Renesas Synergy™ プラットフォーム

UART 通信フレームワークモジュールガイド

R11AN0192JU0102
Rev.1.02
2018.11.14

本資料は英語版を翻訳した参考資料です。本資料の第 6 章までは Renesas Synergy SSP v1.5.0 Users Manual 英語版の第 4 章 Module Guide を参照に翻訳されています。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

要旨（Introduction）

本モジュールガイドは、ユーザがモジュールを効果的に使用してシステムが開発できるようになることを目的としています。このモジュールガイドを習得することで、開発システムへのモジュールの追加とターゲットアプリケーション向けの正確な設定（configuration）ができ、さらに付属のアプリケーションプロジェクトコードを参照して、効率的なコード記述が行えるようになります。

より詳細な API や、より高度なモジュール使用法を記述した他のアプリケーションプロジェクト例もルネサス WEB サイト（本書末尾の「参考文献」の項を参照）から入手でき、より複雑な設計に役立ちます。

UART 通信フレームワークモジュール（UART Communications Framework Module）は、ThreadX®RTOS を使用した通信アプリケーション向けのハイレベル API です。 sf_uart_comms が UART コミュニケーションフレームワークモジュールとして実装されています。 UART コミュニケーションフレームワークモジュールは、Synergy MCU の SCI ペリフェラルを使用します。 ユーザ定義のコールバックを作成して、送信が完了したか、文字が受信されたことを示すことができます。

目次 （Contents）

1. UART 通信フレームワークモジュールの機能（UART Communications Framework Module Features）

このモジュールは、ThreadX® 対応の通信フレームワークです。 ThreadX オブジェクトを使用して、スレッドセーフな操作を保証します。

UART 通信プロトコルのサポート
独占的アクセス（exclusive access）を行うチャネルロックのサポート
ThreadX® 対応の実装

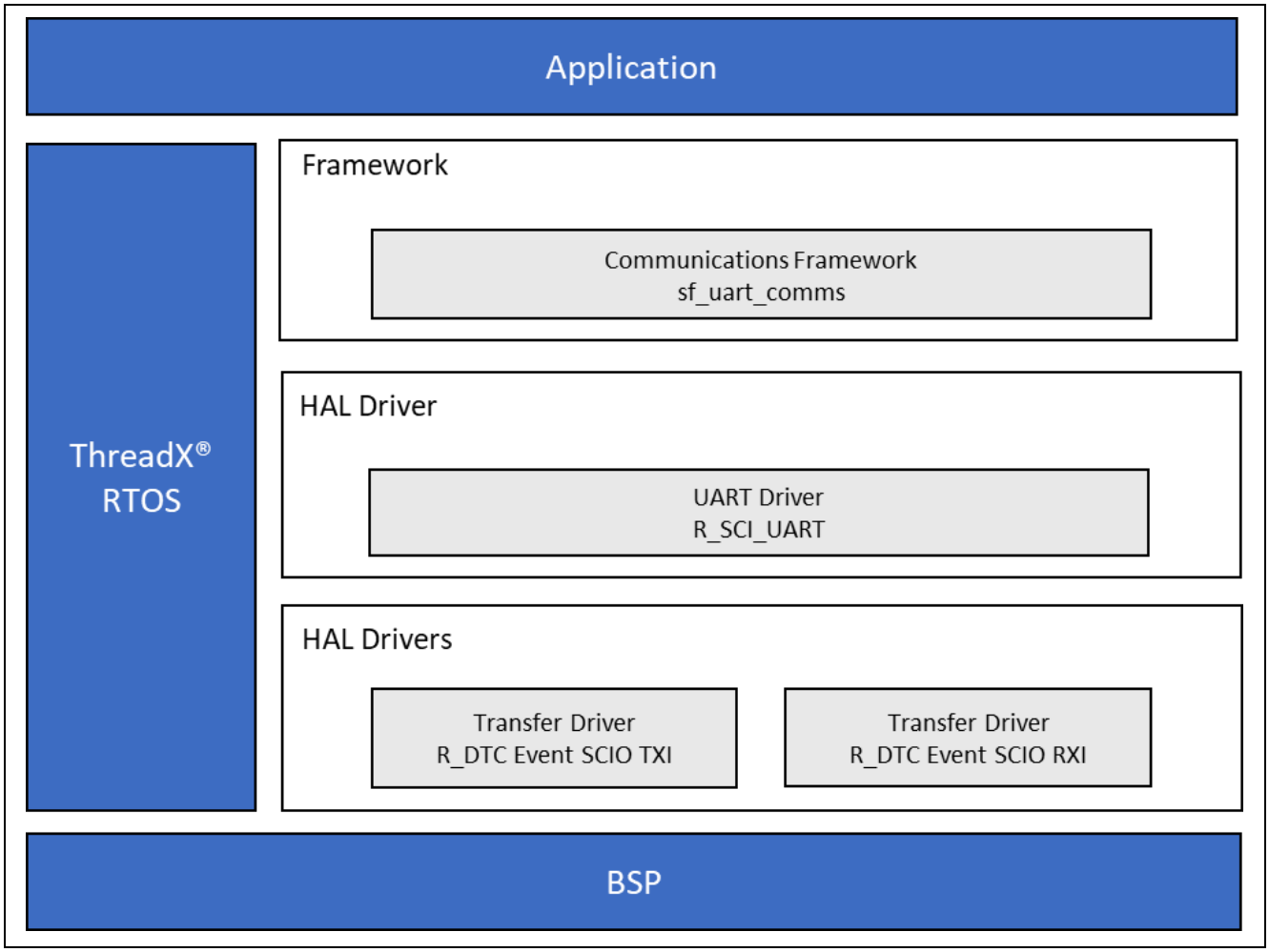

図 1　UART 通信フレームワークモジュールのブロック図

2. UART 通信フレームワークモジュール API の概要（UART Communications Framework Module APIs Overview）

UART 通信フレームワークモジュールは、open、close、read、write、その他の API 機能を定義します。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明を、次の表に示します。ステータス戻り値（status returns value）の表は API 要約表の後にあります。

表 1　UART 通信フレームワークモジュールの API 要約

Function Name	API 呼び出しの例と説明
.open	`g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg);` 通信ドライバを初期化します。
.close	`g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl);` 通信ドライバをクリーンアップします。
.read	`g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout);` 通信ドライバからデータを読み取ります。この呼び出しは、要求された数のバイトを読み取った後、またはドライバへのアクセスを待っている間にタイムアウトが発生した場合に復帰します。
.write	`g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout);` 通信ドライバにデータを書き込みます。この呼び出しは、要求された数のバイトを読み取った後、またはドライバへのアクセスを待っている間にタイムアウトが発生した場合に復帰します。
.lock	`g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout);` 通信ドライバをロックします。通信ドライバへの排他的アクセス（exclusive access）を予約します。
.unlock	`g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type);` 通信ドライバをロック解除します。通信ドライバへの排他的アクセスを解放します。
.versionGet	`g_sf_comms0.p_api->versionGet(&version);` ドライババージョンを指定された version に格納します。

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

表 2　ステータス戻り値

Name	説明
SSP_SUCCESS	チャネルが正常に開かれました。
SSP_ERR_IN_USE	チャネルは既に使用中です。
SSP_ERR_ASSERTION	UART 制御ブロックまたは設定構造体へのポインタが NULL です。
SSP_ERR_HW_LOCKED	チャネルがロックされています。
SSP_ERR_INVALID_MODE	チャネルが非 UART モード用に使用されているか、設定されているモードが誤っています。
SSP_ERR_INVALID_ARGUMENT	設定構造体に無効なパラメータ設定値が見つかりました。
SSP_ERR_QUEUE_UNAVAILABLE	送信キューと受信キューのいずれかまたは両方が開かれていません。
SSP_ERR_INTERNAL	内部エラーが発生しました。
SSP_ERR_TIMEOUT	タイムアウトエラー。
SSP_ERR_INSUFFICIENT_DATA	受信循環バッファに十分なデータがありません。

