Renesas Synergy™ プラットフォーム

コンソールフレームワークモジュールガイド

R11AN0110JU0101
Rev.1.01
2019.06.21

(注1)本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。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。

(注 2)本資料の第 6 章まで(要旨除く)の日本語訳は、「Synergy™ Software Package (SSP) v1.5.0 ユーザーズマニュアル モジュール概要編(参考資料)」の第 4 章「モジュールの概要」に掲載されていますのでそちらを参照ください 。

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。

コンソールフレームワーク(Console Framework)は、ThreadX® RTOS を使用した、メニュー形式のコンソールコマンドラインインタフェース (command line interface : CLI) 向け API です。コンソールフレームワークモジュールは、UART、USB、イーサネット Telnet 接続に対応するハードウェアオプションに接続するロウレベル通信インタフェース(communications interface)を使用します。コンソールフレームワークモジュールには、ユーザ定義のコマンドメニューや各種 API があります。これにより、プロンプト表示、メニューコマンドの識別とそれらのコマンドに対応するコールバックの発行、および入力文字列の読み取り、書き込み、解析を実行することができます。

目次

1. **コンソールフレームワークモジュールの特徴（Console Framework Module Features）**
2. **コンソールフレームワークモジュール API の概要（Console Framework Module APIs Overview）**
3. **コンソールフレームワークモジュールの動作の概要（Console Framework Module Operational Overview）**
4. **アプリケーションへのコンソールフレームワークモジュールの組み込み（Including the Console Framework Module in an Application）**
5. **コンソールフレームワークモジュールの構成（Configuring the Console Framework Module）**
6. **アプリケーションでのコンソールフレームワークモジュールの使用（Using the Console Framework Module in an Application）**
7. **コンソール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 Console Framework Module Application Project）**

このモジュールガイドに関連するアプリケーションプロジェクトでは、サンプルアプリケーションに関する手順を示しています。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、コンソール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。また、完成した設計で、コンソールフレームワークモジュール API を示すために使用している (console_framework_callback.h と console_framework_mg_api.h 内の) コードを確認することもできます。

このアプリケーションプロジェクトは、コンソールフレームワークモジュール API の一般的な使用方法を示しています。(次の図に、このアプリケーションプロジェクトのコンソールフレームワークスタックを示します。)コンソールフレームワークのスタックを g_sf_console0 というスレッドに追加し、ローレベルドライバとして USB 実装が選択されます。SK-S7G2 キットを BSP として使用しており、各種のインターコネクトリソースはすべて設定済みです。Windows 10 を使用している場合、ISDE プロンプトが指定する USB 割り込みを有効にすることと、 [Device Class] を [Miscellaneous] に変更することです。次の表に、必要な設定を示します。

