_t::response
```

Brief description

User will update this variable, Pass SSP_SUCCESS if user accepts the remote request.

12.1.324 sf_ble_gatt_notif_ind_event_data_t

```
typedef struct{
    uint16_t  handle
    uint8_t *  p_data
    uint16_t  data_length
} sf_ble_gatt_notif_ind_event_data_t
```

12.1.324.1 handle

```
uint16_t sf_ble_gatt_notif_ind_event_data_t::handle
```

Brief description

Characteristic value or descriptor handle.

12.1.324.2 p_data

```
uint8_t* sf_ble_gatt_notif_ind_event_data_t::p_data
```

Brief description

Pointer to data.

12.1.324.3 data_length

```
uint16_t sf_ble_gatt_notif_ind_event_data_t::data_length
```

Brief description

Data length.

12.1.325 sf_ble_hrp_api_hrmeas_t

```
typedef struct{
    uint8_t  flags
    uint8_t  rr_interval_num
    uint16_t  heart_rate_measure
    uint16_t  energy_expended
    uint16_t  rr_interval[SF_BLE_PRF_HRP_API_RR_INTERVAL_BUFF_LEN]
} sf_ble_hrp_api_hrmeas_t
```

12.1.326 sf_ble_hrp_api_meas_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_hrp_api_hrmeas_t  measurements_info
} sf_ble_hrp_api_meas_ntf_t
```

12.1.327 sf_ble_hrp_cp_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_hrp_api_hrcp_t  control_point_value
} sf_ble_hrp_cp_change_t
```

12.1.327.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.327.2 control_point_value

`sf_ble_prf_hrp_api_hrcp_t::control_point_value`

Brief description

Control point value.

12.1.328 sf_ble_info_t

```
typedef struct{
    sf_ble_chipset_info_t  chipset
```

```
    uint16_t  rssi
} sf_ble_info_t
```

12.1.328.1 chipset

`sf_ble_chipset_info_t::chipset`

Brief description

Chipset information.

12.1.328.2 rssi

`uint16_t sf_ble_info_t::rssi`

Brief description

RSSI value.

12.1.329 sf_ble_instance_t

```
typedef struct{
    sf_ble_ctrl_t *  p_ctrl
    sf_ble_cfg_t const  *  p_cfg
    sf_ble_api_t const  *  p_api
} sf_ble_instance_t
```

12.1.329.1 p_ctrl

`sf_ble_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.329.2 p_cfg

`sf_ble_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.329.3 p_api

`sf_ble_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.330 sf_ble_long_attr_info_t

```
typedef struct{
    uint8_t  val_len
    uint8_t  reserved
    uint16_t  attr_hdl
    uint8_t  value[SF_BLE_ATTM_MAX_VALUE]
} sf_ble_long_attr_info_t
```

12.1.330.1 val_len

`uint8_t sf_ble_long_attr_info_t::val_len`

Brief description

size of the value data

12.1.330.2 reserved

`uint8_t sf_ble_long_attr_info_t::reserved`

Brief description

Reserved.

12.1.330.3 attr_hdl

`uint16_t sf_ble_long_attr_info_t::attr_hdl`

Brief description

Attribute handle.

12.1.330.4 value

`uint8_t sf_ble_long_attr_info_t::value[SF_BLE_ATTM_MAX_VALUE]`

Brief description

actual value pairs

12.1.331 sf_ble_onboard_profile_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_ble_onboard_profile_ctrl_t *const p_ctrl, const
sf_ble_onboard_profile_cfg_t *p_cfg)
    ssp_err_t(*  close)(sf_ble_onboard_profile_ctrl_t *const p_ctrl)
    ssp_err_t(*  onbpEnable)(sf_ble_onboard_profile_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_profile_callback_t
p_prf_cb, sf_ble_prf_sec_t sec)
    ssp_err_t(*  onbpDisable)(sf_ble_onboard_profile_ctrl_t *const p_ctrl,
sf_ble_conn_handle_t *p_handle, sf_onbp_t profile)
```

```
    ssp_err_t(*  onbpServerWriteData)(sf_ble_onboard_profile_ctrl_t *const
p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t
characteristics, const void *p_data)
    ssp_err_t(*  onbpServerSendNotification)(sf_ble_onboard_profile_ctrl_t
*const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile,
sf_ble_onbp_char_t characteristics, const void *p_data)
    ssp_err_t(*  onbpServerSendIndication)(sf_ble_onboard_profile_ctrl_t *const
p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t
characteristics, const void *p_data)
    ssp_err_t(*  onbpClientWriteCCCD)(sf_ble_onboard_profile_ctrl_t *const
p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t
cccd_char, sf_ble_cccd_val_t cccd_val)
    ssp_err_t(*  onbpClientWriteChar)(sf_ble_onboard_profile_ctrl_t *const
p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t
characteristics, const void *p_data)
    ssp_err_t(*  onbpClientReadChar)(sf_ble_onboard_profile_ctrl_t *const
p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t
characteristics)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_ble_onboard_profile_api_t
```

12.1.332 sf_ble_onboard_profile_cccd_changed_t

```
typedef struct{
    sf_ble_conn_handle_t  conn_handle
    sf_ble_onbp_char_t  char_code
    sf_ble_cccd_val_t  cccd_val
    uint8_t  inst_idx
} sf_ble_onboard_profile_cccd_changed_t
```

12.1.332.1 conn_handle

`sf_ble_conn_handle_t::conn_handle`

Brief description

Connection handle.

12.1.332.2 char_code

`sf_ble_onbp_char_t::char_code`

Brief description

CCCD type that has been changed by the remote node.

12.1.332.3 cccd_val

`sf_ble_cccd_val_t::cccd_val`

Brief description

CCCD value.

12.1.332.4 inst_idx

`uint8_t sf_ble_onboard_profile_cccd_changed_t::inst_idx`

Brief description

Instance index, Applicable for HOGP.

12.1.333 sf_ble_onboard_profile_cfg_t

```
typedef struct{
    sf_ble_instance_t const  *  p_low_lvl_ble
    void *  p_extend
} sf_ble_onboard_profile_cfg_t
```

12.1.333.1 p_low_lvl_ble

`sf_ble_instance_t::p_low_lvl_ble`

Brief description

Low level BLE Interface.

12.1.333.2 p_extend

`void* sf_ble_onboard_profile_cfg_t::p_extend`

Brief description

Extended configuration.

12.1.334 sf_ble_onboard_profile_ctrl_t

```
typedef struct{
    sf_ble_instance_t *  p_low_lvl_ble
} sf_ble_onboard_profile_ctrl_t
```

12.1.334.1 p_low_lvl_ble

`sf_ble_instance_t::p_low_lvl_ble`

Brief description

Low level BLE Framework information needed by On-Board Profile.

12.1.335 sf_ble_onboard_profile_instance_t

```
typedef struct{
    sf_ble_onboard_profile_ctrl_t *  p_ctrl
    sf_ble_onboard_profile_cfg_t const  *  p_cfg
    sf_ble_onboard_profile_api_t const  *  p_api
} sf_ble_onboard_profile_instance_t
```

12.1.335.1 p_ctrl

sf_ble_onboard_profile_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.335.2 p_cfg

sf_ble_onboard_profile_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.335.3 p_api

sf_ble_onboard_profile_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.336 sf_ble_prf_alert_status_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_alert_status  alert_status
} sf_ble_prf_alert_status_ntf_t
```

12.1.336.1 conhdl

sf_ble_conn_handle_t::conhdl

Brief description

Connection handle.

12.1.336.2 alert_status

sf_ble_prf_alert_status::alert_status

Brief description

Ringer control point value.

12.1.337 sf_ble_prf_cts_curr_time_t

```
typedef struct{
    sf_ble_prf_cts_date_time_t  stamp
    uint8_t  day_of_week
    uint8_t  fractions256
    uint8_t  adjust_reason
    uint8_t  reserved
} sf_ble_prf_cts_curr_time_t
```

12.1.337.1 stamp

`sf_ble_prf_cts_date_time_t::stamp`

Brief description

Date time info.

12.1.337.2 day_of_week

`uint8_t sf_ble_prf_cts_curr_time_t::day_of_week`

Brief description

Day of week.

12.1.337.3 fractions256

`uint8_t sf_ble_prf_cts_curr_time_t::fractions256`

Brief description

Fraction value.

12.1.337.4 adjust_reason

`uint8_t sf_ble_prf_cts_curr_time_t::adjust_reason`

Brief description

Adjust reason.

12.1.337.5 reserved

`uint8_t sf_ble_prf_cts_curr_time_t::reserved`

Brief description

Reserved.

12.1.338 sf_ble_prf_cts_date_time_t

```
typedef struct{
    uint16_t  year
    uint8_t  month
    uint8_t  day
    uint8_t  hour
    uint8_t  min
    uint8_t  sec
    uint8_t  reserved
} sf_ble_prf_cts_date_time_t
```

12.1.338.1 year

`uint16_t sf_ble_prf_cts_date_time_t::year`

Brief description

Year value.

12.1.338.2 month

`uint8_t sf_ble_prf_cts_date_time_t::month`

Brief description

Month value.

12.1.338.3 day

`uint8_t sf_ble_prf_cts_date_time_t::day`

Brief description

Day value.

12.1.338.4 hour

`uint8_t sf_ble_prf_cts_date_time_t::hour`

Brief description

Hour value.

12.1.338.5 min

`uint8_t sf_ble_prf_cts_date_time_t::min`

Brief description

Minute value.

12.1.338.6 sec

`uint8_t sf_ble_prf_cts_date_time_t::sec`

Brief description

Second value.

12.1.338.7 reserved

`uint8_t sf_ble_prf_cts_date_time_t::reserved`

Brief description

Reserved.

12.1.339 sf_ble_prf_dis_pnpID_t

```
typedef struct{
    uint8_t  vendorIdSource
    uint16_t  vendorId
    uint16_t  productId
    uint16_t  productVersion
} sf_ble_prf_dis_pnpID_t
```

12.1.339.1 vendorIdSource

`uint8_t sf_ble_prf_dis_pnpID_t::vendorIdSource`

Brief description

Vendor ID source.

12.1.339.2 vendorId

`uint16_t sf_ble_prf_dis_pnpID_t::vendorId`

Brief description

Vendor ID.

12.1.339.3 productId

`uint16_t sf_ble_prf_dis_pnpID_t::productId`

Brief description

Product ID.

12.1.339.4 productVersion

`uint16_t sf_ble_prf_dis_pnpID_t::productVersion`

Brief description

Version of Product.

12.1.340 sf_ble_prf_hid_change_event_t

```
typedef struct{
    uint16_t  conhdl
    uint8_t  inst_idx
    sf_ble_prf_hid_protocol_mode_t  protocol_mode_val
    sf_ble_prf_hid_ctrl_point_val_t  control_point_val
    union{}                 ble_prf_value
} sf_ble_prf_hid_change_event_t
```

12.1.340.1 conhdl

`uint16_t sf_ble_prf_hid_change_event_t::conhdl`

Brief description

Connection handle.

12.1.340.2 inst_idx

`uint8_t sf_ble_prf_hid_change_event_t::inst_idx`

Brief description

Instance Index.

12.1.340.3 protocol_mode_val

`sf_ble_prf_hid_protocol_mode_t::protocol_mode_val`

Brief description

Protocol Mode.

12.1.340.4 control_point_val

`sf_ble_prf_hid_ctrl_point_val_t::control_point_val`

Brief description

HID Control Point.

12.1.340.5 ble_prf_value

`See source code for the definition of this member.`

Brief description

Value which is changed by client.

12.1.341 sf_ble_prf_hid_report_desc_t

```
typedef struct{
    sf_ble_hidd_device_type_t  device_type
    uint8_t  report_type
    uint8_t  value[SF_BLE_PRF_HIDS_REPORT_MAX]
    uint16_t  value_size
} sf_ble_prf_hid_report_desc_t
```

12.1.341.1 device_type

`sf_ble_hidd_device_type_t::device_type`

Brief description

Device type.

12.1.341.2 report_type

`uint8_t sf_ble_prf_hid_report_desc_t::report_type`

Brief description

Report type.

12.1.341.3 value

`uint8_t sf_ble_prf_hid_report_desc_t::value[SF_BLE_PRF_HIDS_REPORT_MAX]`

Brief description

Report values.

12.1.341.4 value_size

`uint16_t sf_ble_prf_hid_report_desc_t::value_size`

Brief description

Report size.

12.1.342 sf_ble_prf_hid_report_ind_t

```
typedef struct{
    uint16_t  conhdl
    uint8_t  inst_idx
    uint8_t  reserved
    sf_ble_prf_hid_report_desc_t  report
} sf_ble_prf_hid_report_ind_t
```

12.1.342.1 conhdl

`uint16_t sf_ble_prf_hid_report_ind_t::conhdl`

Brief description

Connection handle.

12.1.342.2 inst_idx

`uint8_t sf_ble_prf_hid_report_ind_t::inst_idx`

Brief description

Instance Index.

12.1.342.3 reserved

`uint8_t sf_ble_prf_hid_report_ind_t::reserved`

Brief description

Reserved.

12.1.342.4 report

`sf_ble_prf_hid_report_desc_t::report`

Brief description

Report received from either BHOST or RHOST.

12.1.343 sf_ble_prf_htp_temp_info_ind_t

```
typedef struct{
    uint16_t  conhdl
    sf_ble_prf_htp_temp_info_t  temp_info
} sf_ble_prf_htp_temp_info_ind_t
```

12.1.343.1 conhdl

`uint16_t sf_ble_prf_htp_temp_info_ind_t::conhdl`

Brief description

Connection handle.

12.1.343.2 temp_info

`sf_ble_prf_htp_temp_info_t::temp_info`

Brief description

Thermometer info.

12.1.344 sf_ble_prf_htp_temp_info_t

```
typedef struct{
    uint8_t  flag_stable_meas
    uint8_t  flags
    int32_t  temp_val
    sf_ble_prf_cts_date_time_t  stamp
    sf_ble_prf_htp_temp_type_t  type
    uint8_t  reserved
} sf_ble_prf_htp_temp_info_t
```

12.1.344.1 flag_stable_meas

`uint8_t sf_ble_prf_htp_temp_info_t::flag_stable_meas`

Brief description

Stable or intermediary type of temperature.

12.1.344.2 flags

`uint8_t sf_ble_prf_htp_temp_info_t::flags`

Brief description

flags

12.1.344.3 temp_val

`int32_t sf_ble_prf_htp_temp_info_t::temp_val`

Brief description

temp value

12.1.344.4 stamp

`sf_ble_prf_cts_date_time_t::stamp`

Brief description

time stamp

12.1.344.5 type

`sf_ble_prf_htp_temp_type_t::type`

Brief description

type

12.1.344.6 reserved

uint8_t sf_ble_prf_htp_temp_info_t::reserved

Brief description

Reserved.

12.1.345 sf_ble_prf_ias_alert_lvl_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_ias_alert_type_t  alert_lvl
} sf_ble_prf_ias_alert_lvl_change_t
```

12.1.345.1 conhdl

sf_ble_conn_handle_t::conhdl

Brief description

Connection handle.

12.1.345.2 alert_lvl

sf_ble_prf_ias_alert_type_t::alert_lvl

Brief description

Control point value.

12.1.346 sf_ble_prf_ndcs_time_dst_t

```
typedef struct{
    sf_ble_prf_cts_date_time_t  stamp
    uint8_t  dst_offset
    uint8_t  reserved
} sf_ble_prf_ndcs_time_dst_t
```

12.1.346.1 stamp

sf_ble_prf_cts_date_time_t::stamp

Brief description

Current time stamp.

12.1.346.2 dst_offset

uint8_t sf_ble_prf_ndcs_time_dst_t::dst_offset

Brief description

DST Offset.

12.1.346.3 reserved

`uint8_t sf_ble_prf_ndcs_time_dst_t::reserved`

Brief description

Reserved.

12.1.347 sf_ble_prf_ringer_cp_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_ringer_cp_t  ringer_cp
} sf_ble_prf_ringer_cp_change_t
```

12.1.347.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.347.2 ringer_cp

`sf_ble_prf_ringer_cp_t::ringer_cp`

Brief description

Ringer control point value.

12.1.348 sf_ble_prf_ringer_setting_ntf_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_ringer_setting_t  ringer_setting
} sf_ble_prf_ringer_setting_ntf_t
```

12.1.348.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.348.2 ringer_setting

`sf_ble_prf_ringer_setting_t::ringer_setting`

Brief description

Ringer control point value.

12.1.349 sf_ble_prf_rtus_time_updt_state_t

```
typedef struct{
    uint8_t  current_state
    uint8_t  update_result
} sf_ble_prf_rtus_time_updt_state_t
```

12.1.349.1 current_state

`uint8_t sf_ble_prf_rtus_time_::current_state`

Brief description

Current state of Reference time.

12.1.349.2 update_result

`uint8_t sf_ble_prf_rtus_time_::update_result`

Brief description

Result of update.

12.1.350 sf_ble_prf_scps_scan_intv_t

```
typedef struct{
    uint16_t  le_scan_interval
    uint16_t  le_scan_window
} sf_ble_prf_scps_scan_intv_t
```

12.1.350.1 le_scan_interval

`uint16_t sf_ble_prf_scps_scan_intv_t::le_scan_interval`

Brief description

scan interval value

12.1.350.2 le_scan_window

`uint16_t sf_ble_prf_scps_scan_intv_t::le_scan_window`

Brief description

scan window value

12.1.351 sf_ble_prf_tip_write_data_t

```
typedef struct{
    sf_ble_prf_cts_curr_time_t  current_time
    sf_ble_cts_local_time_t  local_time
    sf_ble_cts_ref_time_t  ref_time
    sf_ble_prf_ndcs_time_dst_t  next_dst
    sf_ble_prf_rtus_time_updt_state_t  update_state
} sf_ble_prf_tip_write_data_t
```

12.1.351.1 current_time

sf_ble_prf_cts_curr_time_t::current_time

Brief description

Current Time Information.

12.1.351.2 local_time

sf_ble_cts_local_time_t::local_time

Brief description

Local Time Information.

12.1.351.3 ref_time

sf_ble_cts_ref_time_t::ref_time

Brief description

Reference Time Information.

12.1.351.4 next_dst

sf_ble_prf_ndcs_time_dst_t::next_dst

Brief description

Next DST Information.

12.1.351.5 update_state

sf_ble_prf_rtus_time_updt_state_t::update_state

Brief description

Update Status Information.

12.1.352 sf_ble_provisioning_t

```
typedef struct{
    uint8_t  gap_name[SF_BLE_MAX_GAP_NAME_LEN]
    uint8_t  bcast_mode
    sf_ble_bonding_mode_t  bonding_mode
    sf_ble_sec_info_t  sec_info
    sf_ble_gap_role_t  gap_role
    sf_ble_callback_t  p_ble_callback
} sf_ble_provisioning_t
```

12.1.352.1 gap_name

`uint8_t sf_ble_provisioning_t::gap_name[SF_BLE_MAX_GAP_NAME_LEN]`

Brief description

GAP name.

12.1.352.2 bcast_mode

`uint8_t sf_ble_provisioning_t::bcast_mode`

Brief description

broadcast mode

12.1.352.3 bonding_mode

`sf_ble_bonding_mode_t::bonding_mode`

Brief description

bonding mode

12.1.352.4 sec_info

`sf_ble_sec_info_t::sec_info`

Brief description

Security information.

12.1.352.5 gap_role

`sf_ble_gap_role_t::gap_role`

Brief description

GAP role (master/slave)

12.1.352.6 p_ble_callback

`sf_ble_callback_t::p_ble_callback`

Brief description

GAP user event callback.

12.1.353 sf_ble_scan_info_t

```
typedef struct{
    uint16_t  total_scan_duration
    sf_ble_scan_mode_t  scan_mode
    sf_ble_disc_type_t  discovery_type
    sf_ble_addr_type_t  address_type
    sf_ble_adv_filt_type_t  filt_policy
} sf_ble_scan_info_t
```

12.1.353.1 total_scan_duration

`uint16_t sf_ble_scan_info_t::total_scan_duration`

Brief description

BLE total scan duration in milliseconds.

12.1.353.2 scan_mode

`sf_ble_scan_mode_t::scan_mode`

Brief description

BLE scan mode.

12.1.353.3 discovery_type

`sf_ble_disc_type_t::discovery_type`

Brief description

Used only in active scan for specifying discovery type.

12.1.353.4 address_type

`sf_ble_addr_type_t::address_type`

Brief description

BLE address type.

12.1.353.5 filt_policy

`sf_ble_adv_filt_type_t::filt_policy`

Brief description

Scan Filter policy.

12.1.354 sf_ble_scan_t

```
typedef struct{
    sf_ble_addr_type_t  addr_type
    uint8_t  bd_addr[SF_BLE_ADDR_LEN]
    uint16_t  data_len
    uint8_t  data[SF_BLE_MAX_BLE_ADV_DATA_LEN]
    int16_t  rssi
} sf_ble_scan_t
```

12.1.354.1 addr_type

`sf_ble_addr_type_t::addr_type`

Brief description

Address type of remote BLE device.

12.1.354.2 bd_addr

`uint8_t sf_ble_scan_t::bd_addr[SF_BLE_ADDR_LEN]`

Brief description

Remote BLE address.

12.1.354.3 data_len

`uint16_t sf_ble_scan_t::data_len`

Brief description

Scan data length.

12.1.354.4 data

`uint8_t sf_ble_scan_t::data[SF_BLE_MAX_BLE_ADV_DATA_LEN]`

Brief description

Scan data.

12.1.354.5 rssi

`int16_t sf_ble_scan_t::rssi`

Brief description

RSSI value.

12.1.355 sf_ble_scps_scan_intv_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_scps_scan_intv_t  scan_interval_window_val
} sf_ble_scps_scan_intv_change_t
```

12.1.355.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.355.2 scan_interval_window_val

`sf_ble_prf_scps_scan_intv_t::scan_interval_window_val`

Brief description

Scan Interval window.

12.1.356 sf_ble_sec_enc_start_ind_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    uint8_t  status
    uint8_t  key_size
    sf_ble_auth_type_t  auth_type
    uint8_t  bonding_status
} sf_ble_sec_enc_start_ind_t
```

12.1.356.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle of the remote device.

12.1.356.2 status

`uint8_t sf_ble_sec_enc_start_ind_t::status`

Brief description

Result of encryption start, 0 = SUCCESS else error.

12.1.356.3 key_size

`uint8_t sf_ble_sec_enc_start_ind_t::key_size`

Brief description

Key Size.

12.1.356.4 auth_type

`sf_ble_auth_type_t::auth_type`

Brief description

Security properties.

12.1.356.5 bonding_status

`uint8_t sf_ble_sec_enc_start_ind_t::bonding_status`

Brief description

Bonding Status, 0 = Unbonded, 1 = Bonded.

12.1.357 sf_ble_sec_info_t

```
typedef struct{
    sf_ble_sec_mode_t  sec_mode
    uint8_t  id_key[SF_BLE_SEC_KEY_LEN]
    uint8_t  csrk_key[SF_BLE_SEC_KEY_LEN]
    uint8_t  ltk_key[SF_BLE_SEC_KEY_LEN]
    uint8_t  rand_num[SF_BLE_RAND_NUM_LENGTH]
    uint16_t  ediv
} sf_ble_sec_info_t
```

12.1.357.1 sec_mode

`sf_ble_sec_mode_t::sec_mode`

Brief description

security mode

12.1.357.2 id_key

`uint8_t sf_ble_sec_info_t::id_key[SF_BLE_SEC_KEY_LEN]`

Brief description

GAP Identity Resolving Security key.

12.1.357.3 csrk_key

`uint8_t sf_ble_sec_info_t::csrk_key[SF_BLE_SEC_KEY_LEN]`

Brief description

GAP Connection Signature Resolving Security key.

12.1.357.4 ltk_key

`uint8_t sf_ble_sec_info_t::ltk_key[SF_BLE_SEC_KEY_LEN]`

Brief description

GAP Connection Long term Security key.

12.1.357.5 rand_num

`uint8_t sf_ble_sec_info_t::rand_num[SF_BLE_RAND_NUM_LENGTH]`

Brief description

GAP Connection Random number for Long term Security key.

12.1.357.6 ediv

`uint16_t sf_ble_sec_info_t::ediv`

Brief description

GAP Connection Encrypted Diversifier for Long term Security key.

12.1.358 sf_ble_service_discovery_req_t

```
typedef struct{
    sf_ble_uuid_t  uuid
    uint16_t  start_handle
    uint16_t  end_handle
    sf_ble_service_discovery_t  discovery_type
} sf_ble_service_discovery_req_t
```

12.1.358.1 uuid

`sf_ble_uuid_t::uuid`

Brief description

Service UUID.

12.1.358.2 start_handle

`uint16_t sf_ble_service_discovery_req_t::start_handle`

Brief description

Discovery start handle.

12.1.358.3 end_handle

`uint16_t sf_ble_service_discovery_req_t::end_handle`

Brief description

Discovery end handle.

12.1.358.4 discovery_type

`sf_ble_service_discovery_t::discovery_type`

Brief description

Service discovery type.

12.1.359 sf_ble_service_discovery_rsp_t

```
typedef struct{
    uint16_t  service_handle
    sf_ble_uuid_t  uuid
    uint16_t  start_handle
    uint16_t  end_handle
} sf_ble_service_discovery_rsp_t
```

12.1.359.1 service_handle

`uint16_t sf_ble_service_discovery_rsp_t::service_handle`

Brief description

Service handle.

12.1.359.2 uuid

`sf_ble_uuid_t::uuid`

Brief description

Service UUID.

12.1.359.3 start_handle

`uint16_t sf_ble_service_discovery_rsp_t::start_handle`

Brief description

Start handle of sub-attributes handle range.

12.1.359.4 end_handle

`uint16_t sf_ble_service_discovery_rsp_t::end_handle`

Brief description

End handle of sub-attributes handle range.

12.1.360 sf_ble_sm_enc_info_t

```
typedef struct{
    sf_ble_conn_handle_t  conn_idx
    uint16_t  ediv
    uint8_t  rand_num[SF_BLE_RAND_NUM_LENGTH]
    uint8_t  ltk_key[SF_BLE_SEC_KEY_LEN]
} sf_ble_sm_enc_info_t
```

12.1.360.1 conn_idx

`sf_ble_conn_handle_t::conn_idx`

Brief description

connection index

12.1.360.2 ediv

`uint16_t sf_ble_sm_enc_info_t::ediv`

Brief description

BLE Security EDIV.

12.1.360.3 rand_num

`uint8_t sf_ble_sm_enc_info_t::rand_num[SF_BLE_RAND_NUM_LENGTH]`

Brief description

Random number for security.

12.1.360.4 ltk_key

`uint8_t sf_ble_sm_enc_info_t::ltk_key[SF_BLE_SEC_KEY_LEN]`

Brief description

BLE long term security key.

12.1.361 sf_ble_sm_key_ind_t

```
typedef struct{
    uint8_t  conn_idx
```

```
    sf_ble_key_dist_t  key_code
    uint16_t  ediv
    uint8_t  rand_num[SF_BLE_RAND_NUM_LENGTH]
    uint8_t  ltk_key[SF_BLE_SEC_KEY_LEN]
} sf_ble_sm_key_ind_t
```

12.1.361.1 conn_idx

`uint8_t sf_ble_sm_key_ind_t::conn_idx`

Brief description

connection index

12.1.361.2 key_code

`sf_ble_key_dist_t::key_code`

Brief description

type of security key

12.1.361.3 ediv

`uint16_t sf_ble_sm_key_ind_t::ediv`

Brief description

BLE Security EDIV.

12.1.361.4 rand_num

`uint8_t sf_ble_sm_key_ind_t::rand_num[SF_BLE_RAND_NUM_LENGTH]`

Brief description

Random number for security.

12.1.361.5 ltk_key

`uint8_t sf_ble_sm_key_ind_t::ltk_key[SF_BLE_SEC_KEY_LEN]`

Brief description

BLE long term security key.

12.1.362 sf_ble_sm_tk_ind_t

```
typedef struct{
    sf_ble_sm_tk_info_t  tk_info
    uint8_t  tk_req_status
    uint8_t  tk_key[SF_BLE_SEC_KEY_LEN]
} sf_ble_sm_tk_ind_t
```

12.1.362.1 tk_info

`sf_ble_sm_tk_info_t::tk_info`

Brief description

input parameter,information to app about temporary key request

12.1.362.2 tk_req_status

`uint8_t sf_ble_sm_tk_ind_t::tk_req_status`

Brief description

output parameter: request status, application has to set this in callback event whether request is valid or not

12.1.362.3 tk_key

`uint8_t sf_ble_sm_tk_ind_t::tk_key[SF_BLE_SEC_KEY_LEN]`

Brief description

application has to set this in callback event temporary key for encryption

12.1.363 sf_ble_sm_tk_info_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    uint8_t  disp_en
} sf_ble_sm_tk_info_t
```

12.1.363.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.363.2 disp_en

`uint8_t sf_ble_sm_tk_info_t::disp_en`

Brief description

Whether to enable display.

12.1.364 sf_ble_svc_attribute_t

```
typedef struct{
    uint16_t  attr_handle
    uint16_t  attr_type
    uint16_t  parent_svc_handle
```

```
    uint8_t *  p_attr_value
    sf_ble_uuid_length_t  attr_value_len
} sf_ble_svc_attribute_t
```

12.1.364.1 attr_handle

`uint16_t sf_ble_svc_attribute_t::attr_handle`

Brief description

Service Handle.

12.1.364.2 attr_type

`uint16_t sf_ble_svc_attribute_t::attr_type`

Brief description

Attribute type such as SF_BLE_GATT_PRI_SERVICE, SF_BLE_GATT_INCLUDE_SERVICE.

12.1.364.3 parent_svc_handle

`uint16_t sf_ble_svc_attribute_t::parent_svc_handle`

Brief description

parent_svc_handle is service handle of parent service which is already registered

Detailed description

If service is included service,

12.1.364.4 p_attr_value

`uint8_t* sf_ble_svc_attribute_t::p_attr_value`

Brief description

Service UUID Pointer.

12.1.364.5 attr_value_len

`sf_ble_uuid_length_t::attr_value_len`

Brief description

UUID Length.

12.1.365 sf_ble_tip_cp_change_t

```
typedef struct{
    sf_ble_conn_handle_t  conhdl
    sf_ble_prf_tip_time_ctrl_point_t  control_point_value
} sf_ble_tip_cp_change_t
```

12.1.365.1 conhdl

`sf_ble_conn_handle_t::conhdl`

Brief description

Connection handle.

12.1.365.2 control_point_value

`sf_ble_prf_tip_time_ctrl_point_t::control_point_value`

Brief description

Time Update Control point value.

12.1.366 sf_ble_uuid_t

```
typedef struct{
    uint16_t  uuid16
    uint32_t  uuid32
    uint8_t  uuid128[SF_BLE_128BITS_UUID_LENGTH]
    union{}                 union{}
    sf_ble_uuid_length_t  uuid_length
} sf_ble_uuid_t
```

12.1.366.1 uuid16

`uint16_t sf_ble_uuid_t::uuid16`

Brief description

16 bit UUID

12.1.366.2 uuid32

`uint32_t sf_ble_uuid_t::uuid32`

Brief description

32 bit UUID

12.1.366.3 uuid128

`uint8_t sf_ble_uuid_t::uuid128[SF_BLE_128BITS_UUID_LENGTH]`

Brief description

128 bit UUID

12.1.366.4 union{}

`See source code for the definition of this member.`

12.1.366.5 uuid_length

`sf_ble_uuid_length_t::uuid_length`

Brief description

Service UUID length.

12.1.367 sf_ble_write_cmd_event_data_t

```
typedef struct{
    uint16_t  handle
    uint16_t  offset
    sf_ble_write_response_t  response_required
    uint8_t *  p_data
    uint16_t  data_length
} sf_ble_write_cmd_event_data_t
```

12.1.367.1 handle

`uint16_t sf_ble_write_cmd_event_data_t::handle`

Brief description

Characteristic value or descriptor handle.

12.1.367.2 offset

`uint16_t sf_ble_write_cmd_event_data_t::offset`

Brief description

Offset for long Characteristic value.

12.1.367.3 response_required

`sf_ble_write_response_t::response_required`

Brief description

Write response required or not.

12.1.367.4 p_data

`uint8_t* sf_ble_write_cmd_event_data_t::p_data`

Brief description

Pointer to data.

12.1.367.5 data_length

`uint16_t sf_ble_write_cmd_event_data_t::data_length`

Brief description

Data length.

12.1.368 sf_block_media_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_block_media_ctrl_t *p_ctrl, sf_block_media_cfg_t const
*const p_cfg)
    ssp_err_t(*  read)(sf_block_media_ctrl_t *p_ctrl, uint8_t *const p_dest,
uint32_t const start_sector, uint32_t const sector_count)
    ssp_err_t(*  write)(sf_block_media_ctrl_t *p_ctrl, uint8_t const *const
p_src, uint32_t const start_sector, uint32_t const sector_count)
    ssp_err_t(*  ioctl)(sf_block_media_ctrl_t *p_ctrl, ssp_command_t const
command, void *p_data)
    ssp_err_t(*  close)(sf_block_media_ctrl_t *p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_block_media_api_t
```

12.1.369 sf_block_media_cfg_t

```
typedef struct{
    uint32_t  block_size
    void const  *  p_extend
} sf_block_media_cfg_t
```

12.1.369.1 block_size

`uint32_t sf_block_media_cfg_t::block_size`

Brief description

Block size in bytes.

12.1.369.2 p_extend

`void const* sf_block_media_cfg_t::p_extend`

Brief description

Instance dependent configuration.

12.1.370 sf_block_media_instance_t

```
typedef struct{
    sf_block_media_ctrl_t *  p_ctrl
    sf_block_media_cfg_t const  *  p_cfg
    sf_block_media_api_t const  *  p_api
} sf_block_media_instance_t
```

12.1.370.1 p_ctrl

`sf_block_media_ctrl_t::p_ctrl`

Brief description

Block media pointer to device driver control structure.

12.1.370.2 p_cfg

`sf_block_media_cfg_t::p_cfg`

Brief description

Block media pointer to device driver configuration structure.

12.1.370.3 p_api

`sf_block_media_api_t::p_api`

Brief description

Block media pointer to device driver api structure.

12.1.371 sf_block_media_on_sdmmc_cfg_t

```
typedef struct{
    sdmmc_instance_t const  *const  p_lower_lvl_sdmmc
} sf_block_media_on_sdmmc_cfg_t
```

12.1.371.1 p_lower_lvl_sdmmc

`sdmmc_instance_t::p_lower_lvl_sdmmc`

Brief description

Pointer to SDMMC instance structure.

12.1.372 sf_block_media_sdmmc_instance_ctrl_t

```
typedef struct{
    uint32_t  block_size
    sdmmc_instance_t *  p_lower_lvl_sdmmc
    TX_EVENT_FLAGS_GROUP  eventflag
    uint32_t  open
} sf_block_media_sdmmc_instance_ctrl_t
```

12.1.372.1 block_size

`uint32_t ::block_size`

Brief description

Block size in bytes.

12.1.372.2 p_lower_lvl_sdmmc

`sdmmc_instance_t::p_lower_lvl_sdmmc`

Brief description

Pointer to SDMMC instance structure.

12.1.372.3 eventflag

`TX_EVENT_FLAGS_GROUP::eventflag`

Brief description

Pointer to the event flag object for SDMMC data transfer.

12.1.372.4 open

`uint32_t ::open`

Brief description

Used to determine if framework is initialized.