SSP_ERR_RXBUF_OVERFLOW	受信キューオーバーフロー。
SSP_ERR_OVERFLOW	ハードウェアオーバフロー。
SSP_ERR_FRAMING	フレーミングエラー。
SSP_ERR_PARITY	パリティエラー。
SSP_ERR_INSUFFICIENT_SPACE	送信循環バッファに十分なスペースがありません。

注： ローレベルドライバ（Lower-level drivers）は、一般的なエラーコード（common error codes）を返すことがあります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

3. UART 通信フレームワークモジュールの動作の概要（UART Communications Framework Module Operational Overview）

UART フレームワークは、UART 標準プロトコルを使用した使いやすい通信フレームワークです。 このフレームワークは、スレッドセーフ（thread safe）な方法で UART デバイスからデータを読み書きする高水準の API 関数や、アプリケーションが UART チャネルをスレッドにロック（およびロック解除）する API 関数を提供します。 これは、複数のアプリケーションスレッドが同一 UART デバイスとの通信を行おうとする時や、コンテキストスイッチが UART（Kermit など）上に実装された高水準アプリケーションプロトコルや状態マシンを変動させる可能性がある場合に特に有効です。

3.1 UART 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項（UART Communications Framework Module Important Operational Notes and Limitations）

3.1.1 UART 通信フレームワークモジュールの動作に関する注意事項と制限事項（UART Communications Framework Module Operational Notes）

UART フレームワークモジュールは、全てのチャネルでリエントラント（reentrant）です。
UART フレームワークは、r_sci_uart モジュールの UART ドライバを使用して UART デバイスと通信をおこないます。 UART ドライバは、DTC ドライバを追加することによって拡張が可能で（ISDE の Synergy Platform コンフィグレータを使用）、CPU を中断することなく UART デバイスとの間で読み取り、または書き込みトランザクションを実行します。UART フレームワークを使用して UART デバイスからデータを読み取る場合、データを読み取るために UART ドライバのコールバック機能が使用されます（DTC モジュールのサポートがコンフィグレータを介して UART ドライバに追加されている場合でも）。 これは、ドライバが DTC を使用してデバイスからデータを読み取ったときに発生する可能性のあるタイミングと同期に関する潜在的な問題を回避します。 UART フレームワークを使用して UART デバイスにデータを書き込むときは、DTC を使用してトランザクションを実行します（ドライバが DTC を使用するように設定されている場合）。

3.1.2 UART 通信フレームワークモジュールの制限事項（UART Communications Framework Module Limitations）

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

4. アプリケーションへの UART 通信フレームワークモジュールの組み込み（Including the UART Communications Framework Module in an Application）

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

注： この章を理解するには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーション作成時の重要な各手順の取り扱い方を修得してください。

UART 通信フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（通信フレームワークのデフォルト名は g_sf_comms0 です。この名前は、対応する [Properties] ウィンドウで変更できます。）

表 3　通信フレームワークの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_uart_comms	Threads	New Stack > Framework > Connectivity > Communications Framework on sf_uart_comms

次の図に示すように、sf_uart_comms の UART 通信フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の設定情報を必要とするドライバは、ボックスのテキストが赤くハイライト表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、一度追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルドライバの選択が必要です。これらはオプションの場合と推奨の場合があります（ブロック内でテキストによって示されます）。ローレベルモジュールの追加が必要な場合は、モジュールの説明に **Add** というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、**[New]** アイコンが表示され、可能な選択肢が表示されます。

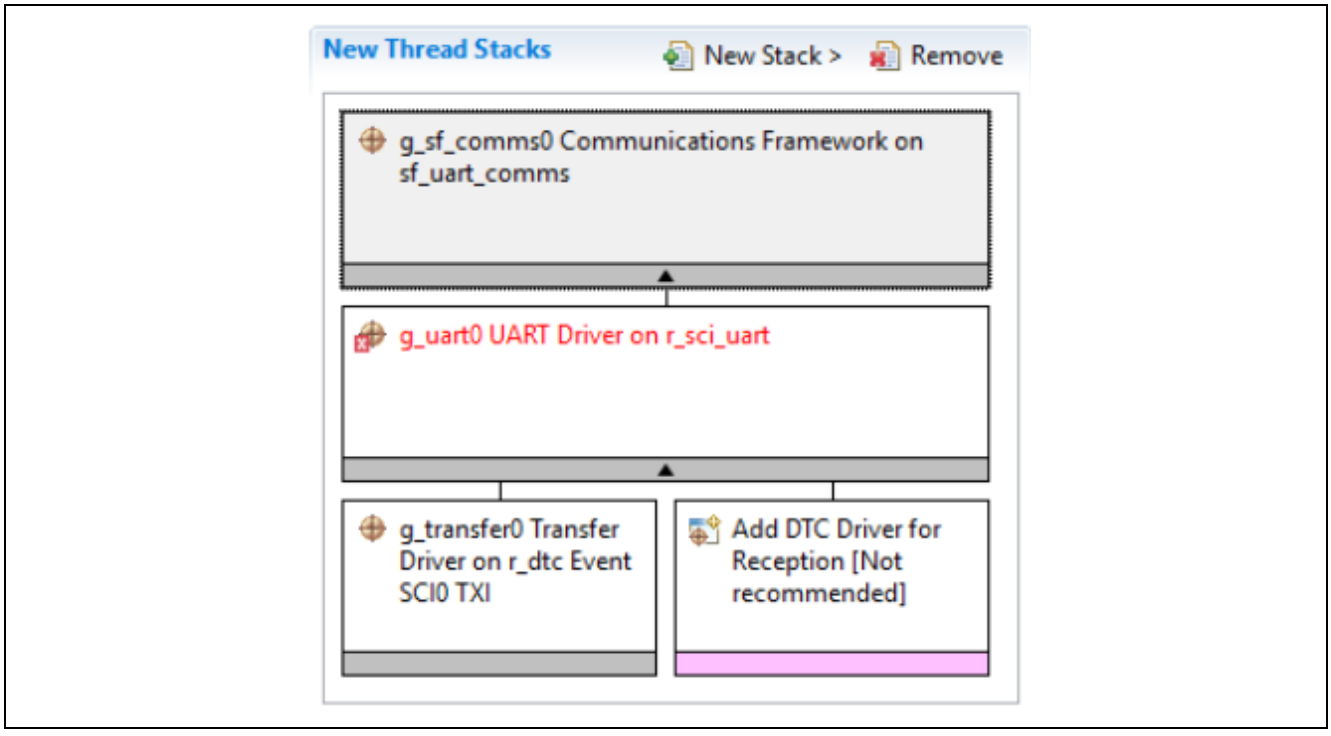

図 2　UART 通信フレームワークモジュールのスタック

5. UART 通信フレームワークモジュールの構成（Configuring the UART Communications Framework Module）

UART 通信フレームワークモジュールはシステム要求に合わせた設定（configured）が必要です。[SSP configuration] ウィンドウでは、割り込みや動作モードなどの必須の設定選択項目が自動的に識別されます。これらは、ローレベルモジュールが正常に作動するために設定される必要があります。変更しても競合が発生しないプロパティのみが変更可能ですが、その他のプロパティは「ロック」されていて変更できません。これらは ISDE 内の [Properties] ウィンドウで「ロック」されているプロパティとして、ロックアイコンで識別されます。この方法により構成プロセスが簡略化され、従来の手動による構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにあります。この内容を下表に示します。

