Renesas Synergy™プラットフォーム
SCE HAL モジュールガイド

R11AN0088JU0101
Rev1.01
2019.09.12

（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 SCE HAL 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「参考情報」の章を参照）から入手でき、より複雑な設計に役立ちます。

セキュア暗号化エンジン（Secure Cryptographic Engine、SCE）HAL モジュールは、乱数生成（random number generation）、ダイジェスト（digest）計算（hash：ハッシュ）、データ暗号化と復号（data encryption and decryption）や、デジタル署名と検証（digital signing and verification）を行うためのハイレベル API（high-level API）です。このモジュールは、r_sce に実装されています。SCE は、暗号化の機能を実行する専用ハードウェアブロックです。Synergy MCU シリーズにより、SCE が提供する機能が異なります。

目次

1. SCE HAL Module Features....3
2. SCE APIs Overview....3
3. SCE HAL Module Operational Overview....3
3.1 SCE HAL Module Operational Notes and Limitations....3
4. Including the SCE HAL Module in an Application....3
5. Configuring the SCE HAL Module....3
6. Using the SCE HAL Module in an Application....3
7. SCE HAL モジュールのアプリケーションプロジェクト（The SCE HAL Module Application Project）....3
8. ターゲットアプリケーションに対応する SCE HAL モジュールのカスタマイズ（Customizing the SCE HAL Module for a Target Application）....7
8.1 AES 鍵の生成（AES key generation）....8
8.2 RSA 鍵の生成（RSA key generation）....9
8.3 DSA 鍵の生成（DSA key generation）....10
9. SCE HAL 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 SCE HAL Module Application Project）....11
9.1 AES と RSA による暗号化の結果の表示方法（AES and RSA encryption result presentation）....12
9.2 RSA と DSA による署名の結果の表示方法（RSA and DSA signature result presentation）....13
9.3 HASH と TRNG による結果の表示方法（HASH and TRNG result presentation）....14
10. SCE HAL モジュールのまとめ（SCE HAL Module Conclusion）....15
11. SCE HAL モジュールの次の手順（SCE HAL Module Next Steps）....15
12. SCE HAL モジュールの参考情報（SCE HAL Module Reference Information）....15

1. SCE HAL Module Features
2. SCE APIs Overview
3. SCE HAL Module Operational Overview
3.1 SCE HAL Module Operational Notes and Limitations
4. Including the SCE HAL Module in an Application
5. Configuring the SCE HAL Module
6. Using the SCE HAL Module in an Application
7. SCE HAL モジュールのアプリケーションプロジェクト（The SCE HAL 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」の章に掲載されているリンクから入手可能です。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、SCE HAL 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、システムにおける SCE API を理解するために、コード（`hal_entry.c`、`sce_aes_mg`、`sce_dsa_mg.c`、`sce_hash_mg.c`、`sce_rsa_encryption_mg.c`、`sce_rsa_signature_mg.c`、`sce_trng_mg.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、SCE 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は、データ暗号化と復号、デジタル署名と検証、データハッシュの計算（data-hash calculation）、乱数生成のデモを実行します。以下の表に、このアプリケーションプロジェクトに必要なソフトウェアおよびハードウェアのバージョンを示します。

表 17　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）
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 Renesas 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