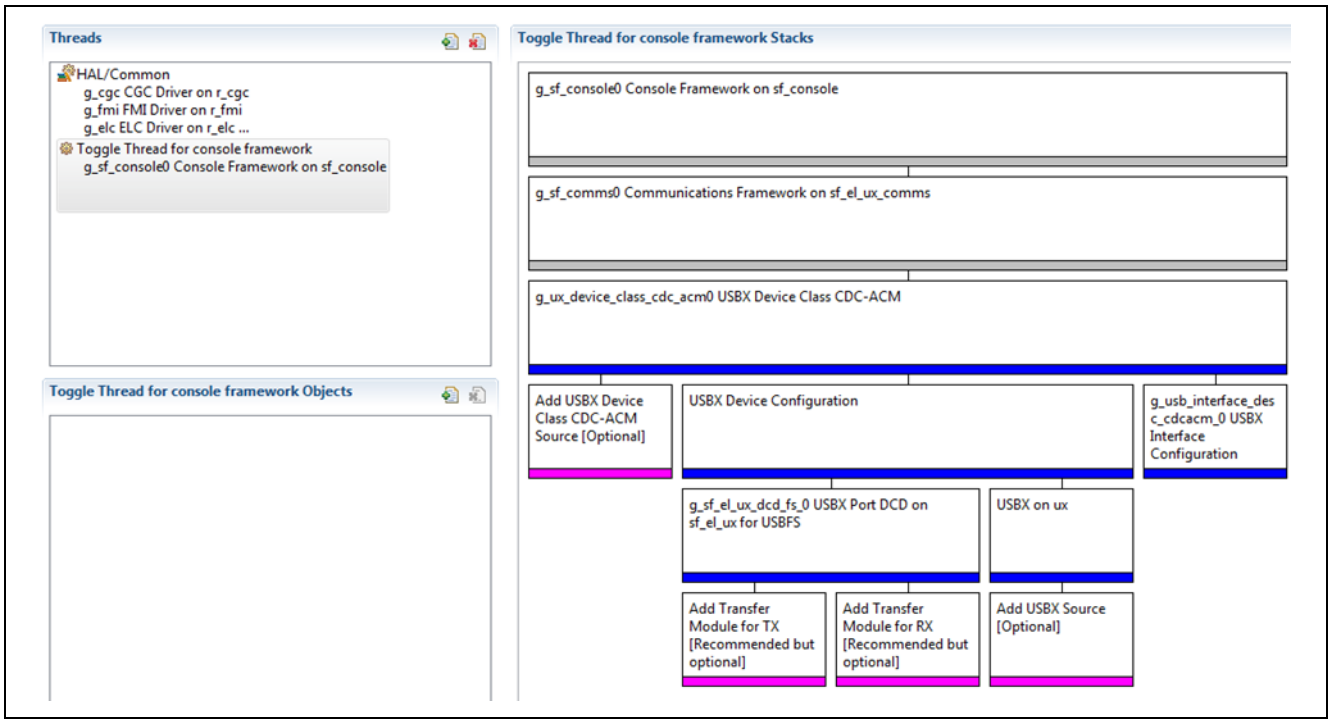

図 7.　アプリケーションプロジェクトのコンソールフレームワークスタック: USB 通信オプションが選択済み

表 24　アプリケーションプロジェクトへの、sf_el_ux 上の USBX Port DCD HS または FS の構成設定の追加

Resource	ISDE Tab	プロパティ / 構成設定
g_sf_el_ux_dcd_fs0 USBX Port DCD on sf_el_ux orUSBFS	Threads Stack/ Properties	Full Speed Interrupt Priority / Priority 4

LED をトグルさせる（点灯と消灯を繰り返す）コンソールフレームワークのシンプルな使い方では、コードは 1 ページに収まります。このシンプルな設計では、prompt API が TOGGLE コマンドを検出した場合、この API は TOGGLE コマンドに関連付けられているコールバックを呼び出し、ターゲットキット上の LED をトグルします。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ソフトウェアとハードウェアのバージョンを示します。また、表の後の図で、LED トグルコンソールフレームワーク（LED toggle console framework）実装のシンプルなブロック図を示します。

表 25　アプリケーションプロジェクトで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)
SSP	1.2.0 またはそれ以降	Synergy ソフトウェアプラットフォーム
IAR EW for Synergy	7.71.2 またはそれ以降	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.3.1 またはそれ以降	Synergy Standalone Configurator
SK-S7G2	v3.0 ,v3.1 またはそれ以降	スタータキット