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

表 4　sf_uart_comms の UART 通信フレームワークモジュールのスタックの設定

ISDE のプロパティ	値	説明
Parameter Checking	Default (BSP), Enabled, Disabled, Default: Default (BSP),	パラメータチェックを含めるかどうかを選択
Read Input Queue Size (4-Byte Words)	Default: 15	データ受信キューのバッファサイズ。sf_uart_comms はキュー管理に ThreadX キューを利用します。
Name	g_sf_comms0	UART 通信フレームワークモジュールの名前
Name of generated initialization function	sf_comms_init0	生成された初期化関数選択の名前
Auto Initialization	Enable, Disable Default: Enable	自動初期化の選択

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

UART 通信フレームワークのローレベルドライバの設定（Configuration Settings for the UART Communications Framework Lower-Level Modules）

通常、デフォルト設定から変更する必要があるローレベルドライバは少なく、これらはスレッドスタックブロックに赤いテキストで表示されます。一部の設定プロパティは、フレームワークが適切に作動するために特定の値に設定されており、それらはロックされていてユーザは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

表 5　r_sci_uart 上の UART HAL モジュールの設定

ISDE のプロパティ	値	説明
External RTS Operation	Enable, Disable Default: Disable	RTS 信号として使用する IOPORT ピンを有効にします。RTS 機能では、この設定パラメータを「Enable」(有効) にし、設定「Name of UART callback function for the RTS external pin control」(RTS 外部ピン制御のための UART コールバック関数名) を指定します。
Reception	Enable, Disable Default: Enable	SCI 上のすべての UART チャネルについて UART の受信を有効または無効にします。この設定パラメータに「Disable」(無効) を設定すると、コードサイズが小さくなります。UART 受信のためのコード部分がコンパイルされないためです。このパラメータは、個々の UARTT チャネルに対しては設定できません。
Transmission	Enable, Disable Default: Enable	SCI 上のすべてのチャネルについて UART 送信を有効または無効にします。この設定に「Disable」(無効) を設定すると、UART 送信のためのコード部分がコンパイルから除外されるため、コードサイズが小さくなりますが、この設定に「Disable」(無効) を設定できるのは、UART ポートとして機能する他の SCI チャネルが送信を行わない場合だけです。
Parameter Checking	Default (BSP), Enabled, Disabled Default: Default (BSP)	パラメータエラーチェックを有効または無効にします。
Name	g_uart0	SCI 上の UART モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Channel	Default: 0	SCI チャネル番号 (ch)
Baud Rate	Default: 9600	ボーレートの選択 (bps)
Data Bits	7 bits, 8 bits, 9 bits Default: 8 bits	UART データビット
Parity	None, Odd, Even Default: None	UART パリティビット
Stop Bits	1 bit, 2 bits Default: 1 bit	UART ストップビット

CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	SCI チャネル n の CTSn/RTSn ピンに対して CTS または RTS を選択します。SCI ハードウェアは、このピン上で CTS と RTS のいずれかの制御信号をサポートしますが、両方はサポートしません。CTS と RTS の両方を使用するアプリケーションでは、この設定パラメータに「CTS」を選択し、設定「External RTS Operation」(外部 RTS 操作) を有効にし、設定「Name of UART callback function for the RTS external pin control」(RTS 外部ピン制御のための UART コールバック関数名) を指定します。
Name of UART callback function to be defined by user	(Lock) NULL	名前は有効な C シンボルである必要があります。フレームワークから使用されるためロックされています。
Name of UART callback function for the RTS external pin control to be defined by user	Default: NULL	名前は有効な C シンボルである必要があります。
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	ボーレートクロック発信器ブロックで使用するクロックソースを選択します。
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	選択したチャネル n に対し SCKn ピン上でボーレートクロックを出力するためのオプション設定。
Start bit detection	Falling Edge, Low Level Default: Falling Edge	受信におけるスタートビット検出モード。この設定には、通常は「Falling Edge」(立ち下がりエッジ) を設定します。
Noise Cancel	Enable, Disable Default: Disable	RXDn ピン上でデジタルノイズキャンセルを有効にします。SCI のデジタルノイズフィルターブロックは、2 ステージのフリップフロップ回路で構成されます。詳細については、Renesas Synergy ハードウェアマニュアルのノイズキャンセルの章を参照してください。
Bit Rate Modulation Enable	Enable, Disable Default: Enable	ビットレート変調有効化の選択。
Receive FIFO Trigger Level	One, Max Default: Max	受信 FIFO トリガレベルの選択。
Receive Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	受信割り込み優先順位の選択。
Transmit Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	送信割り込み優先順位の選択。

Transmit End Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	送信終了割り込み優先順位の選択。
Error Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	エラー割り込み優先順位の選択。

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

表 6　r_dtc Event SCI0 TXI 上の転送ドライバの設定

ISDE のプロパティ	値	説明
Parameter Checking	Default (BSP), Enabled, Disabled Default: Default (BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	Default: .ssp_dtc_vector_table	DTC ベクタテーブルを維持するためのリンカセクション
Name	Default: g_transfer0	モジュール名
Mode	(Locked) Normal	モードの選択 (ロックされています)
Transfer Size	(Locked) 1 Byte	転送サイズの選択(ロックされています)
Destination Address Mode	(Locked) Fixed	宛先アドレスモードの選択(ロックされています)
Source Address Mode	(Locked) Incremented	ソースアドレスモードの選択(ロックされています)
Repeat Area (Unused in Normal Mode)	(Locked) Source	リピートエリアの選択(ロックされています)
Interrupt Frequency	(Locked) After all transfers have completed	割り込み頻度の選択(ロックされています)
Destination Pointer	(Locked) NULL	宛先ポインタの選択(ロックされています)
Source Pointer	(Locked) NULL	ソースポインタの選択(ロックされています)
Number of Transfers	(Locked) 0	転送回数の選択(ロックされています)
Number of Blocks (Valid only in Block Mode)	(Locked) 0	ブロック数の選択(ロックされています)
Activation Source (Must enable IRQ)	(Locked) Event SCI0 TXI	アクティベーションソースの選択(ロックされています)
Auto Enable	(Locked) False	自動有効化の選択(ロックされています)
Callback (Only valid with Software start)	(Locked) NULL	コールバックの選択(ロックされています)
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

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

表 7　r_dtc Event SCI0 RXI 上の転送ドライバの設定

ISDE のプロパティ	値	説明
Parameter Checking	Default (BSP), Enabled, Disabled Default: Default (BSP)	パラメータチェック用のコードをビルドに含めるかどうかを選択
Software Start	Enabled, Disabled Default: Disabled	ソフトウェアスタートの選択
Linker section to keep DTC vector table	Default: .ssp_dtc_vector_table	DTC ベクタテーブルを維持するためのリンカセクション
Name	Default: g_transfer1	モジュール名
Mode	(Locked) Normal	モードの選択 (ロックされています)
Transfer Size	(Locked) 1 Byte	転送サイズの選択 (ロックされています)
Destination Address Mode	(Locked) Incremented	宛先アドレスモードの選択 (ロックされています)
Source Address Mode	(Locked) Fixed	ソースアドレスモードの選択 (ロックされています)
Repeat Area (Unused in Normal Mode)	(Locked) Destination	リピートエリアの選択 (ロックされています)
Interrupt Frequency	(Locked) After all transfers have completed	割り込み頻度の選択 (ロックされています)
Destination Pointer	(Locked) NULL	宛先ポインタの選択 (ロックされています)
Source Pointer	(Locked) NULL	ソースポインタの選択 (ロックされています)
Number of Transfers	(Locked) 0	転送回数の選択 (ロックされています)
Number of Blocks (Valid only in Block Mode)	(Locked) 0	ブロック数の選択 (ロックされています)
Activation Source (Must enable IRQ)	(Locked) Event SCI0 RXI	アクティベーションソースの選択 (ロックされています)
Auto Enable	(Locked) False	自動有効化の選択 (ロックされています)
Callback (Only valid with Software start)	(Locked) NULL	コールバックの選択 (ロックされています)
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1~14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC ソフトウェアイベント割り込み優先順位の選択。