以下の図に、このアプリケーションプロジェクトのシンプルなフローを示します。

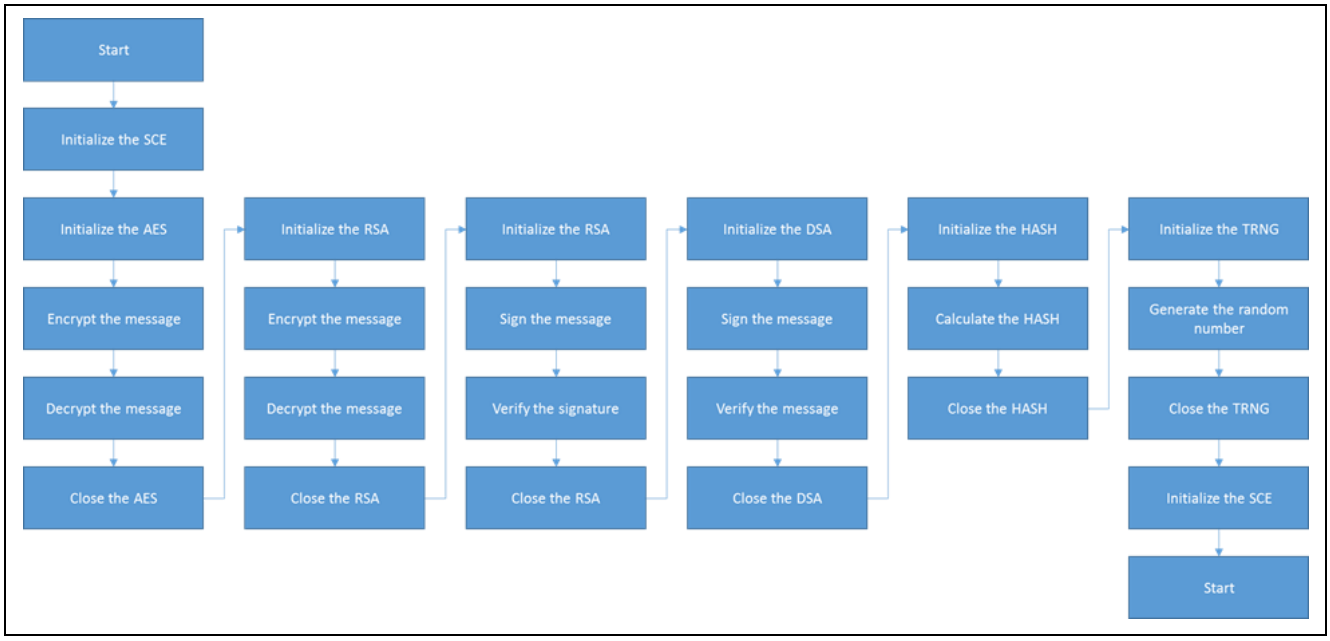

図 4　SCE HAL モジュールのアプリケーションプロジェクトのフロー

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入手することができます。各ファイル（hal_entry.c、sce_aes_mg、sce_dsa_mg.c、sce_hash_mg.c、sce_rsa_encruption_mg.c、sce_rsa_signature_mg.c、sce_trng_mg.c)は、このプロジェクトを ISDE にインポートするとプロジェクト内に配置されます。ISDE でこのファイルを開き、API の使い方のガイドを受けることができます。

hal_entry.c の最初のセクションは自動生成されたヘッダファイルを含んでおり、複数の SCE インスタンス構造体（SCE instance structures）を参照しています。hal_entry.c 関数の中で、操作に使用するいくつかのサンプルデータを作成します。正しい形式で作成した後は、SCE HAL モジュールを開き、下記の 6 つのサンプル暗号化関数を呼び出します。

- aes_example
- rsa_encryption_example
- rsa_signature_example
- dsa_example
- hash_example
- trng_example

aes_example は、sce_aes_mg.c ファイル内にあります。この関数は最初に、AES 関数が必要とする複数の AES 鍵（AES キー）と AES 初期化ベクタ（AES initial vector）を宣言します。SCE AES ドライバを開いた後、必要なパラメータを指定して g_sce_aes_0.p_api->encrypt() API を呼び出し、サンプルデータを暗号化します。そして暗号化プロセスの結果を encrypted_message 配列に格納します。この API 呼び出しの前と後にブレークポイント（breakpoint）を設定すると、暗号化プロセスの結果を表示できます。

その後、必要なパラメータを指定して g_sce_aes_0.p_api->decrypt() API を呼び出し、暗号化済みメッセージを復号します。そして復号プロセスの結果を decrypted_message 配列に格納します。この API 呼び出しの前と後にブレークポイントを設定すると、復号プロセスの結果を表示できます。

aes_example の最後のセクションは、開始時のメッセージ（starting message）と復号済みメッセージを比較します。これらのメッセージが互いに一致すると、アプリケーションは実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

sce_aes サンプルが正常に完了した後、sce_rsa_encryption_mg.c ファイル内の rsa_encryption_example を実行します。この関数は最初に、RSA 関数が必要とする RSA 公開鍵（public key）と秘密鍵（private key）を作成するための指数（exponent）と約数（modulus、法）を宣言します。

SCE RSA HAL モジュールを開いた後、必要なパラメータを指定して g_sce_rsa_0.p_api->encrypt() API を呼び出し、サンプルデータを暗号化します。そして暗号化プロセスの結果を encrypted_message 配列（array）に格納します。この API 呼び出しの前と後にブレークポイント（breakpoint）を設定すると、暗号化プロセスの結果を表示することができます。

その後、必須のパラメータを指定して g_sce_rsa_0.p_api->decrypt() API を呼び出し、暗号化済みメッセージを復号します。そして復号化プロセスの結果を decrypted_message 配列に格納します。この API 呼び出しの前と後にブレークポイントを設定すると、復号化プロセスの結果を表示できます。

rsa_example の最後のセクションは、開始時のメッセージ（starting message）と復号済みメッセージを比較します。これらのメッセージが互いに一致すると、アプリケーションの実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

rsa_encryption_example が正常に完了した後、sce_rsa_signature_mg.c ファイル内の rsa_signature_example を実行します。この関数は最初に、RSA の署名関数（sign function）と検証関数（verify function）が必要とする RSA 公開鍵（public key）と秘密鍵（private key）を作成するための指数（exponent）と約数（modulus、法）を宣言します。SCE RSA HAL モジュールを開いた後、必要なパラメータを指定して g_sce_rsa_0.p_api->sign() API を呼び出し、サンプルデータに署名します。そして、署名プロセスの結果を signature 配列に格納します。その後、必要なパラメータを指定して g_sce_rsa_0.p_api->verify() API を呼び出し、既に作成した署名を検証します。検証プロセスが成功すると、アプリケーションは実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