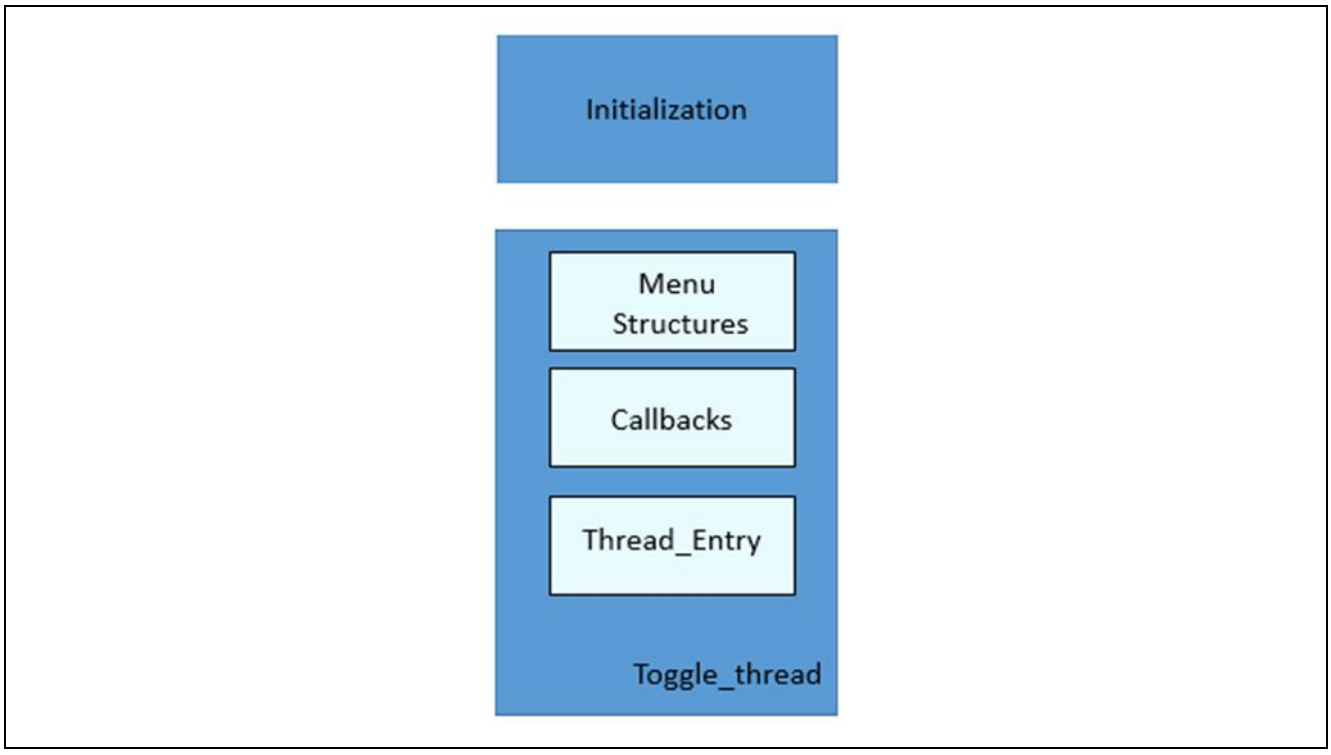

図 8.　コンソールフレームワークの Toggle コマンド実装のブロック図

toggle_thread_entry.c ファイルは、このプロジェクトを ISDE にインポートした後、プロジェクト内に置かれます。ISDE でこのファイルを開き、API の主な使い方に関する以下の説明を参照できます。Toggle コマンドコンソールフレームワークをベースとする実装は、最初にシステムの初期化を行います。ここでは、open コマンドと他の事務的な関数（housekeeping functions）を実行します。トグルスレッドルーチン（toggle-thread routine）は、Toggle コンソールフレームワークの残りの機能を実装します。これらの要素は、コンソールフレームワークの実装全般で使用します。このサンプルプロジェクトでは、toggle_thread.c ファイルの先頭に、(コマンドとメニューを定義する) メニュー構造体（menu structure）が実装されています。その後に、コマンドに関連するコールバックが続きます。このサンプルでは、Toggle コマンドに関連する一つのコールバックが、オンボード LED を駆動する出力信号をトグルします。最後に、thread_entry ルーチンを呼び出します。このサンプルで必要なのは、prompt API を 1 回呼び出すことのみです。

toggle_thread_entry.c 内にあるコードの最初のセクションはヘッダファイルと、prompt API の呼び出しを持っています。コールバック関数、メニュー、コマンドは console_framework_callback.h ファイル内で定義されています。このサンプルでは、コマンド構造体は、TOGGLE コマンド、ヘルプの説明テキスト、コールバック関数の定義、および NULL コンテキストエントリ（NULL context entry）を含む一つのエントリが存在します。これより複雑なシステムであっても、多くのメニュー項目をこの構造体に簡単に追加することができます。

console_framework_callback.h 内にある次のセクションのコードは、ルートメニュー構造体（root menu structure）を記述しています。このサンプルでは、メニューを 1 個だけ使用し、サブメニューが存在しないため、NULL エントリがあります。メニュー名は **Command** であり、メニューの数は、コマンド配列全体のサイズ（size of the entire command array）を一つの配列要素（single array element）のサイズで割った値に等しくなります。メニュー構造体は、コマンド構造体配列の最初の要素へのポインタがあります。以上でこのサンプルに対応するメニュー全体を定義しました。