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

注：　Default は、DTC を利用しません。

5.1 UART 通信フレームワークモジュールのクロック構成（UART Communications Framework Module Clock Configuration）

UART 通信フレームワークは PCLKB をクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、ランタイムで CGC インタフェースを使用します。

5.2 UART 通信フレームワークモジュールのピン構成（UART Communications Framework Module Pin Configuration）

UART 通信フレームワークは、MCU のピンを使用し、選択したローレベル実装に基づいて外部デバイスと通信をおこないます。I/O ピンは、外部デバイスの要件に合うように選択して、構成する必要があります。次の表では[SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はローレベル実装のピンの選択例を示します。

注 : 動作モードの選択によって、使用可能な周辺回路信号と必要な MCU ピンが決定されます。

表 8　sf_uart_comms 上の UART 通信フレームワークモジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select **Peripherals > Connectivity: SCI > SCI8**

表 9　sf_uart_comms の UART 通信フレームワークモジュールのピン設定

Pin Configuration Property	設定値	説明
Pin Group Selection	Mixed, _A Only, _B Only	ピングループの選択
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, Smart Card Default: Disabled	UART 受信部実装では動作モードとして非同期 UART を選択します。
TXD_MOSI	None, P105 Default: None	TXD ピンに割り当てるピンを指定します。
RXD_MISO	None, P104 Default: None	RXD ピンに割り当てるピンを指定します。

注：設定例は、Synergy S7G2 MCU ファミリと SK-S7G2 キットを使用するプロジェクトの設定です。 他の Synergy MCU と Synergy Kits では、使用可能なピン構成の設定が異なる場合があります。

表 10　sf_uart_comms 上の UART 通信フレームワークのピン選択

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select **Peripherals > Connectivity: SCI > SCI3**

注：　上記の選択シーケンスは選択した実装に対応する例です。ターゲットハードウェアによっては他の選択シーケンスも可能です。

表 11　sf_uart_comms 上の UART 通信フレームワークモジュールのピン設定

Pin Configuration Property	設定値	説明
Pin Group Selection	Mixed, _A Only, _B Only Default: Mixed	ピングループの選択
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, Smart Card Default: Disabled	UART トランスミッタ実装では動作モードとして非同期 UART を選択します。
TXD_MOSI	None, P707, P409 Default: None	TXD ピンに割り当てるピンを指定します。
RXD_MISO	None, P706, P408 Default: None	RXD ピンに割り当てるピンを指定します。

注：これらの選択シーケンスは、選択された実装の例です。 ターゲットハードウェアによっては、他の方法も可能です。

6. アプリケーションでの UART 通信フレームワークモジュールの使用（Using the UART Communications Framework Module in an Application）

アプリケーションで UART 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1. `open` API を使用して UART 通信フレームワークを初期化します。
2. `lock` API を使用して連続通信のためにチャネルをロックします （必要な場合）。
3. `read` API を使用してデータを受信します。
4. `write` API を使用してデータを送信します。
5. `unlock` API を使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6. `close` API を使用して、チャネルを閉じます。