rsa_signature_example が正常に完了した後、sce_dsa_mg.c ファイル内の dsa_example を実行します。この関数は最初に、DSA の署名関数と検証関数が必要とする DSA 公開鍵と秘密鍵を作成するための複数の鍵と複数のドメイン変数（domain variable）を宣言します。SCE DSA HAL モジュールを開いた後、必要なパラメータを指定して g_sce_dsa_0.p_api->hashSign() API を呼び出し、サンプルデータに署名します。そして、署名プロセスの結果を signed_message 配列に格納します。その後、必要なパラメータを指定して g_sce_dsa_0.p_api->verify() API を呼び出し、既に作成した署名を検証します。検証プロセスが成功すると、アプリケーションは実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

dsa_example が正常に完了した後、sce_hash_mg.c ファイル内の hash_example を実行します。この関数は最初に、ハッシュ関数が必要とする格納領域（storage area）を宣言します。SCE HASH ドライバを開いた後、必要なパラメータを指定して g_sce_hash_0.p_api->hashUpdate() API を呼び出し、サンプルデータのハッシュを生成します。hashUpdate が成功すると、アプリケーションは実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

hash_example が正常に完了した後、sce_trng_mg.c ファイル内の trng_example を実行します。この関数は最初に、乱数生成関数（random number generation function）が必要とするサイズ（size）と格納領域（storage area）を宣言します。SCE TRNG HAL モジュールを開いた後、必要なパラメータを指定して g_sce_trng_0.p_api->read() API を呼び出し、指定された数の乱数を生成します。read 関数が成功すると、アプリケーションは実行を続行します。それ以外の場合、このアプリケーションは while(1)ループ内で待機します。

アプリケーションはここまでで SCE 関数の実行を完了し、while(1)ループ内で待機します。

SCE HAL モジュールアプリケーションを構築するための重要な要素が選択され、指定したスタックの設定を行います。このアプリケーションプロジェクトでは、以下の 5 個のスタックを使用します。

- r_sce_aes 上の AES HAL モジュール
- r_sce_hash 上の HASH HAL モジュール
- r_sce_rsa 上の RSA HAL モジュール
- r_sce_dsa 上の DSA HAL モジュール
- r_sce_trng 上の TRNG ドライバ

以下の表に、このサンプルアプリケーションで使用するスタックの設定を示します。

表 18　SCE HAL 層のインタフェース API の概要

ドライバ	モジュール	プロパティ	値
AES	g_sce_aes_0 AES Driver on r_sce_aes （r_sce_aes 上の g_sce_aes_0 AES ドライバ）	Name（名前）	g_sce_aes_0
		Key Length（鍵の長さ）	128
		Chaining Mode （チェーンモード）	ECB
RSA	g_sce_rsa_0 RSA Driver on r_sce_rsa （r_sce_rsa 上の g_sce_rsa_0 RSA ドライバ）	Name（名前）	g_sce_rsa_0
		Key Length（鍵の長さ）	1024
DSA	g_sce_dsa_0 DSA Driver on r_sce_dsa （r_sce_dsa 上の g_sce_dsa_0 DSA ドライバ）	Name（名前）	g_sce_dsa_0
		Key Length（鍵の長さ）	（1024、160）
HASH	g_sce_hash_0 HASH Driver on r_sce_hash (r_sce_hash 上の g_sce_hash_0 HASH ドライバ)	Name（名前）	g_sce_hash_0
		Key Length（鍵の長さ）	SHA256
TRNG	g_sce_trng TRNG Driver on r_sce_trng （r_sce_trng 上の g_sce_trng TRNG ドライバ）	Name（名前）	g_sce_trng_0
		Max. Attempts （最大試行回数）	2

8. ターゲットアプリケーションに対応する SCE HAL モジュールのカスタマイズ（Customizing the SCE HAL 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の値に対してユーザが変更を加えます。たとえば、AES の鍵の長さとチェーンモードを変更することがあります。

データの暗号化、復号、署名、検証を実行するには、選択した方法と鍵長（key length）に応じて、一連の鍵が必要です。このアプリケーションプロジェクトで使用する複数の鍵は、以下の場所に掲載されている OpenSSL ツールキットを使用して生成されたものです。
http://gnuwin32.sourceforge.net/packages/openssl.htm.