12.1.373 sf_cellular_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_cellular_ctrl_t *p_ctrl, sf_cellular_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(sf_cellular_ctrl_t *const p_ctrl)
    ssp_err_t(*  provisioningGet)(sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_provisioning_t *const p_cellular_provisioning)
    ssp_err_t(*  provisioningSet)(sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_provisioning_t const *const p_cellular_provisioning)
    ssp_err_t(*  infoGet)(sf_cellular_ctrl_t *const p_ctrl, sf_cellular_info_t
*const p_cellular_info)
    ssp_err_t(*  statisticsGet)(sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_stats_t *const p_cellular_device_stats)
    ssp_err_t(*  transmit)(sf_cellular_ctrl_t *const p_ctrl, uint8_t *const
p_buf, uint32_t length)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
    ssp_err_t(*  networkConnect)(sf_cellular_ctrl_t *const p_ctrl)
    ssp_err_t(*  networkDisconnect)(sf_cellular_ctrl_t *const p_ctrl)
    ssp_err_t(*  networkStatusGet)(sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_network_status_t *p_network_status)
    ssp_err_t(*  simPinSet)(sf_cellular_ctrl_t *const p_ctrl, uint8_t *const
p_old_pin, uint8_t *const p_new_pin)
    ssp_err_t(*  simLock)(sf_cellular_ctrl_t *const p_ctrl, uint8_t *const p_pin)
    ssp_err_t(*  simUnlock)(sf_cellular_ctrl_t *const p_ctrl, uint8_t *const
```

```
p_pin)
    ssp_err_t(*  simIDGet)(sf_cellular_ctrl_t *const p_ctrl, uint8_t *p_sim_id)
    ssp_err_t(*  fotaCheck)(sf_cellular_ctrl_t *const p_ctrl, void *p_fotacheck)
    ssp_err_t(*  fotaStart)(sf_cellular_ctrl_t *const p_ctrl, void *p_fotastart)
    ssp_err_t(*  fotaStop)(sf_cellular_ctrl_t *const p_ctrl, void *p_fotastop)
    ssp_err_t(*  reset)(sf_cellular_ctrl_t *const p_ctrl,
sf_cellular_reset_type_t reset_type)
} sf_cellular_api_t
```

12.1.374 sf_cellular_at_cmd_set_t

```
typedef struct{
    uint8_t *  p_cmd
    uint8_t *  p_success_resp
    uint16_t  max_resp_length
    uint8_t  retry
    uint16_t  retry_delay
} sf_cellular_at_cmd_set_t
```

12.1.374.1 p_cmd

`uint8_t* sf_cellular_at_cmd_set_t::p_cmd`

Brief description

AT Command.

12.1.374.2 p_success_resp

`uint8_t* sf_cellular_at_cmd_set_t::p_success_resp`

Brief description

Success response string.

12.1.374.3 max_resp_length

`uint16_t sf_cellular_at_cmd_set_t::max_resp_length`

Brief description

Maximum length of expected response.

12.1.374.4 retry

`uint8_t sf_cellular_at_cmd_set_t::retry`

Brief description

Retry count.

12.1.374.5 retry_delay

`uint16_t sf_cellular_at_cmd_set_t::retry_delay`

Brief description

Delay between AT command retry.

12.1.375 sf_cellular_callback_args_t

```
typedef struct{
    sf_cellular_event_t  event
    uint8_t *  p_data
    uint32_t  length
    void const  *  p_context
} sf_cellular_callback_args_t
```

12.1.375.1 event

`sf_cellular_event_t::event`

Brief description

Event Code.

12.1.375.2 p_data

`uint8_t* sf_cellular_callback_args_t::p_data`

Brief description

Pointer to received data.

12.1.375.3 length

`uint32_t sf_cellular_callback_args_t::length`

Brief description

Receive Data Length.

12.1.375.4 p_context

`void const* sf_cellular_callback_args_t::p_context`

Brief description

Context Provided to user during callback.

12.1.376 sf_cellular_cfg_t

```
typedef struct{
    sf_cellular_op_select_mode_t  op_select_mode
```

```
    sf_cellular_op_t  op
    uint16_t  num_pref_ops
    sf_cellular_op_t  pref_ops[SF_CELLULAR_MAX_PREFFERED_OPERATOR_COUNT]
    sf_cellular_timezone_update_mode_t  tz_upd_mode
    uint8_t *  p_sim_pin
    uint8_t *  p_puk_pin
    ssp_err_t(*  p_prov_callback)(sf_cellular_callback_args_t *p_args)
    void(*  p_recv_callback)(sf_cellular_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
    sf_cellular_at_cmd_set_t const  *  p_cmd_set
} sf_cellular_cfg_t
```

12.1.376.1 op_select_mode

`sf_cellular_op_select_mode_t::op_select_mode`

Brief description

Cellular Operator selection mode.

12.1.376.2 op

`sf_cellular_op_t::op`

Brief description

Cellular operator. Valid when operator selection mode is manual.

12.1.376.3 num_pref_ops

`uint16_t sf_cellular_cfg_t::num_pref_ops`

Brief description

Number of preferred operators in the pref_ops array.

12.1.376.4 pref_ops

`sf_cellular_op_t::pref_ops`

Brief description

Array of structures describing preferred operators.

12.1.376.5 tz_upd_mode

`sf_cellular_timezone_update_mode_t::tz_upd_mode`

Brief description

TimeZone update mode policy.

12.1.376.6 p_sim_pin

```
uint8_t* sf_cellular_cfg_t::p_sim_pin
```

Brief description

SIM Pin.

12.1.376.7 p_puk_pin

```
uint8_t* sf_cellular_cfg_t::p_puk_pin
```

Brief description

PUK Pin.

12.1.376.8 p_prov_callback

```
ssp_err_t(* sf_cellular_cfg_t::p_prov_callback) (sf_cellular_callback_args_t
*p_args)
```

Detailed description

Pointer to provisioning callback function, used in NSAL

12.1.376.9 p_recv_callback

```
void(* sf_cellular_cfg_t::p_recv_callback) (sf_cellular_callback_args_t *p_args)
```

Detailed description

This is the receive callback function used by NetX which will take a data packet from the Cellular module and hand it over to NetX for further processing.

12.1.376.10 p_context

```
void const* sf_cellular_cfg_t::p_context
```

Brief description

User defined context passed into callback function.

12.1.376.11 p_extend

```
void const* sf_cellular_cfg_t::p_extend
```

Brief description

Instance specific configuration.

12.1.376.12 p_cmd_set

```
sf_cellular_at_cmd_set_t::p_cmd_set
```

Brief description

Instance specific command set.

12.1.377 sf_cellular_ctrl_t

```
typedef struct{
    void *  p_driver_handle
} sf_cellular_ctrl_t
```

12.1.377.1 p_driver_handle

`void* sf_cellular_ctrl_t::p_driver_handle`

Brief description

Stores information required by underlying Cellular device driver.

12.1.378 sf_cellular_info_t

```
typedef struct{
    uint8_t  mfg_name[SF_CELLULAR_MFG_NAME_LEN]
    uint8_t  chipset[SF_CELLULAR_CHIPSET_LEN]
    uint8_t  fw_version[SF_CELLULAR_FWVERSION_LEN]
    uint8_t  imei[SF_CELLULAR_IMEI_LEN]
    uint16_t  rssi
    uint16_t  ber
    uint8_t  ip_addr[SF_CELLULAR_IP_ADDR_LEN]
} sf_cellular_info_t
```

12.1.378.1 mfg_name

`uint8_t sf_cellular_info_t::mfg_name[SF_CELLULAR_MFG_NAME_LEN]`

Brief description

Manufacturer name.

12.1.378.2 chipset

`uint8_t sf_cellular_info_t::chipset[SF_CELLULAR_CHIPSET_LEN]`

Brief description

Pointer to string showing Cellular chipset/driver information.

12.1.378.3 fw_version

`uint8_t sf_cellular_info_t::fw_version[SF_CELLULAR_FWVERSION_LEN]`

Brief description

Cellular firmware version.

12.1.378.4 imei

`uint8_t sf_cellular_info_t::imei[SF_CELLULAR_IMEI_LEN]`

Brief description

IMEI number.

12.1.378.5 rssi

`uint16_t sf_cellular_info_t::rssi`

Brief description

Received signal strength indication.

12.1.378.6 ber

`uint16_t sf_cellular_info_t::ber`

Brief description

Bit rate error.

12.1.378.7 ip_addr

`uint8_t sf_cellular_info_t::ip_addr[SF_CELLULAR_IP_ADDR_LEN]`

Brief description

IP address.

12.1.379 sf_cellular_instance_t

```
typedef struct{
    sf_cellular_ctrl_t *  p_ctrl
    sf_cellular_cfg_t const  *  p_cfg
    sf_cellular_api_t const  *  p_api
} sf_cellular_instance_t
```

12.1.379.1 p_ctrl

`sf_cellular_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.379.2 p_cfg

`sf_cellular_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.379.3 p_api

`sf_cellular_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.380 sf_cellular_network_status_t

```
typedef struct{
    uint16_t  country_code
    uint16_t  operator_code
    int16_t  rssi
    uint8_t  cid[SF_CELLULAR_CID_LEN]
    uint8_t  imsi[SF_CELLULAR_IMSI_LEN]
    uint8_t  op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]
    uint8_t  service_domain
    uint8_t  active_band
} sf_cellular_network_status_t
```

12.1.380.1 country_code

`uint16_t sf_cellular_network_status_t::country_code`

Brief description

Country code.

12.1.380.2 operator_code

`uint16_t sf_cellular_network_status_t::operator_code`

Brief description

Operator code.

12.1.380.3 rssi

`int16_t sf_cellular_network_status_t::rssi`

Brief description

RSSI.

12.1.380.4 cid

`uint8_t sf_cellular_network_status_t::cid[SF_CELLULAR_CID_LEN]`

Brief description

Cell ID.

12.1.380.5 imsi

`uint8_t sf_cellular_network_status_t::imsi[SF_CELLULAR_IMSI_LEN]`

Brief description

IMSI.

12.1.380.6 op_name

`uint8_t sf_cellular_network_status_t::op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]`

Brief description

Operator name.

12.1.380.7 service_domain

`uint8_t sf_cellular_network_status_t::service_domain`

Brief description

Service Domain.

12.1.380.8 active_band

`uint8_t sf_cellular_network_status_t::active_band`

Brief description

Active Band.

12.1.381 sf_cellular_nsal_cfg_t

```
typedef struct{
    uint8_t *  p_ppp_stack
    uint32_t  ppp_stack_size
    UINT  priority
    NX_PPP *  p_ppp
    NX_IP *  p_ip
    NX_PACKET_POOL *  p_ppp_packet_pool
    void(*  p_ppp_invalid_packet_cb)(NX_PACKET *p_packet_ptr)
    void(*  p_ppp_send_byte)(UCHAR byte)
    void(*  p_link_down_cb)(NX_PPP *p_ppp_ptr)
    void(*  p_link_up_cb)(NX_PPP *p_ppp_ptr)
    sf_cellular_auth_type_t  auth_type
    UINT(*  p_chap_get_challenge_cb)(CHAR *p_rand_value, CHAR *p_id, CHAR
*p_name)
    UINT(*  p_chap_get_responder_cb)(CHAR *p_system, CHAR *p_name, CHAR
*p_secret)
```

```
    UINT(*  p_chap_get_verify_cb)(CHAR *p_system, CHAR *p_name, CHAR *p_secret)
    UINT(*  p_pap_generate_login)(CHAR *p_name, CHAR *p_password)
    UINT(*  p_pap_verify_login)(CHAR *p_name, CHAR *p_password)
    uint32_t  local_ip
    uint32_t  peer_ip
    void const  *  p_extend
} sf_cellular_nsal_cfg_t
```

12.1.381.1 p_ppp_stack

`uint8_t* sf_cellular_nsal_cfg_t::p_ppp_stack`

Brief description

PPP Stack.

12.1.381.2 ppp_stack_size

`uint32_t sf_cellular_nsal_cfg_t::ppp_stack_size`

Brief description

PPP Stack size.

12.1.381.3 priority

`UINT sf_cellular_nsal_cfg_t::priority`

Brief description

PPP Thread Priority.

12.1.381.4 p_ppp

`NX_PPP::p_ppp`

Brief description

NetX PPP Interface.

12.1.381.5 p_ip

`NX_IP::p_ip`

Brief description

NetX IP Interface.

12.1.381.6 p_ppp_packet_pool

`NX_PACKET_POOL::p_ppp_packet_pool`

Brief description

NetX Packet pool for internal.

12.1.381.7 p_ppp_invalid_packet_cb

```
void(* sf_cellular_nsal_cfg_t::p_ppp_invalid_packet_cb) (NX_PACKET *p_packet_ptr)
```

Brief description

Callback handler for Invalid packet.

12.1.381.8 p_ppp_send_byte

```
void(* sf_cellular_nsal_cfg_t::p_ppp_send_byte) (UCHAR byte)
```

Brief description

PPP Send byte callback function.

12.1.381.9 p_link_down_cb

```
void(* sf_cellular_nsal_cfg_t::p_link_down_cb) (NX_PPP *p_ppp_ptr)
```

Brief description

PPP Link down notification callback.

Detailed description

Link Notification callback function

12.1.381.10 p_link_up_cb

```
void(* sf_cellular_nsal_cfg_t::p_link_up_cb) (NX_PPP *p_ppp_ptr)
```

Brief description

PPP Link up notification callback.

12.1.381.11 auth_type

```
sf_cellular_auth_type_t::auth_type
```

Brief description

Authentication Type.

12.1.381.12 p_chap_get_challenge_cb

```
UINT(* sf_cellular_nsal_cfg_t::p_chap_get_challenge_cb) (CHAR *p_rand_value, CHAR
*p_id, CHAR *p_name)
```

Brief description

Get challenge notification callback.

Detailed description

CHAP Callback Function

12.1.381.13 p_chap_get_responder_cb

```
UINT(* sf_cellular_nsal_cfg_t::p_chap_get_responder_cb) (CHAR *p_system, CHAR
*p_name, CHAR *p_secret)
```

Brief description

Get Responder notification callback.

12.1.381.14 p_chap_get_verify_cb

```
UINT(* sf_cellular_nsal_cfg_t::p_chap_get_verify_cb) (CHAR *p_system, CHAR
*p_name, CHAR *p_secret)
```

Brief description

Get Chap verification callback.

12.1.381.15 p_pap_generate_login

```
UINT(* sf_cellular_nsal_cfg_t::p_pap_generate_login) (CHAR *p_name, CHAR
*p_password)
```

Brief description

PAP Authentication generate login callback.

Detailed description

PAP Callback Function

12.1.381.16 p_pap_verify_login

```
UINT(* sf_cellular_nsal_cfg_t::p_pap_verify_login) (CHAR *p_name, CHAR
*p_password)
```

Brief description

PAP authentication verification callback.

12.1.381.17 local_ip

```
uint32_t sf_cellular_nsal_cfg_t::local_ip
```

Brief description

Local IP Address.

12.1.381.18 peer_ip

```
uint32_t sf_cellular_nsal_cfg_t::peer_ip
```

Brief description

Peer IP Address.

12.1.381.19 p_extend

```
void const* sf_cellular_nsal_cfg_t::p_extend
```

Brief description

Instance specific configuration.

12.1.382 sf_cellular_op_t

```
typedef struct{
    sf_cellular_op_name_format_t  op_name_format
    uint8_t  op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]
} sf_cellular_op_t
```

12.1.382.1 op_name_format

```
sf_cellular_op_name_format_t::op_name_format
```

Brief description

Cellular operator name format.

12.1.382.2 op_name

```
uint8_t sf_cellular_op_t::op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]
```

Brief description

Cellular operator name.

12.1.383 sf_cellular_provisioning_t

```
typedef struct{
    uint8_t  apn[SF_CELLULAR_MAX_STRING_LEN]
    sf_cellular_auth_type_t  auth_type
    uint8_t  username[SF_CELLULAR_MAX_STRING_LEN]
    uint8_t  password[SF_CELLULAR_MAX_STRING_LEN]
    sf_cellular_airplane_mode_t  airplane_mode
    uint8_t  context_id
    sf_cellular_pdp_type_t  pdp_type
} sf_cellular_provisioning_t
```

12.1.383.1 apn

```
uint8_t sf_cellular_provisioning_t::apn[SF_CELLULAR_MAX_STRING_LEN]
```

Brief description

Access Point Name.

12.1.383.2 auth_type

`sf_cellular_auth_type_t::auth_type`

Brief description

Authentication type: PAP/CHAP.

12.1.383.3 username

`uint8_t sf_cellular_provisioning_t::username[SF_CELLULAR_MAX_STRING_LEN]`

Brief description

User name used for authentication.

12.1.383.4 password

`uint8_t sf_cellular_provisioning_t::password[SF_CELLULAR_MAX_STRING_LEN]`

Brief description

Password used for authentication.

12.1.383.5 airplane_mode

`sf_cellular_airplane_mode_t::airplane_mode`

Brief description

Airplane mode.

12.1.383.6 context_id

`uint8_t sf_cellular_provisioning_t::context_id`

Brief description

Context ID to be used for connection.

12.1.383.7 pdp_type

`sf_cellular_pdp_type_t::pdp_type`

Brief description

PDP Type for Context.

12.1.384 sf_cellular_socket_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_cellular_socket_ctrl_t *p_ctrl,
sf_cellular_socket_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(sf_cellular_socket_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_cellular_socket_api_t
```

12.1.385 sf_cellular_socket_cfg_t

```
typedef struct{
    sf_cellular_instance_t *  p_lower_lvl_cellular
    void *  p_extend
} sf_cellular_socket_cfg_t
```

12.1.385.1 p_lower_lvl_cellular

`sf_cellular_instance_t::p_lower_lvl_cellular`

Brief description

Pointer to SF on-chip stack instance.

12.1.385.2 p_extend

`void* sf_cellular_socket_cfg_t::p_extend`

Brief description

Extended configuration.

12.1.386 sf_cellular_socket_ctrl_t

```
typedef struct{
    sf_cellular_instance_t *  p_lower_lvl_cellular
} sf_cellular_socket_ctrl_t
```

12.1.386.1 p_lower_lvl_cellular

`sf_cellular_instance_t::p_lower_lvl_cellular`

Brief description

low level cellular interface

12.1.387 sf_cellular_socket_instance_t

```
typedef struct{
    sf_cellular_socket_ctrl_t *  p_ctrl
    sf_cellular_socket_cfg_t const  *  p_cfg
    sf_cellular_socket_api_t const  *  p_api
} sf_cellular_socket_instance_t
```

12.1.387.1 p_ctrl

`sf_cellular_socket_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.387.2 p_cfg

`sf_cellular_socket_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.387.3 p_api

`sf_cellular_socket_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.388 sf_cellular_stats_t

```
typedef struct{
    uint32_t  rx_bytes
    uint32_t  tx_bytes
    uint32_t  rx_err
    uint32_t  tx_err
} sf_cellular_stats_t
```

12.1.388.1 rx_bytes

`uint32_t sf_cellular_stats_t::rx_bytes`

Brief description

Bytes received successfully.

12.1.388.2 tx_bytes

`uint32_t sf_cellular_stats_t::tx_bytes`

Brief description

Bytes transmitted successfully.

12.1.388.3 rx_err

`uint32_t sf_cellular_stats_t::rx_err`

Brief description

Bytes receive errors.

12.1.388.4 tx_err

```
uint32_t sf_cellular_stats_t::tx_err
```

Brief description

Bytes transmit errors.

12.1.389 sf_comms_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_comms_ctrl_t *const p_ctrl, sf_comms_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(sf_comms_ctrl_t *const p_ctrl)
    ssp_err_t(*  read)(sf_comms_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const bytes, UINT const timeout)
    ssp_err_t(*  write)(sf_comms_ctrl_t *const p_ctrl, uint8_t const *const
p_src, uint32_t const bytes, UINT const timeout)
    ssp_err_t(*  lock)(sf_comms_ctrl_t *const p_ctrl, sf_comms_lock_t lock_type,
UINT timeout)
    ssp_err_t(*  unlock)(sf_comms_ctrl_t *const p_ctrl, sf_comms_lock_t
lock_type)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_comms_api_t
```

12.1.390 sf_comms_cfg_t

```
typedef struct{
    void const  *  p_extend
} sf_comms_cfg_t
```

12.1.390.1 p_extend

```
void const* sf_comms_cfg_t::p_extend
```

Brief description

Pointer to lower level communications control structure.

12.1.391 sf_comms_instance_t

```
typedef struct{
    sf_comms_ctrl_t *  p_ctrl
    sf_comms_cfg_t const  *  p_cfg
    sf_comms_api_t const  *  p_api
} sf_comms_instance_t
```

12.1.391.1 p_ctrl

`sf_comms_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.391.2 p_cfg

`sf_comms_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.391.3 p_api

`sf_comms_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.392 sf_console_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_console_ctrl_t *const p_ctrl, sf_console_cfg_t const
*const p_cfg)
    ssp_err_t(*  close)(sf_console_ctrl_t *const p_ctrl)
    ssp_err_t(*  prompt)(sf_console_ctrl_t *const p_ctrl, sf_console_menu_t const
*const p_menu, UINT const timeout)
    ssp_err_t(*  parse)(sf_console_ctrl_t *const p_ctrl, sf_console_menu_t const
*const p_cmd_list, uint8_t const *const p_input, uint32_t const bytes)
    ssp_err_t(*  read)(sf_console_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const bytes, uint32_t const timeout)
    ssp_err_t(*  write)(sf_console_ctrl_t *const p_ctrl, uint8_t const *const
p_src, uint32_t const timeout)
    ssp_err_t(*  argumentFind)(uint8_t const *const p_arg, uint8_t const *const
p_str, int32_t *const p_index, int32_t *const p_data)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_console_api_t
```

12.1.393 sf_console_callback_args_t

```
typedef struct{
    sf_console_ctrl_t *  p_ctrl
    uint8_t const  *  p_remaining_string
    uint8_t const  *  context
    uint32_t  bytes
} sf_console_callback_args_t
```

12.1.393.1 p_ctrl

`sf_console_ctrl_t::p_ctrl`

Brief description

Pointer to console that received the command that caused this callback.

12.1.393.2 p_remaining_string

`uint8_t const* sf_console_callback_args_t::p_remaining_string`

Brief description

String remaining after parsing command.

12.1.393.3 context

`uint8_t const* sf_console_callback_args_t::context`

Brief description

Pointer to user provided data.

12.1.393.4 bytes

`uint32_t sf_console_callback_args_t::bytes`

Brief description

The number of bytes remaining in the input string.

12.1.394 sf_console_cfg_t

```
typedef struct{
    sf_comms_instance_t const  *  p_comms
    sf_console_menu_t const  *  p_initial_menu
    bool  echo
    bool  autostart
} sf_console_cfg_t
```

12.1.394.1 p_comms

`sf_comms_instance_t::p_comms`

Brief description

Pointer to communications driver instance.

12.1.394.2 p_initial_menu

`sf_console_menu_t::p_initial_menu`

Brief description

First menu to print during Open.

12.1.394.3 echo

`bool sf_console_cfg_t::echo`

Brief description

Whether to echo input commands to transmitter.

12.1.394.4 autostart

`bool sf_console_cfg_t::autostart`

Brief description

If true, prompt will occur with p_initial_menu after initialization.

12.1.395 sf_console_command_t

```
typedef struct{
    uint8_t *  command
    uint8_t *  help
    void(*  callback)(sf_console_callback_args_t *p_args)
    void const  *  context
} sf_console_command_t
```

12.1.395.1 command

`uint8_t* sf_console_command_t::command`

Brief description

Command string.

12.1.395.2 help

`uint8_t* sf_console_command_t::help`

Brief description

Description of command.

12.1.395.3 callback

`void(* sf_console_command_t::callback) (sf_console_callback_args_t *p_args)`

Brief description

Callback to call when command is selected.

12.1.395.4 context

```
void const* sf_console_command_t::context
```

Brief description

User provided context passed into callback.

12.1.396 sf_console_instance_ctrl_t

```
typedef struct{
    sf_console_menu_t const  *  p_current_menu
    sf_comms_instance_t const  *  p_comms
    uint8_t  new_line
    bool  echo
    uint8_t  input[SF_CONSOLE_MAX_INPUT_LENGTH]
} sf_console_instance_ctrl_t
```

12.1.396.1 p_current_menu

```
sf_console_menu_t::p_current_menu
```

Brief description

Current menu is stored here.

12.1.396.2 p_comms

```
sf_comms_instance_t::p_comms
```

Brief description

Pointer to communications driver instance.

12.1.396.3 new_line

```
uint8_t ::new_line
```

Brief description

Whether to echo input commands to transmitter.

12.1.396.4 echo

```
bool ::echo
```

Brief description

Whether to echo input commands to transmitter.

12.1.396.5 input

```
uint8_t ::input[SF_CONSOLE_MAX_INPUT_LENGTH]
```

Brief description

Input buffer used to store user input.

12.1.397 sf_console_instance_t

```
typedef struct{
    sf_console_ctrl_t *  p_ctrl
    sf_console_cfg_t const  *  p_cfg
    sf_console_api_t const  *  p_api
} sf_console_instance_t
```

12.1.397.1 p_ctrl

`sf_console_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.397.2 p_cfg

`sf_console_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.397.3 p_api

`sf_console_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.398 sf_console_menu_t

```
typedef struct{
    struct st_sf_console_menu const  *  menu_prev
    uint8_t const  *  menu_name
    uint32_t  num_commands
    sf_console_command_t const  *  command_list
} sf_console_menu_t
```

12.1.398.1 menu_prev

`struct st_sf_console_menu const* sf_console_menu_t::menu_prev`

Brief description

Previous menu.

12.1.398.2 menu_name

`uint8_t const* sf_console_menu_t::menu_name`

Brief description

Menu name, used as a prompt.

12.1.398.3 num_commands

`uint32_t sf_console_menu_t::num_commands`

Brief description

Number of commands in this menu.

12.1.398.4 command_list

`sf_console_command_t::command_list`

Brief description

Pointer to an array of commands of length num_commands.

12.1.399 sf_crypto_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_crypto_ctrl_t *const p_ctrl, sf_crypto_cfg_t const
*const p_cfg)
    ssp_err_t(*  close)(sf_crypto_ctrl_t *const p_ctrl)
    ssp_err_t(*  lock)(sf_crypto_ctrl_t *const p_ctrl)
    ssp_err_t(*  unlock)(sf_crypto_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
    ssp_err_t(*  statusGet)(sf_crypto_ctrl_t *const p_ctrl, sf_crypto_state_t
*p_status)
} sf_crypto_api_t
```

12.1.400 sf_crypto_callback_args_t

```
typedef struct{
    sf_crypto_event_t  event
    ssp_err_t  error
} sf_crypto_callback_args_t
```

12.1.400.1 event

`sf_crypto_event_t::event`

Brief description

Event code of the low level hardware.

12.1.400.2 error

`::error`

Brief description

Error code if SF_CRYPTO_EVENT_ERROR.

12.1.401 sf_crypto_cfg_t

```
typedef struct{
    uint32_t  wait_option
    crypto_instance_t *  p_lower_lvl_crypto
    void const  *  p_extend
    void const  *  p_context
    void *  p_memory_pool
    uint32_t  memory_pool_size
    sf_crypto_close_option_t  close_option
} sf_crypto_cfg_t
```

12.1.401.1 wait_option

`uint32_t ::wait_option`

Brief description

Wait option for RTOS service calls.

12.1.401.2 p_lower_lvl_crypto

`crypto_instance_t::p_lower_lvl_crypto`

Brief description

Pointer to a low-level Crypto engine HAL driver instance.

12.1.401.3 p_extend

`void const* ::p_extend`

Brief description

Extension parameter for hardware specific settings.

12.1.401.4 p_context

`void const* ::p_context`

Brief description

Placeholder for user data.

12.1.401.5 p_memory_pool

```
void* ::p_memory_pool
```

Brief description

Byte pool address.

12.1.401.6 memory_pool_size

```
uint32_t ::memory_pool_size
```

Brief description

Byte pool size.

12.1.401.7 close_option

```
sf_crypto_close_option_t::close_option
```

Brief description

Close option.

12.1.402 sf_crypto_data_handle_t

```
typedef struct{
    uint8_t *  p_data
    uint32_t  data_length
} sf_crypto_data_handle_t
```

12.1.402.1 p_data

```
uint8_t* ::p_data
```

Brief description

Pointer to data.

12.1.402.2 data_length

```
uint32_t ::data_length
```

Brief description

The length of data pointed by p_data.

12.1.403 sf_crypto_hash_api_t

```
typedef struct{
    ssp_err_t(*  open>)(sf_crypto_hash_ctrl_t *const p_ctrl, sf_crypto_hash_cfg_t
const *const p_cfg)
    ssp_err_t(*  close)(sf_crypto_hash_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  hashInit)(sf_crypto_hash_ctrl_t *const p_ctrl)
    ssp_err_t(*  hashUpdate)(sf_crypto_hash_ctrl_t *const p_ctrl,
sf_crypto_data_handle_t const *const p_data_in)
    ssp_err_t(*  hashFinal)(sf_crypto_hash_ctrl_t *const p_ctrl,
sf_crypto_data_handle_t *const p_msg_digest, uint32_t *p_size)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_crypto_hash_api_t
```

12.1.404 sf_crypto_hash_callback_args_t

```
typedef struct{
    ssp_err_t  error
} sf_crypto_hash_callback_args_t
```

12.1.404.1 error

::error

Brief description

Error code if SF_CRYPTO_EVENT_ERROR.

12.1.405 sf_crypto_hash_cfg_t

```
typedef struct{
    sf_crypto_hash_type_t  hash_type
    sf_crypto_instance_t *  p_lower_lvl_crypto_common
    hash_instance_t *  p_lower_lvl_instance
    void *  p_extend
} sf_crypto_hash_cfg_t
```

12.1.405.1 hash_type

sf_crypto_hash_type_t::hash_type

Brief description

HASH algorithm type.

12.1.405.2 p_lower_lvl_crypto_common

sf_crypto_instance_t::p_lower_lvl_crypto_common

Brief description

Pointer to a Crypto Framework common instance.

12.1.405.3 p_lower_lvl_instance

`hash_instance_t::p_lower_lvl_instance`

Brief description

pointer to HASH lower-level module instance

12.1.405.4 p_extend

`void* ::p_extend`

Brief description

Pointer to an optional configuration for Crypto HAL module.

12.1.406 sf_crypto_hash_context_t

```
typedef struct{
    uint8_t *  p_message_digest
    uint8_t *  p_message_buffer
    uint64_t  message_bytes
    uint32_t  message_bytes_buffered
} sf_crypto_hash_context_t
```

12.1.406.1 p_message_digest

`uint8_t* ::p_message_digest`

Brief description

Intermediate digest stored buffer.

12.1.406.2 p_message_buffer

`uint8_t* ::p_message_buffer`

Brief description

Intermediate message data stored buffer.

12.1.406.3 message_bytes

`uint64_t ::message_bytes`

Brief description

Number of bytes from user data processed.

12.1.406.4 message_bytes_buffered

`uint32_t ::message_bytes_buffered`

Brief description

Number of bytes buffered in the message data stored buffer.

12.1.407 sf_crypto_hash_instance_ctrl_t

```
typedef struct{
    sf_crypto_hash_state_t  status
    sf_crypto_hash_type_t  hash_type
    sf_crypto_hash_context_t  hash_context
    sf_crypto_instance_t *  p_lower_lvl_crypto_common
    hash_instance_t *  p_lower_lvl_instance
} sf_crypto_hash_instance_ctrl_t
```

12.1.407.1 status

`sf_crypto_hash_state_t::status`

Brief description

Module status.

12.1.407.2 hash_type

`sf_crypto_hash_type_t::hash_type`

Brief description

HASH algorithm type.

12.1.407.3 hash_context

`sf_crypto_hash_context_t::hash_context`

Brief description

Context for calculating message digest.

12.1.407.4 p_lower_lvl_crypto_common

`sf_crypto_instance_t::p_lower_lvl_crypto_common`

Brief description

Pointer to a Crypto Framework common instance.

12.1.407.5 p_lower_lvl_instance

`hash_instance_t::p_lower_lvl_instance`

Brief description

pointer to lower-level crypto module control structure

12.1.408 sf_crypto_hash_instance_t

```
typedef struct{
    sf_crypto_hash_ctrl_t *  p_ctrl
    sf_crypto_hash_cfg_t *  p_cfg
    sf_crypto_hash_api_t const  *  p_api
} sf_crypto_hash_instance_t
```

12.1.408.1 p_ctrl

sf_crypto_hash_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.408.2 p_cfg

sf_crypto_hash_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.408.3 p_api

sf_crypto_hash_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.409 sf_crypto_instance_ctrl_t

```
typedef struct{
    sf_crypto_state_t  status
    TX_MUTEX  mutex
    TX_SEMAPHORE  semaphore
    TX_BYTE_POOL  byte_pool
    uint32_t  wait_option
    uint32_t  open_counter
    void *  p_lower_lvl_crypto
    void(*  p_callback)(sf_crypto_callback_args_t *p_args)
    void *  p_context
    sf_crypto_close_option_t  close_option
} sf_crypto_instance_ctrl_t
```

12.1.409.1 status

sf_crypto_state_t::status

Brief description

Module status.

12.1.409.2 mutex

`TX_MUTEX::mutex`

Brief description

Mutex used in the Crypto Framework.

12.1.409.3 semaphore

`TX_SEMAPHORE::semaphore`

Brief description

Semaphore used in the Crypto Framework (Reserve)

12.1.409.4 byte_pool

`TX_BYTE_POOL::byte_pool`

Brief description

Byte pool used in the Crypto Framework.

12.1.409.5 wait_option

`uint32_t ::wait_option`

Brief description

Wait time option used for RTOS service calls.

12.1.409.6 open_counter

`uint32_t ::open_counter`

Brief description

Counter to keep the number of SF_CRYPTO_XXX opened.

12.1.409.7 p_lower_lvl_crypto

`void* ::p_lower_lvl_crypto`

Brief description

Pointer to a low-level Crypto engine HAL driver instance.

12.1.409.8 p_callback

`void(* ::p_callback) ( *p_args)`

Brief description

Pointer to callback function.

12.1.409.9 p_context

`void* ::p_context`

Brief description

Pointer to a context.

12.1.409.10 close_option

`sf_crypto_close_option_t::close_option`

Brief description

Close option.

12.1.410 sf_crypto_instance_t

```
typedef struct{
    sf_crypto_ctrl_t *  p_ctrl
    sf_crypto_cfg_t const  *  p_cfg
    sf_crypto_api_t const  *  p_api
} sf_crypto_instance_t
```

12.1.410.1 p_ctrl

`sf_crypto_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.410.2 p_cfg

`sf_crypto_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.410.3 p_api

`sf_crypto_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.411 sf_crypto_key_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_crypto_key_ctrl_t *const p_ctrl, sf_crypto_key_cfg_t
const *const p_cfg)
    ssp_err_t(*  close)(sf_crypto_key_ctrl_t *const p_ctrl)
    ssp_err_t(*  keyGenerate)(sf_crypto_key_ctrl_t *const p_ctrl, sf_crypto_key_t
*const p_secret_key, sf_crypto_key_t *const p_public_key)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_crypto_key_api_t
```

12.1.412 sf_crypto_key_cfg_t

```
typedef struct{
    sf_crypto_key_type_t  key_type
    sf_crypto_key_size_t  key_size
    sf_crypto_data_handle_t  domain_params
    sf_crypto_instance_t *  p_lower_lvl_crypto_common
    void const  *  p_extend
} sf_crypto_key_cfg_t
```

12.1.412.1 key_type

sf_crypto_key_type_t::key_type

Brief description

Key type to be generated.

12.1.412.2 key_size

sf_crypto_key_size_t::key_size

Brief description

Key size to be generated.

12.1.412.3 domain_params

sf_crypto_data_handle_t::domain_params

Detailed description

Pointer to domain parameters for the requested key type. Structure contains, pointer to the data (contains the domain data) and data length. Should set to NULL for RSA and AES.

12.1.412.4 p_lower_lvl_crypto_common

sf_crypto_instance_t::p_lower_lvl_crypto_common

Brief description

Pointer to a Crypto Framework common instance.

12.1.412.5 p_extend

`void const* ::p_extend`

Brief description

Extension parameter for hardware specific settings (Future purpose).

12.1.413 sf_crypto_key_instance_ctrl_t

```
typedef struct{
    sf_crypto_key_state_t  status
    sf_crypto_key_type_t  key_type
    sf_crypto_key_size_t  key_size
    sf_crypto_data_handle_t  domain_params
    sf_crypto_instance_ctrl_t *  p_fwk_common_ctrl
    sf_crypto_api_t *  p_fwk_common_api
    void *  p_hal_ctrl
    void *  p_hal_api
} sf_crypto_key_instance_ctrl_t
```

12.1.413.1 status

`sf_crypto_key_state_t::status`

Brief description

Module status.

12.1.413.2 key_type

`sf_crypto_key_type_t::key_type`

Brief description

Key type.

12.1.413.3 key_size

`sf_crypto_key_size_t::key_size`

Brief description

Key size.

12.1.413.4 domain_params

`sf_crypto_data_handle_t::domain_params`

Brief description

Domain parameters structure with pointer to data and length.

12.1.413.5 p_fwk_common_ctrl

`sf_crypto_instance_ctrl_t::p_fwk_common_ctrl`

Brief description

Pointer to the Crypto Framework Common instance.

12.1.413.6 p_fwk_common_api

`sf_crypto_api_t::p_fwk_common_api`

Brief description

Pointer to the Crypto Framework Common instance.

12.1.413.7 p_hal_ctrl

`void* ::p_hal_ctrl`

Brief description

pointer to Crypto module control structure

12.1.413.8 p_hal_api

`void* ::p_hal_api`

Brief description

pointer to Crypto module API structure

12.1.414 sf_crypto_key_instance_t

```
typedef struct{
    sf_crypto_key_ctrl_t *  p_ctrl
    sf_crypto_key_cfg_t *  p_cfg
    sf_crypto_key_api_t const  *  p_api
} sf_crypto_key_instance_t
```

12.1.414.1 p_ctrl

`sf_crypto_key_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.414.2 p_cfg

sf_crypto_key_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.414.3 p_api

sf_crypto_key_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.415 sf_crypto_trng_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_crypto_trng_ctrl_t *const p_ctrl, sf_crypto_trng_cfg_t
const *const p_cfg)
    ssp_err_t(*  close)(sf_crypto_trng_ctrl_t *const p_ctrl)
    ssp_err_t(*  randomNumberGenerate)(sf_crypto_trng_ctrl_t *const p_ctrl,
sf_crypto_data_handle_t *const p_random_number_buff)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_crypto_trng_api_t
```

12.1.416 sf_crypto_trng_cfg_t

```
typedef struct{
    sf_crypto_instance_t *  p_lower_lvl_common
    trng_instance_t *  p_lower_lvl_instance
    void *  p_extend
} sf_crypto_trng_cfg_t
```

12.1.416.1 p_lower_lvl_common

sf_crypto_instance_t::p_lower_lvl_common

Brief description

Pointer to a Crypto Framework common instance.

12.1.416.2 p_lower_lvl_instance

trng_instance_t::p_lower_lvl_instance

Brief description

Pointer to Crypto TRNG HAL instance.

12.1.416.3 p_extend

`void* ::p_extend`

Brief description

Pointer to an optional configuration for HW specific settings.

12.1.417 sf_crypto_trng_instance_t

```
typedef struct{
    sf_crypto_trng_ctrl_t *  p_ctrl
    sf_crypto_trng_cfg_t *  p_cfg
    sf_crypto_trng_api_t const  *  p_api
} sf_crypto_trng_instance_t
```

12.1.417.1 p_ctrl

`sf_crypto_trng_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.417.2 p_cfg

`sf_crypto_trng_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.417.3 p_api

`sf_crypto_trng_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.418 sf_el_fx_t

```
typedef struct{
    sf_block_media_instance_t *  p_lower_lvl_block_media
    void *  p_extend
} sf_el_fx_t
```

12.1.418.1 p_lower_lvl_block_media

`sf_block_media_instance_t::p_lower_lvl_block_media`

Brief description

Lower level block media pointer.

12.1.418.2 p_extend

```
void* ::p_extend
```

12.1.419 sf_el_gx_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_el_gx_ctrl_t *const p_ctrl, sf_el_gx_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(sf_el_gx_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
    UINT(*  setup)(GX_DISPLAY *p_display)
    ssp_err_t(*  canvasInit)(sf_el_gx_ctrl_t *const p_ctrl, GX_WINDOW_ROOT
*p_window_root)
} sf_el_gx_api_t
```

12.1.420 sf_el_gx_callback_args_t

```
typedef struct{
    sf_el_gx_device_t  device
    sf_el_gx_event_t  event
    uint32_t  error
} sf_el_gx_callback_args_t
```

12.1.420.1 device

```
sf_el_gx_device_t::device
```

Brief description

Device code.