上記の手順を、次の図の通常の動作フロー図に示します。

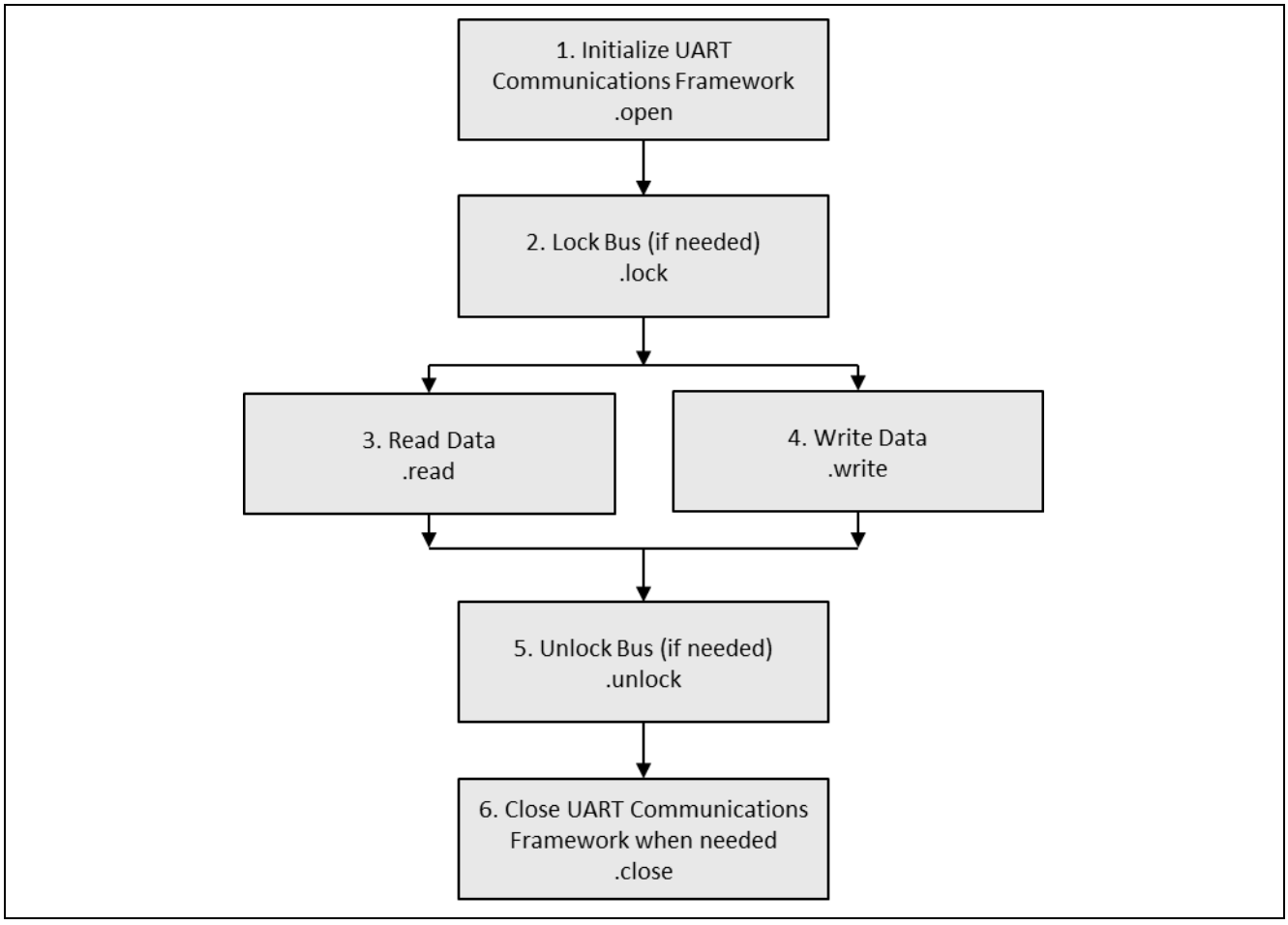

図 3　通常の UART 通信フレームワークアプリケーションのフロー図

7. UART 通信フレームワークモジュールのアプリケーションプロジェクト（The UART Communications Framework Module Application Project）

このモジュールガイドで説明するアプリケーションプロジェクトを実際に使って、設計全体の手順を示します。このプロジェクトは、このドキュメントの末尾にある「参考情報」の章の説明に掲載されているリンクから入手します。ISDE でアプリケーションプロジェクトをインポートして開き、通信フレームワークモジュールに対応する設定項目を表示することができます。UART_FW_MG_AP_NORMAL_OPERATION プロジェクトは、このフレームワークを使用した UART データの単純な送受信を行う方法を示します。追加プロジェクト UART_FW_MG_AP_HARDWARE_CONTROL は、ハードウェアフロー制御（RTS/CTS）の使用方法を示す目的で提供されています。ハードウェアフロー制御の追加のみが、二つのプロジェクトの違いです。この章では、UART 通信フレームワークを使用した、単純な送受信を行う方法を示します。また、完成した設計で、通信フレームワーク API を示しているコードを見ることもできます。

このアプリケーションプロジェクトは、通信フレームワーク API （Communication Framework API ）の一般的な使用方法を示します。このプロジェクトは 2 つのスレッドで構成されています。1 つはデータを送信し、もう 1 つはデータを受信します。送信スレッド（transmitter thread）は定期的に「Hello World」と書き込み、受信スレッド（receiver thread）は UART データの受信を継続的に待つように設定してあります。このフレームワークは、送信と受信すべての割り込みを処理します。したがって、データの送受信にユーザ指定のコールバック関数を必要としません。受信スレッドが UART データを受信した時点で、共通のセミホスト機能を使用して、そのデータをデバッグコンソールに出力します。次の表は、このアプリケーションプロジェクトが使用する関連ソフトウェアとハードウェアの対象バージョンを示します。

表 12　このアプリケーションプロジェクトが使用するソフトウェアとハードウェアのリソース

リソース	リビジョン	説明
e2 studio	5.3.1 またはそれ以降	統合ソリューション開発環境 (ISDE)
IAR Workbench	7.71.2 またはそれ以降	IAR Workbench IDE for Synergy Platform
SSP	1.2.0 またはそれ以降	Synergy ソフトウェアパッケージ
SSC	5.3.1 またはそれ以降	Synergy Standalone Configurator
SK-S7G2	v3.0 と v3.1	スタータキット

次の図に、このアプリケーションプロジェクトの基本的なフロー図を示します。

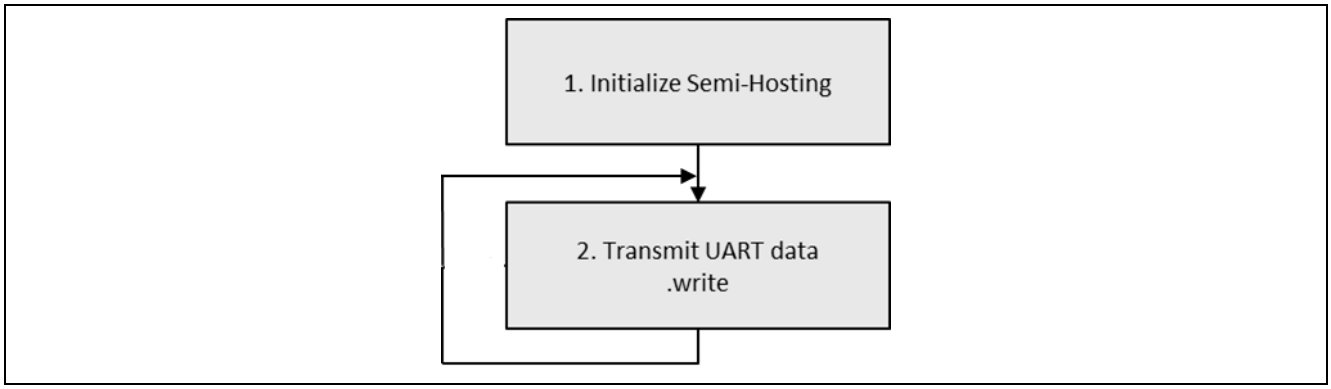

図 4　UART 通信フレームワークモジュールのアプリケーションプロジェクトで使用する送信スレッドのフロー図

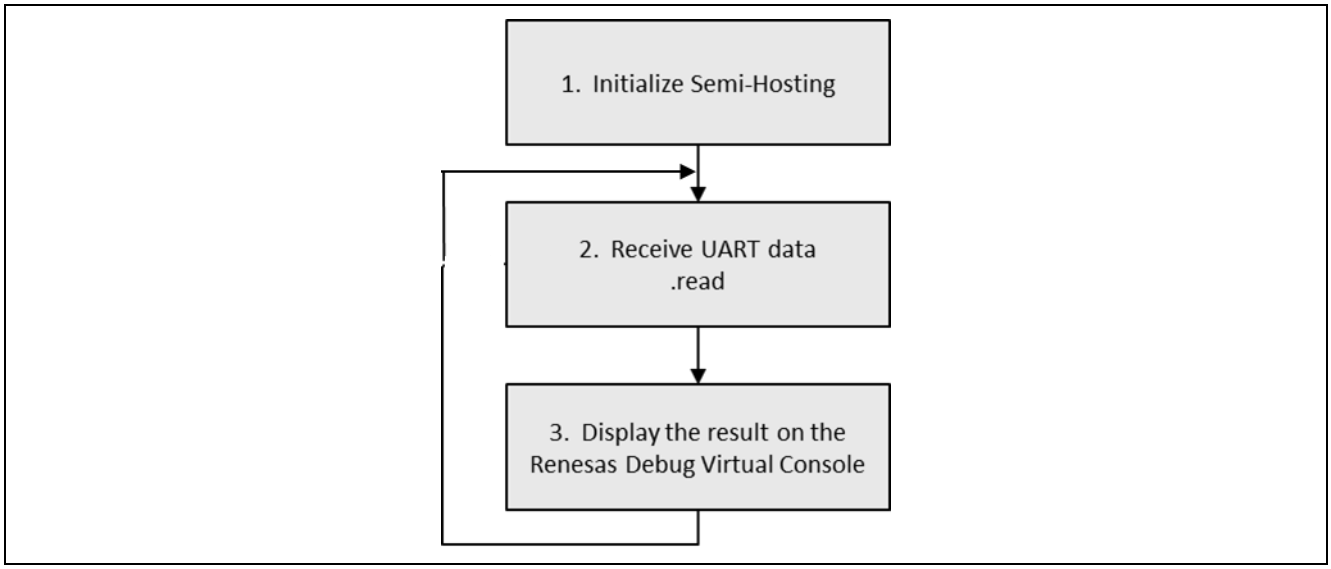

図 5　UART 通信フレームワークモジュールのアプリケーションプロジェクトで使用する受信スレッドのフロー図

アプリケーションプロジェクト全体は、このドキュメントの末尾にある「参考情報」の章に掲載されているリンクにあります。<UART_Framework_MG>.c ファイルは、このプロジェクトを一旦 ISDE にインポートすると、プロジェクト内に格納されます。ISDE でこのファイルを開くことができますし、API の主な使い方を確認するための説明も参照できます。