以下のページで、鍵を生成した詳細な方法を示します。

バイナリ形式の zip ファイルをダウンロードし、展開します。

コマンドウィンドウを開き、openssl フォルダ、次いで bin フォルダに移動します。

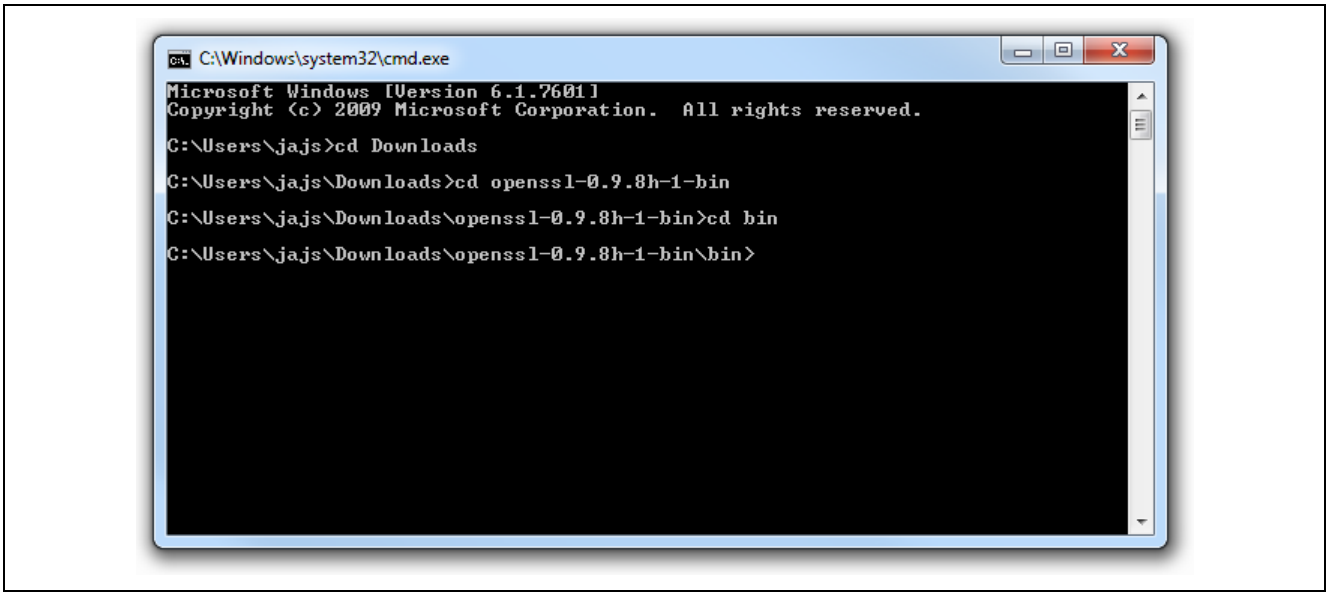

図 5　openssl のバイナリ（bin）フォルダへの移動

8.1 AES 鍵の生成（AES key generation）

AES の鍵と初期ベクタの生成は、`openssl aes-128-ecb -P` コマンドを使用して行います（この例では、鍵の長さが 128 ビット、ECB チェーンモード）。ユーザは、パスワードの入力と確認を求められます。パスワードに対応するハッシュの計算と生成を実行するために、このパスワードが使用されます。入力するパスワードは任意の値でかまいません。AES アルゴリズムを使用してデータの暗号化や復号を行うときに、ここで生成された key（鍵）と IV（Initialization Vector、初期化ベクタ）を使用する必要があります。16 進形式で定義した 32 ビット長の整数配列に対応する適切な形式が使用されていることに注意してください。

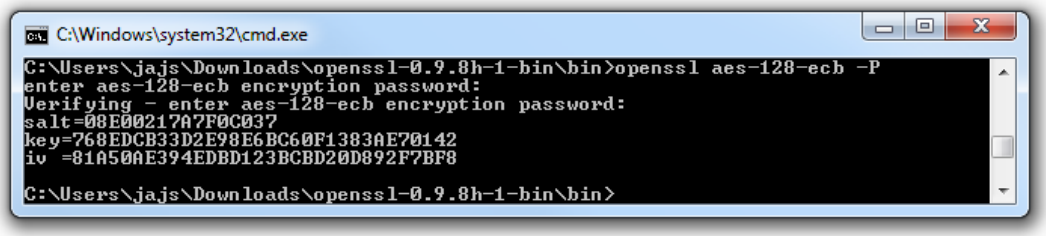

図 6 AES の鍵と初期化ベクタ（IV）

8.2 RSA 鍵の生成（RSA key generation）

OpenSSL を使用して RSA 鍵を生成するには、`openssl genrsa -out rsa-openssl.pem 1024` コマンド（鍵の長さが 1,024 ビットの場合）、次に `openssl rsa -in rsa-openssl.pem -pubout -outform DER -text` コマンドを実行します。このプログラムは DER 形式でデータを画面に出力します（1 バイトの 16 進数をコロンで区切った形式）。この形式を、C ベースの 32 ビット 16 進数の数値表現に変換する必要があります。秘密鍵（private key）と公開鍵（public key）を生成するためにこれらのデータを組み合わせる例は、前の章に掲載されています。いくつかの数値が `00` というバイトで始まっている場合、以下の図のようになります。このバイトは無視する必要があります。