注: 開発者用サンプルプロジェクトでは、より複雑なメニュー構造体を使用しています。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」の章の参照先にあります。シンプルなサンプルの基本的なメニュー構造体を理解した後、開発者用サンプルコードで使用しているメニューを参照し、複数レベルのメニューの実装、コマンド引数（command arguments）の使用法、プロジェクト全体での分散コマンド構造体（distributed command structures）の使用法、read と write の各 API の使用法を確認することができます。

次のセクションのコードは、toggle コマンドを実装するコールバック関数を定義しています。コンソールフレームワークのすべてのコマンドは、コールバック関数の形で実装しています。toggle コマンドは LedsGet BSP 呼び出し（LedsGet BSP call）を使用して LED のリストを取得します。その後、最初の LED ドライバ端子の値を読み取り、その値を反転して同じ端子に書き込むことで、LED をトグルします。

最後のセクションのコードは、スレッドエントリ（thread entry）であり、シンプルなコンソールフレームワーク全体を 1 行のコードで実装しています。prompt API はインスタンス制御構造体 (g_sf_console0.p_ctrl) をメイン引数として呼び出されます。インスタンス構造体（instance structure）は、スレッドの初期化中に自動的に open API により生成され、このコードサブセクション（code sub-section）には表示されません。他の引数（other arguments）は、プロンプトに対応する有効なメニューコマンドへのポインタと NULL です。前者は、このサンプルでは必要ではありません。後者は、コマンドを 1 個だけ使用するため NULL となっています。より複雑なコンテキスト依存メニュー（context sensitive menu）において、このポインタは使用可能な複数のコマンドのサブセットと、待ち時間を識別します。(このサンプルでは、待ち時間は FOREVER です。プロンプトに対するユーザの応答にタイムアウトを適用しないからです。一方で、たとえば自動化したテストシステムでは、エラーチェックメカニズムとして、コマンドとコマンドの間に既知の最大遅延時間を使用することが考えられます。)

8. ターゲットアプリケーションに対応するコンソールフレームワークモジュールのカスタマイズ（Customizing the Console Framework Module for a Target Application）

コンソールフレームワークは、さまざまなメニュー、さまざまなスレッドエントリ名、さまざまなインスタンス名、および通信フレームワークスタックモジュールに対応する様々な設定項目に合わせてカスタマイズすることができます。 (または異なる通信フレームワークに合わせて)

8.1 メニュー構造体（Menu Structures）

コンソールフレームワークは主に、メニュー構造体とコマンド構造体を使用してカスタマイズします。他のアプリケーションにおいても、これらの構造体を容易に変更できます。他のシステムでこのサンプルコードを動作させるために必要と

なる変更は、ヘッダファイル名やインスタンス名を変更となります。また、他の通信フレームワークを使用する場合、低レベルモジュールを設定する必要もあります。

8.2　スレッドのエントリ名（Thread Entry Name）

スレッドエントリ名が、ヘッダファイルと整合した名前であることを確認してください。このサンプルでは、関連するプロパティエントリ内で定義したスレッド名は、toggle_thread です。必要なインクルードファイル定義は次のとおりです。

#include toggle_thread.h.

8.3　インスタンス名（Instance Name）

API 呼び出し（API call）に対応するインスタンス名は、設定の際に割り当てた名前に一致している必要があります。このサンプルで、インスタンス名は g_sf_console0 と定義してあり、API 呼び出しは、API 定義の場合 g_sf_console0 と、p_ctrl の場合は下記のインスタンス定義を使用します。

```
g_sf_console0.p_api->prompt(g_sf_console0.p_ctrl, NULL,TX_WAIT_FOREVER);
```

8.4　通信フレームワークモジュールの選択と設定（Communications Framework Module Selection and Configuration）

コンソールフレームワークは通信フレームワークに対する標準的なインタフェースを使用します。このため、次の図に示すコンソールフレームワークスタックの場合のように、カスタムアプリケーションは通信フレームワークを追加するときにさまざまなオプション (Telnet、USB、または UART) を選択できます。通信フレームワークを変更する場合、Toggle コマンドコンソールフレームワークを編集する必要はありません。新しい通信フレームワークが必要とする設定の差は、コンフィギュレータを使用して入力することです。