このプロジェクトは、以下の 2 つの UART 通信フレームワークモジュールを使用します。

1. g_sf_comms0：トランスミッタスレッドである Uart TX Thread がデータを送信するときに使用します。

- g_sf_comms0 は、SCI チャネル 8 を使用するように設定します。
- P105 を UART の送信ピンとして使用します。

2. g_sf_comms1：レシーバスレッドである Uart RX Thread がデータを受信するときに使用します。

- g_sf_comms1 は、SCI チャネル 3 を使用するように設定します。
- P706 を UART の受信ピンとして使用します。

このプロジェクトでは、UART_TX_entry.c の g_sf_comms0 がデータを送信し、UART_RX_entry.c の g_sf_comms1 がそのデータを受信します。**プロジェクトが動作するように、g_sf_comms0 の送信ピンである P105 を、g_sf_comms1 の受信ピンである P706 に接続する必要があります。**

UART_Framework_MG.c では、sf_uart_comms 関数に対応する、抽象化した関数（abstracted functions）を複数記述しています。UART_Framework_MG.c にある関数を使用して、このガイドに示す UART 機能を実行します。

Uart TX Thread は UART_TX_entry.c 内の処理を実行し、その中から UART_FW_TX() を呼び出します。この関数は、UART_TX_AP.c 内にあります。ここで、sf_comms0_write 関数を使用して、「Hello World」を定期的に送信します。この関数は、送信しようとするデータを SF_UART_COMMS モジュールに書き込み、このモジュールは UART 経由でそのデータを送信します。書き込みが成功した場合、Renesas Virtual Debug Console (ルネサスデバッグ仮想コンソール) に「UART transmit」(UART 送信) と出力し、成功したことを示します。タイムアウトパラメータとして、TX_WAIT_FOREVER を指定してあります。したがって、このスレッドは UART ドライバへのアクセスを取得するまで、待ち状態になります。

Uart RX Thread は UART_RX_entry.c 内の処理を実行し、その中から UART_FW_RX() を呼び出します。この関数は、UART_RX_AP.c 内にあります。ここで、この関数は while 無限ループ（infinite loop）に入ります。このループは、「Hello World」に対応できる 14 バイトの読み取り長（read length）、および

TX_WAIT_FOREVER のタイムアウトを指定して sf_comms1_read を呼び出します。したがって、このスレッドは UART から 14 バイトを読み取るまで、待ち状態になります。UART データを受信した時点で、このデータもコンソールに出力します。その後、レシーバスレッドは、UART データを再度受信するために保留状態になります。

注： この説明は、Synergy ソフトウェアパッケージ内のデバッグコンソールで `printf()` を使用する方法をユーザが理解していることを想定しています。このような経験がない場合、このドキュメントの末尾にある「参考情報」の章で紹介しているナレッジベースの記事、「How do I Use Printf() with the Debug Console in the Synergy Software Package」(Synergy ソフトウェアパッケージのデバッグコンソールで Printf() を使用する方法)を参照してください。代わりに、デバッグモードで変数ウォッチ機能を使用して結果を表示することもできます。

対象ボードや MCU の必要な動作と物理プロパティをサポートするために、このアプリケーションプロジェクトではいくつかの重要なプロパティを設定しています。以下にそれらのプロパティと、このプロジェクトで設定した値を示します。実践的な練習として、このアプリケーションプロジェクトを開き、[Properties] ウィンドウでこれらの設定を表示することもできます。

表 13　アプリケーションプロジェクト (Receive) に対応する通信フレームワークの設定項目

ISDE のプロパティ	設定済みの値
Read Input Queue Size (4-Byte Words)	15
Name	g_sf_comms1

表 14　アプリケーションプロジェクト (Receive) に対応する通信フレームワーク – ローレベル UART ドライバの設定項目

ISDE のプロパティ	設定済みの値
Name	g_uart1
Channel	3
Baud Rate	9600
Data Bits	8bits
Parity	None
Stop Bits	1 bit
CTS/RTS Selection	RTS(CTS is disabled)
Receive Interrupt Priority	Priority 3
Transmit Interrupt Priority	Priority 3
Transmit End Interrupt Priority	Priority 3
Error Interrupt Priority	Priority 3

表 15　アプリケーションプロジェクト (Transmit) に対応する通信フレームワークの設定項目

ISDE のプロパティ	設定済みの値
Read Input Queue Size (4-Byte Words)	15
Name	g_sf_comms0

表 16　アプリケーションプロジェクト (Transmit) に対応する通信フレームワーク – ローレベル UART ドライバの設定項目

ISDE のプロパティ	設定済みの値
Name	g_uart0
Channel	8
Baud Rate	9600
Data Bits	8bits
Parity	None
Stop Bits	1 bit
CTS/RTS Selection	RTS(CTS is disabled)
Receive Interrupt Priority	Priority 3
Transmit Interrupt Priority	Priority 3
Transmit End Interrupt Priority	Priority 3
Error Interrupt Priority	Priority 3

UART ピンも設定する必要があります。このアプリケーションで使用できるように、SCI3 と SCI8 を設定済みです。

表 17　UART SCI3 のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select Peripherals > Connectivity: SCI > SCI3

表 18　UART SCI3 のピン構成設定

Pin Configuration Property	設定済みの値
Pin Group Selection	Mixed
Operation Mode	Asynchronous UART
TXD_MOSI	P707
RXD_MISO	P706

表 19　UART SCI8 のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select Peripherals > Connectivity: SCI > SCI8

表 20　UART SCI8 のピン構成設定

Pin Configuration Property	設定済みの値
Pin Group Selection	Mixed
Operation Mode	Asynchronous UART
TXD_MOSI	P105
RXD_MISO	P104

8. 対象アプリケーションの UART 通信フレームワークモジュールのカスタマイズ（Customizing the UART Communications Framework Module for a Target Application）

いくつかの設定項目は通常、アプリケーションプロジェクトで示している値に対し、ユーザが変更を加えます。たとえば、名前、チャネル、ボーレート、データビット数、パリティ、ストップビット数、または CTS/RTS 選択を更新することにより、ユーザは通信フレームワークの構成を容易に変更できます。

カスタマイズ可能な 1 つの方法は、ターゲットアプリケーションに対してハードウェアフロー制御を実装することです。SCI ハードウェアモジュールが一度にサポートするハードウェアフロー制御は、RTS と CTS のいずれかの信号のみです。CTS と RTS は CTSn/RTSn ピンにマルチプレクスされているので、ユーザは使用状況に応じて、どちらか 1 つのハードウェアフロー制御信号を排他的に使用することができます。SCI ドライバモジュールで ローレベル UART を使用した場合、RTS 信号に対する追加のピンを有効にする仕様拡張ができ、CTS と RTS 両方の信号を制御することができます。このモードを有効にするには、UART 通信フレームワークを次のように設定します。

表 21　アプリケーションプロジェクト (Receive) に対応する通信フレームワークの設定項目

ISDE のプロパティ	設定済みの値
Read Input Queue Size (4-Byte Words)	15
Name	g_sf_comms1

表 22　アプリケーションプロジェクト (Receive) に対応する通信フレームワーク – ローレベル UART ドライバの設定項目

ISDE のプロパティ	設定済みの値
External RTS Operation	**Enable**
Reception	Enable
Transmission	Enable
Parameter Checking	Default (BSP)
Name	g_uart1
Channel	3
Baud Rate	9600
Data Bits	8bits
Parity	None
Stop Bits	1 bit
CTS/RTS Selection	CTS
Name of UART callback function for the RTS external pin	**uart_rts_cb**
Receive Interrupt Priority	Priority 3
Transmit Interrupt Priority	Priority 3
Transmit End Interrupt Priority	Priority 3
Error Interrupt Priority	Priority 3

