Renesas Synergy™ プラットフォーム

メッセージングフレームワークモジュールガイド

R11AN0096JU0103
Rev.1.03
2019.05.15

(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。

メッセージングフレームワーク（Messaging Framework）モジュールは、sf_message に実装（implement）されており、複数のスレッド間でメッセージを受け渡しする簡易型（lightweight）のイベント駆動型フレームワーク API（event driven framework API）を内蔵しています。メッセージングフレームワークモジュールを使用して、アプリケーションは複数のスレッド間でメッセージを交換することができます。このフレームワークは、メッセージ受け渡しのために ThreadX メッセージキュープリミティブ（message-queue primitive）を使用しますが、これはThreadX RTOS メッセージキューサービス（message-queue services）単体で使用するよりもより多くの利点があります。メッセージングフレームワーク API は、純粋なソフトウェア API であり、どのハードウェア周辺回路（hardware peripheral）にもアクセスしません。イベント生成側スレッド（event-producer thread）とメッセージサブスクライバスレッド（message-subscriber thread）はメッセージングフレームワークコールバックを通じて、メッセージ受け渡しが完了した後にハンドシェイク（handshake）を実施することができます。

[Messaging] (メッセージング) タブを使用して、独自のカスタムイベントクラス（custom event class）、イベント、メッセージングフレームワークモジュールのサブスクライバを作成することや、またはタッチパネルフレームワーク（Touch Panel Framework）モジュールが使用するタッチイベントのような事前設定済みイベント（preconfigured events）をカスタマイズすることができます。

目次

7. メッセージング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 Messaging 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うことで、設計全体の手順を体験することができます。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」章に掲載されているリンクにあります。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、メッセージング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、メッセージング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コード (`producer_thread_entry.c`、`led1_thread_entry.c`、`led2_thread_entry.c`、および `led3_thread_entry.c`)を確認することもできます。

本アプリケーションプロジェクトは、メッセージングフレームワークモジュール API の標準的な使用方法を示します。このアプリケーションプロジェクトのproducer thread entry はメッセージングフレームワークを初期化（initialize）し、ランダムなメッセージを周期的に選択します。このメッセージは、どの LED を点灯または消灯させるかを決定する情報を持っています。各 LED は独自のハンドラスレッド（handler thread）を持っており、このスレッドは対応のダイオード（LED）を制御します。これらのスレッド内で、メッセージの受信と処理を行います。スレッドはコマンドを実行し、新しいコマンドを待ちます。生成側スレッドは送信したメッセージを出力し、ハンドラスレッドは共通のセミホスト機能（common semihosting function）を使用して、受信したメッセージ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対応ソフトウェアおよびハードウェアのターゲットバージョンを示します。

表 1　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	6.2.1 またはそれ以降	統合ソリューション開発環境 (ISDE)
SSP	1.5.0 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	8.23.1 またはそれ以降	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 またはそれ以降	Synergy Standalone Configurator
SK-S7G2	v3.0 から v3.3	スタータキット