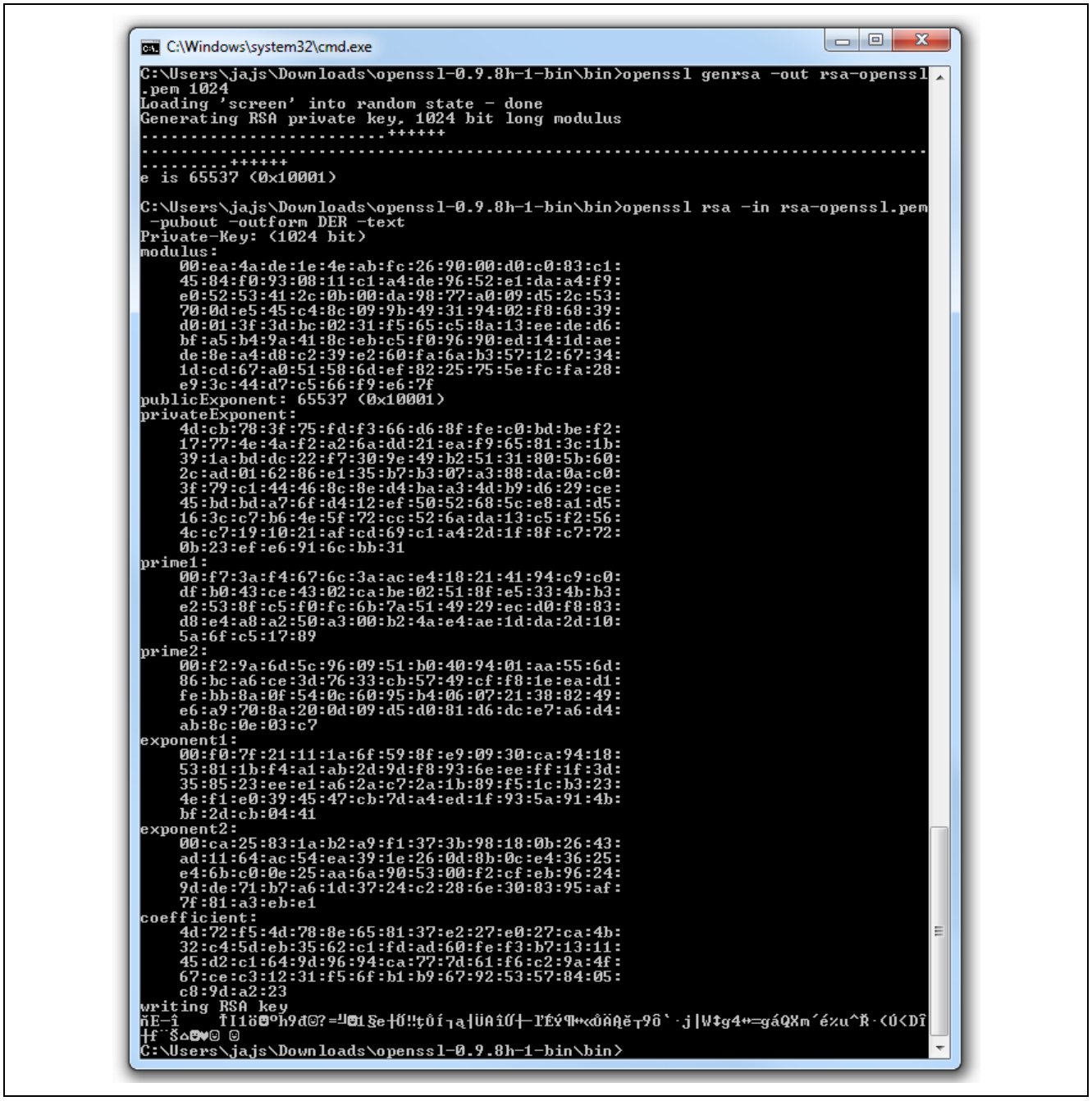

図 7　RSA の鍵と初期化ベクタ（IV）

8.3 DSA 鍵の生成（DSA key generation）

DSA の署名と検証に使用する鍵とパラメータを生成するために、以下の一連のコマンドを使用します。`openssl dsaparam -out dsa-param-openssl.pem 1024`、次に openssl gendsa -out dsa-openssl.pem dsa-param-openssl.pem、最後に `openssl dsa -in dsa-openssl.pem -pubout -outform DER -text`、です。このプログラムは DER 形式でデータを画面に出力します。この形式を、C ベースの 32 ビット 16 進数の数値表現に変換する必要があります。秘密鍵（private key）と公開鍵（public key）を生成するためにこれらのデータを組み合わせる例は、前の章に掲載されています。1 つの数値が `00` というバイトで始まっている場合、以下の図のようになります。このバイトは無視する必要があります。