12.1.420.2 event

```
sf_el_gx_event_t::event
```

Brief description

Event code of the low level hardware.

12.1.420.3 error

```
uint32_t sf_el_gx_callback_args_t::error
```

Brief description

Error code if SF_EL_GX_EVENT_ERROR.

12.1.421 sf_el_gx_cfg_t

```
typedef struct{
    display_instance_t *  p_display_instance
    display_runtime_cfg_t *  p_display_runtime_cfg
    void *  p_canvas
    void *  p_framebuffer_a
    void *  p_framebuffer_b
    void(*  p_callback)(sf_el_gx_callback_args_t *p_args)
    void *  p_context
    void *  p_jpegbuffer
    uint32_t  jpegbuffer_size
    uint16_t  rotation_angle
    void *  p_sf_jpeg_decode_instance
} sf_el_gx_cfg_t
```

12.1.421.1 p_display_instance

`display_instance_t::p_display_instance`

Brief description

Pointer to a display instance.

12.1.421.2 p_display_runtime_cfg

`display_runtime_cfg_t::p_display_runtime_cfg`

Brief description

Pointer to a runtime display configuration.

12.1.421.3 p_canvas

`void* sf_el_gx_cfg_t::p_canvas`

Brief description

Pointer to a canvas(reserved)

12.1.421.4 p_framebuffer_a

`void* sf_el_gx_cfg_t::p_framebuffer_a`

Brief description

Pointer to a frame buffer(A)

12.1.421.5 p_framebuffer_b

`void* sf_el_gx_cfg_t::p_framebuffer_b`

Brief description

Pointer to a frame buffer(B)

12.1.421.6 p_callback

```
void(* sf_el_gx_cfg_t::p_callback) (sf_el_gx_callback_args_t *p_args)
```

Brief description

Pointer to callback function.

12.1.421.7 p_context

```
void* sf_el_gx_cfg_t::p_context
```

Brief description

Pointer to a context.

12.1.421.8 p_jpegbuffer

```
void* sf_el_gx_cfg_t::p_jpegbuffer
```

Brief description

Pointer to a JPEG work buffer.

12.1.421.9 jpegbuffer_size

```
uint32_t sf_el_gx_cfg_t::jpegbuffer_size
```

Brief description

Size of a JPEG work buffer.

12.1.421.10 rotation_angle

```
uint16_t sf_el_gx_cfg_t::rotation_angle
```

Brief description

Screen rotation angle(0/90/270)

12.1.421.11 p_sf_jpeg_decode_instance

```
void* sf_el_gx_cfg_t::p_sf_jpeg_decode_instance
```

Brief description

Pointer to a JPEG framework instance.

12.1.422 sf_el_gx_instance_ctrl_t

```
typedef struct{
    GX_DISPLAY *  p_display
```

```
    display_instance_t *  p_display_instance
    display_runtime_cfg_t *  p_display_runtime_cfg
    void *  p_canvas
    void *  p_framebuffer_read
    void *  p_framebuffer_write
    void(*  p_callback)(sf_el_gx_callback_args_t *p_args)
    void *  p_context
    TX_SEMAPHORE  semaphore
    bool  rendering_enable
    bool  display_list_flushed
    sf_el_gx_state_t  state
    void *  p_jpegbuffer
    uint32_t  jpegbuffer_size
    uint16_t  rotation_angle
    void *  p_sf_jpeg_decode_instance
} sf_el_gx_instance_ctrl_t
```

12.1.422.1 p_display

`GX_DISPLAY::p_display`

Brief description

Pointer to the GUIX display context.

12.1.422.2 p_display_instance

`display_instance_t::p_display_instance`

Brief description

Pointer to a display instance.

12.1.422.3 p_display_runtime_cfg

`display_runtime_cfg_t::p_display_runtime_cfg`

Brief description

Pointer to a runtime display configuration.

12.1.422.4 p_canvas

`void* ::p_canvas`

Brief description

Pointer to a canvas(reserved)

12.1.422.5 p_framebuffer_read

`void* ::p_framebuffer_read`

Brief description

Pointer to a frame buffer (for displaying)

12.1.422.6 p_framebuffer_write

```
void* ::p_framebuffer_write
```

Brief description

Pointer to a frame buffer (for rendering)

12.1.422.7 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Pointer to callback function.

12.1.422.8 p_context

```
void* ::p_context
```

Brief description

Pointer to a context.

12.1.422.9 semaphore

```
TX_SEMAPHORE::semaphore
```

Brief description

Semaphore for the frame buffer flip sync.

12.1.422.10 rendering_enable

```
bool ::rendering_enable
```

Brief description

Sync flag between Rendering and displaying.

12.1.422.11 display_list_flushed

```
bool ::display_list_flushed
```

Brief description

Flag to show the display list is flushed.

12.1.422.12 state

```
sf_el_gx_state_t::state
```

Brief description

State of this module.

12.1.422.13 p_jpegbuffer

```
void* ::p_jpegbuffer
```

Brief description

Pointer to a JPEG work buffer.

12.1.422.14 jpegbuffer_size

```
uint32_t ::jpegbuffer_size
```

Brief description

Size of a JPEG work buffer.

12.1.422.15 rotation_angle

```
uint16_t ::rotation_angle
```

Brief description

Screen rotation angle(0/90/270)

12.1.422.16 p_sf_jpeg_decode_instance

```
void* ::p_sf_jpeg_decode_instance
```

Brief description

Pointer to a JPEG framework instance.

12.1.423 sf_el_gx_instance_t

```
typedef struct{
    sf_el_gx_ctrl_t *  p_ctrl
    sf_el_gx_cfg_t const  *  p_cfg
    sf_el_gx_api_t const  *  p_api
} sf_el_gx_instance_t
```

12.1.423.1 p_ctrl

```
sf_el_gx_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.423.2 p_cfg

sf_el_gx_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.423.3 p_api

sf_el_gx_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.424 sf_el_nx_comms_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    NX_PACKET *  p_current_packet
    uint32_t  packet_index
    TX_MUTEX  mutex[2]
    NX_PACKET_POOL  pool
    uint8_t  pool_memory[SF_EL_NX_COMMS_PACKET_POOL_MEMORY_SIZE_BYTES]
    NX_IP  ip
    uint8_t  ip_memory[SF_EL_NX_COMMS_IP_MEMORY_SIZE_BYTES]
    uint8_t  arp_memory[SF_EL_NX_COMMS_ARP_MEMORY_SIZE_BYTES]
    NX_TELNET_SERVER  telnet_server
    uint8_t  telnet_server_memory[SF_EL_NX_COMMS_TELNET_SERVER_MEMORY_SIZE_BYTES
]
    UINT  logical_connection
    TX_EVENT_FLAGS_GROUP  available
    TX_QUEUE  queue
    uint8_t  queue_memory[SF_EL_NX_COMMS_QUEUE_MEMORY_SIZE_BYTES]
} sf_el_nx_comms_instance_ctrl_t
```

12.1.424.1 open

uint32_t ::open

12.1.424.2 p_current_packet

NX_PACKET::p_current_packet

12.1.424.3 packet_index

uint32_t ::packet_index

12.1.424.4 mutex

```
TX_MUTEX::mutex
```

12.1.424.5 pool

```
NX_PACKET_POOL::pool
```

12.1.424.6 pool_memory

```
uint8_t ::pool_memory[SF_EL_NX_COMMS_PACKET_POOL_MEMORY_SIZE_BYTES]
```

12.1.424.7 ip

```
NX_IP::ip
```

12.1.424.8 ip_memory

```
uint8_t ::ip_memory[SF_EL_NX_COMMS_IP_MEMORY_SIZE_BYTES]
```

12.1.424.9 arp_memory

```
uint8_t ::arp_memory[SF_EL_NX_COMMS_ARP_MEMORY_SIZE_BYTES]
```

12.1.424.10 telnet_server

```
NX_TELNET_SERVER::telnet_server
```

12.1.424.11 telnet_server_memory

```
uint8_t ::telnet_server_memory[SF_EL_NX_COMMS_TELNET_SERVER_MEMORY_SIZE_BYTES]
```

12.1.424.12 logical_connection

```
UINT ::logical_connection
```

12.1.424.13 available

```
TX_EVENT_FLAGS_GROUP::available
```

Brief description

Flag to tell if this connection is available or not.

12.1.424.14 queue

```
TX_QUEUE::queue
```

Brief description

Queue of received bytes.

12.1.424.15 queue_memory

```
uint8_t ::queue_memory[SF_EL_NX_COMMS_QUEUE_MEMORY_SIZE_BYTES]
```

12.1.425 sf_el_nx_comms_on_comms_cfg_t

```
typedef struct{
    uint32_t  ip_address
    uint32_t  subnet_mask
    VOID(*  driver)(NX_IP_DRIVER *driver_req_ptr)
} sf_el_nx_comms_on_comms_cfg_t
```

12.1.425.1 ip_address

```
uint32_t sf_::ip_address
```

12.1.425.2 subnet_mask

```
uint32_t sf_::subnet_mask
```

12.1.425.3 driver

```
VOID(* sf_::driver) (NX_IP_DRIVER *driver_req_ptr)
```

12.1.426 sf_el_ux_comms_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    TX_MUTEX  mutex[2]
    UX_SLAVE_CLASS_CDC_ACM *  p_cdc
    uint32_t  leftover_length
    uint32_t  index
    uint8_t  memory[SF_EL_UX_COMMS_CFG_USB_MEMORY_SIZE_BYTES]
    uint8_t  rx_memory[SF_EL_UX_COMMS_CFG_BUFFER_MAX_LENGTH]
    TX_SEMAPHORE  semaphore
} sf_el_ux_comms_instance_ctrl_t
```

12.1.426.1 open

```
uint32_t ::open
```

12.1.426.2 mutex

TX_MUTEX::mutex

12.1.426.3 p_cdc

UX_SLAVE_CLASS_CDC_ACM::p_cdc

12.1.426.4 leftover_length

uint32_t ::leftover_length

12.1.426.5 index

uint32_t ::index

12.1.426.6 memory

uint8_t ::memory[SF_EL_UX_COMMS_CFG_USB_MEMORY_SIZE_BYTES]

12.1.426.7 rx_memory

uint8_t ::rx_memory

12.1.426.8 semaphore

TX_SEMAPHORE::semaphore

12.1.427 sf_external_irq_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_external_irq_ctrl_t *const p_ctrl,
sf_external_irq_cfg_t const *const p_cfg)
    ssp_err_t(*  wait)(sf_external_irq_ctrl_t *const p_ctrl, ULONG const timeout)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
    ssp_err_t(*  close)(sf_external_irq_ctrl_t *const p_ctrl)
} sf_external_irq_api_t
```

12.1.428 sf_external_irq_cfg_t

```
typedef struct{
    external_irq_instance_t const  *  p_lower_lvl_irq
    sf_external_irq_event_t  event
} sf_external_irq_cfg_t
```

12.1.428.1 p_lower_lvl_irq

external_irq_instance_t::p_lower_lvl_irq

Detailed description

All info needed to work with lower layer

12.1.428.2 event

sf_external_irq_event_t::event

Brief description

Select what happens when the external IRQ is triggered.

12.1.429 sf_external_irq_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    TX_MUTEX  mutex
    TX_SEMAPHORE  semaphore
    external_irq_instance_t const  *  p_lower_lvl_irq
    bool  callback_used
} sf_external_irq_instance_ctrl_t
```

12.1.429.1 open

uint32_t ::open

Brief description

Used by driver to check if control block is valid.

12.1.429.2 mutex

TX_MUTEX::mutex

Brief description

Mutex used to protect access to lower level driver hardware registers.

12.1.429.3 semaphore

TX_SEMAPHORE::semaphore

Brief description

Semaphore used for SF_EXTERNAL_IRQ_Wait.

12.1.429.4 p_lower_lvl_irq

external_irq_instance_t::p_lower_lvl_irq

Brief description

Pointer to lower level driver instance.

12.1.429.5 callback_used

```
bool ::callback_used
```

Brief description

Used by driver to check if wait can be used.

12.1.430 sf_external_irq_instance_t

```
typedef struct{
    sf_external_irq_ctrl_t *  p_ctrl
    sf_external_irq_cfg_t const  *  p_cfg
    sf_external_irq_api_t const  *  p_api
} sf_external_irq_instance_t
```

12.1.430.1 p_ctrl

```
sf_external_irq_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.430.2 p_cfg

```
sf_external_irq_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.430.3 p_api

```
sf_external_irq_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.431 sf_i2c_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_i2c_ctrl_t *p_ctrl, sf_i2c_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(sf_i2c_ctrl_t *const p_ctrl, uint8_t *const p_dest,
uint32_t const bytes, bool const restart, uint32_t const timeout)
    ssp_err_t(*  write)(sf_i2c_ctrl_t *const p_ctrl, uint8_t *const p_src,
uint32_t const bytes, bool const restart, uint32_t const timeout)
```

```
    ssp_err_t(*  reset)(sf_i2c_ctrl_t *const p_ctrl, uint32_t const timeout)
    ssp_err_t(*  close)(sf_i2c_ctrl_t *const p_ctrl)
    ssp_err_t(*  lock)(sf_i2c_ctrl_t *const p_ctrl)
    ssp_err_t(*  unlock)(sf_i2c_ctrl_t *const p_ctrl)
    ssp_err_t(*  version)(ssp_version_t *const p_version)
} sf_i2c_api_t
```

12.1.432 sf_i2c_bus_t

```
typedef struct{
    uint8_t  channel
    TX_MUTEX *  p_lock_mutex
    TX_EVENT_FLAGS_GROUP *  p_sync_eventflag
    sf_i2c_ctrl_t **  pp_curr_ctrl
    uint8_t *  p_bus_name
    i2c_api_master_t const  *  p_lower_lvl_api
    uint8_t  device_count
    sf_i2c_ctrl_t **  pp_curr_bus_ctrl
} sf_i2c_bus_t
```

12.1.432.1 channel

`uint8_t sf_i2c_bus_t::channel`

Brief description

Channel.

12.1.432.2 p_lock_mutex

`TX_MUTEX::p_lock_mutex`

Brief description

Lock mutex handle for this channel.

12.1.432.3 p_sync_eventflag

`TX_EVENT_FLAGS_GROUP::p_sync_eventflag`

Brief description

Pointer to the event flag object for I2C data transfer.

12.1.432.4 pp_curr_ctrl

`sf_i2c_ctrl_t::pp_curr_ctrl`

Brief description

Current device using the bus (by switching the address)

12.1.432.5 p_bus_name

`uint8_t* sf_i2c_bus_t::p_bus_name`

Brief description

User-supplied name to identify the bus. Useful for debugging.

12.1.432.6 p_lower_lvl_api

`i2c_api_master_t::p_lower_lvl_api`

Brief description

Pointer to I2C HAL interface to be used in the framework.

12.1.432.7 device_count

`uint8_t sf_i2c_bus_t::device_count`

Brief description

Number of devices on the bus; initialize to 0.

12.1.432.8 pp_curr_bus_ctrl

`sf_i2c_ctrl_t::pp_curr_bus_ctrl`

Brief description

Device configured on the bus (low level configuration)

12.1.433 sf_i2c_cfg_t

```
typedef struct{
    sf_i2c_bus_t *  p_bus
    i2c_cfg_t const  *  p_lower_lvl_cfg
} sf_i2c_cfg_t
```

12.1.433.1 p_bus

`sf_i2c_bus_t::p_bus`

Brief description

Bus used by the device.

12.1.433.2 p_lower_lvl_cfg

`i2c_cfg_t::p_lower_lvl_cfg`

Brief description

Pointer to I2C HAL configuration.

12.1.434 sf_i2c_instance_ctrl_t

```
typedef struct{
    sf_i2c_bus_t *  p_bus
    i2c_master_instance_t const  *  p_lower_lvl_i2c
    i2c_cfg_t  lower_lvl_cfg
    i2c_ctrl_t *  p_lower_lvl_ctrl
    sf_i2c_dev_state_t  dev_state
    bool  locked
    bool  restarted
} sf_i2c_instance_ctrl_t
```

12.1.434.1 p_bus

sf_i2c_bus_t::p_bus

Brief description

Bus using this device. Copy from configuration structure.

12.1.434.2 p_lower_lvl_i2c

i2c_master_instance_t::p_lower_lvl_i2c

Brief description

I2C instance.

12.1.434.3 lower_lvl_cfg

i2c_cfg_t::lower_lvl_cfg

Brief description

Used to reconfigure I2C driver.

12.1.434.4 p_lower_lvl_ctrl

i2c_ctrl_t::p_lower_lvl_ctrl

Brief description

I2C peripheral control block.

12.1.434.5 dev_state

sf_i2c_dev_state_t::dev_state

Brief description

Device status.

12.1.434.6 locked

`bool ::locked`

Brief description

Lock and unlock bus for a device.

12.1.434.7 restarted

`bool ::restarted`

Brief description

Indicates whether device issued a restart.

12.1.435 sf_i2c_instance_t

```
typedef struct{
    sf_i2c_ctrl_t *  p_ctrl
    sf_i2c_cfg_t const  *  p_cfg
    sf_i2c_api_t const  *  p_api
} sf_i2c_instance_t
```

12.1.435.1 p_ctrl

`sf_i2c_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.435.2 p_cfg

`sf_i2c_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.435.3 p_api

`sf_i2c_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.436 sf_jpeg_decode_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_jpeg_decode_ctrl_t *const p_ctrl, sf_jpeg_decode_cfg_t
const *const p_cfg)
```

```
    ssp_err_t(*  inputBufferSet)(sf_jpeg_decode_ctrl_t *const p_ctrl, void *const
p_buffer, uint32_t const buffer_size)
    ssp_err_t(*  outputBufferSet)(sf_jpeg_decode_ctrl_t *const p_ctrl, void
*p_buffer, uint32_t buffer_size)
    ssp_err_t(*  linesDecodedGet)(sf_jpeg_decode_ctrl_t *const p_ctrl, uint32_t
*const p_lines)
    ssp_err_t(*  horizontalStrideSet)(sf_jpeg_decode_ctrl_t *const p_ctrl,
uint32_t horizontal_stride)
    ssp_err_t(*  imageSubsampleSet)(sf_jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_subsample_t horizontal_subsample, jpeg_decode_subsample_t
vertical_subsample)
    ssp_err_t(*  wait)(sf_jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_status_t
*const p_status, uint32_t timeout)
    ssp_err_t(*  statusGet)(sf_jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_status_t *const p_status)
    ssp_err_t(*  imageSizeGet)(sf_jpeg_decode_ctrl_t *const p_ctrl, uint16_t
*p_horizontal_size, uint16_t *p_vertical_size)
    ssp_err_t(*  pixelFormatGet)(sf_jpeg_decode_ctrl_t *const p_ctrl,
jpeg_decode_color_space_t *const p_color_space)
    ssp_err_t(*  close)(sf_jpeg_decode_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_jpeg_decode_api_t
```

12.1.437 sf_jpeg_decode_cfg_t

```
typedef struct{
    jpeg_decode_instance_t const  *  p_lower_lvl_jpeg_decode
} sf_jpeg_decode_cfg_t
```

12.1.437.1 p_lower_lvl_jpeg_decode

`jpeg_decode_instance_t::p_lower_lvl_jpeg_decode`

Detailed description

Pointer to a driver structure that implements this interface. Pre-configured driver structures are located in r_jpeg_decode.c and extern'ed in r_jpeg_decode.h.

12.1.438 sf_jpeg_decode_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    uint32_t  state
    TX_MUTEX  mutex
    TX_EVENT_FLAGS_GROUP  events
    jpeg_decode_instance_t const  *  p_lower_lvl_jpeg_decode
} sf_jpeg_decode_instance_ctrl_t
```

12.1.438.1 open

`uint32_t ::open`

Brief description

Indicate whether the driver is open.

12.1.438.2 state

`uint32_t ::state`

Brief description

Used by driver to check if pointer to control block is valid.

12.1.438.3 mutex

`TX_MUTEX::mutex`

Brief description

Mutex used to protect access to lower level driver hardware.

12.1.438.4 events

`TX_EVENT_FLAGS_GROUP::events`

Brief description

Event flags used by the HAL driver to notify the framework driver of.

12.1.438.5 p_lower_lvl_jpeg_decode

`jpeg_decode_instance_t::p_lower_lvl_jpeg_decode`

Brief description

Pointer to lower level instance.

12.1.439 sf_jpeg_decode_instance_t

```
typedef struct{
    sf_jpeg_decode_ctrl_t *  p_ctrl
    sf_jpeg_decode_cfg_t const  *  p_cfg
    sf_jpeg_decode_api_t const  *  p_api
} sf_jpeg_decode_instance_t
```

12.1.439.1 p_ctrl

`sf_jpeg_decode_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.439.2 p_cfg

`sf_jpeg_decode_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.439.3 p_api

`sf_jpeg_decode_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.440 sf_message_acquire_cfg_t

```
typedef struct{
    bool  buffer_keep
} sf_message_acquire_cfg_t
```

12.1.440.1 buffer_keep

`bool sf_message_acquire_cfg_t::buffer_keep`

Brief description

Buffer keep option.

12.1.441 sf_message_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_message_ctrl_t *const p_ctrl, sf_message_cfg_t const
*const p_cfg)
    ssp_err_t(*  close)(sf_message_ctrl_t *const p_ctrl)
    ssp_err_t(*  bufferAcquire)(sf_message_ctrl_t const *const p_ctrl,
sf_message_header_t **pp_buffer, sf_message_acquire_cfg_t const *const
p_acquire_cfg, uint32_t const wait_option)
    ssp_err_t(*  bufferRelease)(sf_message_ctrl_t *const p_ctrl,
sf_message_header_t *const p_buffer, sf_message_release_option_t const option)
    ssp_err_t(*  post)(sf_message_ctrl_t *const p_ctrl, sf_message_header_t const
*const p_buffer, sf_message_post_cfg_t const *const p_post_cfg,
sf_message_post_err_t *const p_post_err, uint32_t const wait_option)
    ssp_err_t(*  pend)(sf_message_ctrl_t const *const p_ctrl, TX_QUEUE const
*const p_queue, sf_message_header_t **pp_buffer, uint32_t const wait_option)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_message_api_t
```

12.1.442 sf_message_buffer_ctrl_t

```
typedef struct{
    void(*  p_callback)(sf_message_callback_args_t *)
    void const  *  p_context
    struct sf_message_buffer_ctrl_t::st_buffer_ctrl_flag  flag_b
} sf_message_buffer_ctrl_t
```

12.1.442.1 p_callback

```
void(* sf_message_buffer_ctrl_t::p_callback) (sf_message_callback_args_t *)
```

Brief description

Optional user callback function.

12.1.442.2 p_context

```
void const* sf_message_buffer_ctrl_t::p_context
```

Brief description

Context provided to user during callback.

12.1.442.3 flag_b

```
sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::flag_b
```

12.1.443 sf_message_buffer_ctrl_t::st_buffer_ctrl_flag

```
typedef struct{
    uint32_t  semaphore
    uint32_t  buffer_keep
    uint32_t  nak_response
    uint32_t  reserved
    uint32_t  in_use
} sf_message_buffer_ctrl_t::st_buffer_ctrl_flag
```

12.1.443.1 semaphore

```
uint32_t sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::semaphore
```

Brief description

Counting semaphore to prevent a buffer from being released.

12.1.443.2 buffer_keep

```
uint32_t sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::buffer_keep
```

Brief description

Buffer keep request.

12.1.443.3 nak_response

`uint32_t sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::nak_response`

Brief description

NAK (ORed logic for multiple subscribers)

12.1.443.4 reserved

`uint32_t sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::reserved`

Brief description

Reserved bits.

12.1.443.5 in_use

`uint32_t sf_message_buffer_ctrl_t::st_buffer_ctrl_flag::in_use`

Brief description

Buffer in-use.

12.1.444 sf_message_callback_args_t

```
typedef struct{
    sf_message_callback_event_t  event
    void const  *  p_context
} sf_message_callback_args_t
```

12.1.444.1 event

`sf_message_callback_event_t::event`

Brief description

Event code.

12.1.444.2 p_context

`void const* sf_message_callback_args_t::p_context`

Brief description

Context provided to user during callback.

12.1.445 sf_message_cfg_t

```
typedef struct{
    void *  p_work_memory_start
    uint32_t  work_memory_size_bytes
    uint32_t  buffer_size
    sf_message_subscriber_list_t **  pp_subscriber_lists
    uint8_t *  p_block_pool_name
} sf_message_cfg_t
```

12.1.445.1 p_work_memory_start

```
void* sf_message_cfg_t::p_work_memory_start
```

Brief description

Start address of the memory area.

12.1.445.2 work_memory_size_bytes

```
uint32_t sf_message_cfg_t::work_memory_size_bytes
```

Brief description

Size of working memory area in bytes.

12.1.445.3 buffer_size

```
uint32_t sf_message_cfg_t::buffer_size
```

Brief description

Bytes of the message block.

12.1.445.4 pp_subscriber_lists

```
sf_message_subscriber_list_t::pp_subscriber_lists
```

Brief description

Pointer array to the subscriber lists.

12.1.445.5 p_block_pool_name

```
uint8_t* sf_message_cfg_t::p_block_pool_name
```

Brief description

Pointer to the block pool name.

12.1.446 sf_message_header_t

```
typedef struct{
    uint32_t  event
    uint32_t  class_code
    uint32_t  class_instance
    uint32_t  code
    struct{}                event_b
    union{}                 union{}
} sf_message_header_t
```

12.1.446.1 event

```
uint32_t sf_message_header_t::event
```

12.1.446.2 class_code

```
uint32_t sf_message_header_t::class_code
```

Brief description

Event class code.

12.1.446.3 class_instance

```
uint32_t sf_message_header_t::class_instance
```

Brief description

Event class instance number.

12.1.446.4 code

```
uint32_t sf_message_header_t::code
```

Brief description

Event code.

12.1.446.5 event_b

```
See source code for the definition of this member.
```

12.1.446.6 union{}

```
See source code for the definition of this member.
```

12.1.447 sf_message_instance_ctrl_t

```
typedef struct{
    TX_BLOCK_POOL  block_pool
    sf_message_subscriber_list_t **  pp_subscriber_lists
    uint32_t  buffer_size
    uint32_t  number_of_buffers
    uint16_t  number_of_subscriber_groups
    sf_message_state_t  state
} sf_message_instance_ctrl_t
```

12.1.447.1 block_pool

TX_BLOCK_POOL::block_pool

Brief description

Pointer to the memory block pool control.

12.1.447.2 pp_subscriber_lists

sf_message_subscriber_list_t::pp_subscriber_lists

Brief description

Pointer array to the subscriber lists.

12.1.447.3 buffer_size

uint32_t ::buffer_size

Brief description

Bytes of the message buffer.

12.1.447.4 number_of_buffers

uint32_t ::number_of_buffers

Brief description

The number of allocated buffers.

12.1.447.5 number_of_subscriber_groups

uint16_t ::number_of_subscriber_groups

Brief description

The number of subscriber groups.

12.1.447.6 state

`sf_message_state_t::state`

Brief description

Status of the message framework.

12.1.448 sf_message_instance_range_t

```
typedef struct{
    uint8_t  start
    uint8_t  end
} sf_message_instance_range_t
```

12.1.448.1 start

`uint8_t sf_message_instance_range_t::start`

Brief description

Start of the event class instance range.

12.1.448.2 end

`uint8_t sf_message_instance_range_t::end`

Brief description

End of the event class instance range.

12.1.449 sf_message_instance_t

```
typedef struct{
    sf_message_ctrl_t *  p_ctrl
    sf_message_cfg_t const  *  p_cfg
    sf_message_api_t const  *  p_api
} sf_message_instance_t
```

12.1.449.1 p_ctrl

`sf_message_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.449.2 p_cfg

`sf_message_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.449.3 p_api

`sf_message_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.450 sf_message_post_cfg_t

```
typedef struct{
    sf_message_priority_t  priority
    void(*  p_callback)(sf_message_callback_args_t *)
    void const  *  p_context
} sf_message_post_cfg_t
```

12.1.450.1 priority

`sf_message_priority_t::priority`

Brief description

Message priority.

12.1.450.2 p_callback

`void(* sf_message_post_cfg_t::p_callback) (sf_message_callback_args_t *)`

Brief description

User callback function.

12.1.450.3 p_context

`void const* sf_message_post_cfg_t::p_context`

Brief description

Context provided to user during callback.

12.1.451 sf_message_post_err_t

```
typedef struct{
    TX_QUEUE *  p_queue
} sf_message_post_err_t
```

12.1.451.1 p_queue

`TX_QUEUE::p_queue`

Brief description

Queue.

12.1.452 sf_message_subscriber_list_t

```
typedef struct{
    sf_message_event_class_t  event_class
    uint16_t  number_of_nodes
    sf_message_subscriber_t **  pp_subscriber_group
} sf_message_subscriber_list_t
```

12.1.452.1 event_class

```
sf_message_event_class_t sf_message_subscriber_list_t::event_class
```

Brief description

Event class code.

12.1.452.2 number_of_nodes

```
uint16_t sf_message_subscriber_list_t::number_of_nodes
```

Brief description

Number of nodes in the subscriber group.

12.1.452.3 pp_subscriber_group

```
sf_message_subscriber_t::pp_subscriber_group
```

Brief description

Subscriber group for the event class.

12.1.453 sf_message_subscriber_t

```
typedef struct{
    TX_QUEUE *  p_queue
    sf_message_instance_range_t  instance_range
} sf_message_subscriber_t
```

12.1.453.1 p_queue

```
TX_QUEUE::p_queue
```

Brief description

Pointer to the message queue for subscriber thread.

12.1.453.2 instance_range

`sf_message_instance_range_t::instance_range`

Brief description

Range of the event class instance to receive message.

12.1.454 sf_power_profiles_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_power_profiles_ctrl_t *const p_ctrl,
sf_power_profiles_cfg_t const *const p_cfg)
    ssp_err_t(*  sleep)(sf_power_profiles_ctrl_t *const p_ctrl)
    ssp_err_t(*  close)(sf_power_profiles_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_power_profiles_api_t
```

12.1.455 sf_power_profiles_callback_args_t

```
typedef struct{
    sf_power_profiles_event_t  event
    void *  p_context
} sf_power_profiles_callback_args_t
```

12.1.455.1 event

`sf_power_profiles_event_t::event`

Brief description

Power profiles callback event.

12.1.455.2 p_context

`void* sf_power_profiles_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.456 sf_power_profiles_cfg_t

```
typedef struct{
    ioport_cfg_t const  *const  p_wake_ioport_pin_tbl
    ioport_cfg_t const  *const  p_sleep_ioport_pin_tbl
    sf_power_profiles_mode_t  operating_mode
    bool  retain_output_signals
    lpm_instance_t const  *const  p_lower_lvl_lpm
    rtc_instance_t const  *const  p_lower_lvl_rtc
```

```
    void(*  p_callback)(sf_power_profiles_callback_args_t *p_args)
    void *  p_context
} sf_power_profiles_cfg_t
```

12.1.456.1 p_wake_ioport_pin_tbl

`ioport_cfg_t::p_wake_ioport_pin_tbl`

Brief description

Pointer to ioport settings for wakeup.

12.1.456.2 p_sleep_ioport_pin_tbl

`ioport_cfg_t::p_sleep_ioport_pin_tbl`

Brief description

Pointer to ioport settings for sleep.

12.1.456.3 operating_mode

`sf_power_profiles_mode_t::operating_mode`

Brief description

Power profile mode to use.

12.1.456.4 retain_output_signals

`bool sf_power_profiles_cfg_t::retain_output_signals`

Detailed description

(Future implementation:) True (Default) ==> address bus and bus control signals retain the output state False ==> signals are set to high-impedance state

12.1.456.5 p_lower_lvl_lpm

`lpm_instance_t::p_lower_lvl_lpm`

Brief description

Pointer to the LPM instance.

12.1.456.6 p_lower_lvl_rtc

`rtc_instance_t::p_lower_lvl_rtc`

Brief description

Pointer to the RTC instance (if any)

12.1.456.7 p_callback

```
void(* sf_power_profiles_cfg_t::p_callback) (sf_power_profiles_callback_args_t
*p_args)
```

Brief description

Callback function.

12.1.456.8 p_context

```
void* sf_power_profiles_cfg_t::p_context
```

Brief description

Placeholder for user data.

12.1.457 sf_power_profiles_ctrl_t

```
typedef struct{
    uint32_t  open
    ioport_cfg_t const  *  p_wake_ioport_pin_tbl
    ioport_cfg_t const  *  p_sleep_ioport_pin_tbl
    sf_power_profiles_mode_t  operating_mode
    lpm_api_t const  *  p_api_lpm
    rtc_api_t const  *  p_api_rtc
    rtc_ctrl_t *  p_ctrl_rtc
    void(*  p_callback)(sf_power_profiles_callback_args_t *p_args)
    void *  p_context
} sf_power_profiles_ctrl_t
```

12.1.457.1 open

```
uint32_t sf_power_profiles_ctrl_t::open
```

Brief description

Used by driver to check if pointer to control block is valid.

12.1.457.2 p_wake_ioport_pin_tbl

```
ioport_cfg_t::p_wake_ioport_pin_tbl
```

Brief description

Pointer to ioport settings for wakeup.

12.1.457.3 p_sleep_ioport_pin_tbl

```
ioport_cfg_t::p_sleep_ioport_pin_tbl
```

Brief description

Pointer to ioport settings for sleep.

12.1.457.4 operating_mode

`sf_power_profiles_mode_t::operating_mode`

Brief description

Power profile mode to use.

12.1.457.5 p_api_lpm

`lpm_api_t::p_api_lpm`

Brief description

Pointer to lower level Low Power driver function pointers.

12.1.457.6 p_api_rtc

`rtc_api_t::p_api_rtc`

Brief description

Pointer to lower level RTC driver function pointers.

12.1.457.7 p_ctrl_rtc

`rtc_ctrl_t::p_ctrl_rtc`

Brief description

Pointer to lower level RTC driver control block.

12.1.457.8 p_callback

```
void(* sf_power_profiles_ctrl_t::p_callback) (sf_power_profiles_callback_args_t
*p_args)
```

Brief description

Callback function.

12.1.457.9 p_context

`void* sf_power_profiles_ctrl_t::p_context`

Brief description

Placeholder for user data.

12.1.458 sf_power_profiles_instance_t

```
typedef struct{
    sf_power_profiles_ctrl_t *  p_ctrl
    sf_power_profiles_cfg_t const  *  p_cfg
    sf_power_profiles_api_t const  *  p_api
} sf_power_profiles_instance_t
```

12.1.458.1 p_ctrl

sf_power_profiles_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.458.2 p_cfg

sf_power_profiles_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.458.3 p_api

sf_power_profiles_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.459 sf_slider_on_ctsu_cfg_t

```
typedef struct{
    const sf_slider_type_t  type
    uint32_t  num_slider_channels
    ctsu_channel_pair_t const  *  p_slider_channels
    int32_t const  *  p_normalization
    int32_t *  p_channel_average
    uint32_t *  p_offset
    uint16_t *const  p_slider_scount
    uint16_t *const  p_slider_baseline
    int32_t *const  p_slider_delta
    sf_touch_ctsu_slider_id_t  id
    int32_t  max_slider_value
    const uint16_t  slider_norm_max
    const int32_t  slider_threshold
    const int32_t  channel_average_weight
    const int32_t  prev_sum_weight
```

```
    const int32_t  cutoff
} sf_slider_on_ctsu_cfg_t
```

12.1.459.1 type

`sf_slider_type_t::type`

Detailed description

Linear or circular (wheel)

12.1.459.2 num_slider_channels

`uint32_t ::num_slider_channels`

12.1.459.3 p_slider_channels

`ctsu_channel_pair_t::p_slider_channels`

Detailed description

Define the channel/channel pairs which make up a slider.

12.1.459.4 p_normalization

`int32_t const* ::p_normalization`

12.1.459.5 p_channel_average

`int32_t* ::p_channel_average`

12.1.459.6 p_offset

`uint32_t* ::p_offset`

12.1.459.7 p_slider_scount

`uint16_t* const ::p_slider_scount`

12.1.459.8 p_slider_baseline

`uint16_t* const ::p_slider_baseline`

12.1.459.9 p_slider_delta

`int32_t* const ::p_slider_delta`

12.1.459.10 id

`sf_touch_ctsu_slider_id_t::id`

Detailed description

Unique identifier for slider

12.1.459.11 max_slider_value

`int32_t ::max_slider_value`

Detailed description

Maximum slider value, must be greater than 0; Minimum is always 0

12.1.459.12 slider_norm_max

`const uint16_t ::slider_norm_max`

Detailed description

Individual slider settings that can be modified by the user. Should be the same as in st_sf_touch_ctsu_slider_hdlTOUCH_SLIDER_CFG_NORM_MAX

12.1.459.13 slider_threshold

`const int32_t ::slider_threshold`

Detailed description

Touch threshold. Ensure that value is greater than 0

12.1.459.14 channel_average_weight

`const int32_t ::channel_average_weight`

Detailed description

Weight of running average of counts for each channel

12.1.459.15 prev_sum_weight

`const int32_t ::prev_sum_weight`

Detailed description

Weight of running average of previous sum in position calculation, must be greater than 0

12.1.459.16 cutoff

`const int32_t ::cutoff`

Detailed description

Defines how far below prev_sum running average to get before "SF_TOUCH_SLIDER_STATE_RELEASED" detected

12.1.460 sf_socket_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_socket_ctrl_t *p_ctrl, sf_socket_cfg_t const *const
p_cfg)
    ssp_err_t(*  close)(sf_socket_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_socket_api_t
```

12.1.461 sf_socket_cfg_t

```
typedef struct{
    sf_wifi_onchip_stack_instance_t *  p_lower_lvl_onchip_wifi
    void *  p_extend
} sf_socket_cfg_t
```

12.1.461.1 p_lower_lvl_onchip_wifi

`sf_wifi_onchip_stack_instance_t::p_lower_lvl_onchip_wifi`

Brief description

Pointer to SF on-chip stack instance.