以下の図に、本アプリケーションプロジェクトの簡単なフローを示します。

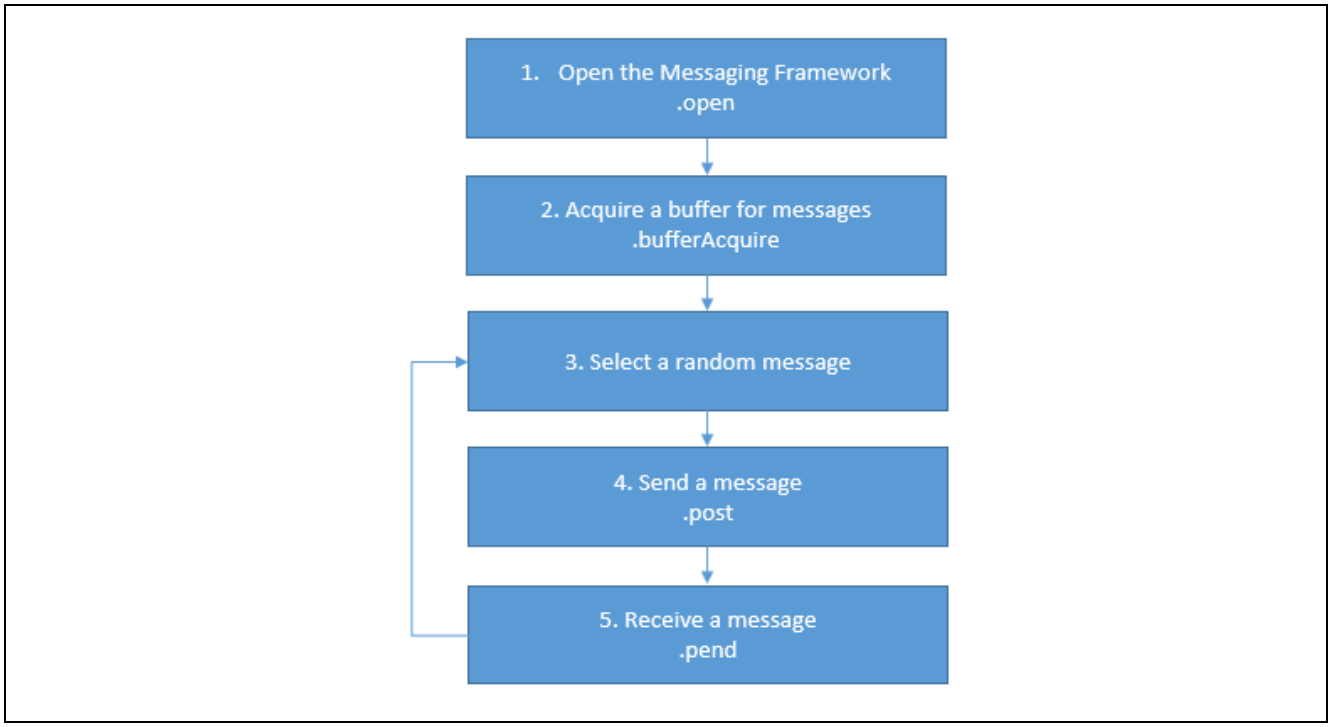

図 1　メッセージングフレームワークモジュールのアプリケーションプロジェクトのフロー

アプリケーションプロジェクト全体は、このドキュメント末尾の「参考情報」の章に掲載されているリンクから参照することができます。`producer_thread_entry.c`、`led1_thread_entry.c`、`led2_thread_entry.c`、および `led3_thread_entry.c` の各ファイルは、このプロジェクトを ISDE にインポートすることにより、プロジェクト内に配置されます。ISDE でこれらの各ファイルを開き、API の使い方のガイドを受けることができます。

`producer_thread_entry.c` の最初のセクションはヘッダファイルであり、メッセージングフレームワークインスタンス構造体（Messaging Framework instance structure）と、セミホスト機能が `printf()` を使用して結果を表示するために使用するコードセクション（code section）を参照しています。次のセクションは、疑似乱数（pseudo-random number）の生成機能（generator）に関係するマクロ定数の定義（maco constant message）を記述しています。疑似乱数の生成機能を使用して、メッセージを選択します。その後に、生成側スレッドに関連するグローバル変数（global variables ）を定義し、関数プロトタイプ（function prototype）が続きます。

続くセクションには関数定義（function definitions）があります。最初の関数は、変数が格納しているバイナリ状態（点灯か消灯か）をトグル（toggle、入れ替え）します。この値に基づいて、2 つのイベントのうちいずれかが返されます。そしてランダムなメッセージを選択するための関数が続きます。選択の対象になる 4 つのイベントクラス（event class）が存在し、各クラスはいずれも 25% の確率で選択できます。セミホスト機能（semi-hosting capability）が有効になっている場合、単一のイベントコードを単一の文字列リテラル（string literal）にマッピングするシンプルな関数が定義されます。最後の関数はランダムなメッセージを 1 つ取得し、メッセージングフレームワークインスタンス（Messaging Framework instance）を経由してそのメッセージを送信します。セミホスト機能を使用して短い情報メッセージを表示し、どのイベントを送信したかを示します。