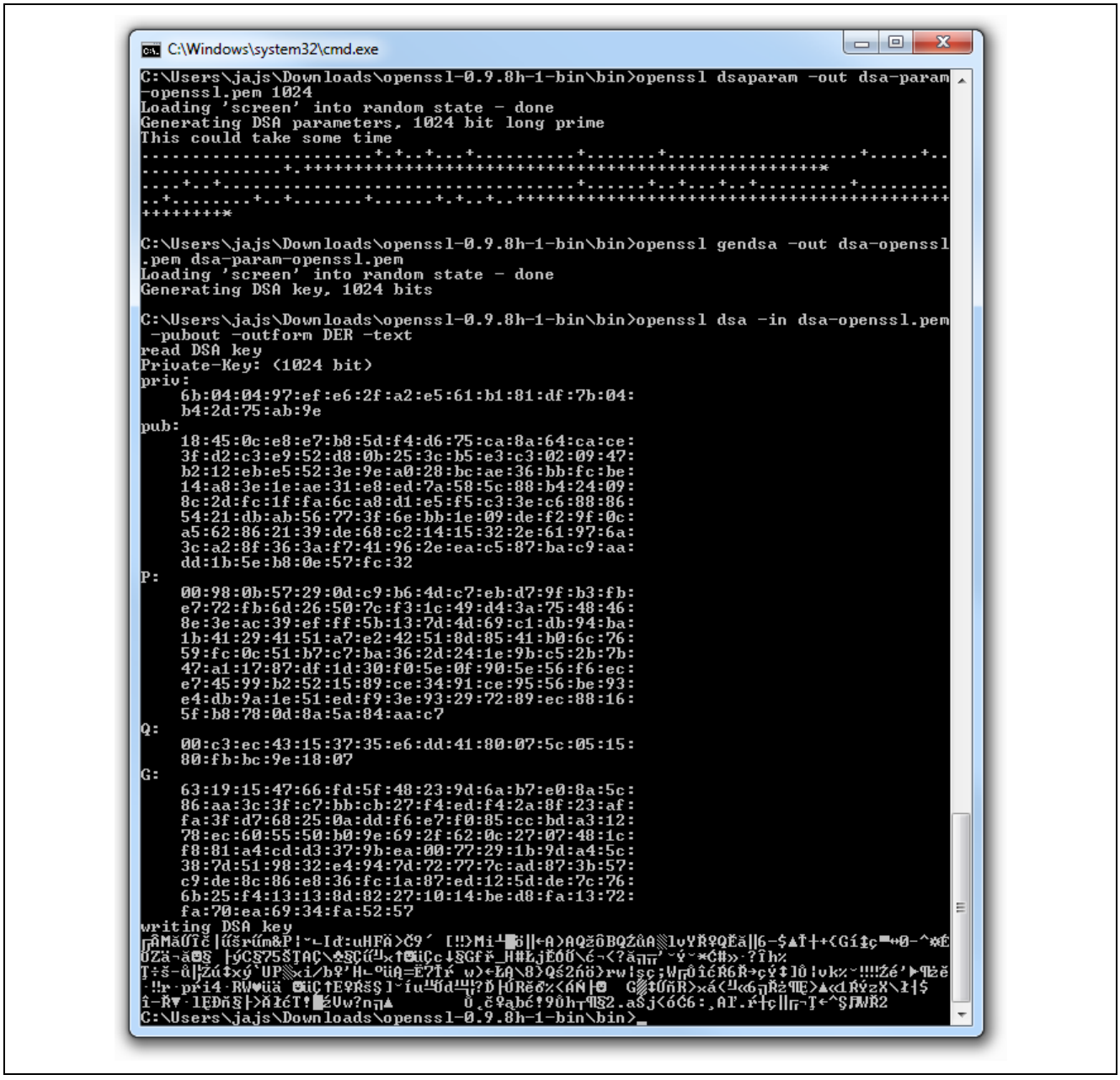

図 8 DSA の鍵とパラメータ

9. SCE HAL 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 SCE HAL Module Application Project）

SCE HAL 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、本プロジェクトの ISDE へのインポート、コンパイル（compile）、およびデバッグ（debug）を容易に実行することができます。

新しいプロジェクト内で SCE アプリケーションを実装する場合、ターゲットキット上で行う定義、設定、ファイルの自動生成、コードの追加、コンパイル、デバッグは、以下の手順に従います。このガイドに示す手順に従うことで、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。

【注意】Renesas Synergy™プラットフォーム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』の最初にあるいくつかの章を参照してください。

SCE HAL モジュール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. `sce_hal` という名称で SK-S7G2 ボード（S7G2-BSP）グループに対応する新しいプロジェクトを作成します。
2. S7G2-SK BSP プロジェクトテンプレート（project template）を選択し、プロジェクトを作成します。
3. 生成したプロジェクトの `Configuration.xml` を開き、**[Threads]**タブを選択します。
4. **[New Stack] > [Driver] > [Crypto]**を選択して[HAL/Common]スレッドで 5 個の SCE スタックを追加し、これらのスタックのパラメータを設定します。
5. **[Generate Project Content]**ボタンをクリックします。
6. 付属のプロジェクトファイル `hal_entry.c` からコードを追加するか、生成された `hal_entry.c` ファイルに上書きする形でコピーします。
7. `sce_aes_mg.c`、`sce_dsa_mg.c`、`sce_functions_mg.h`、`sce_hash_mg.c`、`sce_rsa_encryption_mg.c`、`sce_rsa_signature_mg.c`、`sce_trng_mg.c` の各ファイルを、プロジェクトディレクトリ内にあるプロジェクトの **src** フォルダにコピーします。
8. 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。
9. アプリケーションのデバッグを開始します。

9.1 AES と RSA による暗号化の結果の表示方法（AES and RSA encryption result presentation）

データを復号した後は、復号済みメッセージと元のメッセージを比較します。比較結果が異なる場合、このプログラムは無限ループを開始します。復号済みメッセージを検討するには、aes.c または rsa.c ファイルの中でデータの復号化の後にブレークポイントを配置し、decrypted_message 変数に注目します。