12.1.461.2 p_extend

`void* sf_socket_cfg_t::p_extend`

Brief description

Extended configuration.

12.1.462 sf_socket_ctrl_t

```
typedef struct{
    sf_wifi_onchip_stack_instance_t *  p_lower_lvl_onchip_wifi
} sf_socket_ctrl_t
```

12.1.462.1 p_lower_lvl_onchip_wifi

`sf_wifi_onchip_stack_instance_t::p_lower_lvl_onchip_wifi`

Brief description

low level wifi interface

12.1.463 sf_socket_instance_t

```
typedef struct{
    sf_socket_ctrl_t *  p_ctrl
    sf_socket_cfg_t const  *  p_cfg
    sf_socket_api_t const  *  p_api
} sf_socket_instance_t
```

12.1.463.1 p_ctrl

sf_socket_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.463.2 p_cfg

sf_socket_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.463.3 p_api

sf_socket_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.464 sf_spi_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_spi_ctrl_t *p_ctrl, sf_spi_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(sf_spi_ctrl_t *const p_ctrl, void *const p_dest, uint32_t
const length, spi_bit_width_t const bit_width, uint32_t const timeout)
    ssp_err_t(*  write)(sf_spi_ctrl_t *const p_ctrl, void *const p_src, uint32_t
const length, spi_bit_width_t const bit_width, uint32_t const timeout)
    ssp_err_t(*  writeRead)(sf_spi_ctrl_t *const p_ctrl, void *const p_src, void
*const p_dest, uint32_t const length, spi_bit_width_t const bit_width, uint32_t
const timeout)
    ssp_err_t(*  close)(sf_spi_ctrl_t *const p_ctrl)
    ssp_err_t(*  lock)(sf_spi_ctrl_t *const p_ctrl)
    ssp_err_t(*  unlock)(sf_spi_ctrl_t *const p_ctrl)
    ssp_err_t(*  version)(ssp_version_t *const p_version)
} sf_spi_api_t
```

12.1.465 sf_spi_bus_t

```
typedef struct{
    uint8_t  channel
    uint32_t  freq_hz_min
    TX_MUTEX *  p_lock_mutex
    TX_EVENT_FLAGS_GROUP *  p_sync_eventflag
    sf_spi_ctrl_t **  pp_curr_ctrl
    uint8_t *  p_bus_name
    spi_api_t const  *  p_lower_lvl_api
    uint8_t  device_count
} sf_spi_bus_t
```

12.1.465.1 channel

`uint8_t sf_spi_bus_t::channel`

Brief description

Channel.

12.1.465.2 freq_hz_min

`uint32_t sf_spi_bus_t::freq_hz_min`

Brief description

Bus min frequency supported.

12.1.465.3 p_lock_mutex

`TX_MUTEX::p_lock_mutex`

Brief description

Lock mutex handle for this channel.

12.1.465.4 p_sync_eventflag

`TX_EVENT_FLAGS_GROUP::p_sync_eventflag`

Brief description

Pointer to the event flag object for SPI data transfer.

12.1.465.5 pp_curr_ctrl

`sf_spi_ctrl_t::pp_curr_ctrl`

Brief description

Current device using the bus.

12.1.465.6 p_bus_name

`uint8_t* sf_spi_bus_t::p_bus_name`

Brief description

peripheral name SCI_SPI/RSPI

12.1.465.7 p_lower_lvl_api

`spi_api_t::p_lower_lvl_api`

Brief description

Pointer to SPI HAL interface to be used in the framework.

12.1.465.8 device_count

`uint8_t sf_spi_bus_t::device_count`

Brief description

Number of devices on the bus, initialize to 0.

12.1.466 sf_spi_cfg_t

```
typedef struct{
    sf_spi_bus_t *  p_bus
    ioport_port_pin_t  chip_select
    ioport_level_t  chip_select_level_active
    spi_cfg_t const  *  p_lower_lvl_cfg
} sf_spi_cfg_t
```

12.1.466.1 p_bus

`sf_spi_bus_t::p_bus`

Brief description

Bus used by the device.

12.1.466.2 chip_select

`ioport_port_pin_t::chip_select`

Brief description

Chip select for this device.

12.1.466.3 chip_select_level_active

`ioport_level_t::chip_select_level_active`

Brief description

Polarity of CS, active High or Low.

12.1.466.4 p_lower_lvl_cfg

`spi_cfg_t::p_lower_lvl_cfg`

Brief description

Pointer to SPI HAL configuration.

12.1.467 sf_spi_instance_ctrl_t

```
typedef struct{
    sf_spi_bus_t *  p_bus
    ioport_port_pin_t  chip_select
    ioport_level_t  chip_select_level_active
    spi_cfg_t  lower_lvl_cfg
    spi_ctrl_t *  p_lower_lvl_ctrl
    sf_spi_dev_state_t  dev_state
    bool  locked
} sf_spi_instance_ctrl_t
```

12.1.467.1 p_bus

`sf_spi_bus_t::p_bus`

Brief description

Bus using this device (copy from cfg)

12.1.467.2 chip_select

`ioport_port_pin_t::chip_select`

Brief description

Chip select for this device (copy from cfg)

12.1.467.3 chip_select_level_active

`ioport_level_t::chip_select_level_active`

Brief description

Polarity of CS, active High or Low (copy from cfg)

12.1.467.4 lower_lvl_cfg

`spi_cfg_t::lower_lvl_cfg`

Brief description

SPI peripheral configuration, use for bus reconfiguration.

12.1.467.5 p_lower_lvl_ctrl

`spi_ctrl_t::p_lower_lvl_ctrl`

Brief description

SPI peripheral control block.

12.1.467.6 dev_state

`sf_spi_dev_state_t::dev_state`

Brief description

Device status.

12.1.467.7 locked

`bool ::locked`

Brief description

Lock and unlock bus for a device.

12.1.468 sf_spi_instance_t

```
typedef struct{
    sf_spi_ctrl_t *  p_ctrl
    sf_spi_cfg_t const  *  p_cfg
    sf_spi_api_t const  *  p_api
} sf_spi_instance_t
```

12.1.468.1 p_ctrl

`sf_spi_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.468.2 p_cfg

`sf_spi_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.468.3 p_api

`sf_spi_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.469 sf_thread_monitor_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_thread_monitor_ctrl_t *const p_ctrl,
sf_thread_monitor_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(sf_thread_monitor_ctrl_t *const p_ctrl)
    ssp_err_t(*  threadRegister)(sf_thread_monitor_ctrl_t *const p_ctrl,
sf_thread_monitor_counter_min_max_t const *p_counter_min_max)
    ssp_err_t(*  threadUnregister)(sf_thread_monitor_ctrl_t *const p_ctrl)
    ssp_err_t(*  countIncrement)(sf_thread_monitor_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_thread_monitor_api_t
```

12.1.470 sf_thread_monitor_cfg_t

```
typedef struct{
    wdt_instance_t const  *  p_lower_lvl_wdt
    bool  profiling_mode_enabled
    UINT  priority
} sf_thread_monitor_cfg_t
```

12.1.470.1 p_lower_lvl_wdt

`wdt_instance_t::p_lower_lvl_wdt`

Brief description

Pointer to lower level watchdog instance.

12.1.470.2 profiling_mode_enabled

`bool sf_thread_monitor_cfg_t::profiling_mode_enabled`

Brief description

Enables or disables profiling mode.

12.1.470.3 priority

`UINT sf_thread_monitor_cfg_t::priority`

Brief description

Priority of thread monitor thread.

12.1.471 sf_thread_monitor_counter_min_max_t

```
typedef struct{
    uint32_t  minimum_count
    uint32_t  maximum_count
} sf_thread_monitor_counter_min_max_t
```

12.1.471.1 minimum_count

```
uint32_t sf_thread_monitor_counter_min_max_t::minimum_count
```

Detailed description

Minimum expected count value. If the current count is less than this value the watchdog will reset.

12.1.471.2 maximum_count

```
uint32_t sf_thread_monitor_counter_min_max_t::maximum_count
```

Detailed description

Maximum expected count value. If the current count is more than this value the watchdog will reset

12.1.472 sf_thread_monitor_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    wdt_instance_t const  *  p_lower_lvl_wdt
    uint32_t  timeout_period_msec
    uint32_t  timeout_period_watchdog_clocks
    bool  profiling_mode_enabled
    TX_MUTEX  mutex
    uint32_t  profiling_mode_check
    sf_thread_monitor_thread_counter_t  thread_counters[THREAD_MONITOR_CFG_MAX_N
UMBER_OF_THREADS]
    TX_THREAD  thread
    void const  *  p_extend
    uint8_t  stack[THREAD_MONITOR_THREAD_STACK_SIZE]
} sf_thread_monitor_instance_ctrl_t
```

12.1.472.1 open

```
uint32_t ::open
```

Detailed description

Used by driver to check if the control structure is valid

12.1.472.2 p_lower_lvl_wdt

`wdt_instance_t::p_lower_lvl_wdt`

Detailed description

Pointer to interface structure for the watchdog peripheral

12.1.472.3 timeout_period_msec

`uint32_t ::timeout_period_msec`

Detailed description

Time in milliseconds of the watchdog timeout period. Used to calculate the period of the monitoring thread.

12.1.472.4 timeout_period_watchdog_clocks

`uint32_t ::timeout_period_watchdog_clocks`

Detailed description

Maximum count value of the watchdog. Used to synchronise to the count.

12.1.472.5 profiling_mode_enabled

`bool ::profiling_mode_enabled`

Detailed description

Used by the driver to check if profiling mode is enabled.

12.1.472.6 mutex

`TX_MUTEX::mutex`

Brief description

Mutex to protect access to the thread counters.

12.1.472.7 profiling_mode_check

`uint32_t ::profiling_mode_check`

Detailed description

Value used to verify profiling mode is enabled when prfiling_mode_enabled == true.

12.1.472.8 thread_counters

`sf_thread_monitor_thread_counter_t::thread_counters`

Detailed description

Data storage for the thread counter information.

12.1.472.9 thread

`TX_THREAD::thread`

Brief description

Thread monitor thread.

12.1.472.10 p_extend

`void const* ::p_extend`

Brief description

Extended configuration data.

12.1.472.11 stack

`uint8_t ::stack[THREAD_MONITOR_THREAD_STACK_SIZE]`

Detailed description

Stack for thread monitor thread.

12.1.473 sf_thread_monitor_instance_t

```
typedef struct{
    sf_thread_monitor_ctrl_t *  p_ctrl
    sf_thread_monitor_cfg_t const  *  p_cfg
    sf_thread_monitor_api_t const  *  p_api
} sf_thread_monitor_instance_t
```

12.1.473.1 p_ctrl

`sf_thread_monitor_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.473.2 p_cfg

`sf_thread_monitor_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.473.3 p_api

`sf_thread_monitor_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.474 sf_thread_monitor_thread_counter_t

```
typedef struct{
    uint32_t  current_count
    uint32_t  minimum_count
    uint32_t  maximum_count
    bool  active
    TX_THREAD *  p_thread
} sf_thread_monitor_thread_counter_t
```

12.1.474.1 current_count

`uint32_t sf_thread_monitor_thread_counter_t::current_count`

Brief description

Current count value for a thread.

12.1.474.2 minimum_count

`uint32_t sf_thread_monitor_thread_counter_t::minimum_count`

Detailed description

Minimum expected count value. If the current count is less than this value the watchdog will reset.

12.1.474.3 maximum_count

`uint32_t sf_thread_monitor_thread_counter_t::maximum_count`

Detailed description

Maximum expected count value. If the current count is more than this value the watchdog will reset

12.1.474.4 active

`bool sf_thread_monitor_thread_counter_t::active`

Detailed description

Indicates to the monitoring thread whether this count data is currently active. When a thread is registered this value will be set to false as the count is likely to be a partial count and so should not be monitored. This value will be set to true by the thread monitor thread when it clears all counts to zero.

12.1.474.5 p_thread

`TX_THREAD::p_thread`

Brief description

Pointer to thread for this counter data.

12.1.475 sf_touch_ctsu_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_touch_ctsu_ctrl_t *const p_ctrl, sf_touch_ctsu_cfg_t
const *const p_cfg)
    ssp_err_t(*  read)(sf_touch_ctsu_ctrl_t *const p_ctrl, void *p_dest,
ctsu_read_t opts, const ctsu_channel_pair_t *channels, const uint16_t count)
    ssp_err_t(*  close)(sf_touch_ctsu_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_touch_ctsu_api_t
```

12.1.476 sf_touch_ctsu_button_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_touch_ctsu_button_ctrl_t *const p_ctrl,
sf_touch_ctsu_button_cfg_t const *const p_cfg)
    ssp_err_t(*  enable)(sf_touch_ctsu_button_ctrl_t *const p_ctrl,
sf_touch_ctsu_button_id const button_id)
    ssp_err_t(*  disable)(sf_touch_ctsu_button_ctrl_t *const p_ctrl,
sf_touch_ctsu_button_id const button_id)
    ssp_err_t(*  close)(sf_touch_ctsu_button_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_touch_ctsu_button_api_t
```

12.1.477 sf_touch_ctsu_button_callback_args_t

```
typedef struct{
    sf_touch_ctsu_button_id  id
    sf_touch_button_state_t  event
    void const  *  p_context
} sf_touch_ctsu_button_callback_args_t
```

12.1.477.1 id

`sf_touch_ctsu_button_id::id`

Brief description

Unique id for button.

12.1.477.2 event

`sf_touch_button_state_t::event`

Brief description

Button callback event.

12.1.477.3 p_context

```
void const* sf_touch_ctsu_button_callback_args_t::p_context
```

Brief description

Placeholder for user data.

12.1.478 sf_touch_ctsu_button_cfg_t

```
typedef struct{
    sf_touch_ctsu_instance_t const  *const  p_lower_lvl_touch_framework
    uint32_t  button_count
    sf_touch_ctsu_button_individual_t **  pp_button_cfgs
    void(*  p_callback)(sf_touch_ctsu_button_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} sf_touch_ctsu_button_cfg_t
```

12.1.478.1 p_lower_lvl_touch_framework

```
sf_touch_ctsu_instance_t::p_lower_lvl_touch_framework
```

Brief description

Pointer to the Touch Framework instance.

12.1.478.2 button_count

```
uint32_t sf_touch_ctsu_button_cfg_t::button_count
```

Brief description

Number of buttons in configuration.

12.1.478.3 pp_button_cfgs

```
sf_touch_ctsu_button_individual_t::pp_button_cfgs
```

Brief description

Pointer to button configurations.

12.1.478.4 p_callback

```
void(* sf_touch_ctsu_button_cfg_t::p_callback)
(sf_touch_ctsu_button_callback_args_t *p_args)
```

Brief description

Callback function.

12.1.478.5 p_context

```
void const* sf_touch_ctsu_button_cfg_t::p_context
```

Brief description

Placeholder for user data.

12.1.478.6 p_extend

```
void const* sf_touch_ctsu_button_cfg_t::p_extend
```

Brief description

Extension parameter for instance specific settings.

12.1.479 sf_touch_ctsu_button_hdl

```
typedef struct{
    sf_touch_ctsu_button_individual_t  button_cfg
    sf_touch_button_state_t  state
    int16_t  offset
    sf_touch_button_state_t  prev_state
    uint32_t  debounce_counter
    uint32_t  active_event_counter
    uint32_t  press_down_counter
    uint32_t  open
} sf_touch_ctsu_button_hdl
```

12.1.479.1 button_cfg

```
sf_touch_ctsu_button_individual_t::button_cfg
```

Brief description

Individual button configuration.

12.1.479.2 state

```
sf_touch_button_state_t::state
```

Brief description

Represents the current state of the button.

12.1.479.3 offset

```
int16_t sf_touch_ctsu_button_hdl::offset
```

Brief description

Offset in the results array and bit offset in the binary.

12.1.479.4 prev_state

`sf_touch_button_state_t::prev_state`

Brief description

Holds previous state of the button.

12.1.479.5 debounce_counter

`uint32_t sf_touch_ctsu_button_hdl::debounce_counter`

Brief description

Debounce counter.

12.1.479.6 active_event_counter

`uint32_t sf_touch_ctsu_button_hdl::active_event_counter`

Brief description

Amount of time for which button is spending between states.

12.1.479.7 press_down_counter

`uint32_t sf_touch_ctsu_button_hdl::press_down_counter`

Brief description

Time for which button is held down.

12.1.479.8 open

`uint32_t sf_touch_ctsu_button_hdl::open`

Brief description

Indicate that button has been opened.

12.1.480 sf_touch_ctsu_button_individual_t

```
typedef struct{
    ctsu_channel_pair_t  button_channel
    uint8_t  release_enable
    uint8_t  press_enable
    uint8_t  repeat_enable
    uint8_t  shorthold_enable
    uint8_t  longhold_enable
    uint8_t  byte
    union{}                event_enable
    uint16_t  debounce_threshold
```

```
    sf_touch_ctsu_button_id  id
} sf_touch_ctsu_button_individual_t
```

12.1.480.1 button_channel

`ctsu_channel_pair_t::button_channel`

Brief description

Define the channel/channel pairs which make up a button.

12.1.480.2 release_enable

`uint8_t sf_touch_ctsu_button_individual_t::release_enable`

Brief description

enable release events

12.1.480.3 press_enable

`uint8_t sf_touch_ctsu_button_individual_t::press_enable`

Brief description

enable press events

12.1.480.4 repeat_enable

`uint8_t sf_touch_ctsu_button_individual_t::repeat_enable`

Brief description

enable repeat events

12.1.480.5 shorthold_enable

`uint8_t sf_touch_ctsu_button_individual_t::shorthold_enable`

Brief description

enable short hold events

12.1.480.6 longhold_enable

`uint8_t sf_touch_ctsu_button_individual_t::longhold_enable`

Brief description

enable long hold events

12.1.480.7 byte

`uint8_t sf_touch_ctsu_button_individual_t::byte`

Brief description

Byte representation of events enabled.

12.1.480.8 event_enable

See source code for the definition of this member.

12.1.480.9 debounce_threshold

uint16_t sf_touch_ctsu_button_individual_t::debounce_threshold

Brief description

Number of consecutive times a button is determined as touched before state changes.

12.1.480.10 id

sf_touch_ctsu_button_id::id

Brief description

Unique id for button.

12.1.481 sf_touch_ctsu_button_instance_ctrl_t

```
typedef struct{
    uint32_t  opened
    sf_touch_ctsu_button_hdl *  p_button_hdl
    uint32_t  button_count
    sf_touch_ctsu_instance_t const  *  p_lower_lvl_ctsu
    void const  *  p_context
    void(*  p_callback)(sf_touch_ctsu_button_callback_args_t *p_args)
} sf_touch_ctsu_button_instance_ctrl_t
```

12.1.481.1 opened

uint32_t ::opened

Brief description

Save the initialization state.

12.1.481.2 p_button_hdl

sf_touch_ctsu_button_hdl::p_button_hdl

Brief description

Pointer to the button handle.

12.1.481.3 button_count

```
uint32_t ::button_count
```

Brief description

Button Count.

12.1.481.4 p_lower_lvl_ctsu

```
sf_touch_ctsu_instance_t::p_lower_lvl_ctsu
```

Brief description

Pointer to the lower level instance.

12.1.481.5 p_context

```
void const* ::p_context
```

Brief description

Placeholder for user data.

12.1.481.6 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback function.

12.1.482 sf_touch_ctsu_button_instance_t

```
typedef struct{
    sf_touch_ctsu_button_ctrl_t *  p_ctrl
    sf_touch_ctsu_button_cfg_t const  *  p_cfg
    sf_touch_ctsu_button_api_t const  *  p_api
} sf_touch_ctsu_button_instance_t
```

12.1.482.1 p_ctrl

```
sf_touch_ctsu_button_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.482.2 p_cfg

```
sf_touch_ctsu_button_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.482.3 p_api

`sf_touch_ctsu_button_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.483 sf_touch_ctsu_callback_args_t

```
typedef struct{
    void *  p_context
} sf_touch_ctsu_callback_args_t
```

12.1.483.1 p_context

`void* sf_touch_ctsu_callback_args_t::p_context`

12.1.484 sf_touch_ctsu_cfg_t

```
typedef struct{
    UINT  priority
    uint16_t  update_hz
    void(*  p_callback)(sf_touch_ctsu_callback_args_t *p_args)
    void *  p_context
    ctsu_instance_t *  p_ctsu_instance
} sf_touch_ctsu_cfg_t
```

12.1.484.1 priority

`UINT sf_touch_ctsu_cfg_t::priority`

Brief description

Priority of the touch panel thread.

12.1.484.2 update_hz

`uint16_t sf_touch_ctsu_cfg_t::update_hz`

Brief description

The frequency to run the scans. This is set by the latest open() call.

12.1.484.3 p_callback

`void(* sf_touch_ctsu_cfg_t::p_callback) (sf_touch_ctsu_callback_args_t *p_args)`

Brief description

Callback function;.

12.1.484.4 p_context

```
void* sf_touch_ctsu_cfg_t::p_context
```

Brief description

Arguments to the callback.

12.1.484.5 p_ctsu_instance

```
ctsu_instance_t::p_ctsu_instance
```

Brief description

CTSU instance.

12.1.485 sf_touch_ctsu_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    uint16_t  update_hz
    ctsu_instance_t *  p_lower_lvl_instance
    uint32_t  cb_index
} sf_touch_ctsu_instance_ctrl_t
```

12.1.485.1 open

```
uint32_t ::open
```

Brief description

Used by driver to check if control block is valid.

12.1.485.2 update_hz

```
uint16_t ::update_hz
```

Brief description

The frequency to run the scans at.

12.1.485.3 p_lower_lvl_instance

```
ctsu_instance_t::p_lower_lvl_instance
```

Brief description

Pointer to CTSU instance.

12.1.485.4 cb_index

```
uint32_t ::cb_index
```

Brief description

Indicates the index in callback registry table.

12.1.486 sf_touch_ctsu_instance_t

```
typedef struct{
    sf_touch_ctsu_ctrl_t *  p_ctrl
    sf_touch_ctsu_cfg_t const  *  p_cfg
    sf_touch_ctsu_api_t const  *  p_api
} sf_touch_ctsu_instance_t
```

12.1.486.1 p_ctrl

```
sf_touch_ctsu_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.486.2 p_cfg

```
sf_touch_ctsu_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.486.3 p_api

```
sf_touch_ctsu_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.487 sf_touch_ctsu_slider_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_touch_ctsu_slider_ctrl_t *const p_ctrl,
sf_touch_ctsu_slider_cfg_t const *const p_cfg)
    ssp_err_t(*  enable)(sf_touch_ctsu_slider_ctrl_t *const p_ctrl,
sf_touch_ctsu_slider_id_t const slider_id)
    ssp_err_t(*  disable)(sf_touch_ctsu_slider_ctrl_t *const p_ctrl,
sf_touch_ctsu_slider_id_t const slider_id)
    ssp_err_t(*  close)(sf_touch_ctsu_slider_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_touch_ctsu_slider_api_t
```

12.1.488 sf_touch_ctsu_slider_callback_args_t

```
typedef struct{
    sf_touch_ctsu_slider_id_t  id
    uint32_t  event
    uint32_t  current_position
    void const  *  p_context
} sf_touch_ctsu_slider_callback_args_t
```

12.1.488.1 id

`sf_touch_ctsu_slider_id_t::id`

Brief description

Unique slider identifier.

12.1.488.2 event

`uint32_t sf_touch_ctsu_slider_callback_args_t::event`

Brief description

Slider callback event.

12.1.488.3 current_position

`uint32_t sf_touch_ctsu_slider_callback_args_t::current_position`

Brief description

Current slider position.

12.1.488.4 p_context

`void const* sf_touch_ctsu_slider_callback_args_t::p_context`

Brief description

Placeholder for user data.

12.1.489 sf_touch_ctsu_slider_cfg_t

```
typedef struct{
    sf_touch_ctsu_instance_t *  p_lower_lvl_touch_framework
    uint32_t  slider_count
    void(*  p_callback)(sf_touch_ctsu_slider_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} sf_touch_ctsu_slider_cfg_t
```

12.1.489.1 p_lower_lvl_touch_framework

```
sf_touch_ctsu_instance_t::p_lower_lvl_touch_framework
```

Detailed description

Pointer to the Touch Framework instance

12.1.489.2 slider_count

```
uint32_t sf_touch_ctsu_slider_cfg_t::slider_count
```

Detailed description

Number of sliders in configuration

12.1.489.3 p_callback

```
void(* sf_touch_ctsu_slider_cfg_t::p_callback)
(sf_touch_ctsu_slider_callback_args_t *p_args)
```

Detailed description

Callback function

12.1.489.4 p_context

```
void const* sf_touch_ctsu_slider_cfg_t::p_context
```

Detailed description

Placeholder for user data

12.1.489.5 p_extend

```
void const* sf_touch_ctsu_slider_cfg_t::p_extend
```

Detailed description

Extension parameter for instance specific settings.

12.1.490 sf_touch_ctsu_slider_instance_ctrl_t

```
typedef struct{
    uint32_t  opened
    uint32_t  slider_count
    sf_touch_ctsu_instance_t const  *  p_lower_lvl_ctsu
    void const  *  p_context
    void(*  p_callback)(sf_touch_ctsu_slider_callback_args_t *p_args)
} sf_touch_ctsu_slider_instance_ctrl_t
```

12.1.490.1 opened

```
uint32_t ::opened
```

Brief description

Save the initialization state of the framework.

12.1.490.2 slider_count

```
uint32_t ::slider_count
```

Brief description

Slider Count.

12.1.490.3 p_lower_lvl_ctsu

```
sf_touch_ctsu_instance_t::p_lower_lvl_ctsu
```

Brief description

Pointer to the lower level instance.

12.1.490.4 p_context

```
void const* ::p_context
```

Brief description

Placeholder for user data.

12.1.490.5 p_callback

```
void(* ::p_callback) ( *p_args)
```

Detailed description

Function to call when an event occurs

12.1.491 sf_touch_ctsu_slider_instance_t

```
typedef struct{
    sf_touch_ctsu_slider_ctrl_t *  p_ctrl
    sf_touch_ctsu_slider_cfg_t const  *  p_cfg
    sf_touch_ctsu_slider_api_t const  *  p_api
} sf_touch_ctsu_slider_instance_t
```

12.1.491.1 p_ctrl

```
sf_touch_ctsu_slider_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.491.2 p_cfg

`sf_touch_ctsu_slider_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.491.3 p_api

`sf_touch_ctsu_slider_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.492 sf_touch_panel_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_touch_panel_ctrl_t *const p_ctrl, sf_touch_panel_cfg_t
const *const p_cfg)
    ssp_err_t(*  calibrate)(sf_touch_panel_ctrl_t *const p_ctrl,
sf_touch_panel_calibrate_t const *const p_expected, sf_touch_panel_payload_t
const *const p_actual, ULONG timeout)
    ssp_err_t(*  start)(sf_touch_panel_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(sf_touch_panel_ctrl_t *const p_ctrl)
    ssp_err_t(*  reset)(sf_touch_panel_ctrl_t *const p_ctrl)
    ssp_err_t(*  close)(sf_touch_panel_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_touch_panel_api_t
```

12.1.493 sf_touch_panel_calibrate_t

```
typedef struct{
    uint16_t  x
    uint16_t  y
    uint16_t  tolerance_pixels
    void const  *  p_extend
} sf_touch_panel_calibrate_t
```

12.1.493.1 x

`uint16_t sf_touch_panel_calibrate_t::x`

Brief description

Expected x coordinate.

12.1.493.2 y

```
uint16_t sf_touch_panel_calibrate_t::y
```

Brief description

Expected y coordinate.

12.1.493.3 tolerance_pixels

```
uint16_t sf_touch_panel_calibrate_t::tolerance_pixels
```

Detailed description

Acceptable linear deviation from the expected coordinates in pixels.

12.1.493.4 p_extend

```
void const* sf_touch_panel_calibrate_t::p_extend
```

Detailed description

Pointer to hardware specific extension. See sf_touch_panel_<instance>_cfg_t in sf_touch_panel_<instance>.h.

12.1.494 sf_touch_panel_cfg_t

```
typedef struct{
    uint16_t  hsize_pixels
    uint16_t  vsize_pixels
    UINT  priority
    sf_message_instance_t const  *  p_message
    uint8_t  event_class_instance
    uint16_t  update_hz
    uint16_t  rotation_angle
    void const  *  p_extend
} sf_touch_panel_cfg_t
```

12.1.494.1 hsize_pixels

```
uint16_t sf_touch_panel_cfg_t::hsize_pixels
```

Brief description

Horizontal size of screen in pixels.

12.1.494.2 vsize_pixels

```
uint16_t sf_touch_panel_cfg_t::vsize_pixels
```

Brief description

Vertical size of screen in pixels.

12.1.494.3 priority

`UINT sf_touch_panel_cfg_t::priority`

Brief description

Priority of the touch panel thread.

12.1.494.4 p_message

`sf_message_instance_t::p_message`

Brief description

Pointer to messaging framework control block.

12.1.494.5 event_class_instance

`uint8_t sf_touch_panel_cfg_t::event_class_instance`

Brief description

Event class instance number for posting touch event class messages.

12.1.494.6 update_hz

`uint16_t sf_touch_panel_cfg_t::update_hz`

Detailed description

The frequency to report repeat (SF_TOUCH_PANEL_EVENT_DOWN or SF_TOUCH_PANEL_EVENT_HOLD) touch events in Hertz. ***NOTE:*** *This will be converted to RTOS ticks in the driver and rounded up to the nearest integer value of RTOS ticks.*

12.1.494.7 rotation_angle

`uint16_t sf_touch_panel_cfg_t::rotation_angle`

Brief description

Touch coordinate rotation angle(0/90/180/270)

12.1.494.8 p_extend

`void const* sf_touch_panel_cfg_t::p_extend`

Detailed description

Pointer to hardware specific extension. See sf_touch_panel_<instance>.h.

12.1.495 sf_touch_panel_i2c_cfg_t

```
typedef struct{
    i2c_master_instance_t const  *  p_lower_lvl_i2c
    sf_external_irq_instance_t const  *  p_lower_lvl_irq
```

```
    ioport_port_pin_t  pin
    sf_touch_panel_i2c_chip_t const  *const  p_chip
} sf_touch_panel_i2c_cfg_t
```

12.1.495.1 p_lower_lvl_i2c

`i2c_master_instance_t::p_lower_lvl_i2c`

Detailed description

Pointer to lower level I2C.

12.1.495.2 p_lower_lvl_irq

`sf_external_irq_instance_t::p_lower_lvl_irq`

Detailed description

Pointer to lower level external IRQ.

12.1.495.3 pin

`ioport_port_pin_t::pin`

Detailed description

Port pin connected to reset line on touch controller chip. Set to SF_TOUCH_PANEL_I2C_RESET_PIN_UNUSED if unused.

12.1.495.4 p_chip

`sf_touch_panel_i2c_chip_t::p_chip`

Detailed description

Selected touch controller chip.

12.1.496 sf_touch_panel_i2c_chip_t

```
typedef struct{
    ssp_err_t(*  payloadGet)(sf_touch_panel_ctrl_t *const p_ctrl,
sf_touch_panel_payload_t *const p_payload)
    ssp_err_t(*  reset)(sf_touch_panel_ctrl_t *const p_ctrl)
} sf_touch_panel_i2c_chip_t
```

12.1.496.1 payloadGet

`(* ::payloadGet) ( *const p_ctrl, *const p_payload)`

Detailed description

Reads the touch chip and fills in the touch payload data.

Table 1916:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. This control structure is initialized in this function.
p_payload	out	Pointer to the payload to data structure. Touch data provided should be processed to logical pixel values.

Parameter p_ctrl

Parameter p_payload

12.1.496.2 reset

```
(* ::reset) ( *const p_ctrl)
```

Detailed description

Resets the touch chip.

Table 1917:Parameters

Name	Direction	Description
p_ctrl	inout	Pointer to a structure allocated by user. This control structure is initialized in this function.

Parameter p_ctrl

12.1.497 sf_touch_panel_i2c_instance_ctrl_t

```
typedef struct{
    uint32_t  open
    uint16_t  hsize_pixels
    uint16_t  vsize_pixels
    sf_message_instance_t const  *  p_message
    uint8_t  event_class_instance
    sf_touch_panel_payload_t *  p_payload
    sf_touch_panel_payload_t  last_payload
    TX_MUTEX  mutex
    TX_EVENT_FLAGS_GROUP  flags
    TX_THREAD  thread
    ioport_port_pin_t  pin
```

```
    i2c_master_instance_t const  *  p_lower_lvl_i2c
    sf_external_irq_instance_t const  *  p_lower_lvl_irq
    uint8_t  stack[SF_TOUCH_PANEL_I2C_STACK_SIZE]
    sf_touch_panel_i2c_chip_t const  *  p_chip
    uint16_t  update_hz
    uint16_t  rotation_angle
} sf_touch_panel_i2c_instance_ctrl_t
```

12.1.497.1 open

`uint32_t ::open`

Brief description

Used by driver to check if control block is valid.

12.1.497.2 hsize_pixels

`uint16_t ::hsize_pixels`

Brief description

Horizontal size of screen in pixels.

12.1.497.3 vsize_pixels

`uint16_t ::vsize_pixels`

Brief description

Vertical size of screen in pixels.

12.1.497.4 p_message

`sf_message_instance_t::p_message`

Brief description

Pointer to messaging framework control block.

12.1.497.5 event_class_instance

`uint8_t ::event_class_instance`

Brief description

Event class instance number for posting touch event class messages.

12.1.497.6 p_payload

`sf_touch_panel_payload_t::p_payload`

Brief description

Pointer to buffer used to store payload.

12.1.497.7 last_payload

`sf_touch_panel_payload_t::last_payload`

Brief description

Stores most recent payload put on queue for comparison.

12.1.497.8 mutex

`TX_MUTEX::mutex`

Brief description

Mutex used to protect access to shared resources.

12.1.497.9 flags

`TX_EVENT_FLAGS_GROUP::flags`

Brief description

Event flags for internal communication.

12.1.497.10 thread

`TX_THREAD::thread`

Brief description

Main touch panel thread.

12.1.497.11 pin

`ioport_port_pin_t::pin`

Brief description

Reset pin.

12.1.497.12 p_lower_lvl_i2c

`i2c_master_instance_t::p_lower_lvl_i2c`

Brief description

Lower level I2C.

12.1.497.13 p_lower_lvl_irq

`sf_external_irq_instance_t::p_lower_lvl_irq`

Brief description

Lower level external IRQ.

12.1.497.14 stack

```
uint8_t ::stack[SF_TOUCH_PANEL_I2C_STACK_SIZE]
```

Brief description

Stack for touch panel thread.

12.1.497.15 p_chip

```
sf_touch_panel_i2c_chip_t::p_chip
```

Brief description

Chip specific functions and definitions.