表 23　アプリケーションプロジェクト (Receive) に対応する通信フレームワークの設定項目

ISDE のプロパティ	設定済みの値
Read Input Queue Size (4-Byte Words)	15
Name	g_sf_comms0

表 24　アプリケーションプロジェクト (Receive) に対応する通信フレームワーク – ローレベル UART ドライバの設定項目

ISDE のプロパティ	設定済みの値
External RTS Operation	**Enable**
Reception	Enable
Transmission	Enable
Parameter Checking	Default (BSP)
Name	g_uart0
Channel	8
Baud Rate	9600
Data Bits	8bits
Parity	None
Stop Bits	1 bit
CTS/RTS Selection	**CTS**
Name of UART callback function for the RTS external pin	NULL
Receive Interrupt Priority	Priority 3
Transmit Interrupt Priority	Priority 3
Transmit End Interrupt Priority	Priority 3
Error Interrupt Priority	Priority 3

7 章で設定済みの UART ピンに加えて、1 本の GPIO ピンを外部 RTS ピンとして設定します。このプロジェクトでは、P511 を設定しました。**[Pins]** タブに移動し、**[Ports] > [P511] > [Mode]: [Output mode (Initial High)]** を選択します。

また、SCI 3 (UART データの送信に使用) で使用するように CTS ピンを設定します。

表 25　UART SCI3 のピン選択シーケンス

Resource	ISDE Tab	Pin Selection Sequence
UART	Pins	Select Peripherals > Connectivity: SCI > SCI3

表 26　UART SCI3 のピン構成設定

Pin Configuration Property	設定済みの値
Pin Group Selection	Mixed
Operation Mode	Custom
TXD_MOSI	P105
RXD_MISO	P104
CTS	P107

RTS 外部ピンを制御するために、コールバック関数を使用します。この関数は、`UART_RX_AP.c` ファイルの末尾にあります。UART SCI ドライバの RXI ISR が発生して RTS ピンをハイに設定したときに、このコールバック関数が呼び出されます。また、この関数は、終了する前に RTS ピンをローに設定します。

ハードウェア制御 `UART_MG_AP_HARDWARE_CONTROL` を使用する UART に対応するアプリケーションプロジェクト全体は、このドキュメントの末尾にある「参考情報」の章に掲載されているリンクにあります。ISDE でこのプロジェクトを開き、該当する設定項目の確認と、RTS 外部ピンに対応するコールバック関数の使用方法を観察することができます。

このアプリケーションプロジェクトを実行するときは、P511 (RTS) を P107 (CTS) に接続してください。RTS ピンからの LOW 出力が CTS に入力されることで、受信側が受信可能であることを送信側に伝えます。

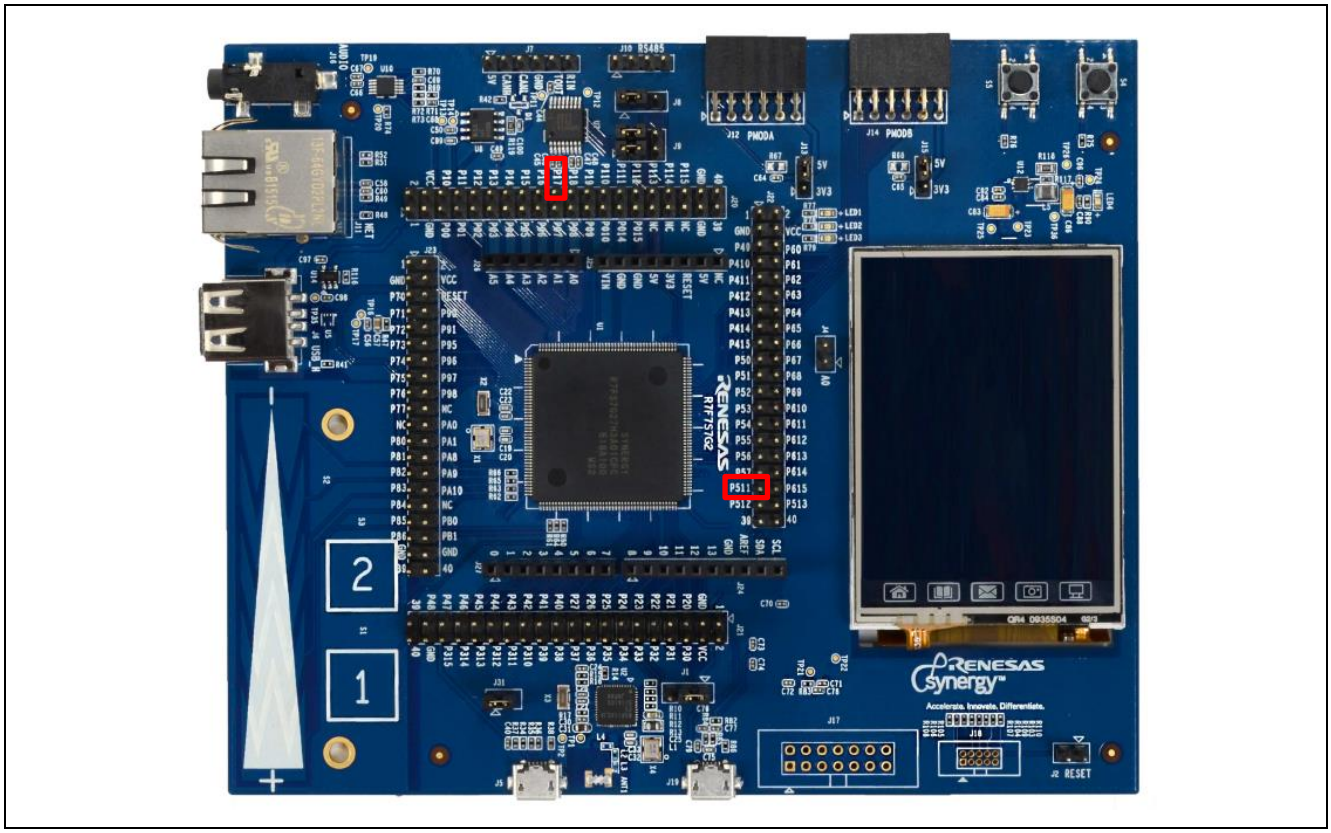

図 6　SK-S7G2 MCU に接続する複数のピン

9.　UART 通信フレームワークモジュールのアプリケーションプロジェクトの実行（Running the UART Communications Framework Module Application Project）

UART 通信フレームワークのアプリケーションプロジェクトを実行し、対象キットでその動作を確認ために、ISDE にこのプロジェクトをインポートし、コンパイルしてデバッグを実行します。e² studio または IAR Embedded Workbench® for Renesas Synergy™ にプロジェクトをインポートし、アプリケーションをビルドして実行する手順については、『Synergy プロジェクトインポートガイド』(r11an0023eu0116-synergy-ssp-import-guide.pdf、このパッケージに付属) を参照してください。新しいプロジェクト内で通信フレームワークアプリケーションを実装するには、下記の対象キットの定義、設定、ファイルの自動生成、コードの追加、コンパイル、デバッグを行うための手順に従います。これらの手順に従うことで SSP を使用する開発プロセスがより実践的に身に付きます。また、このガイド全体を読み通すことで理論の習得に役立ちます。