最後のセクションは、スレッドエントリ関数（thread-entry function）です。この関数は、必要に応じてセミホスト機能を初期化し、`bufferAcquire` API を使用してメッセージバッファ（message buffer）を取得します。無限 `while` ループ内で、メッセージの選択と送信（投稿、posting）を実行する関数を呼び出した後、ティック（tick）数回分の時間にわたってスレッドをスリープ状態に置きます。

LED スレッドのエントリファイル（`led1_thread_entry.c`、`led2_thread_entry.c`、または `led3_thread_entry.c` が該当）の最初のセクションはヘッダファイルを持っていて、このヘッダファイルは、スレッドに関連する必要な変数（thread-related variables）や、LED スレッド設定構造体（LED thread configuration structure）を含む各種構造体（structures）を参照します。その後、グローバル変数の宣言（declaration）が続きます。最初は、受信したメッセージへのポインタとして使用する変数です。2 番目はスレッドの設定項目（thread configuration setting）を格納する変数で、制御対象 LED のインデックス（序数、index）、LED 固有のイベントクラス（event class）、ダイオード（LED）を点灯または消灯させるためのイベントコード（event code）などが該当します。続くセクションは、スレッドエントリ関数です。最初に、LED 設定を初期化します。次に「無限」ループ内でメッセージを受信します。メッセージが使用可能である場合、スレッド固有の設定項目を使用して、メッセージ処理用の関数を呼び出します。その関数はイベントの型（イベントクラスとコード）を検証し、適切なコマンドを実行します。

注記：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 `printf()` 関数に関する経験が少ない場合、このドキュメントの末尾にある「参考情報」の章に記載されている、「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）というナレッジベースの記事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。

ターゲットボードと MCU の必須の操作と物理プロパティ（physical properties）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。以下の表に、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。実際にこのアプリケーションプロジェクトを開き、[Properties]（プロパティ）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 2　アプリケーションプロジェクトに対応するメッセージングフレームワークの設定項目

ISDE のプロパティ	設定値
Parameter Checking (パラメータチェック)	Default (BSP) (デフォルト (BSP))
Message Queue Depth (Total number of messages to be `enqueued` in a Message Queue) (メッセージキューの深さ (メッセージキュー内に配置できるメッセージの総数))	16
Name (名前)	g_sf_message
Work memory size in bytes (バイト単位の作業メモリサイズ)	2048
Pointer to subscriber list array (サブスクライバリスト配列へのポインタ)	p_subscriber_lists
Name of the block pool internally used in the messaging framework (メッセージングフレームワークの内部で使用するブロックプールの名前)	sf_msg_blk_pool

イベントクラス、イベント、サブスクライバリストを設定するには、[Messaging]（メッセージング）タブを開き、以下の設定を使用します。

表 3　アプリケーションプロジェクトに対応する LED1 イベントクラスの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_CLASS_LED1
Name (名前)	LED1
Payload header file (ペイロードヘッダファイル)	led1_api.h
Payload (ペイロード)	led1_payload
Payload type (ペイロードの型)	led1_payload_t

表 4　アプリケーションプロジェクトに対応する LED2 イベントクラスの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_CLASS_LED2
Name (名前)	LED2
Payload header file (ペイロードヘッダファイル)	led2_api.h
Payload (ペイロード)	led2_payload
Payload type (ペイロードの型)	led2_payload_t

表 5　アプリケーションプロジェクトに対応する LED3 イベントクラスの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_CLASS_LED3
Name (名前)	LED3
Payload header file (ペイロードヘッダファイル)	led3_api.h
Payload (ペイロード)	led3_payload
Payload type (ペイロードの型)	led3_payload_t

表 6　アプリケーションプロジェクトに対応する LED All (LED 全体) イベントクラスの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_CLASS_LED_ALL
Name (名前)	LED All
Payload header file (ペイロードヘッダファイル)	led_all_api.h
Payload (ペイロード)	led_all_payload
Payload type (ペイロードの型)	led_all_payload_t