12.1.497.16 update_hz

```
uint16_t ::update_hz
```

Detailed description

The frequency to report repeat (SF_TOUCH_PANEL_EVENT_DOWN or SF_TOUCH_PANEL_EVENT_HOLD) touch events in Hertz. ***NOTE:*** *This will be converted to RTOS ticks in the driver and rounded up to the nearest integer value of RTOS ticks.*

12.1.497.17 rotation_angle

```
uint16_t ::rotation_angle
```

Brief description

Touch coordinate rotation angle(0/90/180/270)

12.1.498 sf_touch_panel_instance_t

```
typedef struct{
    sf_touch_panel_ctrl_t *  p_ctrl
    sf_touch_panel_cfg_t const  *  p_cfg
    sf_touch_panel_api_t const  *  p_api
} sf_touch_panel_instance_t
```

12.1.498.1 p_ctrl

```
sf_touch_panel_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.498.2 p_cfg

sf_touch_panel_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.498.3 p_api

sf_touch_panel_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.499 sf_touch_panel_payload_t

```
typedef struct{
    sf_message_header_t  header
    int16_t  x
    int16_t  y
    sf_touch_panel_event_t  event_type
} sf_touch_panel_payload_t
```

12.1.499.1 header

sf_message_header_t::header

Brief description

Required header for messaging framework.

12.1.499.2 x

int16_t sf_touch_panel_payload_t::x

Brief description

X coordinate.

12.1.499.3 y

int16_t sf_touch_panel_payload_t::y

Brief description

Y coordinate.

12.1.499.4 event_type

sf_touch_panel_event_t::event_type

Brief description

Touch event type.

12.1.500 sf_uart_comms_cfg_t

```
typedef struct{
    uart_instance_t const  *  p_lower_lvl_uart
} sf_uart_comms_cfg_t
```

12.1.500.1 p_lower_lvl_uart

`uart_instance_t::p_lower_lvl_uart`

Detailed description

Pointer to UART Driver instance

12.1.501 sf_uart_comms_instance_ctrl_t

```
typedef struct{
    uint32_t  state
    uart_instance_t const  *  p_lower_lvl_uart
    TX_MUTEX  mutex[2]
    TX_EVENT_FLAGS_GROUP  eventflag[2]
    TX_QUEUE  queue
    uint32_t  queue_mem[SF_UART_COMMS_CFG_QUEUE_SIZE_WORDS]
} sf_uart_comms_instance_ctrl_t
```

12.1.501.1 state

`uint32_t ::state`

Brief description

UART status.

12.1.501.2 p_lower_lvl_uart

`uart_instance_t::p_lower_lvl_uart`

Brief description

Pointer to UART interface (copied from cfg)

12.1.501.3 mutex

`TX_MUTEX::mutex`

Brief description

Pointer to the mutex object for UART resource mutual exclusion.

12.1.501.4 eventflag

`TX_EVENT_FLAGS_GROUP::eventflag`

Brief description

Pointer to the event flag object for UART data transfer.

12.1.501.5 queue

`TX_QUEUE::queue`

Brief description

Queue for reading.

12.1.501.6 queue_mem

`uint32_t ::queue_mem[SF_UART_COMMS_CFG_QUEUE_SIZE_WORDS]`

Brief description

Queue memory.

12.1.502 sf_wifi_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_wifi_ctrl_t *p_ctrl, sf_wifi_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(sf_wifi_ctrl_t *const p_ctrl)
    ssp_err_t(*  multicastListAdd)(sf_wifi_ctrl_t *const p_ctrl, uint8_t const
*const p_mac_addr)
    ssp_err_t(*  multicastListDelete)(sf_wifi_ctrl_t *const p_ctrl, uint8_t const
*const p_mac_addr)
    ssp_err_t(*  statisticsGet)(sf_wifi_ctrl_t *const p_ctrl, sf_wifi_stats_t
*const p_wifi_device_stats)
    ssp_err_t(*  transmit)(sf_wifi_ctrl_t *const p_ctrl, uint8_t *const p_buf,
uint32_t length)
    ssp_err_t(*  provisioningSet)(sf_wifi_ctrl_t *const p_ctrl,
sf_wifi_provisioning_t const *const p_wifi_provisioning)
    ssp_err_t(*  provisioningGet)(sf_wifi_ctrl_t *const p_ctrl,
sf_wifi_provisioning_t *const p_wifi_provisioning)
    ssp_err_t(*  infoGet)(sf_wifi_ctrl_t *const p_ctrl, sf_wifi_info_t *const
p_wifi_info)
    ssp_err_t(*  scan)(sf_wifi_ctrl_t *const p_ctrl, sf_wifi_scan_t *const
p_scan, uint8_t *const p_cnt)
    ssp_err_t(*  ACLAdd)(sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const
p_mac)
    ssp_err_t(*  ACLDelete)(sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const
p_mac)
    ssp_err_t(*  macAddressGet)(sf_wifi_ctrl_t *const p_ctrl, uint8_t *const
p_mac)
```

```
    ssp_err_t(*  macAddressSet)(sf_wifi_ctrl_t *const p_ctrl, uint8_t const
*const p_mac)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_wifi_api_t
```

12.1.503 sf_wifi_callback_args_t

```
typedef struct{
    sf_wifi_event_t  event
    uint8_t *  p_data
    uint32_t  length
    void const  *  p_context
} sf_wifi_callback_args_t
```

12.1.503.1 event

`sf_wifi_event_t::event`

Brief description

Event code.

12.1.503.2 p_data

`uint8_t* sf_wifi_callback_args_t::p_data`

Brief description

Packet data.

12.1.503.3 length

`uint32_t sf_wifi_callback_args_t::length`

Brief description

Packet Data length.

12.1.503.4 p_context

`void const* sf_wifi_callback_args_t::p_context`

Brief description

Context provided to user during callback.

12.1.504 sf_wifi_cfg_t

```
typedef struct{
    uint8_t  mac_addr[6]
    sf_wifi_interface_hw_mode_t  hw_mode
```

```
    uint8_t  tx_power
    sf_wifi_rts_t  rts
    uint16_t  fragmentation
    uint8_t  dtim
    sf_wifi_high_throughput_t  high_throughput
    sf_wifi_preamble_t  preamble
    sf_wifi_wmm_t  wmm
    uint8_t  max_stations
    sf_wifi_ssid_broadcast_t  ssid_broadcast
    sf_wifi_access_control_t  access_control
    uint32_t  beacon
    uint32_t  station_inactivity_timeout
    sf_wifi_wds_t  wds
    void *  p_buffer_pool_rx
    sf_wifi_mandatory_high_throughput_t  req_high_throughput
    void(*  p_callback)(sf_wifi_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} sf_wifi_cfg_t
```

12.1.504.1 mac_addr

`uint8_t sf_wifi_cfg_t::mac_addr[6]`

Brief description

MAC address of WiFi Device.

12.1.504.2 hw_mode

`sf_wifi_interface_hw_mode_t::hw_mode`

Brief description

Modulation type: 11a/b/g/n.

12.1.504.3 tx_power

`uint8_t sf_wifi_cfg_t::tx_power`

Brief description

Sets transmit power in dBm.

12.1.504.4 rts

`sf_wifi_rts_t::rts`

Brief description

RTS/CTS handshake flag.

12.1.504.5 fragmentation

`uint16_t sf_wifi_cfg_t::fragmentation`

Brief description

Fragmentation threshold.

12.1.504.6 dtim

`uint8_t sf_wifi_cfg_t::dtim`

Brief description

Delivery traffic indication message interval.

12.1.504.7 high_throughput

`sf_wifi_high_throughput_t::high_throughput`

Brief description

High-throughput mode. Only valid for 802.11n.

12.1.504.8 preamble

`sf_wifi_preamble_t::preamble`

Brief description

Preamble type.

12.1.504.9 wmm

`sf_wifi_wmm_t::wmm`

Brief description

WiFi Multimedia Mode flag. If enabled, also requires.

12.1.504.10 max_stations

`uint8_t sf_wifi_cfg_t::max_stations`

Brief description

Maximum permitted stations. Valid in AP mode only.

12.1.504.11 ssid_broadcast

`sf_wifi_ssid_broadcast_t::ssid_broadcast`

Brief description

SSID broadcast flag. Valid in AP mode only.

12.1.504.12 access_control

`sf_wifi_access_control_t::access_control`

Brief description

Mode of access control MAC list.

12.1.504.13 beacon

`uint32_t sf_wifi_cfg_t::beacon`

Brief description

Beacon interval. Valid in AP mode only.

12.1.504.14 station_inactivity_timeout

`uint32_t sf_wifi_cfg_t::station_inactivity_timeout`

Brief description

Station inactivity timeout value. Valid in AP mode only.

12.1.504.15 wds

`sf_wifi_wds_t::wds`

Brief description

WDS flag. Valid in AP mode only.

12.1.504.16 p_buffer_pool_rx

`void* sf_wifi_cfg_t::p_buffer_pool_rx`

Brief description

Pointer to Network stack Rx buffer pool.

12.1.504.17 req_high_throughput

`sf_wifi_mandatory_high_throughput_t::req_high_throughput`

Brief description

Only allow HT mode. Valid in AP mode only.

12.1.504.18 p_callback

`void(* sf_wifi_cfg_t::p_callback) (sf_wifi_callback_args_t *p_args)`

Brief description

Pointer to callback function.

12.1.504.19 p_context

```
void const* sf_wifi_cfg_t::p_context
```

Brief description

User defined context passed into callback function.

12.1.504.20 p_extend

```
void const* sf_wifi_cfg_t::p_extend
```

Brief description

Instance specific configuration.

12.1.505 sf_wifi_ctrl_t

```
typedef struct{
    void *  p_driver_handle
} sf_wifi_ctrl_t
```

12.1.505.1 p_driver_handle

```
void* sf_wifi_ctrl_t::p_driver_handle
```

Detailed description

Storage for information needed for each WiFi device driver in the system.

12.1.506 sf_wifi_info_t

```
typedef struct{
    uint8_t *  p_chipset
    uint16_t  rssi
    uint16_t  noise_level
    uint16_t  link_quality
} sf_wifi_info_t
```

12.1.506.1 p_chipset

```
uint8_t* sf_wifi_info_t::p_chipset
```

Brief description

Pointer to sting showing WiFi chipset/driver information.

12.1.506.2 rssi

```
uint16_t sf_wifi_info_t::rssi
```

Brief description

Received signal strength indication.

12.1.506.3 noise_level

`uint16_t sf_wifi_info_t::noise_level`

Brief description

Signal to noise ratio.

12.1.506.4 link_quality

`uint16_t sf_wifi_info_t::link_quality`

Brief description

Signal strength / quality.

12.1.507 sf_wifi_instance_t

```
typedef struct{
    sf_wifi_ctrl_t *  p_ctrl
    sf_wifi_cfg_t const  *  p_cfg
    sf_wifi_api_t const  *  p_api
} sf_wifi_instance_t
```

12.1.507.1 p_ctrl

`sf_wifi_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.507.2 p_cfg

`sf_wifi_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.507.3 p_api

`sf_wifi_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.508 sf_wifi_ip_addr_t

```
typedef struct{
    sf_wifi_ip_addr_version_t  version
    uint32_t  v4
    uint32_t  v6[4]
    union{}                addr
} sf_wifi_ip_addr_t
```

12.1.508.1 version

`sf_wifi_ip_addr_version_t::version`

Brief description

IP Address Version : v4 or v6.

12.1.508.2 v4

`uint32_t sf_wifi_ip_addr_t::v4`

12.1.508.3 v6

`uint32_t sf_wifi_ip_addr_t::v6[4]`

12.1.508.4 addr

`See source code for the definition of this member.`

Brief description

IP address.

12.1.509 sf_wifi_nsal_callback_args_t

```
typedef struct{
    NX_IP *  p_ip
    sf_wifi_nsal_cfg_t *  p_wifi_nsal_cfg
} sf_wifi_nsal_callback_args_t
```

12.1.509.1 p_ip

`NX_IP::p_ip`

Brief description

Pointer to NetX IP interface.

12.1.509.2 p_wifi_nsal_cfg

`sf_wifi_nsal_cfg_t::p_wifi_nsal_cfg`

Brief description

pointer to NSAL configuration

12.1.510 sf_wifi_nsal_cfg_t

```
typedef struct{
    sf_wifi_nsal_zero_copy_t  tx_zero_copy
    sf_wifi_nsal_zero_copy_t  rx_zero_copy
    uint8_t *  p_tx_packet_buffer
} sf_wifi_nsal_cfg_t
```

12.1.510.1 tx_zero_copy

`sf_wifi_nsal_zero_copy_t::tx_zero_copy`

Brief description

Transmit path zero copy support.

12.1.510.2 rx_zero_copy

`sf_wifi_nsal_zero_copy_t::rx_zero_copy`

Brief description

Receive path zero copy support.

12.1.510.3 p_tx_packet_buffer

`uint8_t* sf_wifi_nsal_cfg_t::p_tx_packet_buffer`

Brief description

Pointer to Tx buffer used to consolidate data from chained NetX packets.

12.1.511 sf_wifi_onchip_stack_api_t

```
typedef struct{
    ssp_err_t(*  open)(sf_wifi_onchip_stack_ctrl_t *p_ctrl,
sf_wifi_onchip_stack_cfg_t const *const p_cfg)
    ssp_err_t(*  close)(sf_wifi_onchip_stack_ctrl_t *const p_ctrl)
    ssp_err_t(*  ipAddressCfg)(sf_wifi_onchip_stack_ctrl_t *const p_ctrl,
sf_wifi_onchip_stack_ip_cfg_t *const p_cfg)
    ssp_err_t(*  dhcpServerStart)(sf_wifi_onchip_stack_ctrl_t *const p_ctrl,
sf_wifi_ip_addr_t const *const p_start_ip, sf_wifi_ip_addr_t const *const
p_end_ip)
    ssp_err_t(*  dhcpServerStop)(sf_wifi_onchip_stack_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} sf_wifi_onchip_stack_api_t
```

12.1.512 sf_wifi_onchip_stack_cfg_t

```
typedef struct{
    sf_wifi_instance_t const  *  p_lower_lvl_wifi
    void *  p_extend
} sf_wifi_onchip_stack_cfg_t
```

12.1.512.1 p_lower_lvl_wifi

`sf_wifi_instance_t::p_lower_lvl_wifi`

Brief description

Pointer to SF WiFi instance.

12.1.512.2 p_extend

`void* sf_wifi_onchip_stack_cfg_t::p_extend`

Brief description

Extended configuration.

12.1.513 sf_wifi_onchip_stack_ctrl_t

```
typedef struct{
    sf_wifi_instance_t *  p_lower_lvl_wifi
} sf_wifi_onchip_stack_ctrl_t
```

12.1.513.1 p_lower_lvl_wifi

`sf_wifi_instance_t::p_lower_lvl_wifi`

Brief description

Pointer to SF WiFi instance.

Detailed description

Storage for information needed for each WiFi device driver in the system.

12.1.514 sf_wifi_onchip_stack_instance_t

```
typedef struct{
    sf_wifi_onchip_stack_ctrl_t *  p_ctrl
    sf_wifi_onchip_stack_cfg_t const  *  p_cfg
```

```
    sf_wifi_onchip_stack_api_t const  *  p_api
} sf_wifi_onchip_stack_instance_t
```

12.1.514.1 p_ctrl

sf_wifi_onchip_stack_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.514.2 p_cfg

sf_wifi_onchip_stack_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.514.3 p_api

sf_wifi_onchip_stack_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.515 sf_wifi_onchip_stack_ip_cfg_t

```
typedef struct{
    sf_wifi_onchip_stack_ip_addr_mode_t  ip_addr_mode
    sf_wifi_ip_addr_t  ip_addr
    sf_wifi_ip_addr_t  netmask
    sf_wifi_ip_addr_t  gateway
} sf_wifi_onchip_stack_ip_cfg_t
```

12.1.515.1 ip_addr_mode

sf_wifi_onchip_stack_ip_addr_mode_t::ip_addr_mode

Brief description

Addressing mode.

12.1.515.2 ip_addr

sf_wifi_ip_addr_t::ip_addr

Brief description

Interface IP address.

12.1.515.3 netmask

`sf_wifi_ip_addr_t::netmask`

Brief description

Interface netmask.

12.1.515.4 gateway

`sf_wifi_ip_addr_t::gateway`

Brief description

Interface Gateway.

12.1.516 sf_wifi_provisioning_t

```
typedef struct{
    sf_wifi_interface_mode_t  mode
    uint8_t  channel
    uint8_t  ssid[SF_WIFI_SSID_LENGTH]
    sf_wifi_security_type_t  security
    sf_wifi_encryption_type_t  encryption
    uint8_t  key[SF_WIFI_SECURITY_KEY_LENGTH]
    void(*  p_callback)(sf_wifi_callback_args_t *p_args)
} sf_wifi_provisioning_t
```

12.1.516.1 mode

`sf_wifi_interface_mode_t::mode`

Brief description

Select AP or Client mode.

12.1.516.2 channel

`uint8_t sf_wifi_provisioning_t::channel`

Brief description

RF Channel number.

12.1.516.3 ssid

`uint8_t sf_wifi_provisioning_t::ssid[SF_WIFI_SSID_LENGTH]`

Brief description

SSID.

12.1.516.4 security

`sf_wifi_security_type_t::security`

Brief description

Security type.

12.1.516.5 encryption

`sf_wifi_encryption_type_t::encryption`

Brief description

Encryption type.

12.1.516.6 key

`uint8_t sf_wifi_provisioning_t::key[SF_WIFI_SECURITY_KEY_LENGTH]`

Brief description

Pre-shared key.

12.1.516.7 p_callback

`void(* sf_wifi_provisioning_t::p_callback) (sf_wifi_callback_args_t *p_args)`

Brief description

Pointer to AP Link UP/Down callback function.

12.1.517 sf_wifi_scan_t

```
typedef struct{
    sf_wifi_interface_hw_mode_t  hw_mode
    uint8_t  rssi
    uint8_t  ssid[SF_WIFI_SSID_LENGTH]
    uint8_t  bssid[SF_WIFI_MAC_ADDR_LENGTH]
    uint8_t  channel
    sf_wifi_security_type_t  security
    sf_wifi_encryption_type_t  encryption
    sf_wifi_bss_type_t  bss_type
} sf_wifi_scan_t
```

12.1.517.1 hw_mode

`sf_wifi_interface_hw_mode_t::hw_mode`

Brief description

Hardware mode 802.11a/b/g/n.

12.1.517.2 rssi

`uint8_t sf_wifi_scan_t::rssi`

Brief description

Signal Strength.

12.1.517.3 ssid

`uint8_t sf_wifi_scan_t::ssid[SF_WIFI_SSID_LENGTH]`

Brief description

SSID name.

12.1.517.4 bssid

`uint8_t sf_wifi_scan_t::bssid[SF_WIFI_MAC_ADDR_LENGTH]`

Brief description

Basic Service Set Identification (i.e. MAC address of Access Point)

12.1.517.5 channel

`uint8_t sf_wifi_scan_t::channel`

Brief description

Radio channel that the AP beacon was received on.

12.1.517.6 security

`sf_wifi_security_type_t::security`

Brief description

Security type.

12.1.517.7 encryption

`sf_wifi_encryption_type_t::encryption`

Brief description

Encryption type.

12.1.517.8 bss_type

`sf_wifi_bss_type_t::bss_type`

Brief description

Network type.

12.1.518 sf_wifi_stats_t

```
typedef struct{
    uint32_t  rx_pkts
    uint32_t  tx_pkts
    uint32_t  tx_err
} sf_wifi_stats_t
```

12.1.518.1 rx_pkts

```
uint32_t sf_wifi_stats_t::rx_pkts
```

Brief description

Packets received successfully.

12.1.518.2 tx_pkts

```
uint32_t sf_wifi_stats_t::tx_pkts
```

Brief description

Packets transmitted successfully.

12.1.518.3 tx_err

```
uint32_t sf_wifi_stats_t::tx_err
```

Brief description

Transmit errors.

12.1.519 slcdc_api_t

```
typedef struct{
    ssp_err_t(*  open)(slcdc_ctrl_t *const p_ctrl, slcdc_cfg_t const *const
p_cfg)
    ssp_err_t(*  write)(slcdc_ctrl_t *const p_ctrl, slcdc_size_t const
start_segment, slcdc_size_t const *const p_data, slcdc_size_t const segment_count)
    ssp_err_t(*  modify)(slcdc_ctrl_t *const p_ctrl, slcdc_size_t const segment,
slcdc_size_t const data_mask, slcdc_size_t const data)
    ssp_err_t(*  start)(slcdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  stop)(slcdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  contrastIncrease)(slcdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  contrastDecrease)(slcdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  setdisplayArea)(slcdc_ctrl_t *const p_ctrl, slcdc_display_area_t
const display_area)
    ssp_err_t(*  close)(slcdc_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} slcdc_api_t
```

12.1.520 slcdc_cfg_t

```
typedef struct{
    slcdc_display_clock_t  slcdc_clock
    slcdc_clk_div_t  slcdc_clock_setting
    slcdc_bias_method_t  bias_method
    slcdc_time_slice_t  time_slice
    slcdc_wave_form_t  wave_form
    slcdc_drive_volt_gen_t  drive_volt_gen
} slcdc_cfg_t
```

12.1.520.1 slcdc_clock

`slcdc_display_clock_t::slcdc_clock`

Brief description

LCD clock source (LCDSCKSEL)

12.1.520.2 slcdc_clock_setting

`slcdc_clk_div_t::slcdc_clock_setting`

Brief description

LCD clock setting (LCDC0)

12.1.520.3 bias_method

`slcdc_bias_method_t::bias_method`

Brief description

LCD display bias method select (LBAS bit).

12.1.520.4 time_slice

`slcdc_time_slice_t::time_slice`

Brief description

Time slice of LCD display select (LDTY bit)

12.1.520.5 wave_form

`slcdc_wave_form_t::wave_form`

Brief description

LCD display waveform select (LWAVE bit).

12.1.520.6 drive_volt_gen

`slcdc_drive_volt_gen_t::drive_volt_gen`

Brief description

LCD Drive Voltage Generator Select (MDSTET bit).

12.1.521 slcdc_instance_ctrl_t

```
typedef struct{
    slcdc_display_state_t  state
    slcdc_cfg_t  info
    void const  *  p_context
    R_LCD_Type *  p_reg
} slcdc_instance_ctrl_t
```

12.1.521.1 state

`slcdc_display_state_t::state`

Brief description

Status of SLCD module.

12.1.521.2 info

`slcdc_cfg_t::info`

Brief description

SLCDC config info.

12.1.521.3 p_context

`void const* ::p_context`

Brief description

Pointer to the higher level device context.

12.1.521.4 p_reg

`R_LCD_Type* ::p_reg`

Brief description

Pointer to register base address.

12.1.522 slcdc_instance_t

```
typedef struct{
    slcdc_ctrl_t *  p_ctrl
```

```
    slcdc_cfg_t const  *  p_cfg
    slcdc_api_t const  *  p_api
} slcdc_instance_t
```

12.1.522.1 p_ctrl

`slcdc_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.522.2 p_cfg

`slcdc_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.522.3 p_api

`slcdc_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.523 sockaddr

```
typedef struct{
    short  sin_family
    unsigned short  sin_port
    struct in_addr  sin_addr
    char  sin_zero[8]
    USHORT  sa_family
    UCHAR  sa_data[14]
} sockaddr
```

12.1.523.1 sin_family

`short sockaddr::sin_family`

Brief description

Address family.

12.1.523.2 sin_port

`unsigned short sockaddr::sin_port`

Brief description

Port number.

12.1.523.3 sin_addr

`in_addr::sin_addr`

Brief description

IP Address.

12.1.523.4 sin_zero

`char sockaddr::sin_zero`

Brief description

zero this if you want to

Detailed description

Zero this if you want to.

12.1.523.5 sa_family

`USHORT sockaddr::sa_family`

12.1.523.6 sa_data

`UCHAR sockaddr::sa_data`

12.1.524 sockaddr_in

```
typedef struct{
    uint16_t  sin_family
    uint16_t  sin_port
    struct in_addr  sin_addr
    int8_t  sin_zero[8]
    USHORT  sin_family
    USHORT  sin_port
    CHAR  sin_zero[8]
} sockaddr_in
```

12.1.524.1 sin_family

`USHORT sockaddr_in::sin_family`

Brief description

Internet Protocol (AF_INET)

12.1.524.2 sin_port

```
USHORT sockaddr_in::sin_port
```

Brief description

Address port (16 bits)

12.1.524.3 sin_addr

```
in_addr::sin_addr
```

Brief description

Internet address (32 bits)

12.1.524.4 sin_zero

```
CHAR sockaddr_in::sin_zero
```

Brief description

Not used.

Detailed description

Not used structure member.

12.1.524.5 sin_family

```
USHORT sockaddr_in::sin_family
```

12.1.524.6 sin_port

```
USHORT sockaddr_in::sin_port
```

12.1.524.7 sin_zero

```
CHAR sockaddr_in::sin_zero[8]
```

12.1.525 spi_api_t

```
typedef struct{
    ssp_err_t(*  open)(spi_ctrl_t *p_ctrl, spi_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(spi_ctrl_t *const p_ctrl, void const *p_dest, uint32_t
const length, spi_bit_width_t const bit_width)
    ssp_err_t(*  write)(spi_ctrl_t *const p_ctrl, void const *p_src, uint32_t
const length, spi_bit_width_t const bit_width)
    ssp_err_t(*  writeRead)(spi_ctrl_t *const p_ctrl, void const *p_src, void
const *p_dest, uint32_t const length, spi_bit_width_t const bit_width)
    ssp_err_t(*  close)(spi_ctrl_t *const p_ctrl)
```

```
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} spi_api_t
```

12.1.526 spi_callback_args_t

```
typedef struct{
    uint32_t  channel
    spi_event_t  event
    void const  *  p_context
} spi_callback_args_t
```

12.1.526.1 channel

`uint32_t spi_callback_args_t::channel`

Brief description

Device channel number.

12.1.526.2 event

`spi_event_t::event`

Brief description

Event code.

12.1.526.3 p_context

`void const* spi_callback_args_t::p_context`

Brief description

Context provided to user during callback.

12.1.527 spi_cfg_t

```
typedef struct{
    uint8_t  channel
    uint8_t  rxi_ipl
    uint8_t  txi_ipl
    uint8_t  tei_ipl
    uint8_t  eri_ipl
    spi_mode_t  operating_mode
    spi_clk_phase_t  clk_phase
    spi_clk_polarity_t  clk_polarity
    spi_mode_fault_t  mode_fault
    spi_bit_order_t  bit_order
    uint32_t  bitrate
    transfer_instance_t const  *  p_transfer_tx
```

```
    transfer_instance_t const  *  p_transfer_rx
    void(*  p_callback)(spi_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} spi_cfg_t
```

12.1.527.1 channel

`uint8_t spi_cfg_t::channel`

Brief description

Channel number to be used.

12.1.527.2 rxi_ipl

`uint8_t spi_cfg_t::rxi_ipl`

Brief description

Receive interrupt priority.

12.1.527.3 txi_ipl

`uint8_t spi_cfg_t::txi_ipl`

Brief description

Transmit interrupt priority.

12.1.527.4 tei_ipl

`uint8_t spi_cfg_t::tei_ipl`

Brief description

Transmit end interrupt priority.

12.1.527.5 eri_ipl

`uint8_t spi_cfg_t::eri_ipl`

Brief description

Error interrupt priority.

12.1.527.6 operating_mode

`spi_mode_t::operating_mode`

Brief description

Select master or slave operating mode.

12.1.527.7 clk_phase

`spi_clk_phase_t::clk_phase`

Brief description

Data sampling on odd or even clock edge.

12.1.527.8 clk_polarity

`spi_clk_polarity_t::clk_polarity`

Brief description

Clock level when idle.

12.1.527.9 mode_fault

`spi_mode_fault_t::mode_fault`

Brief description

Mode fault error (master/slave conflict) flag.

12.1.527.10 bit_order

`spi_bit_order_t::bit_order`

Brief description

Select to transmit MSB/LSB first.

12.1.527.11 bitrate

`uint32_t spi_cfg_t::bitrate`

Brief description

Bits Per Second.

12.1.527.12 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Brief description

To use SPI DTC/DMA write transfer, link a DTC/DMA instance here. Set to NULL if unused.

12.1.527.13 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Brief description

To use SPI DTC/DMA read transfer, link a DTC/DMA instance here. Set to NULL if unused.

12.1.527.14 p_callback

```
void(* spi_cfg_t::p_callback) (spi_callback_args_t *p_args)
```

Brief description

Pointer to user callback function.

12.1.527.15 p_context

```
void const* spi_cfg_t::p_context
```

Brief description

User defined context passed to callback function.

12.1.527.16 p_extend

```
void const* spi_cfg_t::p_extend
```

Brief description

Extended SPI hardware dependent configuration.

12.1.528 spi_instance_t

```
typedef struct{
    spi_ctrl_t *  p_ctrl
    spi_cfg_t const  *  p_cfg
    spi_api_t const  *  p_api
} spi_instance_t
```

12.1.528.1 p_ctrl

```
spi_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.528.2 p_cfg

```
spi_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.528.3 p_api

```
spi_api_t::p_api
```

Brief description

Pointer to the API structure for this instance.

12.1.529 spi_on_rspi_cfg_t

```
typedef struct{
    rspi_operation_t  rspi_clksyn
    rspi_communication_t  rspi_comm
    rspi_ssl_polarity_t  ssl_polarity
    rspi_loopback_t  loopback
    rspi_mosi_idle_t  mosi_idle
    rspi_parity_t  parity
    rspi_ssl_select_t  ssl_select
    rspi_ssl_level_keep_t  ssl_level_keep
    rspi_clock_delay_t  clock_delay
    rspi_ssl_negation_delay_t  ssl_neg_delay
    rspi_access_delay_t  access_delay
} spi_on_rspi_cfg_t
```

12.1.529.1 rspi_clksyn

`rspi_operation_t::rspi_clksyn`

Detailed description

Select spi or clock syn mode operation

12.1.529.2 rspi_comm

`rspi_communication_t::rspi_comm`

Detailed description

Select full-duplex or transmit-only communication

12.1.529.3 ssl_polarity

`rspi_ssl_polarity_t::ssl_polarity`

Detailed description

Select SSLn signal polarity

12.1.529.4 loopback

`rspi_loopback_t::loopback`

Detailed description

Select loopback1 and loopback2

12.1.529.5 mosi_idle

`rspi_mosi_idle_t::mosi_idle`

Detailed description

Select mosi idle fixed value and selection

12.1.529.6 parity

`rspi_parity_t::parity`

Detailed description

Select parity and enable/disable parity

12.1.529.7 ssl_select

`rspi_ssl_select_t::ssl_select`

Detailed description

Select which slave to use;0-SSL0;1-SSL1;2-SSL2;3-SSL3

12.1.529.8 ssl_level_keep

`rspi_ssl_level_keep_t::ssl_level_keep`

Detailed description

Select SSL level after transfer completion;0-negate;1-keeps

12.1.529.9 clock_delay

`rspi_clock_delay_t::clock_delay`

Detailed description

Select clock delay from 0 to 7

12.1.529.10 ssl_neg_delay

`rspi_ssl_negation_delay_t::ssl_neg_delay`

Detailed description

Select Slave elect negation delay from 0 to 7

12.1.529.11 access_delay

`rspi_access_delay_t::access_delay`

Detailed description

Select next access delay from 0 to 7

12.1.530 ssi_instance_ctrl_t

```
typedef struct{
    void(*  p_callback)(i2s_callback_args_t *p_args)
    void const  *  p_context
    R_SSI0_Type *  p_reg
    timer_instance_t const  *  p_timer
    transfer_instance_t const  *  p_transfer_tx
    transfer_instance_t const  *  p_transfer_rx
    uint32_t const  *  p_tx_src
    uint32_t  tx_src_bytes
    uint32_t *  p_rx_dest
    uint32_t  rx_dest_bytes
    uint32_t  sampling_freq_hz
    uint8_t  channel
    uint8_t  fifo_access_bytes
    uint8_t  stop_requested_tx
    uint8_t  stop_requested_rx
    uint8_t  tx_in_progress
    uint8_t  tx_in_use
    uint8_t  rx_in_use
    uint8_t  zeros_written
    IRQn_Type  txi_irq
    IRQn_Type  rxi_irq
    IRQn_Type  int_irq
    uint32_t  open
} ssi_instance_ctrl_t
```

12.1.530.1 p_callback

```
void(* ::p_callback) ( *p_args)
```

Detailed description

Callback provided when an I2S ISR occurs. NULL indicates no CPU interrupt.

12.1.530.2 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in i2s_callback_args_t.

12.1.530.3 p_reg

```
R_SSI0_Type* ::p_reg
```

Brief description

Pointer to SSI register base address.

12.1.530.4 p_timer

`timer_instance_t::p_timer`

Brief description

Timer used to generate audio clock.

12.1.530.5 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Brief description

Transfer used for hardware acceleration during write.

12.1.530.6 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Brief description

Transfer used for hardware acceleration during read.

12.1.530.7 p_tx_src

`uint32_t const* ::p_tx_src`

Detailed description

Source buffer pointer used to fill hardware FIFO from transmit ISR.

12.1.530.8 tx_src_bytes

`uint32_t ::tx_src_bytes`

Detailed description

Size of source buffer used to fill hardware FIFO from transmit ISR.

12.1.530.9 p_rx_dest

`uint32_t* ::p_rx_dest`

Detailed description

Destination buffer pointer used to fill from hardware FIFO in receive ISR.

12.1.530.10 rx_dest_bytes

`uint32_t ::rx_dest_bytes`

Detailed description

Size of destination buffer used to fill from hardware FIFO in receive ISR.

12.1.530.11 sampling_freq_hz

```
uint32_t ::sampling_freq_hz
```

Brief description

Sampling frequency in Hertz.

12.1.530.12 channel

```
uint8_t ::channel
```

Brief description

Channel number.

12.1.530.13 fifo_access_bytes

```
uint8_t ::fifo_access_bytes
```

Brief description

Byte access to FIFO.

12.1.530.14 stop_requested_tx

```
uint8_t ::stop_requested_tx
```

Brief description

Stops I2S after transmit is complete.

12.1.530.15 stop_requested_rx

```
uint8_t ::stop_requested_rx
```

Brief description

Stops I2S after receive is complete.

12.1.530.16 tx_in_progress

```
uint8_t ::tx_in_progress
```

Brief description

True if a transmit transfer is in progress.

12.1.530.17 tx_in_use

```
uint8_t ::tx_in_use
```

Brief description

True if a transmission is in progress.

12.1.530.18 rx_in_use

`uint8_t ::rx_in_use`

Brief description

True if a reception is in progress.

12.1.530.19 zeros_written

`uint8_t ::zeros_written`

Brief description

True if zeros have been transmitted.

12.1.530.20 txi_irq

`IRQn_Type ::txi_irq`

Brief description

Transmit IRQ number.

12.1.530.21 rxi_irq

`IRQn_Type ::rxi_irq`

Brief description

Receive IRQ number.

12.1.530.22 int_irq

`IRQn_Type ::int_irq`

Brief description

Idle/Error IRQ number.

12.1.530.23 open

`uint32_t ::open`

Brief description

Whether or not this control block is initialized.

12.1.531 st_sf_audio_playback_instance_ctrl

```
typedef struct{
    uint32_t  open
    TX_THREAD *  p_owner
    void(*  p_callback)(sf_message_callback_args_t *p_args)
```

```
    uint8_t  class_instance
    uint32_t  samples_remaining
    uint32_t  index
    uint32_t  end
    sf_audio_playback_data_t *  p_data[2]
    sf_audio_playback_status_t  status
    sf_audio_playback_common_instance_ctrl_t *  p_common_ctrl
} st_sf_audio_playback_instance_ctrl
```

12.1.531.1 open

`uint32_t ::open`

Brief description

Used to determine if driver is initialized.

12.1.531.2 p_owner

`TX_THREAD::p_owner`

Detailed description

Pointer to thread that began the stream at this index. Used to ensure multiple threads don't interleave data on the same stream.

12.1.531.3 p_callback

`void(* ::p_callback) ( *p_args)`

Detailed description

Callback called when playback of a buffer passed to start is complete.

12.1.531.4 class_instance

`uint8_t ::class_instance`

Brief description

Class instance used to identify the stream to the messaging framework.

12.1.531.5 samples_remaining

`uint32_t ::samples_remaining`

Brief description

Internal state of data samples remaining for this stream.

12.1.531.6 index

`uint32_t ::index`

Brief description

Internal state of current data index for this stream.

12.1.531.7 end

`uint32_t ::end`

Brief description

Used to track completion of looped playback.

12.1.531.8 p_data

`sf_audio_playback_data_t::p_data`

Brief description

Audio data read from queue.

12.1.531.9 status

`sf_audio_playback_status_t::status`

Brief description

Status of current stream.

12.1.531.10 p_common_ctrl

`sf_audio_playback_common_instance_ctrl_t::p_common_ctrl`

Detailed description

Pointer to the hardware control block used by this stream.

12.1.532 st_sf_ble_prf_htp_meas_intv_val_t

```
typedef struct{
    uint16_t  conhdl
    uint16_t  intv
} st_sf_ble_prf_htp_meas_intv_val_t
```

12.1.532.1 conhdl

`uint16_t st_sf_ble_prf_htp_meas_intv_val_t::conhdl`

Brief description

Connection handle.

12.1.532.2 intv

`uint16_t st_sf_ble_prf_htp_meas_intv_val_t::intv`

Brief description

Interval value.

12.1.533 st_sf_touch_ctsu_slider_hdl

```
typedef struct{
    uint32_t  open
    sf_touch_ctsu_slider_id_t  id
    sf_touch_ctsu_slider_state_t  state
    sf_slider_type_t  type
    uint32_t  num_slider_channels
    ctsu_channel_pair_t const  *  p_slider_channels
    int32_t const  *  p_normalization
    int32_t *  p_channel_average
    uint32_t *  p_offset
    uint16_t *  p_slider_scount
    uint16_t *  p_slider_baseline
    int32_t *  p_slider_delta
    uint32_t  position
    int32_t  prev_sum
    int32_t  max_slider_value
    ssp_err_t(*  p_update)(sf_touch_ctsu_slider_hdl_t *const hdl,
sf_touch_ctsu_instance_t const *const p_lower_lvl_touch_framework, uint32_t
*p_pos, sf_touch_ctsu_slider_state_t *const p_state)
    uint64_t  bit_mask[SF_TOUCH_CTSU_SLIDER_BIT_MASK_ARRAY_SIZE]
    uint16_t  slider_norm_max
    int32_t  slider_threshold
    int32_t  channel_average_weight
    int32_t  prev_sum_weight
    int32_t  cutoff
} st_sf_touch_ctsu_slider_hdl
```

12.1.533.1 open

`uint32_t ::open`

Detailed description

Indicate that slider has been opened

12.1.533.2 id

`sf_touch_ctsu_slider_id_t::id`

Detailed description

Unique identifier for slider.

12.1.533.3 state

`sf_touch_ctsu_slider_state_t::state`

Detailed description

Represents the current state of the slider.

12.1.533.4 type

`sf_slider_type_t::type`

Detailed description

Linear or circular (wheel)

12.1.533.5 num_slider_channels

```
uint32_t ::num_slider_channels
```

12.1.533.6 p_slider_channels

`ctsu_channel_pair_t::p_slider_channels`

Detailed description

Define the channel/channel pairs which make up a slider.

12.1.533.7 p_normalization

```
int32_t const* ::p_normalization
```

12.1.533.8 p_channel_average

```
int32_t* ::p_channel_average
```

12.1.533.9 p_offset

```
uint32_t* ::p_offset
```

12.1.533.10 p_slider_scount

```
uint16_t* ::p_slider_scount
```

12.1.533.11 p_slider_baseline

```
uint16_t* ::p_slider_baseline
```

12.1.533.12 p_slider_delta

```
int32_t* ::p_slider_delta
```

12.1.533.13 position

```
uint32_t ::position
```

Detailed description

Calculated position.

12.1.533.14 prev_sum

```
int32_t ::prev_sum
```

Detailed description

Used to store the running average of previous sum in position calculation

12.1.533.15 max_slider_value

```
int32_t ::max_slider_value
```

Detailed description

Maximum slider value, must be greater than 0; Minimum is always 0

12.1.533.16 p_update

```
(* ::p_update) ( *const hdl, const *const p_lower_lvl_touch_framework, uint32_t
*p_pos, *const p_state)
```

Detailed description

Function to calculate position of touch.