注：　Synergy 開発プロセスの基本的な流れを経験したことのあるユーザにとって、以下の手順は十分なものです。これらの手順をまだ理解していない場合、これらの手順を実行する説明について、このドキュメントの末尾にある「参考情報」の章に掲載されている『SSP ユーザーズマニュアル』の「Getting Started with SSP」(SSP 入門) という章を参照してください。

通信フレームワークのアプリケーションプロジェクトを作成し、実行するために、以下の手順に従ってください。

1. **SK-S7G2** に対応する新しい Renesas Synergy プロジェクトを作成し、`UART_FW_MG_AP_NORMAL_OPERATION` という名前を付けます。
2. **[Threads]** タブを選択します。
3. 新しいスレッドを追加し、次のように設定します。
   a. Symbol: `UART RX`
   b. Name: **Uart RX Thread**
4. **Uart RX Thread** に通信フレームワークを追加し、表 10 と 表 11 のように設定します。
5. 別の新しいスレッドを追加し、次のように設定します。
   a. Symbol: `UART_TX`

b. Name: **Uart TX Thread**

6. このスレッドにも UART 通信フレームワークを追加し、関連する表ですでに示した方法で設定します。
7. **[Pins]** タブに移動し、SCI3 と SCI8 に対応するピンが、関連する表ですでに示した方法で設定されていることを確認します。
8. **[Generate Project Content]** ボタンをクリックします。
9. 付属している `UART_RX_entry.c` と `UART_TX_entry.c` からコードを取得し、上書きする形でコピーします。また、他の付属ファイルである `UART_RX_AP.c`、`UART_RX_AP.h`、`UART_TX_AP.c`、`UART_TX_AP.h`、`UART_FRAMEWORK_MG.c`、および `UART_Framework_MG.h` を追加します。
10. プロジェクトをコンパイルします。
11. micro USB ケーブルを SK-S7G2 MCU スタータキットの J19 につなぎ、ホスト PC に接続します。
12. アプリケーションのデバッグを開始します。
13. 次の場所で出力を確認できます。**Renesas Debug Virtual Console** (ルネサスデバッグ仮想コンソール)。

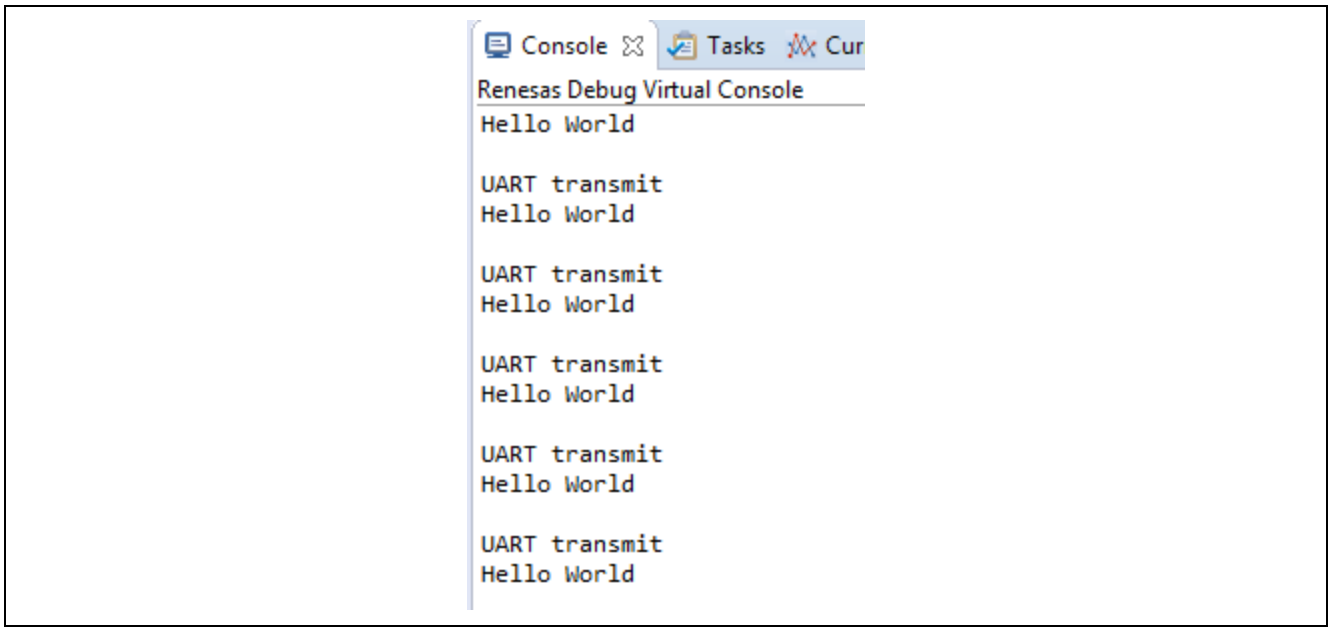

図 6　通信フレームワークのアプリケーションプロジェクトのサンプル出力

10. UART 通信フレームワークモジュールのまとめ（UART Communications Framework Module Conclusion）

このモジュールガイドは、サンプルプロジェクトでモジュールの選択、追加、設定、使用を行うために必要な情報全般を説明しました。従来の組み込みシステムでは、これらの手順を理解することに多くに時間を必要とし、間違いが起こりやすい操作でした。Renesas Synergy プラットフォームの登場により、これらの手順の所要時間が短くなり、設定項目の競合や、ローレベルドライバの誤った選択のような一般的な誤りを防止できるようになりました。アプリケーションプロジェクトで示したように、 高レベル API を使用することで高レベルな制御できますので、従来の開発環境で必要とされた時間が不要になり、開発時間を短縮できます。また、特定の状況ではローレベルドライバを作成することもできます。

11. UART 通信フレームワークモジュールの次の手順（UART Communications Framework Module Next Steps）

シンプルな UART 通信フレームワークのプロジェクトをマスターした後、より複雑なサンプルを確認することができます。他のインタフェースに実装されている通信フレームワークの使用方法を検討することもできます。Renesas Synergy ソフトウェアプラットフォームは、UX とイーサネットの各インタフェース上でもこのフレームワークを実装しています。

12. UART 通信フレームワークモジュールの参考情報（UART Communications Framework Module Reference Information）

『SSP ユーザーズマニュアル』：SSP distribution package の一部として html 形式を入手できるほか、Synergy WEB サイト（https://www.renesas.com/jp/ja/products/synergy.html）の SSP サイト（https://www.renesas.com/jp/ja/products/synergy/software/ssp.html） から PDF 版を入手することもできます。

本プロジェクト（ソフトウェア）は英文ドキュメントとともに、下記サイトから入手できます。未登録の場合は MyRenesas への登録が必要です。

https://www.renesas.com/jp/ja/software/D6002120.html

さらに関連参考資料、リソースに関する最新情報は Synergy WEB サイト（下記）から入手できます。
日本語版：https://www.renesas.com/jp/ja/products/synergy.html

英語版：https://www.renesas.com/us/en/products/synergy.html

ホームページとサポート窓口

サポート:　https://synergygallery.renesas.com/support

テクニカルサポート：

- アメリカ：https://www.renesas.com/en-us/support/contact.html
- ヨーロッパ：https://www.renesas.com/en-eu/support/contact.html
- 日本：https://www.renesas.com/ja-jp/support/contact.html

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.02	2018.11.14		・第 1.02 版 発行 ・第 2 章～6 章：SSP v1.5.0 UM 英文版から和訳 ・第 7 章～：英文版（r11an0192eu0102-synergy-uart-comms-fmwk-mod-guide Rev1.02、発効日 2018 年 1 月 9 日）を和訳 ・第 12 章：シナジーナレッジデータベースの HTTP 変更

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 6.0