表 7　アプリケーションプロジェクトに対応する LED1_ON (LED1 点灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED1_ON
Name (名前)	LED1_ON

表 8　アプリケーションプロジェクトに対応する LED1_OFF (LED1 消灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED1_OFF
Name (名前)	LED1_OFF

表 9　アプリケーションプロジェクトに対応する LED2_ON (LED2 点灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED2_ON
Name (名前)	LED2_ON

表 10 アプリケーションプロジェクトに対応する LED2_OFF (LED2 消灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED2_OFF
Name (名前)	LED2_OFF

表 11 アプリケーションプロジェクトに対応する LED3_ON (LED3 点灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED3_ON
Name (名前)	LED3_ON

表 12 アプリケーションプロジェクトに対応する LED3_OFF (LED3 消灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED3_OFF
Name (名前)	LED3_OFF

表 13　　アプリケーションプロジェクトに対応する LED_ALL_ON (LED 全体点灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED_ALL_ON
Name (名前)	LED_ALL_ON

表 14　　アプリケーションプロジェクトに対応する LED_ALL_OFF (LED 全体消灯) イベントの定義

ISDE のプロパティ	設定値
Symbol (シンボル)	SF_MESSAGE_EVENT_LED_ALL_OFF
Name (名前)	LED_ALL_OFF

表 15 アプリケーションプロジェクトに対応するイベントサブスクライバの設定

Thread\Event Class (スレッド - イベントクラス)	LED1	LED2	LED3	LED All
LED1 Thread	✓			✓
LED2 Thread		✓		✓
LED3 Thread			✓	✓

注記：アプリケーションプロジェクトは、各イベントクラスに対応してただ 1 つのインスタンスを使用するので、すべてのイベントクラスとスレッドの開始値（start value）と終了値（end value）を 0 に設定する必要があります。

8. ターゲットアプリケーションに対応するメッセージングフレームワークモジュールのカスタマイズ（Customizing the Messaging Framework Module for a Target Application）

いくつかの設定項目では通常、アプリケーションプロジェクトが示している値に対し、ユーザが変更を加えます。ユーザは新しいイベントクラスまたはイベントを簡単に追加することができます。また、ユーザはイベントごとにサブスクライバリストを変更することもできます。この作業を行うには、コンフィギュレータの [Messaging] (メッセージング) タブを使用します。

9. メッセージング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 Messaging Framework Module Application Project）

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するだけで、メッセージング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させ、対象キットでその動作を観察することができます。

新しいプロジェクト内で メッセージングフレームワークアプリケーションを実装するには、対象キットで定義付け、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。これらの手順を実践することで、SSP を使用する開発プロセスの習得が容易になります。

注記: 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、『SSP ユーザーズマニュアル』の最初の数章を参照してください。

メッセージング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、実行するには、以下の手順に従ってください。

1. Messaging_FW_MG_AP という名称で SK-S7G2 キット用 Renesas Synergy プロジェクトを作成します。
2. **[Threads]** (スレッド) タブを選択します。
3. 以下の新しいスレッドを追加します。
   Symbol (シンボル)　led1_thread
   Name (名前)　LED1 Thread
4. 次のような新しいスレッドを追加します。
   Symbol (シンボル)　led2_thread
   Name (名前)　LED2 Thread
5. 次のような新しいスレッドを追加します。
   Symbol (シンボル)　led3_thread
   Name (名前)　LED3 Thread
6. 次のような新しいスレッドを追加します。
   Symbol (シンボル)　producer_thread
   Name (名前)　Producer Thread
7. [Messaging Framework] (メッセージングフレームワーク) を [Producer Thread] (生成側スレッド) に追加します。
8. メッセージングフレームワークインスタンスを設定します。
9. **[Messaging]** (メッセージング) タブを選択します。
10. 次のような新しいイベントクラスを追加します。
    Name (名前)　LED1
11. 次のような新しいイベントクラスを追加します。
    Name (名前)　LED2
12. 次のような新しいイベントクラスを追加します。
    Name (名前)　LED3
13. 次のような新しいイベントクラスを追加します。
    Name (名前)　LED All
14. [LED1 Thread] (LED1 スレッド) を [LED1 Subscribers] (LED1 サブスクライバ) と [LED All Subscribers] (LED 全体サブスクライバ) に追加します。
15. [LED2 Thread] (LED2 スレッド) を [LED2 Subscribers] (LED2 サブスクライバ) と [LED All Subscribers] (LED 全体サブスクライバ) に追加します。
16. [LED3 Thread] (LED3 スレッド) を [LED3 Subscribers] (LED3 サブスクライバ) と [LED All Subscribers] (LED 全体サブスクライバ) に追加します。
17. 次のようなイベントを追加します。
    Name (名前)　LED1_ON
18. 次のようなイベントを追加します。
    Name (名前)　LED1_OFF
19. 次のようなイベントを追加します。
    Name (名前)　LED2_ON
20. 次のようなイベントを追加します。
    Name (名前)　LED2_OFF
21. 次のようなイベントを追加します。
    Name (名前)　LED3_ON
22. 次のようなイベントを追加します。
    Name (名前)　LED3_OFF
23. 次のようなイベントを追加します。
    Name (名前)　LED_ALL_ON