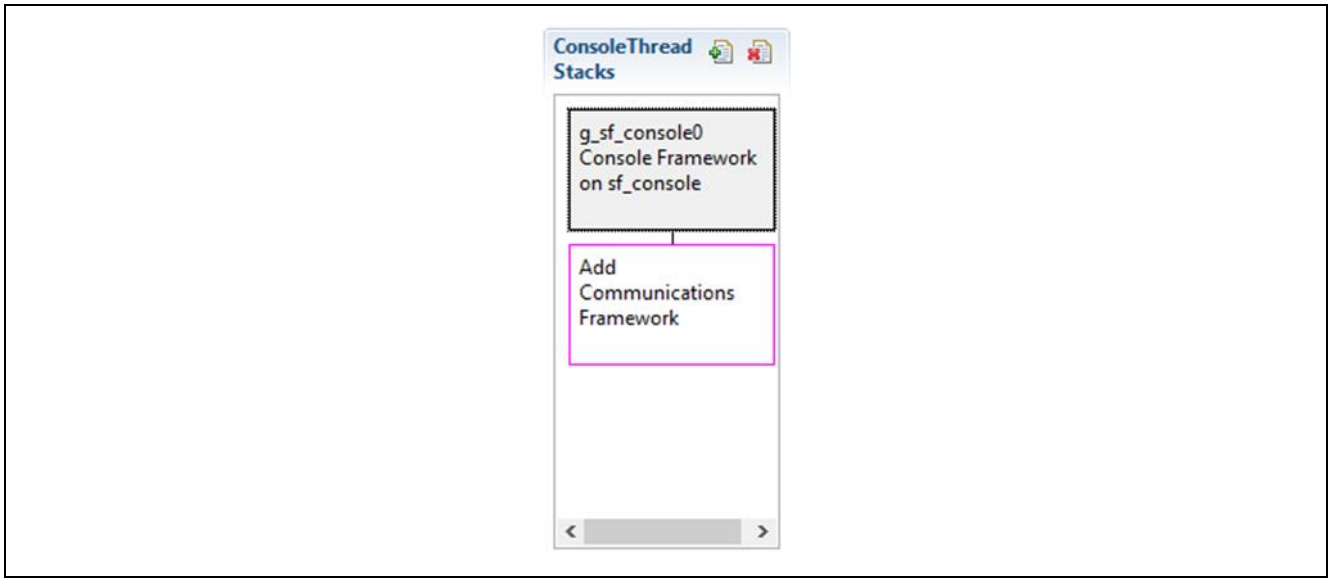

図 9.　コンソールフレームワーク内での通信フレームワークオプションの追加

このドキュメントの「参考情報」の章で説明されているアプリケーションフレームワークのビデオを視聴すると、コンソールの物理接続を USB から Telnet に切り替える作業がどれほど簡単なのか確認できます。設定ウィンドウ内で数分の操作を行うだけで、コンソールの切り替え作業全体が完了します。

9. コンソールフレームワークモジュールのサンプルアプリケーションの実行（Running the Console Framework Module Application Example）

コンソール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を実行し、ターゲットキットでその動作を確認するためには、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。

新しいプロジェクト内でコンソールフレームワークモジュールアプリケーションを実装するには、ターゲットキットで定義、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。実際にこれらの手順に従うことで SSP での開発プロセスをより実践的に習得するのに役立ちます。

- CDC 経由で接続を確立するには、最初に CDC ドライバをインストールする必要があります。CDC ドライバは下記サイトから入手できます。
  Renesas Synergy™ USB CDC ドライバ
  https://www.renesas.com/jp/ja/products/synergy/software/add-ons/usb-cdc-drivers.html

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、以下の手順は十分詳細なものです。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、『SSP ユーザーズマニュアル』の最初にあるいくつかの章を参照してください。

1. 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』（下記WEB）を参照してください。

   英語版:

   https://www.renesas.com/jp/ja/doc/products/renesas-synergy/apn/r11an0023eu0121-synergy-ssp-import-guide.pdf

   日本語版（参考資料）:

   https://www.renesas.com/jp/ja/doc/products/renesas-synergy/apn/r11an0023ju0121-synergy-ssp-import-guide.pdf