12.1.533.17 bit_mask

```
uint64_t ::bit_mask[SF_TOUCH_CTSU_SLIDER_BIT_MASK_ARRAY_SIZE]
```

Detailed description

Bit mask to be used in update function

12.1.533.18 slider_norm_max

```
uint16_t ::slider_norm_max
```

Detailed description

Individual slider settings that can be modified by the user. Should be the same as in sf_slider_on_ctsu_cfg_tTOUCH_SLIDER_CFG_NORM_MAX

12.1.533.19 slider_threshold

```
int32_t ::slider_threshold
```

Detailed description

Touch threshold. Ensure that value is greater than 0

12.1.533.20 channel_average_weight

```
int32_t ::channel_average_weight
```

Detailed description

Weight of running average of counts for each channel

12.1.533.21 prev_sum_weight

```
int32_t ::prev_sum_weight
```

Detailed description

Weight of running average of previous sum in position calculation, must be greater than 0

12.1.533.22 cutoff

```
int32_t ::cutoff
```

Detailed description

Defines how far below prev_sum running average to get before "SF_TOUCH_SLIDER_STATE_RELEASED" detected

12.1.534 tcon_func_t

```
typedef struct{
    void(*  tcon_select)(R_GLCDC_Type *p_glcd_reg, glcd_tcon_signal_select_t)
    void(*  invert)(R_GLCDC_Type *p_glcd_reg)
} tcon_func_t
```

12.1.534.1 tcon_select

```
void(* ::tcon_select) (R_GLCDC_Type *p_glcd_reg, )
```

12.1.534.2 invert

```
void(* ::invert) (R_GLCDC_Type *p_glcd_reg)
```

12.1.535 tdes_api_t

```
typedef struct{
    uint32_t(*  open)(tdes_ctrl_t *const p_ctrl, tdes_cfg_t const *const p_cfg)
```

```
    uint32_t(*  encrypt)(tdes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  decrypt)(tdes_ctrl_t *const p_ctrl, const uint32_t *p_key,
uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
    uint32_t(*  close)(tdes_ctrl_t *const p_ctrl)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} tdes_api_t
```

12.1.536 tdes_cfg_t

```
typedef struct{
    crypto_api_t const  *  p_crypto_api
} tdes_cfg_t
```

12.1.536.1 p_crypto_api

crypto_api_t::p_crypto_api

Brief description

pointer to crypto engine api

12.1.537 tdes_ctrl_t

```
typedef struct{
    crypto_ctrl_t  crypto_ctrl
    crypto_api_t const  *  p_crypto_api
} tdes_ctrl_t
```

12.1.537.1 crypto_ctrl

crypto_ctrl_t::crypto_ctrl

Brief description

pointer to crypto control structure

12.1.537.2 p_crypto_api

crypto_api_t::p_crypto_api

Brief description

pointer to crypto engine API

12.1.538 tdes_instance_t

```
typedef struct{
    tdes_ctrl_t *  p_ctrl
```

```
    tdes_cfg_t const  *  p_cfg
    tdes_api_t const  *  p_api
} tdes_instance_t
```

12.1.538.1 p_ctrl

tdes_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.538.2 p_cfg

tdes_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.538.3 p_api

tdes_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.539 timer_api_t

```
typedef struct{
    ssp_err_t(*  open)(timer_ctrl_t *const p_ctrl, timer_cfg_t const *const
p_cfg)
    ssp_err_t(*  stop)(timer_ctrl_t *const p_ctrl)
    ssp_err_t(*  start)(timer_ctrl_t *const p_ctrl)
    ssp_err_t(*  reset)(timer_ctrl_t *const p_ctrl)
    ssp_err_t(*  counterGet)(timer_ctrl_t *const p_ctrl, timer_size_t *const
p_value)
    ssp_err_t(*  periodSet)(timer_ctrl_t *const p_ctrl, timer_size_t const
period, timer_unit_t const unit)
    ssp_err_t(*  dutyCycleSet)(timer_ctrl_t *const p_ctrl, timer_size_t const
duty_cycle, timer_pwm_unit_t const duty_cycle_unit, uint8_t const pin)
    ssp_err_t(*  infoGet)(timer_ctrl_t *const p_ctrl, timer_info_t *const p_info)
    ssp_err_t(*  close)(timer_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
} timer_api_t
```

12.1.540 timer_callback_args_t

```
typedef struct{
    void const  *  p_context
    timer_event_t  event
} timer_callback_args_t
```

12.1.540.1 p_context

```
void const* timer_callback_args_t::p_context
```

Detailed description

Placeholder for user data. Set in open function in timer_cfg_t.

12.1.540.2 event

```
timer_event_t::event
```

Brief description

The event can be used to identify what caused the callback (overflow or error).

12.1.541 timer_cfg_t

```
typedef struct{
    timer_mode_t  mode
    uint32_t  period
    timer_unit_t  unit
    timer_size_t  duty_cycle
    timer_pwm_unit_t  duty_cycle_unit
    uint8_t  channel
    uint8_t  irq_ipl
    bool  autostart
    void(*  p_callback)(timer_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} timer_cfg_t
```

12.1.541.1 mode

```
timer_mode_t::mode
```

Brief description

Select enumerated value from timer_mode_t.

12.1.541.2 period

```
uint32_t timer_cfg_t::period
```

Detailed description

Defines when the timer should expire. For a free running counter, set to TIMER_MAX_CLOCK with unit TIMER_UNIT_PERIOD_RAW_COUNTS

12.1.541.3 unit

`timer_unit_t::unit`

Brief description

Units of period.

12.1.541.4 duty_cycle

`timer_size_t::duty_cycle`

Brief description

Duty cycle in units duty_cycle_unit.

12.1.541.5 duty_cycle_unit

`timer_pwm_unit_t::duty_cycle_unit`

Brief description

Units of duty_cycle.

12.1.541.6 channel

`uint8_t timer_cfg_t::channel`

Detailed description

Select a channel corresponding to the channel number of the hardware.

12.1.541.7 irq_ipl

`uint8_t timer_cfg_t::irq_ipl`

Brief description

Timer interrupt priority.

12.1.541.8 autostart

`bool timer_cfg_t::autostart`

Detailed description

Whether to start during Open call or not. True: Start during open. False: Don't start during open.

12.1.541.9 p_callback

```
void(* timer_cfg_t::p_callback) (timer_callback_args_t *p_args)
```

Detailed description

Callback provided when a timer ISR occurs. Set to NULL for no CPU interrupt.

12.1.541.10 p_context

```
void const* timer_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in timer_callback_args_t.

12.1.541.11 p_extend

```
void const* timer_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.542 timer_info_t

```
typedef struct{
    timer_direction_t  count_direction
    uint32_t  clock_frequency
    timer_size_t  period_counts
    timer_status_t  status
    elc_event_t  elc_event
} timer_info_t
```

12.1.542.1 count_direction

```
timer_direction_t::count_direction
```

Brief description

Clock counting direction of the timer resource.

12.1.542.2 clock_frequency

```
uint32_t timer_info_t::clock_frequency
```

Brief description

Clock frequency of the timer resource.

12.1.542.3 period_counts

```
timer_size_t::period_counts
```

Brief description

Time in clock counts until timer will expire.

12.1.542.4 status

`timer_status_t::status`

12.1.542.5 elc_event

`elc_event_t::elc_event`

Brief description

ELC event associated with the count operation of the timer resource.

12.1.543 timer_instance_t

```
typedef struct{
    timer_ctrl_t *  p_ctrl
    timer_cfg_t const  *  p_cfg
    timer_api_t const  *  p_api
} timer_instance_t
```

12.1.543.1 p_ctrl

`timer_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.543.2 p_cfg

`timer_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.543.3 p_api

`timer_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.544 timer_on_agt_cfg_t

```
typedef struct{
    agt_count_source_t  count_source
    bool  agto_output_enabled
    bool  agtio_output_enabled
    bool  output_inverted
    bool  agtoa_output_enable
    bool  agtob_output_enable
} timer_on_agt_cfg_t
```

12.1.544.1 count_source

`agt_count_source_t::count_source`

Brief description

AGT channel clock source. Valid values are: AGT_CLOCK_PCLKB, AGT_CLOCK_LOCO, AGT_CLOCK_FSUB.

12.1.544.2 agto_output_enabled

`bool ::agto_output_enabled`

Brief description

AGTO pin is enabled for output compare (true, false)

12.1.544.3 agtio_output_enabled

`bool ::agtio_output_enabled`

Brief description

AGTIO pin is enabled for output compare (true, false)

12.1.544.4 output_inverted

`bool ::output_inverted`

Brief description

Output inverted (true, false)

12.1.544.5 agtoa_output_enable

`bool ::agtoa_output_enable`

Brief description

Enable comparator A output pin (true, false)

12.1.544.6 agtob_output_enable

```
bool ::agtob_output_enable
```

Brief description

Enable comparator B output pin (true, false)

12.1.545 timer_on_gpt_cfg_t

```
typedef struct{
    gpt_output_pin_t  gtioca
    gpt_output_pin_t  gtiocb
} timer_on_gpt_cfg_t
```

12.1.545.1 gtioca

```
gpt_output_pin_t::gtioca
```

Brief description

Configuration for GPT I/O pin A.

12.1.545.2 gtiocb

```
gpt_output_pin_t::gtiocb
```

Brief description

Configuration for GPT I/O pin B.

12.1.546 transfer_api_t

```
typedef struct{
    ssp_err_t(*  open)(transfer_ctrl_t *const p_ctrl, transfer_cfg_t const *const
p_cfg)
    ssp_err_t(*  reset)(transfer_ctrl_t *const p_ctrl, void const *p_src, void
*p_dest, uint16_t const num_transfers)
    ssp_err_t(*  enable)(transfer_ctrl_t *const p_ctrl)
    ssp_err_t(*  disable)(transfer_ctrl_t *const p_ctrl)
    ssp_err_t(*  start)(transfer_ctrl_t *const p_ctrl, transfer_start_mode_t
mode)
    ssp_err_t(*  stop)(transfer_ctrl_t *const p_ctrl)
    ssp_err_t(*  infoGet)(transfer_ctrl_t *const p_ctrl, transfer_properties_t
*const p_info)
    ssp_err_t(*  close)(transfer_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_version)
    ssp_err_t(*  blockReset)(transfer_ctrl_t *const p_ctrl, void const *p_src,
void *p_dest, uint16_t const length, transfer_size_t size, uint16_t const
num_transfers)
} transfer_api_t
```

12.1.547 transfer_callback_args_t

```
typedef struct{
    void const  *  p_context
} transfer_callback_args_t
```

12.1.547.1 p_context

```
void const* transfer_callback_args_t::p_context
```

Brief description

Placeholder for user data. Set in r_transfer_t::open function in transfer_cfg_t.

12.1.548 transfer_cfg_t

```
typedef struct{
    transfer_info_t *  p_info
    elc_event_t  activation_source
    bool  auto_enable
    uint8_t  irq_ipl
    void(*  p_callback)(transfer_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} transfer_cfg_t
```

12.1.548.1 p_info

```
transfer_info_t::p_info
```

Detailed description

Pointer to transfer configuration options. If using chain transfer (DTC only), this can be a pointer to an array of chained transfers that will be completed in order.

12.1.548.2 activation_source

```
elc_event_t::activation_source
```

Detailed description

Select which event will trigger the transfer. ***NOTE:*** *Select ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 for software activation. When using DTC, these events may only be used once each. DMAC uses internal software start when either of these events are selected.*

12.1.548.3 auto_enable

```
bool transfer_cfg_t::auto_enable
```

Detailed description

Select whether the transfer should be enabled after open.

12.1.548.4 irq_ipl

```
uint8_t transfer_cfg_t::irq_ipl
```

Detailed description

Interrupt priority level. ***ATTENTION:*** *Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.*

12.1.548.5 p_callback

```
void(* transfer_cfg_t::p_callback) (transfer_callback_args_t *p_args)
```

Detailed description

Callback for transfer end interrupt. Set to NULL for no CPU interrupt.

ATTENTION: *Unsupported for DTC except when ELC software events are used. DTC transfers trigger the interrupt associated with the activation source.*

12.1.548.6 p_context

```
void const* transfer_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user p_callback in transfer_callback_args_t.

12.1.548.7 p_extend

```
void const* transfer_cfg_t::p_extend
```

Brief description

Extension parameter for hardware specific settings.

12.1.549 transfer_info_t

```
typedef struct{
    uint32_t  __pad0__
    uint32_t  __pad1__
    transfer_addr_mode_t  dest_addr_mode
    transfer_repeat_area_t  repeat_area
    transfer_irq_t  irq
    transfer_chain_mode_t  chain_mode
    uint32_t  __pad2__
    transfer_addr_mode_t  src_addr_mode
    transfer_size_t  size
    transfer_mode_t  mode
    void const  *volatile  p_src
    void *volatile  p_dest
    uint16_t  num_blocks
```

```
    uint16_t  length
} transfer_info_t
```

12.1.549.1 __pad0__

`uint32_t transfer_info_t::__pad0__`

12.1.549.2 __pad1__

`uint32_t transfer_info_t::__pad1__`

12.1.549.3 dest_addr_mode

`transfer_addr_mode_t::dest_addr_mode`

Detailed description

Select what happens to destination pointer after each transfer.

12.1.549.4 repeat_area

`transfer_repeat_area_t::repeat_area`

Detailed description

Select to repeat source or destination area, unused in TRANSFER_MODE_NORMAL.

12.1.549.5 irq

`transfer_irq_t::irq`

Detailed description

Select if interrupts should occur after each individual transfer or after the completion of all planned transfers.

12.1.549.6 chain_mode

`transfer_chain_mode_t::chain_mode`

Detailed description

Select when the chain transfer ends.

12.1.549.7 __pad2__

`uint32_t transfer_info_t::__pad2__`

12.1.549.8 src_addr_mode

`transfer_addr_mode_t::src_addr_mode`

Detailed description

Select what happens to source pointer after each transfer.

12.1.549.9 size

`transfer_size_t::size`

Detailed description

Select number of bytes to transfer at once. length.

12.1.549.10 mode

`transfer_mode_t::mode`

Detailed description

Select mode from transfer_mode_t.

12.1.549.11 p_src

`void const* volatile transfer_info_t::p_src`

Brief description

Source pointer.

12.1.549.12 p_dest

`void* volatile transfer_info_t::p_dest`

Brief description

Destination pointer.

12.1.549.13 num_blocks

`volatile uint16_t transfer_info_t::num_blocks`

Detailed description

Number of blocks to transfer when using TRANSFER_MODE_BLOCK (both DTC an DMAC) and TRANSFER_MODE_REPEAT (DMAC only), unused in other modes.

12.1.549.14 length

`volatile uint16_t transfer_info_t::length`

Detailed description

Length of each transfer. Range limited for TRANSFER_MODE_BLOCK and TRANSFER_MODE_REPEAT, see HAL driver for details.

12.1.550 transfer_instance_t

```
typedef struct{
    transfer_ctrl_t *  p_ctrl
    transfer_cfg_t const  *  p_cfg
    transfer_api_t const  *  p_api
} transfer_instance_t
```

12.1.550.1 p_ctrl

transfer_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.550.2 p_cfg

transfer_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.550.3 p_api

transfer_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.551 transfer_on_dmac_cfg_t

```
typedef struct{
    uint8_t  channel
} transfer_on_dmac_cfg_t
```

12.1.551.1 channel

uint8_t ::channel

Brief description

Channel number, does not apply to all HAL drivers.

12.1.552 transfer_properties_t

```
typedef struct{
    uint32_t  transfer_length_max
    uint16_t  transfer_length_remaining
```

```
    bool  in_progress
} transfer_properties_t
```

12.1.552.1 transfer_length_max

`uint32_t transfer_properties_t::transfer_length_max`

Brief description

Maximum number of transfers.

12.1.552.2 transfer_length_remaining

`uint16_t transfer_properties_t::transfer_length_remaining`

Brief description

Number of transfers remaining.

12.1.552.3 in_progress

`bool transfer_properties_t::in_progress`

Brief description

Whether or not this transfer is in progress.

12.1.553 trng_api_t

```
typedef struct{
    uint32_t(*  open)(trng_ctrl_t *const p_ctrl, trng_cfg_t const *const p_cfg)
    uint32_t(*  read)(trng_ctrl_t *const p_ctrl, uint32_t *const p_rngbuf,
uint32_t nwords)
    uint32_t(*  close)(trng_ctrl_t *const p_ctrl)
    uint32_t(*  versionGet)(ssp_version_t *const p_version)
} trng_api_t
```

12.1.554 trng_cfg_t

```
typedef struct{
    crypto_api_t const  *  p_crypto_api
    uint32_t  nattempts
} trng_cfg_t
```

12.1.554.1 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto API

12.1.554.2 nattempts

`uint32_t trng_cfg_t::nattempts`

Brief description

number of retries when a continuous test failure occurs

12.1.555 trng_ctrl_t

```
typedef struct{
    uint32_t  nattempts
    crypto_ctrl_t *  p_crypto_ctrl
    crypto_api_t const  *  p_crypto_api
    uint32_t  prevbuf[TRNG_REGISTER_SIZE_WORDS]
    uint32_t  currbuf[TRNG_REGISTER_SIZE_WORDS]
} trng_ctrl_t
```

12.1.555.1 nattempts

`uint32_t trng_ctrl_t::nattempts`

Brief description

number of retries

12.1.555.2 p_crypto_ctrl

`crypto_ctrl_t::p_crypto_ctrl`

Brief description

pointer to crypto control structure

12.1.555.3 p_crypto_api

`crypto_api_t::p_crypto_api`

Brief description

pointer to crypto-engine API

12.1.555.4 prevbuf

`uint32_t trng_ctrl_t::prevbuf[TRNG_REGISTER_SIZE_WORDS]`

Brief description

previous random data

12.1.555.5 currbuf

`uint32_t trng_ctrl_t::currbuf[TRNG_REGISTER_SIZE_WORDS]`

Brief description

current random data

12.1.556 trng_instance_t

```
typedef struct{
    trng_ctrl_t *  p_ctrl
    trng_cfg_t const  *  p_cfg
    trng_api_t const  *  p_api
} trng_instance_t
```

12.1.556.1 p_ctrl

`trng_ctrl_t::p_ctrl`

Brief description

Pointer to the control structure for this instance.

12.1.556.2 p_cfg

`trng_cfg_t::p_cfg`

Brief description

Pointer to the configuration structure for this instance.

12.1.556.3 p_api

`trng_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.557 uart_api_t

```
typedef struct{
    ssp_err_t(*  open)(uart_ctrl_t *const p_ctrl, uart_cfg_t const *const p_cfg)
    ssp_err_t(*  read)(uart_ctrl_t *const p_ctrl, uint8_t const *const p_dest,
uint32_t const bytes)
    ssp_err_t(*  write)(uart_ctrl_t *const p_ctrl, uint8_t const *const p_src,
uint32_t const bytes)
    ssp_err_t(*  baudSet)(uart_ctrl_t *const p_ctrl, uint32_t const baudrate)
    ssp_err_t(*  infoGet)(uart_ctrl_t *const p_ctrl, uart_info_t *const p_info)
    ssp_err_t(*  close)(uart_ctrl_t *const p_ctrl)
    ssp_err_t(*  versionGet)(ssp_version_t *p_version)
} uart_api_t
```

12.1.558 uart_callback_args_t

```
typedef struct{
    uint32_t  channel
    uart_event_t  event
    uint32_t  data
    void const  *  p_context
} uart_callback_args_t
```

12.1.558.1 channel

`uint32_t uart_callback_args_t::channel`

Brief description

Device channel number.

12.1.558.2 event

`uart_event_t::event`

Brief description

Event code.

12.1.558.3 data

`uint32_t uart_callback_args_t::data`

Detailed description

Contains the next character received for the events UART_EVENT_RX_CHAR, UART_EVENT_ERR_PARITY, UART_EVENT_ERR_FRAMING, or UART_EVENT_ERR_OVERFLOW. Otherwise unused.

12.1.558.4 p_context

`void const* uart_callback_args_t::p_context`

Brief description

Context provided to user during callback.

12.1.559 uart_cfg_t

```
typedef struct{
    uint8_t  channel
    uint32_t  baud_rate
    uart_data_bits_t  data_bits
    uart_parity_t  parity
    uart_stop_bits_t  stop_bits
    bool  ctsrts_en
    uint8_t  rxi_ipl
```

```
    uint8_t  txi_ipl
    uint8_t  tei_ipl
    uint8_t  eri_ipl
    transfer_instance_t const  *  p_transfer_rx
    transfer_instance_t const  *  p_transfer_tx
    void(*  p_callback)(uart_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} uart_cfg_t
```

12.1.559.1 channel

`uint8_t uart_cfg_t::channel`

Brief description

Select a channel corresponding to the channel number of the hardware.

12.1.559.2 baud_rate

`uint32_t uart_cfg_t::baud_rate`

Brief description

Baud rate, i.e. 9600, 19200, 115200.

12.1.559.3 data_bits

`uart_data_bits_t::data_bits`

Brief description

Data bit length (8 or 7 or 9)

12.1.559.4 parity

`uart_parity_t::parity`

Brief description

Parity type (none or odd or even)

12.1.559.5 stop_bits

`uart_stop_bits_t::stop_bits`

Brief description

Stop bit length (1 or 2)

12.1.559.6 ctsrts_en

`bool uart_cfg_t::ctsrts_en`

Brief description

CTS/RTS hardware flow control enable.

12.1.559.7 rxi_ipl

`uint8_t uart_cfg_t::rxi_ipl`

Brief description

Receive interrupt priority.

12.1.559.8 txi_ipl

`uint8_t uart_cfg_t::txi_ipl`

Brief description

Transmit interrupt priority.

12.1.559.9 tei_ipl

`uint8_t uart_cfg_t::tei_ipl`

Brief description

Transmit end interrupt priority.

12.1.559.10 eri_ipl

`uint8_t uart_cfg_t::eri_ipl`

Brief description

Error interrupt priority.

12.1.559.11 p_transfer_rx

`transfer_instance_t::p_transfer_rx`

Detailed description

Optional transfer instance used to receive multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the read API is limited to one byte at a time.

12.1.559.12 p_transfer_tx

`transfer_instance_t::p_transfer_tx`

Detailed description

Optional transfer instance used to send multiple bytes without interrupts. Set to NULL if unused. If NULL, the number of bytes allowed in the write APIs is limited to one byte at a time.

12.1.559.13 p_callback

```
void(* uart_cfg_t::p_callback) (uart_callback_args_t *p_args)
```

Brief description

Pointer to callback function.

12.1.559.14 p_context

```
void const* uart_cfg_t::p_context
```

Brief description

User defined context passed into callback function.

12.1.559.15 p_extend

```
void const* uart_cfg_t::p_extend
```

Brief description

UART hardware dependent configuration.

12.1.560 uart_info_t

```
typedef struct{
    uint32_t  write_bytes_max
    uint32_t  read_bytes_max
} uart_info_t
```

12.1.560.1 write_bytes_max

```
uint32_t uart_info_t::write_bytes_max
```

Detailed description

Maximum bytes that can be written at this time. Only applies if p_transfer_tx is not NULL.

12.1.560.2 read_bytes_max

```
uint32_t uart_info_t::read_bytes_max
```

Detailed description

Maximum bytes that are available to read at one time. Only applies if p_transfer_rx is not NULL.

12.1.561 uart_instance_t

```
typedef struct{
    uart_ctrl_t *  p_ctrl
    uart_cfg_t const  *  p_cfg
```

```
    uart_api_t const  *  p_api
} uart_instance_t
```

12.1.561.1 p_ctrl

uart_ctrl_t::p_ctrl

Brief description

Pointer to the control structure for this instance.

12.1.561.2 p_cfg

uart_cfg_t::p_cfg

Brief description

Pointer to the configuration structure for this instance.

12.1.561.3 p_api

uart_api_t::p_api

Brief description

Pointer to the API structure for this instance.

12.1.562 uart_on_sci_cfg_t

```
typedef struct{
    sci_clk_src_t  clk_src
    bool  baudclk_out
    bool  rx_edge_start
    bool  noisecancel_en
    sci_uart_rx_fifo_trigger_t  rx_fifo_trigger
    void(*  p_extpin_ctrl)(uint32_t channel, uint32_t level)
    bool  bitrate_modulation
} uart_on_sci_cfg_t
```

12.1.562.1 clk_src

sci_clk_src_t::clk_src

Brief description

Use SCI_CLK_SRC_INT/EXT8X/EXT16X.

12.1.562.2 baudclk_out

bool ::baudclk_out

Brief description

Baud rate clock output enable.

12.1.562.3 rx_edge_start

```
bool ::rx_edge_start
```

Brief description

Start reception on falling edge.

12.1.562.4 noisecancel_en

```
bool ::noisecancel_en
```

Brief description

Noise cancel enable.

12.1.562.5 rx_fifo_trigger

```
sci_uart_rx_fifo_trigger_t::rx_fifo_trigger
```

Detailed description

Receive FIFO trigger level, unused if channel has no FIFO or if DTC is used.

12.1.562.6 p_extpin_ctrl

```
void(* ::p_extpin_ctrl) (uint32_t channel, uint32_t level)
```

Detailed description

Pointer to the user callback function to control external GPIO pin control used as RTS signal.

Table 1918:Parameters

Name	Direction	Description
channel	in	The UART channel used.
level	in	When level is 0, assert RTS. When level is 1, deassert RTS.

Parameter channel

`uint32_t`

Parameter level

`uint32_t`

12.1.562.7 bitrate_modulation

```
bool ::bitrate_modulation
```

Brief description

Bitrate Modulation Function enable or disable.

12.1.563 wdt_api_t

```
typedef struct{
    ssp_err_t(*  cfgGet)(wdt_ctrl_t *const p_ctrl, wdt_cfg_t *const p_cfg)
    ssp_err_t(*  open)(wdt_ctrl_t *const p_ctrl, wdt_cfg_t const *const p_cfg)
    ssp_err_t(*  refresh)(wdt_ctrl_t *const p_ctrl)
    ssp_err_t(*  statusGet)(wdt_ctrl_t *const p_ctrl, wdt_status_t *const
p_status)
    ssp_err_t(*  statusClear)(wdt_ctrl_t *const p_ctrl, const wdt_status_t
status)
    ssp_err_t(*  counterGet)(wdt_ctrl_t *const p_ctrl, uint32_t *const p_count)
    ssp_err_t(*  timeoutGet)(wdt_ctrl_t *const p_ctrl, wdt_timeout_values_t
*const p_timeout)
    ssp_err_t(*  versionGet)(ssp_version_t *const p_data)
} wdt_api_t
```

12.1.564 wdt_callback_args_t

```
typedef struct{
    void const  *  p_context
} wdt_callback_args_t
```

12.1.564.1 p_context

```
void const* wdt_callback_args_t::p_context
```

Brief description

Placeholder for user data. Set in open function in wdt_cfg_t.

12.1.565 wdt_cfg_t

```
typedef struct{
    wdt_start_mode_t  start_mode
    bool  autostart
    wdt_timeout_t  timeout
    wdt_clock_division_t  clock_division
    wdt_window_start_t  window_start
    wdt_window_end_t  window_end
    wdt_reset_control_t  reset_control
    wdt_stop_control_t  stop_control
```

```
    void(*  p_callback)(wdt_callback_args_t *p_args)
    void const  *  p_context
    void const  *  p_extend
} wdt_cfg_t
```

12.1.565.1 start_mode

`wdt_start_mode_t::start_mode`

Brief description

The expected start mode for the WDT.

12.1.565.2 autostart

`bool wdt_cfg_t::autostart`

Detailed description

When true the WDT is started as part of its configuration (register start mode). If false the WDT needs to be started manually by calling the refresh API.

12.1.565.3 timeout

`wdt_timeout_t::timeout`

Brief description

Timeout period.

12.1.565.4 clock_division

`wdt_clock_division_t::clock_division`

Brief description

Clock divider.

12.1.565.5 window_start

`wdt_window_start_t::window_start`

Brief description

Refresh permitted window start position.

12.1.565.6 window_end

`wdt_window_end_t::window_end`

Brief description

Refresh permitted window end position.

12.1.565.7 reset_control

`wdt_reset_control_t::reset_control`

Brief description

Select NMI or reset generated on underflow.

12.1.565.8 stop_control

`wdt_stop_control_t::stop_control`

Brief description

Select whether counter operates in sleep mode.

12.1.565.9 p_callback

```
void(* wdt_cfg_t::p_callback) (wdt_callback_args_t *p_args)
```

Brief description

Callback provided when a WDT NMI ISR occurs.

12.1.565.10 p_context

```
void const* wdt_cfg_t::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

12.1.565.11 p_extend

```
void const* wdt_cfg_t::p_extend
```

Brief description

Placeholder for user extension.

12.1.566 wdt_instance_ctrl_t

```
typedef struct{
    bool  wdt_open
    void const  *  p_context
    R_WDT_Type *  p_reg
    void(*  p_callback)(wdt_callback_args_t *p_args)
} wdt_instance_ctrl_t
```

12.1.566.1 wdt_open

```
bool ::wdt_open
```

Detailed description

Indicates whether the open() API has been successfully called.

12.1.566.2 p_context

```
void const* ::p_context
```

Detailed description

Placeholder for user data. Passed to the user callback in wdt_callback_args_t.

12.1.566.3 p_reg

```
R_WDT_Type* ::p_reg
```

Brief description

Pointer to register base address.

12.1.566.4 p_callback

```
void(* ::p_callback) ( *p_args)
```

Brief description

Callback provided when a WDT NMI ISR occurs.

12.1.567 wdt_instance_t

```
typedef struct{
    wdt_ctrl_t *  p_ctrl
    wdt_cfg_t const  *  p_cfg
    wdt_api_t const  *  p_api
} wdt_instance_t
```

12.1.567.1 p_ctrl

```
wdt_ctrl_t::p_ctrl
```

Brief description

Pointer to the control structure for this instance.

12.1.567.2 p_cfg

```
wdt_cfg_t::p_cfg
```

Brief description

Pointer to the configuration structure for this instance.

12.1.567.3 p_api

`wdt_api_t::p_api`

Brief description

Pointer to the API structure for this instance.

12.1.568 wdt_timeout_values_t

```
typedef struct{
    uint32_t  clock_frequency_hz
    uint32_t  timeout_clocks
} wdt_timeout_values_t
```

12.1.568.1 clock_frequency_hz

`uint32_t wdt_timeout_values_t::clock_frequency_hz`

Brief description

Frequency of watchdog clock after divider.

12.1.568.2 timeout_clocks

`uint32_t wdt_timeout_values_t::timeout_clocks`

Brief description

Timeout period in units of watchdog clock ticks.

12.2 Interface Functions

12.2.1 Interface: adc_api_t

Description: ADC Interface

Function name	Definition
.open	ssp_err_t(* adc_api_t::open) (adc_ctrl_t *const p_ctrl, adc_cfg_t const *const p_cfg)
.scanCfg	ssp_err_t(* adc_api_t::scanCfg) (adc_ctrl_t *const p_ctrl, adc_channel_cfg_t const *const p_channel_cfg)

Function name	Definition
.scanStart	ssp_err_t(* adc_api_t::scanStart) (adc_ctrl_t *const p_ctrl)
.scanStop	ssp_err_t(* adc_api_t::scanStop) (adc_ctrl_t *const p_ctrl)
.scanStatusGet	ssp_err_t(* adc_api_t::scanStatusGet) (adc_ctrl_t *const p_ctrl)
.read	ssp_err_t(* adc_api_t::read) (adc_ctrl_t *const p_ctrl, adc_register_t const reg_id, adc_data_size_t *const p_data)
.sampleStateCountSet	ssp_err_t(* adc_api_t::sampleStateCountSet) (adc_ctrl_t *const p_ctrl, adc_sample_state_t *p_sample)
.close	ssp_err_t(* adc_api_t::close) (adc_ctrl_t *const p_ctrl)
.infoGet	ssp_err_t(* adc_api_t::infoGet) (adc_ctrl_t *const p_ctrl, adc_info_t *const p_adc_info)
.versionGet	ssp_err_t(* adc_api_t::versionGet) (ssp_version_t *const p_version)

12.2.2 Interface: aes_api_t

Description: AES Interface

Function name	Definition
.open	uint32_t(* aes_api_t::open) (aes_ctrl_t *const p_ctrl, aes_cfg_t const *const p_cfg)
.createKey	uint32_t(* aes_api_t::createKey) (aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t *p_key)
.encrypt	uint32_t(* aes_api_t::encrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.addAdditionalAuthenticationData	uint32_t(* aes_api_t::addAdditionalAuthenticationData) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source)

Function name	Definition
.encryptFinal	uint32_t(* aes_api_t::encryptFinal) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t input_num_words, uint32_t *p_source, uint32_t output_num_words, uint32_t *p_dest)
.decrypt	uint32_t(* aes_api_t::decrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t imaxcnt, uint32_t *p_source, uint32_t *p_dest)
.setGcmTag	uint32_t(* aes_api_t::setGcmTag) (aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t *p_source)
.getGcmTag	uint32_t(* aes_api_t::getGcmTag) (aes_ctrl_t *const p_ctrl, uint32_t num_words, uint32_t *p_dest)
.close	uint32_t(* aes_api_t::close) (aes_ctrl_t *const p_ctrl)
.zeroPaddingEncrypt	uint32_t(* aes_api_t::zeroPaddingEncrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t *p_dest)
.zeroPaddingDecrypt	uint32_t(* aes_api_t::zeroPaddingDecrypt) (aes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_bytes, uint32_t *p_source, uint32_t *p_dest)
.versionGet	uint32_t(* aes_api_t::versionGet) (ssp_version_t *const p_version)

12.2.3 Interface: cac_api_t

Description: CAC Interface

Function name	Definition
.open	ssp_err_t(* cac_api_t::open) (cac_ctrl_t *const p_ctrl, cac_cfg_t const *const p_cfg)
.read	ssp_err_t(* cac_api_t::read) (cac_ctrl_t *const p_ctrl, uint8_t *const p_status, uint16_t *const p_counter)
.close	ssp_err_t(* cac_api_t::close) (cac_ctrl_t *const p_ctrl)
.stopMeasurement	ssp_err_t(* cac_api_t::stopMeasurement) (cac_ctrl_t *const p_ctrl)

Function name	Definition
.startMeasurement	ssp_err_t(* cac_api_t::startMeasurement) (cac_ctrl_t *const p_ctrl)
.reset	ssp_err_t(* cac_api_t::reset) (cac_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* cac_api_t::versionGet) (ssp_version_t *p_version)

12.2.4 Interface: can_api_t

Description: CAN Interface

Function name	Definition
.open	ssp_err_t(* can_api_t::open) (can_ctrl_t *const p_ctrl, can_cfg_t const *const p_cfg)
.read	ssp_err_t(* can_api_t::read) (can_ctrl_t *const p_ctrl, uint32_t mailbox, can_frame_t *const p_frame)
.write	ssp_err_t(* can_api_t::write) (can_ctrl_t *const p_ctrl, uint32_t mailbox, can_frame_t *const p_frame)
.close	ssp_err_t(* can_api_t::close) (can_ctrl_t *const p_ctrl)
.control	ssp_err_t(* can_api_t::control) (can_ctrl_t *const p_ctrl, can_command_t const command, void *p_data)
.infoGet	ssp_err_t(* can_api_t::infoGet) (can_ctrl_t *const p_ctrl, can_info_t *const p_info)
.versionGet	ssp_err_t(* can_api_t::versionGet) (ssp_version_t *const p_version)

12.2.5 Interface: cgc_api_t

Description: CGC Interface

Function name	Definition
.init	ssp_err_t(* cgc_api_t::init) (void)

Function name	Definition
.clocksCfg	ssp_err_t(* cgc_api_t::clocksCfg) (cgc_clocks_cfg_t const *const p_clock_cfg)
.clockStart	ssp_err_t(* cgc_api_t::clockStart) (cgc_clock_t clock_source, cgc_clock_cfg_t *p_clock_cfg)
.clockStop	ssp_err_t(* cgc_api_t::clockStop) (cgc_clock_t clock_source)
.systemClockSet	ssp_err_t(* cgc_api_t::systemClockSet) (cgc_clock_t clock_source, cgc_system_clock_cfg_t const *const p_clock_cfg)
.systemClockGet	ssp_err_t(* cgc_api_t::systemClockGet) (cgc_clock_t *p_clock_source, cgc_system_clock_cfg_t *p_set_clock_cfg)
.systemClockFreqGet	ssp_err_t(* cgc_api_t::systemClockFreqGet) (cgc_system_clocks_t clock, uint32_t *p_freq_hz)
.clockCheck	ssp_err_t(* cgc_api_t::clockCheck) (cgc_clock_t clock_source)
.oscStopDetect	ssp_err_t(* cgc_api_t::oscStopDetect) (void(*p_callback)(cgc_callback_args_t *p_args), bool enable)
.oscStopStatusClear	ssp_err_t(* cgc_api_t::oscStopStatusClear) (void)
.busClockOutCfg	ssp_err_t(* cgc_api_t::busClockOutCfg) (cgc_bclockout_dividers_t divider)
.busClockOutEnable	ssp_err_t(* cgc_api_t::busClockOutEnable) (void)
.busClockOutDisable	ssp_err_t(* cgc_api_t::busClockOutDisable) (void)
.clockOutCfg	ssp_err_t(* cgc_api_t::clockOutCfg) (cgc_clock_t clock, cgc_clockout_dividers_t divider)
.clockOutEnable	ssp_err_t(* cgc_api_t::clockOutEnable) (void)
.clockOutDisable	ssp_err_t(* cgc_api_t::clockOutDisable) (void)
.lcdClockCfg	ssp_err_t(* cgc_api_t::lcdClockCfg) (cgc_clock_t clock)
.lcdClockEnable	ssp_err_t(* cgc_api_t::lcdClockEnable) (void)