24. 次のようなイベントを追加します。
    Name (名前)　　LED_ALL_OFF
25. **[Generate Project Content]** (プロジェクトコンテンツの生成) ボタンをクリックします。
26. 付属のプロジェクトファイル `led1_api.h`、`led2_api.h`、`led3_api.h`、`led_all_api.h`、`led_api.c`、`led_api.h`、`semihosting_cfg.h`、`led1_thread_entry.c`、`led2_thread_entry.c`、`led3_thread_entry.c`、`producer_thread_entry.c` からコードを追加するか、生成された同名のファイルに上書きする形でこれらの付属ファイルをコピーします。
27. micro USB ケーブルを SK-S7G2 キットの J19 につなぎ、ホスト PC に接続します。
28. アプリケーションのデバッグを開始します。
29. 出力は、デバッグコンソール (Renesas Debug Virtual Console、Renesas デバッグ仮想コンソール) に表示されます。

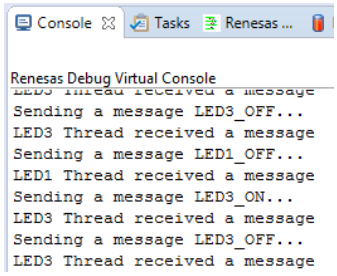

図 2　メッセージングフレームワークのアプリケーションプロジェクトのサンプル出力

10. メッセージングフレームワークモジュールのまとめ（Messaging 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. メッセージングフレームワークモジュールの次の手順（Messaging Framework Module Next Steps）

シンプルなメッセージングフレームワークのプロジェクトをマスターすれば、より複雑なサンプルをレビューできるようになります。『Audio Playback Framework Module Guide』または『Touch Panel Framework Module Guide』を参照することもできます。これらはいずれも、メッセージングフレームワークを活用しています。

ターゲットアプリケーションによっては、シンプルな ThreadX メッセージキューの方が適していることがあります。ThreadX API を確認する場合、『ThreadX ユーザーズマニュアル』が最適な出発点になります。

『SSP ユーザーズマニュアル』と『ThreadX® ユーザーズマニュアル』は、このドキュメントの末尾にある「参考情報」の章で入手することができます。

12. メッセージングフレームワークモジュールの参考情報（Messaging 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、Synergy WEB SSPページ から pdf を入手することもできます。

https://www.renesas.com/jp/ja/products/synergy/software/ssp.html

sf_message モジュールの参考資料やリソースに関する最新版は、以下のSynergy WEBサイトから入手できます。
https://www.renesas.com/jp/ja/products/synergy.htm

Web サイトおよびサポート

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

		改訂内容	
Rev.	**発行日**	**ページ**	**ポイント**
1.03	2019.05.15	−	・初版 ・英語版（R11AN0096JU0103，Rev.1.03，2019.Feb.01）の巻頭と第 7 章以降を翻訳 ・表 5 を英文版に合わせ SSP1.6.0 対応に修正

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)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。