2. 1 本の micro USB ケーブルを SK-S7G2 ボードの J19 につなぎ、もう 1 本の micro USB ケーブルをホストから SK-S7G2 ボードの J5 コネクタにつなぐ方法で、ホスト PC に接続します。
3. アプリケーションのデバッグを開始します。
4. Terra Term アプリケーションを開き、USB CDC とのシリアル通信に対応するシリアルポートに接続します。
5. **?** と入力し、Enter を押して、使用可能なコマンドを表示します。
6. **TOGGLE** と入力し、ボード上にある LED のトグルを開始します。
7. 出力として、ユーザがシリアルコンソールにコマンドを入力したときに LED がトグルします。

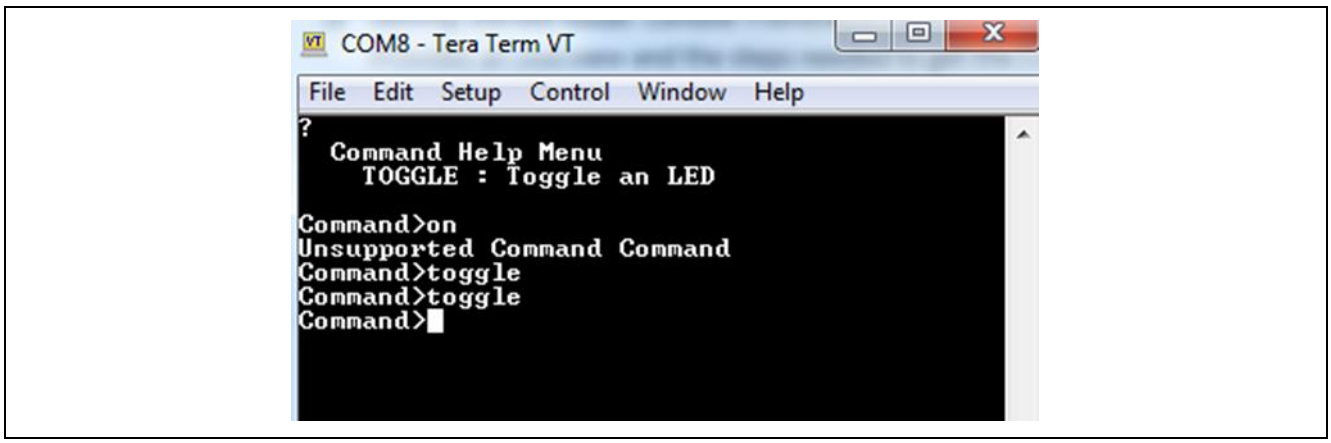

図 10. コンソールフレームワークのアプリケーションプロジェクトのサンプル出力

10. コンソールフレームワークモジュールのまとめ（Console 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背景となる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、また間違いが起こりやすい操作でした。Renesas Synergy プラットフォームにより、これら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など、誤りが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、ハイレベル API を使用することで高いレベルの開発からスタートし、ローレベルドライバを作成するような従来の開発環境で必要とされる時間が不要になり、開発時間を短縮できます。

11. コンソールフレームワークモジュールの次の手順（Console Framework Module Next Steps）

シンプルなコンソールフレームワークのサンプルをマスターした後、より複雑なサンプルを確認することができます。developer example プロジェクトは、コンソールフレームワークのより複雑な機能を複数示す、より複雑なサンプルを提供します。developer example に対応する『入門ガイド』を参照してください。このガイドは、このドキュメントの末尾にある「参考情報」の章の参照先にあります。

12. コンソールフレームワークモジュールの参考情報（Console Framework Module Reference Information）

『SSP ユーザーズマニュアル』：SSP ディストリビューションパッケージの一部として HTML 形式が入手できるほか、Renesas Synergy™ WEBサイトのSSPページ
https://www.renesas.com/jp/ja/products/synergy/software/ssp.htmlから pdf を入手することもできます。

最新版のsf_consoleモジュールの参考資料やリソースへのリンクは、以下の Synergy WEBサイトから入手できます。

https://www.renesas.com/jp/ja/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
1.01	2019.06.21		・初版 ・英文版（R11AN0110EU0101、Rev.1.02、2017.Aug.31）の巻頭と第 7 章以降を翻訳

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/