Function name	Definition
.lcdClockDisable	ssp_err_t(* cgc_api_t::lcdClockDisable) (void)
.sdramClockOutEnable	ssp_err_t(* cgc_api_t::sdramClockOutEnable) (void)
.sdramClockOutDisable	ssp_err_t(* cgc_api_t::sdramClockOutDisable) (void)
.usbClockCfg	ssp_err_t(* cgc_api_t::usbClockCfg) (cgc_usb_clock_div_t divider)
.systickUpdate	ssp_err_t(* cgc_api_t::systickUpdate) (uint32_t period_count, cgc_systick_period_units_t units)
.versionGet	ssp_err_t(* cgc_api_t::versionGet) (ssp_version_t *p_version)

12.2.6 Interface: crc_api_t

Description: CRC Interface

Function name	Definition
.open	ssp_err_t(* crc_api_t::open) (crc_ctrl_t *const p_ctrl, crc_cfg_t const *const p_cfg)
.close	ssp_err_t(* crc_api_t::close) (crc_ctrl_t *const p_ctrl)
.crcResultGet	ssp_err_t(* crc_api_t::crcResultGet) (crc_ctrl_t *const p_ctrl, uint32_t *crc_result)
.snoopEnable	ssp_err_t(* crc_api_t::snoopEnable) (crc_ctrl_t *const p_ctrl, uint32_t crc_seed)
.snoopDisable	ssp_err_t(* crc_api_t::snoopDisable) (crc_ctrl_t *const p_ctrl)
.snoopCfg	ssp_err_t(* crc_api_t::snoopCfg) (crc_ctrl_t *const p_ctrl, crc_snoop_cfg_t *const p_snoop_cfg)
.calculate	ssp_err_t(* crc_api_t::calculate) (crc_ctrl_t *const p_ctrl, void *p_input_buffer, uint32_t num_bytes, uint32_t crc_seed, uint32_t *p_crc_result)
.versionGet	ssp_err_t(* crc_api_t::versionGet) (ssp_version_t *version)

12.2.7 Interface: crypto_api_t

Description: Crypto Interface

Function name	Definition
.open	uint32_t(* crypto_api_t::open) (crypto_ctrl_t *const p_ctrl, crypto_cfg_t const *const p_cfg)
.close	uint32_t(* crypto_api_t::close) (crypto_ctrl_t *const p_ctrl)
.interfaceGet	uint32_t(* crypto_api_t::interfaceGet) (crypto_interface_get_param_t *const interface_info, void *const p_interface)
.statusGet	uint32_t(* crypto_api_t::statusGet) (uint32_t *p_status)
.versionGet	uint32_t(* crypto_api_t::versionGet) (ssp_version_t *const p_version)

12.2.8 Interface: ctsu_api_t

Description: CTSU Interface

Function name	Definition
.open	ssp_err_t(* ctsu_api_t::open) (ctsu_ctrl_t *p_ctrl, ctsu_cfg_t *p_cfg)
.close	ssp_err_t(* ctsu_api_t::close) (ctsu_ctrl_t *p_ctrl, ctsu_close_option_t opts)
.scan	ssp_err_t(* ctsu_api_t::scan) (ctsu_ctrl_t *p_ctrl)
.update	ssp_err_t(* ctsu_api_t::update) (ctsu_ctrl_t *p_ctrl)
.read	ssp_err_t(* ctsu_api_t::read) (ctsu_ctrl_t *p_ctrl, void *p_dest, ctsu_read_t opts, const ctsu_channel_pair_t *channels, const uint16_t count)
.versionGet	ssp_err_t(* ctsu_api_t::versionGet) (ssp_version_t *const p_version)

12.2.9 Interface: dac_api_t

Description: DAC Interface

Function name	Definition
.open	ssp_err_t(* dac_api_t::open) (dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg)
.close	ssp_err_t(* dac_api_t::close) (dac_ctrl_t *p_ctrl)
.write	ssp_err_t(* dac_api_t::write) (dac_ctrl_t *p_ctrl, dac_size_t value)
.start	ssp_err_t(* dac_api_t::start) (dac_ctrl_t *p_ctrl)
.stop	ssp_err_t(* dac_api_t::stop) (dac_ctrl_t *p_ctrl)
.versionGet	ssp_err_t(* dac_api_t::versionGet) (ssp_version_t *p_version)

12.2.10 Interface: display_api_t

Description: Display Interface

Function name	Definition
.open	ssp_err_t(* display_api_t::open) (display_ctrl_t *const p_ctrl, display_cfg_t const *const p_cfg)
.close	ssp_err_t(* display_api_t::close) (display_ctrl_t *const p_ctrl)
.start	ssp_err_t(* display_api_t::start) (display_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* display_api_t::stop) (display_ctrl_t *const p_ctrl)
.layerChange	ssp_err_t(* display_api_t::layerChange) (display_ctrl_t const *const p_ctrl, display_runtime_cfg_t const *const p_cfg, display_frame_layer_t frame)

Function name	Definition
.correction	ssp_err_t(* display_api_t::correction) (display_ctrl_t const *const p_ctrl, display_correction_t const *const p_param)
.clut	ssp_err_t(* display_api_t::clut) (display_ctrl_t const *const p_ctrl, display_clut_cfg_t const *const p_clut_cfg, display_frame_layer_t frame)
.statusGet	ssp_err_t(* display_api_t::statusGet) (display_ctrl_t const *const p_ctrl, display_status_t *const p_status)
.versionGet	ssp_err_t(* display_api_t::versionGet) (ssp_version_t *p_version)

12.2.11 Interface: doc_api_t

Description: DOC Interface

Function name	Definition
.open	ssp_err_t(* doc_api_t::open) (doc_ctrl_t *const p_ctrl, doc_cfg_t const *const p_cfg)
.close	ssp_err_t(* doc_api_t::close) (doc_ctrl_t *const p_ctrl)
.statusGet	ssp_err_t(* doc_api_t::statusGet) (doc_ctrl_t *const p_ctrl, doc_status_t *p_status)
.statusClear	ssp_err_t(* doc_api_t::statusClear) (doc_ctrl_t *const p_ctrl)
.write	ssp_err_t(* doc_api_t::write) (doc_ctrl_t *const p_ctrl, doc_data_t *const p_data)
.inputRegisterWrite	ssp_err_t(* doc_api_t::inputRegisterWrite) (doc_ctrl_t *const p_ctrl, doc_size_t data)
.versionGet	ssp_err_t(* doc_api_t::versionGet) (ssp_version_t *const p_version)

12.2.12 Interface: dsa_api_t

Description: DSA Interface

Function name	Definition
.open	uint32_t(* dsa_api_t::open) (dsa_ctrl_t *const p_ctrl, dsa_cfg_t const *const p_cfg)
.verify	uint32_t(* dsa_api_t::verify) (const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t *p_paddedHash)
.hashVerify	uint32_t(* dsa_api_t::hashVerify) (dsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t *p_paddedHash)
.sign	uint32_t(* dsa_api_t::sign) (const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_paddedHash, uint32_t *p_dest)
.hashSign	uint32_t(* dsa_api_t::hashSign) (dsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_paddedHash, uint32_t *p_dest)
.close	uint32_t(* dsa_api_t::close) (dsa_ctrl_t *const p_ctrl)
.versionGet	uint32_t(* dsa_api_t::versionGet) (ssp_version_t *const p_version)

12.2.13 Interface: elc_api_t

Description: Events and peripheral definitions

Function name	Definition
.init	ssp_err_t(* elc_api_t::init) (elc_cfg_t const *const p_cfg)
.softwareEventGenerate	ssp_err_t(* elc_api_t::softwareEventGenerate) (elc_software_event_t event_num)
.linkSet	ssp_err_t(* elc_api_t::linkSet) (elc_peripheral_t peripheral, elc_event_t signal)
.linkBreak	ssp_err_t(* elc_api_t::linkBreak) (elc_peripheral_t peripheral)

Function name	Definition
.enable	ssp_err_t(* elc_api_t::enable) (void)
.disable	ssp_err_t(* elc_api_t::disable) (void)
.versionGet	ssp_err_t(* elc_api_t::versionGet) (ssp_version_t *const p_version)

12.2.14 Interface: external_irq_api_t

Description: External IRQ Interface

Function name	Definition
.open	ssp_err_t(* external_irq_api_t::open) (external_irq_ctrl_t *const p_ctrl, external_irq_cfg_t const *const p_cfg)
.enable	ssp_err_t(* external_irq_api_t::enable) (external_irq_ctrl_t *const p_ctrl)
.disable	ssp_err_t(* external_irq_api_t::disable) (external_irq_ctrl_t *const p_ctrl)
.triggerSet	ssp_err_t(* external_irq_api_t::triggerSet) (external_irq_ctrl_t *const p_ctrl, external_irq_trigger_t const trigger)
.filterEnable	ssp_err_t(* external_irq_api_t::filterEnable) (external_irq_ctrl_t *const p_ctrl)
.filterDisable	ssp_err_t(* external_irq_api_t::filterDisable) (external_irq_ctrl_t *const p_ctrl)
.close	ssp_err_t(* external_irq_api_t::close) (external_irq_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* external_irq_api_t::versionGet) (ssp_version_t *const p_version)

12.2.15 Interface: flash_api_t

Description: Flash Interface

Function name	Definition
.open	ssp_err_t(* flash_api_t::open) (flash_ctrl_t *const p_ctrl, flash_cfg_t const *const p_cfg)
.write	ssp_err_t(* flash_api_t::write) (flash_ctrl_t *const p_ctrl, uint32_t const src_address, uint32_t const flash_address, uint32_t const num_bytes)
.read	ssp_err_t(* flash_api_t::read) (flash_ctrl_t *const p_ctrl, uint8_t *const p_dest_address, uint32_t const flash_address, uint32_t const num_bytes)
.erase	ssp_err_t(* flash_api_t::erase) (flash_ctrl_t *const p_ctrl, uint32_t const address, uint32_t const num_blocks)
.blankCheck	ssp_err_t(* flash_api_t::blankCheck) (flash_ctrl_t *const p_ctrl, uint32_t const address, uint32_t const num_bytes, flash_result_t *const p_blank_check_result)
.infoGet	ssp_err_t(* flash_api_t::infoGet) (flash_ctrl_t *const p_ctrl, flash_info_t *const p_info)
.close	ssp_err_t(* flash_api_t::close) (flash_ctrl_t *const p_ctrl)
.statusGet	ssp_err_t(* flash_api_t::statusGet) (flash_ctrl_t *const p_ctrl)
.accessWindowSet	ssp_err_t(* flash_api_t::accessWindowSet) (flash_ctrl_t *const p_ctrl, uint32_t const start_addr, uint32_t const end_addr)
.accessWindowClear	ssp_err_t(* flash_api_t::accessWindowClear) (flash_ctrl_t *const p_ctrl)
.reset	ssp_err_t(* flash_api_t::reset) (flash_ctrl_t *const p_ctrl)
.updateFlashClockFreq	ssp_err_t(* flash_api_t::updateFlashClockFreq) (flash_ctrl_t *const p_ctrl)
.startupAreaSelect	ssp_err_t(* flash_api_t::startupAreaSelect) (flash_ctrl_t *const p_ctrl, flash_startup_area_swap_t swap_type, bool is_temporary)

Function name	Definition
.versionGet	ssp_err_t(* flash_api_t::versionGet) (ssp_version_t *p_version)

12.2.16 Interface: fmi_api_t

Description: FMI Interface

Function name	Definition
.init	ssp_err_t(* fmi_api_t::init) (void)
.productInfoGet	ssp_err_t(* fmi_api_t::productInfoGet) (fmi_product_info_t **pp_product_info)
.uniqueIdGet	ssp_err_t(* fmi_api_t::uniqueIdGet) (fmi_unique_id_t *p_unique_id)
.productFeatureGet	ssp_err_t(* fmi_api_t::productFeatureGet) (ssp_feature_t const *const p_feature, fmi_feature_info_t *const p_info)
.eventInfoGet	ssp_err_t(* fmi_api_t::eventInfoGet) (ssp_feature_t const *const p_feature, ssp_signal_t signal, fmi_event_info_t *const p_info)
.versionGet	ssp_err_t(* fmi_api_t::versionGet) (ssp_version_t *const p_version)

12.2.17 Interface: hash_api_t

Description: HASH Algorithm Interface

Function name	Definition
.open	uint32_t(* hash_api_t::open) (hash_ctrl_t *const p_ctrl, hash_cfg_t const *const p_cfg)
.updateHash	uint32_t(* hash_api_t::updateHash) (const uint32_t *p_source, uint32_t num_words, uint32_t *p_dest)

Function name	Definition
.hashUpdate	uint32_t(* hash_api_t::hashUpdate) (hash_ctrl_t *const p_ctrl, const uint32_t *p_source, uint32_t num_words, uint32_t *p_dest)
.close	uint32_t(* hash_api_t::close) (hash_ctrl_t *const p_ctrl)
.versionGet	uint32_t(* hash_api_t::versionGet) (ssp_version_t *const p_version)

12.2.18 Interface: i2s_api_t

Description: I2S Interface

Function name	Definition
.open	ssp_err_t(* i2s_api_t::open) (i2s_ctrl_t *const p_ctrl, i2s_cfg_t const *const p_cfg)
.stop	ssp_err_t(* i2s_api_t::stop) (i2s_ctrl_t *const p_ctrl, i2s_dir_t const dir)
.mute	ssp_err_t(* i2s_api_t::mute) (i2s_ctrl_t *const p_ctrl, i2s_mute_t const mute_enable)
.write	ssp_err_t(* i2s_api_t::write) (i2s_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint16_t const bytes)
.read	ssp_err_t(* i2s_api_t::read) (i2s_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint16_t const bytes)
.writeRead	ssp_err_t(* i2s_api_t::writeRead) (i2s_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint8_t *const p_dest, uint16_t const bytes)
.infoGet	ssp_err_t(* i2s_api_t::infoGet) (i2s_ctrl_t *const p_ctrl, i2s_info_t *const p_info)
.close	ssp_err_t(* i2s_api_t::close) (i2s_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* i2s_api_t::versionGet) (ssp_version_t *const p_version)

12.2.19 Interface: input_capture_api_t

Description: Input Capture Interface

Function name	Definition
.open	ssp_err_t(* input_capture_api_t::open) (input_capture_ctrl_t *const p_ctrl, input_capture_cfg_t const *const p_cfg)
.disable	ssp_err_t(* input_capture_api_t::disable) (input_capture_ctrl_t const *const p_ctrl)
.enable	ssp_err_t(* input_capture_api_t::enable) (input_capture_ctrl_t const *const p_ctrl)
.infoGet	ssp_err_t(* input_capture_api_t::infoGet) (input_capture_ctrl_t const *const p_ctrl, input_capture_info_t *const p_info)
.lastCaptureGet	ssp_err_t(* input_capture_api_t::lastCaptureGet) (input_capture_ctrl_t const *const p_ctrl, input_capture_capture_t *const p_counter)
.close	ssp_err_t(* input_capture_api_t::close) (input_capture_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* input_capture_api_t::versionGet) (ssp_version_t *const p_version)

12.2.20 Interface: ioport_api_t

Description: I/O Port Interface

Function name	Definition
.init	ssp_err_t(* ioport_api_t::init) (const ioport_cfg_t *p_cfg)
.pinsCfg	ssp_err_t(* ioport_api_t::pinsCfg) (const ioport_cfg_t *p_cfg)
.pinCfg	ssp_err_t(* ioport_api_t::pinCfg) (ioport_port_pin_t pin, uint32_t cfg)

Function name	Definition
.pinDirectionSet	ssp_err_t(* ioport_api_t::pinDirectionSet) (ioport_port_pin_t pin, ioport_direction_t direction)
.pinEventInputRead	ssp_err_t(* ioport_api_t::pinEventInputRead) (ioport_port_pin_t pin, ioport_level_t *p_pin_event)
.pinEventOutputWrite	ssp_err_t(* ioport_api_t::pinEventOutputWrite) (ioport_port_pin_t pin, ioport_level_t pin_value)
.pinEthernetModeCfg	ssp_err_t(* ioport_api_t::pinEthernetModeCfg) (ioport_ethernet_channel_t channel, ioport_ethernet_mode_t mode)
.pinRead	ssp_err_t(* ioport_api_t::pinRead) (ioport_port_pin_t pin, ioport_level_t *p_pin_value)
.pinWrite	ssp_err_t(* ioport_api_t::pinWrite) (ioport_port_pin_t pin, ioport_level_t level)
.portDirectionSet	ssp_err_t(* ioport_api_t::portDirectionSet) (ioport_port_t port, ioport_size_t direction_values, ioport_size_t mask)
.portEventInputRead	ssp_err_t(* ioport_api_t::portEventInputRead) (ioport_port_t port, ioport_size_t *p_event_data)
.portEventOutputWrite	ssp_err_t(* ioport_api_t::portEventOutputWrite) (ioport_port_t port, ioport_size_t event_data, ioport_size_t mask_value)
.portRead	ssp_err_t(* ioport_api_t::portRead) (ioport_port_t port, ioport_size_t *p_port_value)
.portWrite	ssp_err_t(* ioport_api_t::portWrite) (ioport_port_t port, ioport_size_t value, ioport_size_t mask)
.versionGet	ssp_err_t(* ioport_api_t::versionGet) (ssp_version_t *p_data)

12.2.21 Interface: jpeg_decode_api_t

Description: JPEG Decode Interface

Function name	Definition
.open	ssp_err_t(* jpeg_decode_api_t::open) (jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_cfg_t const *const p_cfg)
.outputBufferSet	ssp_err_t(* jpeg_decode_api_t::outputBufferSet) (jpeg_decode_ctrl_t *const p_ctrl, void *p_buffer, uint32_t buffer_size)
.horizontalStrideSet	ssp_err_t(* jpeg_decode_api_t::horizontalStrideSet) (jpeg_decode_ctrl_t *const p_ctrl, uint32_t horizontal_stride)
.imageSubsampleSet	ssp_err_t(* jpeg_decode_api_t::imageSubsampleSet) (jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_subsample_t horizontal_subsample, jpeg_decode_subsample_t vertical_subsample)
.inputBufferSet	ssp_err_t(* jpeg_decode_api_t::inputBufferSet) (jpeg_decode_ctrl_t *const p_ctrl, void *p_buffer, uint32_t buffer_size)
.linesDecodedGet	ssp_err_t(* jpeg_decode_api_t::linesDecodedGet) (jpeg_decode_ctrl_t *const p_ctrl, uint32_t *const p_lines)
.imageSizeGet	ssp_err_t(* jpeg_decode_api_t::imageSizeGet) (jpeg_decode_ctrl_t *const p_ctrl, uint16_t *p_horizontal_size, uint16_t *p_vertical_size)
.statusGet	ssp_err_t(* jpeg_decode_api_t::statusGet) (jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_status_t *const p_status)
.close	ssp_err_t(* jpeg_decode_api_t::close) (jpeg_decode_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* jpeg_decode_api_t::versionGet) (ssp_version_t *p_version)
.pixelFormatGet	ssp_err_t(* jpeg_decode_api_t::pixelFormatGet) (jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_color_space_t *const p_color_space)

12.2.22 Interface: keymatrix_api_t

Description: Key Matrix Interface

Function name	Definition
.open	ssp_err_t(* keymatrix_api_t::open) (keymatrix_ctrl_t *const p_ctrl, keymatrix_cfg_t const *const p_cfg)
.enable	ssp_err_t(* keymatrix_api_t::enable) (keymatrix_ctrl_t *const p_ctrl)
.disable	ssp_err_t(* keymatrix_api_t::disable) (keymatrix_ctrl_t *const p_ctrl)
.triggerSet	ssp_err_t(* keymatrix_api_t::triggerSet) (keymatrix_ctrl_t *const p_ctrl, keymatrix_trigger_t const trigger)
.close	ssp_err_t(* keymatrix_api_t::close) (keymatrix_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* keymatrix_api_t::versionGet) (ssp_version_t *const p_version)

12.2.23 Interface: lpm_api_t

Description: Low Power Modes Interface

Function name	Definition
.init	ssp_err_t(* lpm_api_t::init) (lpm_cfg_t const *const p_cfg)
.mstpcrSet	ssp_err_t(* lpm_api_t::mstpcrSet) (uint32_t mstpcra_value, uint32_t mstpcrb_value, uint32_t mstpcrc_value, uint32_t mstpcrd_value)
.mstpcrGet	ssp_err_t(* lpm_api_t::mstpcrGet) (uint32_t *mstpcra_value, uint32_t *mstpcrb_value, uint32_t *mstpcrc_value, uint32_t *mstpcrd_value)
.moduleStop	ssp_err_t(* lpm_api_t::moduleStop) (lpm_mstp_t module)

Function name	Definition
.moduleStart	ssp_err_t(* lpm_api_t::moduleStart) (lpm_mstp_t module)
.operatingPowerModeSet	ssp_err_t(* lpm_api_t::operatingPowerModeSet) (lpm_operating_power_t power_mode, lpm_subosc_t subosc)
.snoozeEnable	ssp_err_t(* lpm_api_t::snoozeEnable) (lpm_snooze_rxd0_t rdx0_mode, lpm_snooze_dtc_t dtc_mode, lpm_snooze_request_t requests, lpm_snooze_end_t triggers)
.snoozeDisable	ssp_err_t(* lpm_api_t::snoozeDisable) (void)
.lowPowerCfg	ssp_err_t(* lpm_api_t::lowPowerCfg) (lpm_low_power_mode_t power_mode, lpm_output_port_enable_t output_port_enable, lpm_power_supply_t power_supply, lpm_io_port_t io_port_state)
.wupenSet	ssp_err_t(* lpm_api_t::wupenSet) (uint32_t wupen_value)
.wupenGet	ssp_err_t(* lpm_api_t::wupenGet) (uint32_t *wupen_value)
.deepStandbyCancelRequestEnable	ssp_err_t(* lpm_api_t::deepStandbyCancelRequestEnable) (lpm_deep_standby_t pin_signal, lpm_cancel_request_edge_t rising_falling)
.deepStandbyCancelRequestDisable	ssp_err_t(* lpm_api_t::deepStandbyCancelRequestDisable) (lpm_deep_standby_t pin_signal)
.lowPowerModeEnter	ssp_err_t(* lpm_api_t::lowPowerModeEnter) (void)
.versionGet	ssp_err_t(* lpm_api_t::versionGet) (ssp_version_t *const p_version)

12.2.24 Interface: lpmv2_api_t

Description: Low Power Modes V2 Interface

Function name	Definition
.init	ssp_err_t(* lpmv2_api_t::init) (void)
.lowPowerCfg	ssp_err_t(* lpmv2_api_t::lowPowerCfg) (lpmv2_cfg_t const *const p_cfg)
.lowPowerModeEnter	ssp_err_t(* lpmv2_api_t::lowPowerModeEnter) (void)
.versionGet	ssp_err_t(* lpmv2_api_t::versionGet) (ssp_version_t *const p_version)

12.2.25 Interface: lvd_api_t

Description: Low Voltage Detection Interface

Function name	Definition
.open	ssp_err_t(* lvd_api_t::open) (lvd_ctrl_t *const p_ctrl, lvd_cfg_t const *const p_cfg)
.statusGet	ssp_err_t(* lvd_api_t::statusGet) (lvd_ctrl_t *const p_ctrl, lvd_status_t *p_lvd_status)
.statusClear	ssp_err_t(* lvd_api_t::statusClear) (lvd_ctrl_t *const p_ctrl)
.close	ssp_err_t(* lvd_api_t::close) (lvd_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* lvd_api_t::versionGet) (ssp_version_t *const p_version)

12.2.26 Interface: pdc_api_t

Description: PDC Interface

Function name	Definition
.open	ssp_err_t(* pdc_api_t::open) (pdc_ctrl_t *const p_ctrl, pdc_cfg_t const *const p_cfg)
.close	ssp_err_t(* pdc_api_t::close) (pdc_ctrl_t *const p_ctrl)

Function name	Definition
.captureStart	ssp_err_t(* pdc_api_t::captureStart) (pdc_ctrl_t *const p_ctrl, uint8_t *const p_buffer)
.stateGet	ssp_err_t(* pdc_api_t::stateGet) (pdc_ctrl_t *const p_ctrl, pdc_state_t *p_state)
.versionGet	ssp_err_t(* pdc_api_t::versionGet) (ssp_version_t *const p_data)

12.2.27 Interface: qspi_api_t

Description: Quad SPI Flash Interface

Function name	Definition
.open	ssp_err_t(* qspi_api_t::open) (qspi_ctrl_t *p_ctrl, qspi_cfg_t const *const p_cfg)
.close	ssp_err_t(* qspi_api_t::close) (qspi_ctrl_t *p_ctrl)
.read	ssp_err_t(* qspi_api_t::read) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint8_t *p_memory_address, uint32_t byte_count)
.pageProgram	ssp_err_t(* qspi_api_t::pageProgram) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint8_t *p_memory_address, uint32_t byte_count)
.erase	ssp_err_t(* qspi_api_t::erase) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address, uint32_t byte_count)
.infoGet	ssp_err_t(* qspi_api_t::infoGet) (qspi_ctrl_t *const p_ctrl, qspi_info_t *const p_info)
.sectorErase	ssp_err_t(* qspi_api_t::sectorErase) (qspi_ctrl_t *p_ctrl, uint8_t *p_device_address)
.statusGet	ssp_err_t(* qspi_api_t::statusGet) (qspi_ctrl_t *p_ctrl, bool *p_write_in_progress)
.bankSelect	ssp_err_t(* qspi_api_t::bankSelect) (uint32_t bank)
.versionGet	ssp_err_t(* qspi_api_t::versionGet) (ssp_version_t *const p_version)

12.2.28 Interface: rsa_api_t

Description: RSA Interface

Function name	Definition
.open	uint32_t(* rsa_api_t::open) (rsa_ctrl_t *const p_ctrl, rsa_cfg_t const *const p_cfg)
.encrypt	uint32_t(* rsa_api_t::encrypt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.decrypt	uint32_t(* rsa_api_t::decrypt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.decryptCrt	uint32_t(* rsa_api_t::decryptCrt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.verify	uint32_t(* rsa_api_t::verify) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_signature, uint32_t *p_padded_hash)
.sign	uint32_t(* rsa_api_t::sign) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t *p_dest)
.signCrt	uint32_t(* rsa_api_t::signCrt) (rsa_ctrl_t *const p_ctrl, const uint32_t *p_key, const uint32_t *p_domain, uint32_t num_words, uint32_t *p_padded_hash, uint32_t *p_dest)
.close	uint32_t(* rsa_api_t::close) (rsa_ctrl_t *const p_ctrl)
.versionGet	uint32_t(* rsa_api_t::versionGet) (ssp_version_t *const p_version)
.keyCreate	uint32_t(* rsa_api_t::keyCreate) (rsa_ctrl_t *const p_ctrl, rsa_key_t *p_private_key, rsa_key_t *p_public_key)

12.2.29 Interface: rtc_api_t

Description: RTC Interface

Function name	Definition
.open	ssp_err_t(* rtc_api_t::open) (rtc_ctrl_t *const p_ctrl, rtc_cfg_t const *const p_cfg)
.close	ssp_err_t(* rtc_api_t::close) (rtc_ctrl_t *const p_ctrl)
.calendarTimeSet	ssp_err_t(* rtc_api_t::calendarTimeSet) (rtc_ctrl_t *const p_ctrl, rtc_time_t *p_time, bool clock_start)
.calendarTimeGet	ssp_err_t(* rtc_api_t::calendarTimeGet) (rtc_ctrl_t *const p_ctrl, rtc_time_t *p_time)
.calendarAlarmSet	ssp_err_t(* rtc_api_t::calendarAlarmSet) (rtc_ctrl_t *const p_ctrl, rtc_alarm_time_t *p_alarm, bool irq_enable_flag)
.calendarAlarmGet	ssp_err_t(* rtc_api_t::calendarAlarmGet) (rtc_ctrl_t *const p_ctrl, rtc_alarm_time_t *p_alarm)
.calendarCounterStart	ssp_err_t(* rtc_api_t::calendarCounterStart) (rtc_ctrl_t *const p_ctrl)
.calendarCounterStop	ssp_err_t(* rtc_api_t::calendarCounterStop) (rtc_ctrl_t *const p_ctrl)
.irqEnable	ssp_err_t(* rtc_api_t::irqEnable) (rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
.irqDisable	ssp_err_t(* rtc_api_t::irqDisable) (rtc_ctrl_t *const p_ctrl, rtc_event_t irq)
.periodicIrqRateSet	ssp_err_t(* rtc_api_t::periodicIrqRateSet) (rtc_ctrl_t *const p_ctrl, rtc_periodic_irq_select_t rate)
.infoGet	ssp_err_t(* rtc_api_t::infoGet) (rtc_ctrl_t *p_ctrl, rtc_info_t *p_rtc_info)
.versionGet	ssp_err_t(* rtc_api_t::versionGet) (ssp_version_t *version)

12.2.30 Interface: sdmmc_api_t

Description: SD/MMC Interface

Function name	Definition
.open	ssp_err_t(* sdmmc_api_t::open) (sdmmc_ctrl_t *const p_ctrl, sdmmc_cfg_t const *const p_cfg)
.close	ssp_err_t(* sdmmc_api_t::close) (sdmmc_ctrl_t *const p_ctrl)
.read	ssp_err_t(* sdmmc_api_t::read) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t const start_sector, uint32_t const sector_count)
.write	ssp_err_t(* sdmmc_api_t::write) (sdmmc_ctrl_t *const p_ctrl, uint8_t const *const p_source, uint32_t const start_sector, uint32_t const sector_count)
.control	ssp_err_t(* sdmmc_api_t::control) (sdmmc_ctrl_t *const p_ctrl, ssp_command_t const command, void *p_data)
.readIo	ssp_err_t(* sdmmc_api_t::readIo) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_data, uint32_t const function, uint32_t const address)
.writeIo	ssp_err_t(* sdmmc_api_t::writeIo) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_data, uint32_t const function, uint32_t const address, sdmmc_io_write_mode_t const read_after_write)
.readIoExt	ssp_err_t(* sdmmc_api_t::readIoExt) (sdmmc_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t const function, uint32_t const address, uint32_t *const count, sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t address_mode)
.writeIoExt	ssp_err_t(* sdmmc_api_t::writeIoExt) (sdmmc_ctrl_t *const p_ctrl, uint8_t const *const p_source, uint32_t const function, uint32_t const address, uint32_t const count, sdmmc_io_transfer_mode_t transfer_mode, sdmmc_io_address_mode_t address_mode)
.ioIntEnable	ssp_err_t(* sdmmc_api_t::ioIntEnable) (sdmmc_ctrl_t *const p_ctrl, bool enable)
.versionGet	ssp_err_t(* sdmmc_api_t::versionGet) (ssp_version_t *const p_version)
.infoGet	ssp_err_t(* sdmmc_api_t::infoGet) (sdmmc_ctrl_t *const p_ctrl, sdmmc_info_t *const p_info)

Function name	Definition
.erase	ssp_err_t(* sdmmc_api_t::erase) (sdmmc_ctrl_t *const p_ctrl, uint32_t const start_sector, uint32_t const sector_count)

12.2.31 Interface: sf_adc_periodic_api_t

Description: ADC Periodic Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_adc_periodic_api_t::open) (sf_adc_periodic_ctrl_t *const p_ctrl, sf_adc_periodic_cfg_t const *const p_cfg)
.start	ssp_err_t(* sf_adc_periodic_api_t::start) (sf_adc_periodic_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* sf_adc_periodic_api_t::stop) (sf_adc_periodic_ctrl_t *const p_ctrl)
.close	ssp_err_t(* sf_adc_periodic_api_t::close) (sf_adc_periodic_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_adc_periodic_api_t::versionGet) (ssp_version_t *const p_version)

12.2.32 Interface: sf_audio_playback_api_t

Description: Audio Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_audio_playback_api_t::open) (sf_audio_playback_ctrl_t *const p_ctrl, sf_audio_playback_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_audio_playback_api_t::close) (sf_audio_playback_ctrl_t *const p_ctrl)
.start	ssp_err_t(* sf_audio_playback_api_t::start) (sf_audio_playback_ctrl_t *const p_ctrl, sf_audio_playback_data_t *const p_data, UINT const timeout)

Function name	Definition
.pause	ssp_err_t(* sf_audio_playback_api_t::pause) (sf_audio_playback_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* sf_audio_playback_api_t::stop) (sf_audio_playback_ctrl_t *const p_ctrl)
.resume	ssp_err_t(* sf_audio_playback_api_t::resume) (sf_audio_playback_ctrl_t *const p_ctrl)
.volumeSet	ssp_err_t(* sf_audio_playback_api_t::volumeSet) (sf_audio_playback_ctrl_t *const p_ctrl, uint8_t const volume)
.versionGet	ssp_err_t(* sf_audio_playback_api_t::versionGet) (ssp_version_t *const p_version)

12.2.33 Interface: sf_audio_playback_hw_api_t

Description: Audio Playback Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_audio_playback_hw_api_t::open) (sf_audio_playback_hw_ctrl_t *const p_ctrl, sf_audio_playback_hw_cfg_t const *const p_cfg)
.start	ssp_err_t(* sf_audio_playback_hw_api_t::start) (sf_audio_playback_hw_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* sf_audio_playback_hw_api_t::stop) (sf_audio_playback_hw_ctrl_t *const p_ctrl)
.play	ssp_err_t(* sf_audio_playback_hw_api_t::play) (sf_audio_playback_hw_ctrl_t *const p_ctrl, int16_t const *const p_buffer, uint32_t length)
.dataTypeGet	ssp_err_t(* sf_audio_playback_hw_api_t::dataTypeGet) (sf_audio_playback_hw_ctrl_t *const p_ctrl, sf_audio_playback_data_type_t *const p_data_type)
.close	ssp_err_t(* sf_audio_playback_hw_api_t::close) (sf_audio_playback_hw_ctrl_t *const p_ctrl)

Function name	Definition
.versionGet	ssp_err_t(* sf_audio_playback_hw_api_t::versionGet) (ssp_version_t *const p_version)

12.2.34 Interface: sf_audio_record_api_t

Description: Audio Recording Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_audio_record_api_t::open) (sf_audio_record_ctrl_t *const p_ctrl, sf_audio_record_cfg_t const *const p_cfg)
.start	ssp_err_t(* sf_audio_record_api_t::start) (sf_audio_record_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* sf_audio_record_api_t::stop) (sf_audio_record_ctrl_t *const p_ctrl)
.infoGet	ssp_err_t(* sf_audio_record_api_t::infoGet) (sf_audio_record_ctrl_t *const p_ctrl, sf_audio_record_info_t *p_info)
.close	ssp_err_t(* sf_audio_record_api_t::close) (sf_audio_record_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_audio_record_api_t::versionGet) (ssp_version_t *const p_version)

12.2.35 Interface: sf_ble_api_t

Description: SF BLE Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_ble_api_t::open) (sf_ble_ctrl_t *const p_ctrl, const sf_ble_cfg_t *p_cfg)
.close	ssp_err_t(* sf_ble_api_t::close) (sf_ble_ctrl_t *const p_ctrl)

Function name	Definition
.infoGet	ssp_err_t(* sf_ble_api_t::infoGet) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_info_t *p_ble_info)
.provisionGet	ssp_err_t(* sf_ble_api_t::provisionGet) (sf_ble_ctrl_t *const p_ctrl, sf_ble_provisioning_t *p_ble_provisioning)
.provisionSet	ssp_err_t(* sf_ble_api_t::provisionSet) (sf_ble_ctrl_t *const p_ctrl, const sf_ble_provisioning_t *p_ble_provisioning)
.scan	ssp_err_t(* sf_ble_api_t::scan) (sf_ble_ctrl_t *const p_ctrl, sf_ble_scan_t *p_scan, uint8_t *p_cnt, sf_ble_scan_info_t *p_scan_info)
.advertisementStart	ssp_err_t(* sf_ble_api_t::advertisementStart) (sf_ble_ctrl_t *const p_ctrl, sf_ble_adv_info_t *const p_advt_info)
.advertisementStop	ssp_err_t(* sf_ble_api_t::advertisementStop) (sf_ble_ctrl_t *const p_ctrl)
.whitelistAdd	ssp_err_t(* sf_ble_api_t::whitelistAdd) (sf_ble_ctrl_t *const p_ctrl, const uint8_t *p_bd_addr)
.whitelistDel	ssp_err_t(* sf_ble_api_t::whitelistDel) (sf_ble_ctrl_t *const p_ctrl, const uint8_t *p_bd_addr)
.bondingStart	ssp_err_t(* sf_ble_api_t::bondingStart) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, const uint8_t *p_bd_addr, sf_ble_bonding_start_t *p_bonding_start)
.bondingResponse	ssp_err_t(* sf_ble_api_t::bondingResponse) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, const uint8_t *p_bd_addr, sf_ble_bonding_response_t *p_bonding_resp)
.startEncryption	ssp_err_t(* sf_ble_api_t::startEncryption) (sf_ble_ctrl_t *const p_ctrl, sf_ble_sm_enc_info_t const *p_enc_info)
.connect	ssp_err_t(* sf_ble_api_t::connect) (sf_ble_ctrl_t *const p_ctrl, sf_ble_connection_t const *const p_conn, sf_ble_conn_handle_t *p_handle)
.listen	ssp_err_t(* sf_ble_api_t::listen) (sf_ble_ctrl_t *const p_ctrl)