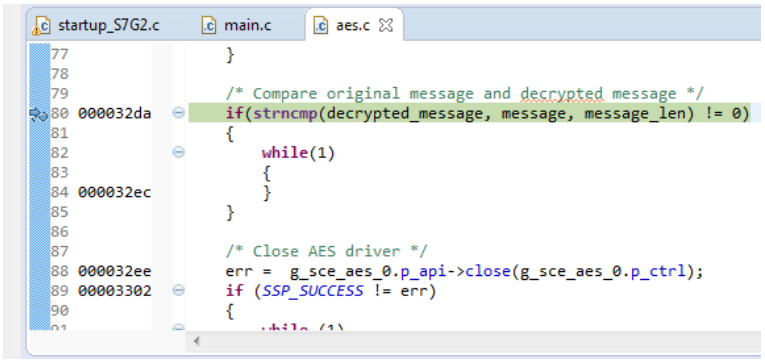

図 9　aes.c ファイル内で、文字列比較の位置に配置したブレークポイント

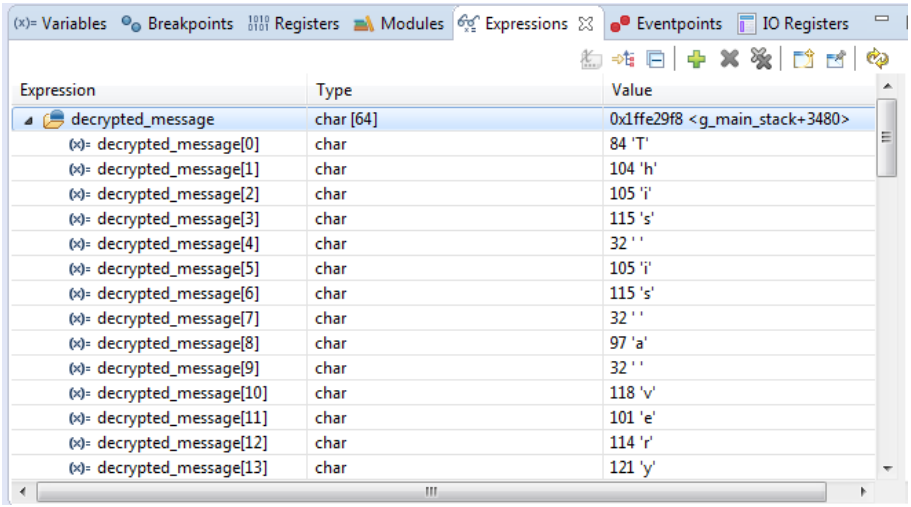

図 10　データの復号化の結果

直前の図に示したように、decrypted_message 変数は、元のメッセージと同じテキストを格納しています。それ以外の場合、このプログラムは無限ループを開始します。

9.2 RSA と DSA による署名の結果の表示方法（RSA and DSA signature result presentation）

データの署名と検証の結果を表示するには、署名を検証した後にブレークポイントを配置し、変数 `err` を確認します。`SSP_SUCCESS` という値は、所望の検証結果（検証の成功）を意味します。それ以外の場合、結果は `SSP_ERR_INVALID_MODE` になり、このプログラムは無限ループを開始します。