Function name	Definition
.disconnect	ssp_err_t(* sf_ble_api_t::disconnect) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle)
.gattAddCustomProfiles	ssp_err_t(* sf_ble_api_t::gattAddCustomProfiles) (sf_ble_ctrl_t *const p_ctrl, sf_ble_svc_attribute_t *p_svc_attr, uint32_t svc_attr_len, sf_ble_char_attribute_t *p_char_attr, uint32_t char_attr_len)
.gattServiceDiscovery	ssp_err_t(* sf_ble_api_t::gattServiceDiscovery) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_service_discovery_req_t const *const p_sf_ble_svc_dscv_req, sf_ble_service_discovery_rsp_t *const p_sf_ble_svc_dscv_rsp, uint32_t *const p_rsp_cnt)
.gattCharDiscovery	ssp_err_t(* sf_ble_api_t::gattCharDiscovery) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_char_discovery_req_t const *const p_sf_ble_char_dscv_req, sf_ble_char_discovery_rsp_t *const p_sf_ble_char_dscv_rsp, uint32_t *const p_rsp_cnt)
.gattCharDescDiscovery	ssp_err_t(* sf_ble_api_t::gattCharDescDiscovery) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, uint16_t start_handle, uint16_t end_handle, sf_ble_char_desc_discovery_rsp_t *const p_sf_ble_chardesc_dscv_rsp, uint32_t *const p_rsp_cnt)
.gattCharRead	ssp_err_t(* sf_ble_api_t::gattCharRead) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_char_read_req_t const *const p_char_read_req, sf_ble_char_read_rsp_t *const p_char_read_rsp)
.gattCharWrite	ssp_err_t(* sf_ble_api_t::gattCharWrite) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_char_write_req_t const *const p_char_write_req)
.gattCharExecuteWrite	ssp_err_t(* sf_ble_api_t::gattCharExecuteWrite) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_ble_execute_write_t execute_flag)
.gattCharWriteLocal	ssp_err_t(* sf_ble_api_t::gattCharWriteLocal) (sf_ble_ctrl_t *const p_ctrl, uint16_t char_handle, uint16_t data_length, uint8_t *const p_data)

Function name	Definition
.gattSendNotify	ssp_err_t(* sf_ble_api_t::gattSendNotify) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
.gattSendIndicate	ssp_err_t(* sf_ble_api_t::gattSendIndicate) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, uint16_t char_handle)
.gattWriteResponse	ssp_err_t(* sf_ble_api_t::gattWriteResponse) (sf_ble_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, uint16_t handle, sf_ble_attribute_error_code_t error_code)
.versionGet	ssp_err_t(* sf_ble_api_t::versionGet) (ssp_version_t *const p_version)

12.2.36 Interface: sf_ble_onboard_profile_api_t

Description: SF BLE On-Board Profile Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_ble_onboard_profile_api_t::open) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, const sf_ble_onboard_profile_cfg_t *p_cfg)
.close	ssp_err_t(* sf_ble_onboard_profile_api_t::close) (sf_ble_onboard_profile_ctrl_t *const p_ctrl)
.onbpEnable	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpEnable) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_profile_callback_t p_prf_cb, sf_ble_prf_sec_t sec)
.onbpDisable	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpDisable) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile)

Function name	Definition
.onbpServerWriteData	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerWriteData) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
.onbpServerSendNotification	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerSendNotification) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
.onbpServerSendIndication	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpServerSendIndication) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
.onbpClientWriteCCCD	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientWriteCCCD) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t cccd_char, sf_ble_cccd_val_t cccd_val)
.onbpClientWriteChar	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientWriteChar) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t characteristics, const void *p_data)
.onbpClientReadChar	ssp_err_t(* sf_ble_onboard_profile_api_t::onbpClientReadChar) (sf_ble_onboard_profile_ctrl_t *const p_ctrl, sf_ble_conn_handle_t *p_handle, sf_onbp_t profile, sf_ble_onbp_char_t characteristics)
.versionGet	ssp_err_t(* sf_ble_onboard_profile_api_t::versionGet) (ssp_version_t *const p_version)

12.2.37 Interface: sf_block_media_api_t

Description: Block Media Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_block_media_api_t::open) (sf_block_media_ctrl_t *p_ctrl, sf_block_media_cfg_t const *const p_cfg)
.read	ssp_err_t(* sf_block_media_api_t::read) (sf_block_media_ctrl_t *p_ctrl, uint8_t *const p_dest, uint32_t const start_sector, uint32_t const sector_count)
.write	ssp_err_t(* sf_block_media_api_t::write) (sf_block_media_ctrl_t *p_ctrl, uint8_t const *const p_src, uint32_t const start_sector, uint32_t const sector_count)
.ioctl	ssp_err_t(* sf_block_media_api_t::ioctl) (sf_block_media_ctrl_t *p_ctrl, ssp_command_t const command, void *p_data)
.close	ssp_err_t(* sf_block_media_api_t::close) (sf_block_media_ctrl_t *p_ctrl)
.versionGet	ssp_err_t(* sf_block_media_api_t::versionGet) (ssp_version_t *const p_version)

12.2.38 Interface: sf_cellular_api_t

Description: SF CELLULAR Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_cellular_api_t::open) (sf_cellular_ctrl_t *p_ctrl, sf_cellular_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_cellular_api_t::close) (sf_cellular_ctrl_t *const p_ctrl)
.provisioningGet	ssp_err_t(* sf_cellular_api_t::provisioningGet) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_provisioning_t *const p_cellular_provisioning)

Function name	Definition
.provisioningSet	ssp_err_t(* sf_cellular_api_t::provisioningSet) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_provisioning_t const *const p_cellular_provisioning)
.infoGet	ssp_err_t(* sf_cellular_api_t::infoGet) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_info_t *const p_cellular_info)
.statisticsGet	ssp_err_t(* sf_cellular_api_t::statisticsGet) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_stats_t *const p_cellular_device_stats)
.transmit	ssp_err_t(* sf_cellular_api_t::transmit) (sf_cellular_ctrl_t *const p_ctrl, uint8_t *const p_buf, uint32_t length)
.versionGet	ssp_err_t(* sf_cellular_api_t::versionGet) (ssp_version_t *const p_version)
.networkConnect	ssp_err_t(* sf_cellular_api_t::networkConnect) (sf_cellular_ctrl_t *const p_ctrl)
.networkDisconnect	ssp_err_t(* sf_cellular_api_t::networkDisconnect) (sf_cellular_ctrl_t *const p_ctrl)
.networkStatusGet	ssp_err_t(* sf_cellular_api_t::networkStatusGet) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_network_status_t *p_network_status)
.simPinSet	ssp_err_t(* sf_cellular_api_t::simPinSet) (sf_cellular_ctrl_t *const p_ctrl, uint8_t *const p_old_pin, uint8_t *const p_new_pin)
.simLock	ssp_err_t(* sf_cellular_api_t::simLock) (sf_cellular_ctrl_t *const p_ctrl, uint8_t *const p_pin)
.simUnlock	ssp_err_t(* sf_cellular_api_t::simUnlock) (sf_cellular_ctrl_t *const p_ctrl, uint8_t *const p_pin)
.simIDGet	ssp_err_t(* sf_cellular_api_t::simIDGet) (sf_cellular_ctrl_t *const p_ctrl, uint8_t *p_sim_id)
.fotaCheck	ssp_err_t(* sf_cellular_api_t::fotaCheck) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotacheck)
.fotaStart	ssp_err_t(* sf_cellular_api_t::fotaStart) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotastart)

Function name	Definition
.fotaStop	ssp_err_t(* sf_cellular_api_t::fotaStop) (sf_cellular_ctrl_t *const p_ctrl, void *p_fotastop)
.reset	ssp_err_t(* sf_cellular_api_t::reset) (sf_cellular_ctrl_t *const p_ctrl, sf_cellular_reset_type_t reset_type)

12.2.39 Interface: sf_cellular_socket_api_t

Description: SF Socket CELLULAR Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_cellular_socket_api_t::open) (sf_cellular_socket_ctrl_t *p_ctrl, sf_cellular_socket_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_cellular_socket_api_t::close) (sf_cellular_socket_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_cellular_socket_api_t::versionGet) (ssp_version_t *const p_version)

12.2.40 Interface: sf_comms_api_t

Description: Communications Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_comms_api_t::open) (sf_comms_ctrl_t *const p_ctrl, sf_comms_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_comms_api_t::close) (sf_comms_ctrl_t *const p_ctrl)
.read	ssp_err_t(* sf_comms_api_t::read) (sf_comms_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t const bytes, UINT const timeout)
.write	ssp_err_t(* sf_comms_api_t::write) (sf_comms_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint32_t const bytes, UINT const timeout)

Function name	Definition
.lock	ssp_err_t(* sf_comms_api_t::lock) (sf_comms_ctrl_t *const p_ctrl, sf_comms_lock_t lock_type, UINT timeout)
.unlock	ssp_err_t(* sf_comms_api_t::unlock) (sf_comms_ctrl_t *const p_ctrl, sf_comms_lock_t lock_type)
.versionGet	ssp_err_t(* sf_comms_api_t::versionGet) (ssp_version_t *const p_version)

12.2.41 Interface: sf_console_api_t

Description: Console Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_console_api_t::open) (sf_console_ctrl_t *const p_ctrl, sf_console_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_console_api_t::close) (sf_console_ctrl_t *const p_ctrl)
.prompt	ssp_err_t(* sf_console_api_t::prompt) (sf_console_ctrl_t *const p_ctrl, sf_console_menu_t const *const p_menu, UINT const timeout)
.parse	ssp_err_t(* sf_console_api_t::parse) (sf_console_ctrl_t *const p_ctrl, sf_console_menu_t const *const p_cmd_list, uint8_t const *const p_input, uint32_t const bytes)
.read	ssp_err_t(* sf_console_api_t::read) (sf_console_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t const bytes, uint32_t const timeout)
.write	ssp_err_t(* sf_console_api_t::write) (sf_console_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint32_t const timeout)
.argumentFind	ssp_err_t(* sf_console_api_t::argumentFind) (uint8_t const *const p_arg, uint8_t const *const p_str, int32_t *const p_index, int32_t *const p_data)

Function name	Definition
.versionGet	ssp_err_t(* sf_console_api_t::versionGet) (ssp_version_t *const p_version)

12.2.42 Interface: sf_crypto_api_t

Function name	Definition
.open	ssp_err_t(* sf_crypto_api_t::open) (sf_crypto_ctrl_t *const p_ctrl, sf_crypto_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_crypto_api_t::close) (sf_crypto_ctrl_t *const p_ctrl)
.lock	ssp_err_t(* sf_crypto_api_t::lock) (sf_crypto_ctrl_t *const p_ctrl)
.unlock	ssp_err_t(* sf_crypto_api_t::unlock) (sf_crypto_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_crypto_api_t::versionGet) (ssp_version_t *const p_version)
.statusGet	ssp_err_t(* sf_crypto_api_t::statusGet) (sf_crypto_ctrl_t *const p_ctrl, sf_crypto_state_t *p_status)

12.2.43 Interface: sf_crypto_hash_api_t

Function name	Definition
.open	ssp_err_t(* sf_crypto_hash_api_t::open) (sf_crypto_hash_ctrl_t *const p_ctrl, sf_crypto_hash_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_crypto_hash_api_t::close) (sf_crypto_hash_ctrl_t *const p_ctrl)
.hashInit	ssp_err_t(* sf_crypto_hash_api_t::hashInit) (sf_crypto_hash_ctrl_t *const p_ctrl)

Function name	Definition
.hashUpdate	ssp_err_t(* sf_crypto_hash_api_t::hashUpdate) (sf_crypto_hash_ctrl_t *const p_ctrl, sf_crypto_data_handle_t const *const p_data_in)
.hashFinal	ssp_err_t(* sf_crypto_hash_api_t::hashFinal) (sf_crypto_hash_ctrl_t *const p_ctrl, sf_crypto_data_handle_t *const p_msg_digest, uint32_t *p_size)
.versionGet	ssp_err_t(* sf_crypto_hash_api_t::versionGet) (ssp_version_t *const p_version)

12.2.44 Interface: sf_crypto_key_api_t

Function name	Definition
.open	ssp_err_t(* sf_crypto_key_api_t::open) (sf_crypto_key_ctrl_t *const p_ctrl, sf_crypto_key_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_crypto_key_api_t::close) (sf_crypto_key_ctrl_t *const p_ctrl)
.keyGenerate	ssp_err_t(* sf_crypto_key_api_t::keyGenerate) (sf_crypto_key_ctrl_t *const p_ctrl, sf_crypto_key_t *const p_secret_key, sf_crypto_key_t *const p_public_key)
.versionGet	ssp_err_t(* sf_crypto_key_api_t::versionGet) (ssp_version_t *const p_version)

12.2.45 Interface: sf_crypto_trng_api_t

Function name	Definition
.open	ssp_err_t(* sf_crypto_trng_api_t::open) (sf_crypto_trng_ctrl_t *const p_ctrl, sf_crypto_trng_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_crypto_trng_api_t::close) (sf_crypto_trng_ctrl_t *const p_ctrl)

Function name	Definition
.randomNumberGenerate	ssp_err_t(* sf_crypto_trng_api_t::randomNumberGenerate) (sf_crypto_trng_ctrl_t *const p_ctrl, sf_crypto_data_handle_t *const p_random_number_buff)
.versionGet	ssp_err_t(* sf_crypto_trng_api_t::versionGet) (ssp_version_t *const p_version)

12.2.46 Interface: sf_el_gx_api_t

Description: GUIX Interface

Function name	Definition
.open	ssp_err_t(* sf_el_gx_api_t::open) (sf_el_gx_ctrl_t *const p_ctrl, sf_el_gx_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_el_gx_api_t::close) (sf_el_gx_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_el_gx_api_t::versionGet) (ssp_version_t *p_version)
.setup	UINT(* sf_el_gx_api_t::setup) (GX_DISPLAY *p_display)
.canvasInit	ssp_err_t(* sf_el_gx_api_t::canvasInit) (sf_el_gx_ctrl_t *const p_ctrl, GX_WINDOW_ROOT *p_window_root)

12.2.47 Interface: sf_external_irq_api_t

Description: External IRQ Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_external_irq_api_t::open) (sf_external_irq_ctrl_t *const p_ctrl, sf_external_irq_cfg_t const *const p_cfg)

Function name	Definition
.wait	ssp_err_t(* sf_external_irq_api_t::wait) (sf_external_irq_ctrl_t *const p_ctrl, ULONG const timeout)
.versionGet	ssp_err_t(* sf_external_irq_api_t::versionGet) (ssp_version_t *const p_version)
.close	ssp_err_t(* sf_external_irq_api_t::close) (sf_external_irq_ctrl_t *const p_ctrl)

12.2.48 Interface: sf_i2c_api_t

Description: I2C Framework

Function name	Definition
.open	ssp_err_t(* sf_i2c_api_t::open) (sf_i2c_ctrl_t *p_ctrl, sf_i2c_cfg_t const *const p_cfg)
.read	ssp_err_t(* sf_i2c_api_t::read) (sf_i2c_ctrl_t *const p_ctrl, uint8_t *const p_dest, uint32_t const bytes, bool const restart, uint32_t const timeout)
.write	ssp_err_t(* sf_i2c_api_t::write) (sf_i2c_ctrl_t *const p_ctrl, uint8_t *const p_src, uint32_t const bytes, bool const restart, uint32_t const timeout)
.reset	ssp_err_t(* sf_i2c_api_t::reset) (sf_i2c_ctrl_t *const p_ctrl, uint32_t const timeout)
.close	ssp_err_t(* sf_i2c_api_t::close) (sf_i2c_ctrl_t *const p_ctrl)
.lock	ssp_err_t(* sf_i2c_api_t::lock) (sf_i2c_ctrl_t *const p_ctrl)
.unlock	ssp_err_t(* sf_i2c_api_t::unlock) (sf_i2c_ctrl_t *const p_ctrl)
.version	ssp_err_t(* sf_i2c_api_t::version) (ssp_version_t *const p_version)

12.2.49 Interface: sf_jpeg_decode_api_t

Description: JPEG Decode Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_jpeg_decode_api_t::open) (sf_jpeg_decode_ctrl_t *const p_ctrl, sf_jpeg_decode_cfg_t const *const p_cfg)
.inputBufferSet	ssp_err_t(* sf_jpeg_decode_api_t::inputBufferSet) (sf_jpeg_decode_ctrl_t *const p_ctrl, void *const p_buffer, uint32_t const buffer_size)
.outputBufferSet	ssp_err_t(* sf_jpeg_decode_api_t::outputBufferSet) (sf_jpeg_decode_ctrl_t *const p_ctrl, void *p_buffer, uint32_t buffer_size)
.linesDecodedGet	ssp_err_t(* sf_jpeg_decode_api_t::linesDecodedGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, uint32_t *const p_lines)
.horizontalStrideSet	ssp_err_t(* sf_jpeg_decode_api_t::horizontalStrideSet) (sf_jpeg_decode_ctrl_t *const p_ctrl, uint32_t horizontal_stride)
.imageSubsampleSet	ssp_err_t(* sf_jpeg_decode_api_t::imageSubsampleSet) (sf_jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_subsample_t horizontal_subsample, jpeg_decode_subsample_t vertical_subsample)
.wait	ssp_err_t(* sf_jpeg_decode_api_t::wait) (sf_jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_status_t *const p_status, uint32_t timeout)
.statusGet	ssp_err_t(* sf_jpeg_decode_api_t::statusGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_status_t *const p_status)
.imageSizeGet	ssp_err_t(* sf_jpeg_decode_api_t::imageSizeGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, uint16_t *p_horizontal_size, uint16_t *p_vertical_size)
.pixelFormatGet	ssp_err_t(* sf_jpeg_decode_api_t::pixelFormatGet) (sf_jpeg_decode_ctrl_t *const p_ctrl, jpeg_decode_color_space_t *const p_color_space)

Function name	Definition
.close	ssp_err_t(* sf_jpeg_decode_api_t::close) (sf_jpeg_decode_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_jpeg_decode_api_t::versionGet) (ssp_version_t *const p_version)

12.2.50 Interface: sf_message_api_t

Description: essaging Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_message_api_t::open) (sf_message_ctrl_t *const p_ctrl, sf_message_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_message_api_t::close) (sf_message_ctrl_t *const p_ctrl)
.bufferAcquire	ssp_err_t(* sf_message_api_t::bufferAcquire) (sf_message_ctrl_t const *const p_ctrl, sf_message_header_t **pp_buffer, sf_message_acquire_cfg_t const *const p_acquire_cfg, uint32_t const wait_option)
.bufferRelease	ssp_err_t(* sf_message_api_t::bufferRelease) (sf_message_ctrl_t *const p_ctrl, sf_message_header_t *const p_buffer, sf_message_release_option_t const option)
.post	ssp_err_t(* sf_message_api_t::post) (sf_message_ctrl_t *const p_ctrl, sf_message_header_t const *const p_buffer, sf_message_post_cfg_t const *const p_post_cfg, sf_message_post_err_t *const p_post_err, uint32_t const wait_option)
.pend	ssp_err_t(* sf_message_api_t::pend) (sf_message_ctrl_t const *const p_ctrl, TX_QUEUE const *const p_queue, sf_message_header_t **pp_buffer, uint32_t const wait_option)
.versionGet	ssp_err_t(* sf_message_api_t::versionGet) (ssp_version_t *const p_version)

12.2.51 Interface: sf_power_profiles_api_t

Description: Power Profiles Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_power_profiles_api_t::open) (sf_power_profiles_ctrl_t *const p_ctrl, sf_power_profiles_cfg_t const *const p_cfg)
.sleep	ssp_err_t(* sf_power_profiles_api_t::sleep) (sf_power_profiles_ctrl_t *const p_ctrl)
.close	ssp_err_t(* sf_power_profiles_api_t::close) (sf_power_profiles_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_power_profiles_api_t::versionGet) (ssp_version_t *const p_version)

12.2.52 Interface: sf_socket_api_t

Description: SF Socket WIFI Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_socket_api_t::open) (sf_socket_ctrl_t *p_ctrl, sf_socket_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_socket_api_t::close) (sf_socket_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_socket_api_t::versionGet) (ssp_version_t *const p_version)

12.2.53 Interface: sf_spi_api_t

Description: SPI Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_spi_api_t::open) (sf_spi_ctrl_t *p_ctrl, sf_spi_cfg_t const *const p_cfg)

Function name	Definition
.read	ssp_err_t(* sf_spi_api_t::read) (sf_spi_ctrl_t *const p_ctrl, void *const p_dest, uint32_t const length, spi_bit_width_t const bit_width, uint32_t const timeout)
.write	ssp_err_t(* sf_spi_api_t::write) (sf_spi_ctrl_t *const p_ctrl, void *const p_src, uint32_t const length, spi_bit_width_t const bit_width, uint32_t const timeout)
.writeRead	ssp_err_t(* sf_spi_api_t::writeRead) (sf_spi_ctrl_t *const p_ctrl, void *const p_src, void *const p_dest, uint32_t const length, spi_bit_width_t const bit_width, uint32_t const timeout)
.close	ssp_err_t(* sf_spi_api_t::close) (sf_spi_ctrl_t *const p_ctrl)
.lock	ssp_err_t(* sf_spi_api_t::lock) (sf_spi_ctrl_t *const p_ctrl)
.unlock	ssp_err_t(* sf_spi_api_t::unlock) (sf_spi_ctrl_t *const p_ctrl)
.version	ssp_err_t(* sf_spi_api_t::version) (ssp_version_t *const p_version)

12.2.54 Interface: sf_thread_monitor_api_t

Description: Thread Monitor Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_thread_monitor_api_t::open) (sf_thread_monitor_ctrl_t *const p_ctrl, sf_thread_monitor_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_thread_monitor_api_t::close) (sf_thread_monitor_ctrl_t *const p_ctrl)
.threadRegister	ssp_err_t(* sf_thread_monitor_api_t::threadRegister) (sf_thread_monitor_ctrl_t *const p_ctrl, sf_thread_monitor_counter_min_max_t const *p_counter_min_max)

Function name	Definition
.threadUnregister	ssp_err_t(* sf_thread_monitor_api_t::threadUnregister) (sf_thread_monitor_ctrl_t *const p_ctrl)
.countIncrement	ssp_err_t(* sf_thread_monitor_api_t::countIncrement) (sf_thread_monitor_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_thread_monitor_api_t::versionGet) (ssp_version_t *const p_version)

12.2.55 Interface: sf_touch_ctsu_api_t

Description: CTSU Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_touch_ctsu_api_t::open) (sf_touch_ctsu_ctrl_t *const p_ctrl, sf_touch_ctsu_cfg_t const *const p_cfg)
.read	ssp_err_t(* sf_touch_ctsu_api_t::read) (sf_touch_ctsu_ctrl_t *const p_ctrl, void *p_dest, ctsu_read_t opts, const ctsu_channel_pair_t *channels, const uint16_t count)
.close	ssp_err_t(* sf_touch_ctsu_api_t::close) (sf_touch_ctsu_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_touch_ctsu_api_t::versionGet) (ssp_version_t *const p_version)

12.2.56 Interface: sf_touch_ctsu_button_api_t

Description: CTSU Button Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_touch_ctsu_button_api_t::open) (sf_touch_ctsu_button_ctrl_t *const p_ctrl, sf_touch_ctsu_button_cfg_t const *const p_cfg)

Function name	Definition
.enable	ssp_err_t(* sf_touch_ctsu_button_api_t::enable) (sf_touch_ctsu_button_ctrl_t *const p_ctrl, sf_touch_ctsu_button_id const button_id)
.disable	ssp_err_t(* sf_touch_ctsu_button_api_t::disable) (sf_touch_ctsu_button_ctrl_t *const p_ctrl, sf_touch_ctsu_button_id const button_id)
.close	ssp_err_t(* sf_touch_ctsu_button_api_t::close) (sf_touch_ctsu_button_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_touch_ctsu_button_api_t::versionGet) (ssp_version_t *const p_version)

12.2.57 Interface: sf_touch_ctsu_slider_api_t

Description: CTSU Slider Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_touch_ctsu_slider_api_t::open) (sf_touch_ctsu_slider_ctrl_t *const p_ctrl, sf_touch_ctsu_slider_cfg_t const *const p_cfg)
.enable	ssp_err_t(* sf_touch_ctsu_slider_api_t::enable) (sf_touch_ctsu_slider_ctrl_t *const p_ctrl, sf_touch_ctsu_slider_id_t const slider_id)
.disable	ssp_err_t(* sf_touch_ctsu_slider_api_t::disable) (sf_touch_ctsu_slider_ctrl_t *const p_ctrl, sf_touch_ctsu_slider_id_t const slider_id)
.close	ssp_err_t(* sf_touch_ctsu_slider_api_t::close) (sf_touch_ctsu_slider_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_touch_ctsu_slider_api_t::versionGet) (ssp_version_t *const p_version)

12.2.58 Interface: sf_touch_panel_api_t

Description: Touch Panel Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_touch_panel_api_t::open) (sf_touch_panel_ctrl_t *const p_ctrl, sf_touch_panel_cfg_t const *const p_cfg)
.calibrate	ssp_err_t(* sf_touch_panel_api_t::calibrate) (sf_touch_panel_ctrl_t *const p_ctrl, sf_touch_panel_calibrate_t const *const p_expected, sf_touch_panel_payload_t const *const p_actual, ULONG timeout)
.start	ssp_err_t(* sf_touch_panel_api_t::start) (sf_touch_panel_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* sf_touch_panel_api_t::stop) (sf_touch_panel_ctrl_t *const p_ctrl)
.reset	ssp_err_t(* sf_touch_panel_api_t::reset) (sf_touch_panel_ctrl_t *const p_ctrl)
.close	ssp_err_t(* sf_touch_panel_api_t::close) (sf_touch_panel_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_touch_panel_api_t::versionGet) (ssp_version_t *const p_version)

12.2.59 Interface: sf_wifi_api_t

Description: SF WIFI Framework Interface

Function name	Definition
.open	ssp_err_t(* sf_wifi_api_t::open) (sf_wifi_ctrl_t *p_ctrl, sf_wifi_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_wifi_api_t::close) (sf_wifi_ctrl_t *const p_ctrl)
.multicastListAdd	ssp_err_t(* sf_wifi_api_t::multicastListAdd) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const p_mac_addr)
.multicastListDelete	ssp_err_t(* sf_wifi_api_t::multicastListDelete) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const p_mac_addr)

Function name	Definition
.statisticsGet	ssp_err_t(* sf_wifi_api_t::statisticsGet) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_stats_t *const p_wifi_device_stats)
.transmit	ssp_err_t(* sf_wifi_api_t::transmit) (sf_wifi_ctrl_t *const p_ctrl, uint8_t *const p_buf, uint32_t length)
.provisioningSet	ssp_err_t(* sf_wifi_api_t::provisioningSet) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_provisioning_t const *const p_wifi_provisioning)
.provisioningGet	ssp_err_t(* sf_wifi_api_t::provisioningGet) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_provisioning_t *const p_wifi_provisioning)
.infoGet	ssp_err_t(* sf_wifi_api_t::infoGet) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_info_t *const p_wifi_info)
.scan	ssp_err_t(* sf_wifi_api_t::scan) (sf_wifi_ctrl_t *const p_ctrl, sf_wifi_scan_t *const p_scan, uint8_t *const p_cnt)
.ACLAdd	ssp_err_t(* sf_wifi_api_t::ACLAdd) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const p_mac)
.ACLDelete	ssp_err_t(* sf_wifi_api_t::ACLDelete) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const p_mac)
.macAddressGet	ssp_err_t(* sf_wifi_api_t::macAddressGet) (sf_wifi_ctrl_t *const p_ctrl, uint8_t *const p_mac)
.macAddressSet	ssp_err_t(* sf_wifi_api_t::macAddressSet) (sf_wifi_ctrl_t *const p_ctrl, uint8_t const *const p_mac)
.versionGet	ssp_err_t(* sf_wifi_api_t::versionGet) (ssp_version_t *const p_version)

12.2.60 Interface: sf_wifi_onchip_stack_api_t

Description: SF WIFI On-Chip Stack Interface

Function name	Definition
.open	ssp_err_t(* sf_wifi_onchip_stack_api_t::open) (sf_wifi_onchip_stack_ctrl_t *p_ctrl, sf_wifi_onchip_stack_cfg_t const *const p_cfg)
.close	ssp_err_t(* sf_wifi_onchip_stack_api_t::close) (sf_wifi_onchip_stack_ctrl_t *const p_ctrl)
.ipAddressCfg	ssp_err_t(* sf_wifi_onchip_stack_api_t::ipAddressCfg) (sf_wifi_onchip_stack_ctrl_t *const p_ctrl, sf_wifi_onchip_stack_ip_cfg_t *const p_cfg)
.dhcpServerStart	ssp_err_t(* sf_wifi_onchip_stack_api_t::dhcpServerStart) (sf_wifi_onchip_stack_ctrl_t *const p_ctrl, sf_wifi_ip_addr_t const *const p_start_ip, sf_wifi_ip_addr_t const *const p_end_ip)
.dhcpServerStop	ssp_err_t(* sf_wifi_onchip_stack_api_t::dhcpServerStop) (sf_wifi_onchip_stack_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* sf_wifi_onchip_stack_api_t::versionGet) (ssp_version_t *const p_version)

12.2.61 Interface: slcdc_api_t

Description: SLCDC Interface

Function name	Definition
.open	ssp_err_t(* slcdc_api_t::open) (slcdc_ctrl_t *const p_ctrl, slcdc_cfg_t const *const p_cfg)
.write	ssp_err_t(* slcdc_api_t::write) (slcdc_ctrl_t *const p_ctrl, slcdc_size_t const start_segment, slcdc_size_t const *const p_data, slcdc_size_t const segment_count)
.modify	ssp_err_t(* slcdc_api_t::modify) (slcdc_ctrl_t *const p_ctrl, slcdc_size_t const segment, slcdc_size_t const data_mask, slcdc_size_t const data)

Function name	Definition
.start	ssp_err_t(* slcdc_api_t::start) (slcdc_ctrl_t *const p_ctrl)
.stop	ssp_err_t(* slcdc_api_t::stop) (slcdc_ctrl_t *const p_ctrl)
.contrastIncrease	ssp_err_t(* slcdc_api_t::contrastIncrease) (slcdc_ctrl_t *const p_ctrl)
.contrastDecrease	ssp_err_t(* slcdc_api_t::contrastDecrease) (slcdc_ctrl_t *const p_ctrl)
.setdisplayArea	ssp_err_t(* slcdc_api_t::setdisplayArea) (slcdc_ctrl_t *const p_ctrl, slcdc_display_area_t const display_area)
.close	ssp_err_t(* slcdc_api_t::close) (slcdc_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* slcdc_api_t::versionGet) (ssp_version_t *p_version)

12.2.62 Interface: spi_api_t

Description: SPI Interface

Function name	Definition
.open	ssp_err_t(* spi_api_t::open) (spi_ctrl_t *p_ctrl, spi_cfg_t const *const p_cfg)
.read	ssp_err_t(* spi_api_t::read) (spi_ctrl_t *const p_ctrl, void const *p_dest, uint32_t const length, spi_bit_width_t const bit_width)
.write	ssp_err_t(* spi_api_t::write) (spi_ctrl_t *const p_ctrl, void const *p_src, uint32_t const length, spi_bit_width_t const bit_width)
.writeRead	ssp_err_t(* spi_api_t::writeRead) (spi_ctrl_t *const p_ctrl, void const *p_src, void const *p_dest, uint32_t const length, spi_bit_width_t const bit_width)
.close	ssp_err_t(* spi_api_t::close) (spi_ctrl_t *const p_ctrl)

Function name	Definition
.versionGet	ssp_err_t(* spi_api_t::versionGet) (ssp_version_t *p_version)

12.2.63 Interface: tdes_api_t

Description: TDES Interface

Function name	Definition
.open	uint32_t(* tdes_api_t::open) (tdes_ctrl_t *const p_ctrl, tdes_cfg_t const *const p_cfg)
.encrypt	uint32_t(* tdes_api_t::encrypt) (tdes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.decrypt	uint32_t(* tdes_api_t::decrypt) (tdes_ctrl_t *const p_ctrl, const uint32_t *p_key, uint32_t *p_iv, uint32_t num_words, uint32_t *p_source, uint32_t *p_dest)
.close	uint32_t(* tdes_api_t::close) (tdes_ctrl_t *const p_ctrl)
.versionGet	uint32_t(* tdes_api_t::versionGet) (ssp_version_t *const p_version)

12.2.64 Interface: timer_api_t

Description: Timer Interface

Function name	Definition
.open	ssp_err_t(* timer_api_t::open) (timer_ctrl_t *const p_ctrl, timer_cfg_t const *const p_cfg)
.stop	ssp_err_t(* timer_api_t::stop) (timer_ctrl_t *const p_ctrl)
.start	ssp_err_t(* timer_api_t::start) (timer_ctrl_t *const p_ctrl)
.reset	ssp_err_t(* timer_api_t::reset) (timer_ctrl_t *const p_ctrl)

Function name	Definition
.counterGet	ssp_err_t(* timer_api_t::counterGet) (timer_ctrl_t *const p_ctrl, timer_size_t *const p_value)
.periodSet	ssp_err_t(* timer_api_t::periodSet) (timer_ctrl_t *const p_ctrl, timer_size_t const period, timer_unit_t const unit)
.dutyCycleSet	ssp_err_t(* timer_api_t::dutyCycleSet) (timer_ctrl_t *const p_ctrl, timer_size_t const duty_cycle, timer_pwm_unit_t const duty_cycle_unit, uint8_t const pin)
.infoGet	ssp_err_t(* timer_api_t::infoGet) (timer_ctrl_t *const p_ctrl, timer_info_t *const p_info)
.close	ssp_err_t(* timer_api_t::close) (timer_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* timer_api_t::versionGet) (ssp_version_t *const p_version)

12.2.65 Interface: transfer_api_t

Description: Transfer Interface

Function name	Definition
.open	ssp_err_t(* transfer_api_t::open) (transfer_ctrl_t *const p_ctrl, transfer_cfg_t const *const p_cfg)
.reset	ssp_err_t(* transfer_api_t::reset) (transfer_ctrl_t *const p_ctrl, void const *p_src, void *p_dest, uint16_t const num_transfers)
.enable	ssp_err_t(* transfer_api_t::enable) (transfer_ctrl_t *const p_ctrl)
.disable	ssp_err_t(* transfer_api_t::disable) (transfer_ctrl_t *const p_ctrl)
.start	ssp_err_t(* transfer_api_t::start) (transfer_ctrl_t *const p_ctrl, transfer_start_mode_t mode)
.stop	ssp_err_t(* transfer_api_t::stop) (transfer_ctrl_t *const p_ctrl)

Function name	Definition
.infoGet	ssp_err_t(* transfer_api_t::infoGet) (transfer_ctrl_t *const p_ctrl, transfer_properties_t *const p_info)
.close	ssp_err_t(* transfer_api_t::close) (transfer_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* transfer_api_t::versionGet) (ssp_version_t *const p_version)
.blockReset	ssp_err_t(* transfer_api_t::blockReset) (transfer_ctrl_t *const p_ctrl, void const *p_src, void *p_dest, uint16_t const length, transfer_size_t size, uint16_t const num_transfers)

12.2.66 Interface: trng_api_t

Description: Random number generation

Function name	Definition
.open	uint32_t(* trng_api_t::open) (trng_ctrl_t *const p_ctrl, trng_cfg_t const *const p_cfg)
.read	uint32_t(* trng_api_t::read) (trng_ctrl_t *const p_ctrl, uint32_t *const p_rngbuf, uint32_t nwords)
.close	uint32_t(* trng_api_t::close) (trng_ctrl_t *const p_ctrl)
.versionGet	uint32_t(* trng_api_t::versionGet) (ssp_version_t *const p_version)

12.2.67 Interface: uart_api_t

Description: UART Interface

Function name	Definition
.open	ssp_err_t(* uart_api_t::open) (uart_ctrl_t *const p_ctrl, uart_cfg_t const *const p_cfg)
.read	ssp_err_t(* uart_api_t::read) (uart_ctrl_t *const p_ctrl, uint8_t const *const p_dest, uint32_t const bytes)

Function name	Definition
.write	ssp_err_t(* uart_api_t::write) (uart_ctrl_t *const p_ctrl, uint8_t const *const p_src, uint32_t const bytes)
.baudSet	ssp_err_t(* uart_api_t::baudSet) (uart_ctrl_t *const p_ctrl, uint32_t const baudrate)
.infoGet	ssp_err_t(* uart_api_t::infoGet) (uart_ctrl_t *const p_ctrl, uart_info_t *const p_info)
.close	ssp_err_t(* uart_api_t::close) (uart_ctrl_t *const p_ctrl)
.versionGet	ssp_err_t(* uart_api_t::versionGet) (ssp_version_t *p_version)

12.2.68 Interface: wdt_api_t

Description: WDT Interface

Function name	Definition
.cfgGet	ssp_err_t(* wdt_api_t::cfgGet) (wdt_ctrl_t *const p_ctrl, wdt_cfg_t *const p_cfg)
.open	ssp_err_t(* wdt_api_t::open) (wdt_ctrl_t *const p_ctrl, wdt_cfg_t const *const p_cfg)
.refresh	ssp_err_t(* wdt_api_t::refresh) (wdt_ctrl_t *const p_ctrl)
.statusGet	ssp_err_t(* wdt_api_t::statusGet) (wdt_ctrl_t *const p_ctrl, wdt_status_t *const p_status)
.statusClear	ssp_err_t(* wdt_api_t::statusClear) (wdt_ctrl_t *const p_ctrl, const wdt_status_t status)
.counterGet	ssp_err_t(* wdt_api_t::counterGet) (wdt_ctrl_t *const p_ctrl, uint32_t *const p_count)
.timeoutGet	ssp_err_t(* wdt_api_t::timeoutGet) (wdt_ctrl_t *const p_ctrl, wdt_timeout_values_t *const p_timeout)
.versionGet	ssp_err_t(* wdt_api_t::versionGet) (ssp_version_t *const p_data)

参考資料

Renesas Synergy™ Software Package（SSP）
v1.3.2ユーザーズマニュアル（参考資料）

発行年月日　2018年4月25日　Rev.1.13

発行　　ルネサス エレクトロニクス株式会社
　　〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

Renesas Synergy™ Software Package（SSP）
v1.3.2ユーザーズマニュアル（参考資料）

Renesas Electronics Corporation

R01US0315JU0113