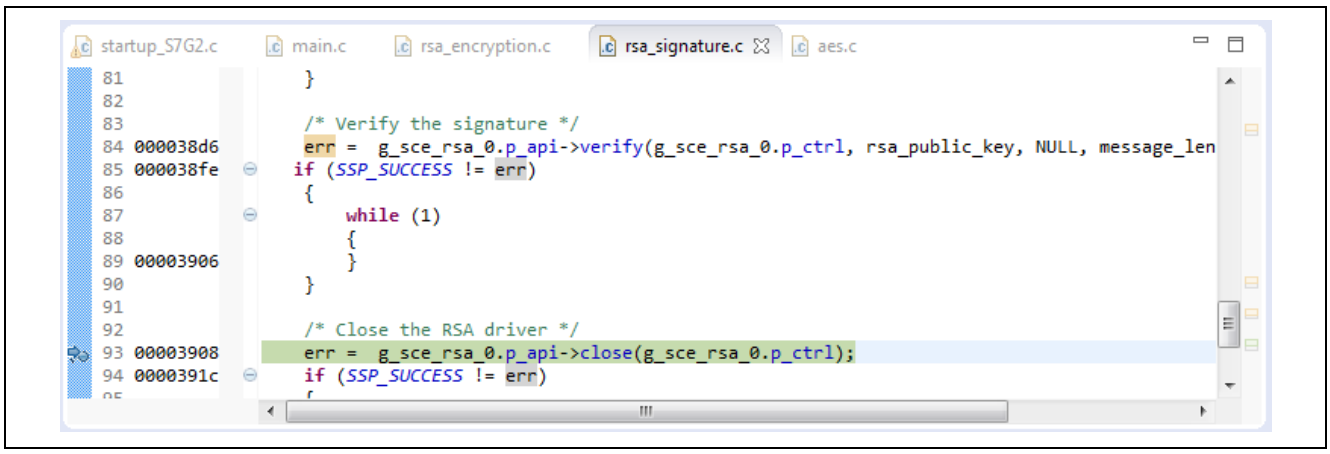

図 11　rsa_signature.c ファイル内で署名の検証の後に配置したブレークポイント

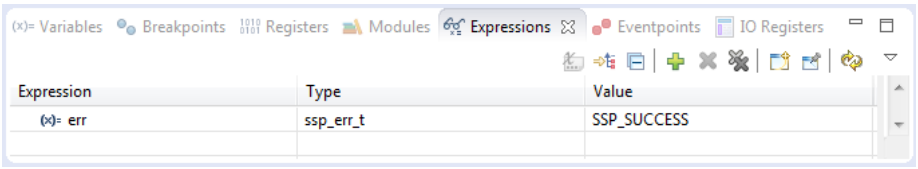

図 12　署名検証の結果

9.3 HASH と TRNG による結果の表示方法（HASH and TRNG result presentation）

ハッシュ関数または乱数生成の結果を表示するには、それらの計算または生成の後にブレークポイントを配置し、`hash` または `random_number` 変数に注目します。

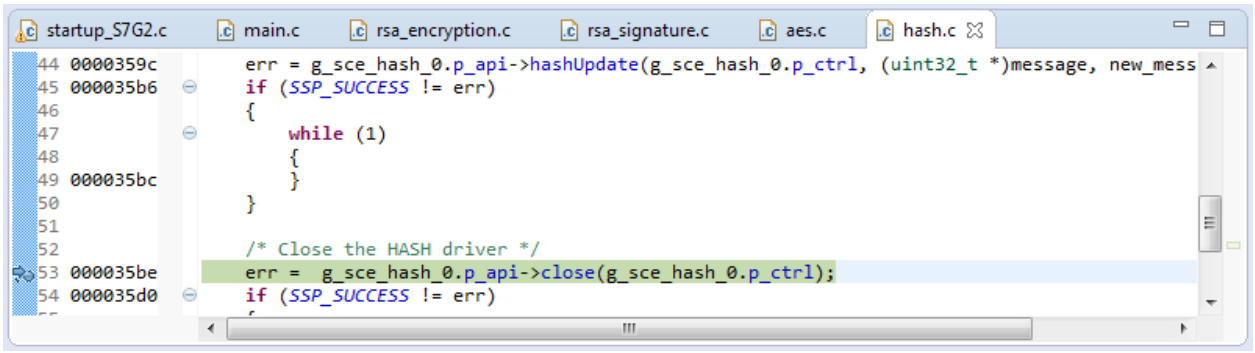

図 13　ハッシュ計算後のブレークポイント

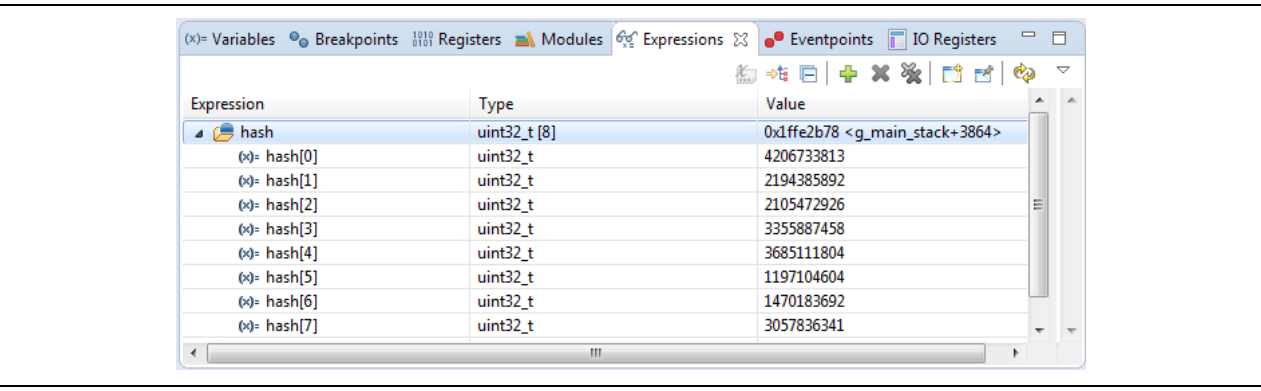

図 14　ハッシュ値

10. SCE HAL モジュールのまとめ（SCE HAL 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の時間を必要とし、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. SCE HAL モジュールの次の手順（SCE HAL Module Next Steps）

シンプルな SCE モジュール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。SCE HAL の使用方法を示す他のアプリケーションプロジェクトとアプリケーションノートは、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 WEB から入手可能です。

12. SCE HAL モジュールの参考情報（SCE HAL 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 r_sce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

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

Rev.	発行日	改訂内容	
		ページ	ポイント
1.01	2019.09.12	—	• 初版 • 英語版（R11AN0088EU0101, Sep.14.17, 2017.09.14）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)

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。