ルネサスマイクロコンピュータ
RSA ライブラリ: ユーザーズマニュアル

要旨

このマニュアルは、ルネサスマイクロコンピュータ用のソフトウェアライブラリ（以下 RSA ライブラリとします）について説明します。

このマニュアルは、RSA ライブラリを使ってアプリケーションプログラムを作成するための情報を提供します。

RSA ライブラリは、本マニュアルの他に、対応マイコン毎に「導入ガイド」を用意しています。

プログラムの ROM/RAM サイズや処理性能、対応マイコン毎の注意事項等をまとめた資料です。本マニュアルと合わせてご参照ください。

RSA ライブラリは次の規格書に示される仕様をもとに作成されたソフトウェアです。本マニュアルと合わせてご参照ください。

PKCS #1 v2.1: RSA Cryptography Standard

動作確認デバイス

ルネサスマイクロコンピュータ

目次

1. 概要

1.1 RSA の概要

RSA に関する情報は以下のサイトに掲載されています。

http://ja.wikipedia.org/wiki/RSA%E6%9A%97%E5%8F%B7

1.2 RSA 署名生成/検証イメージ

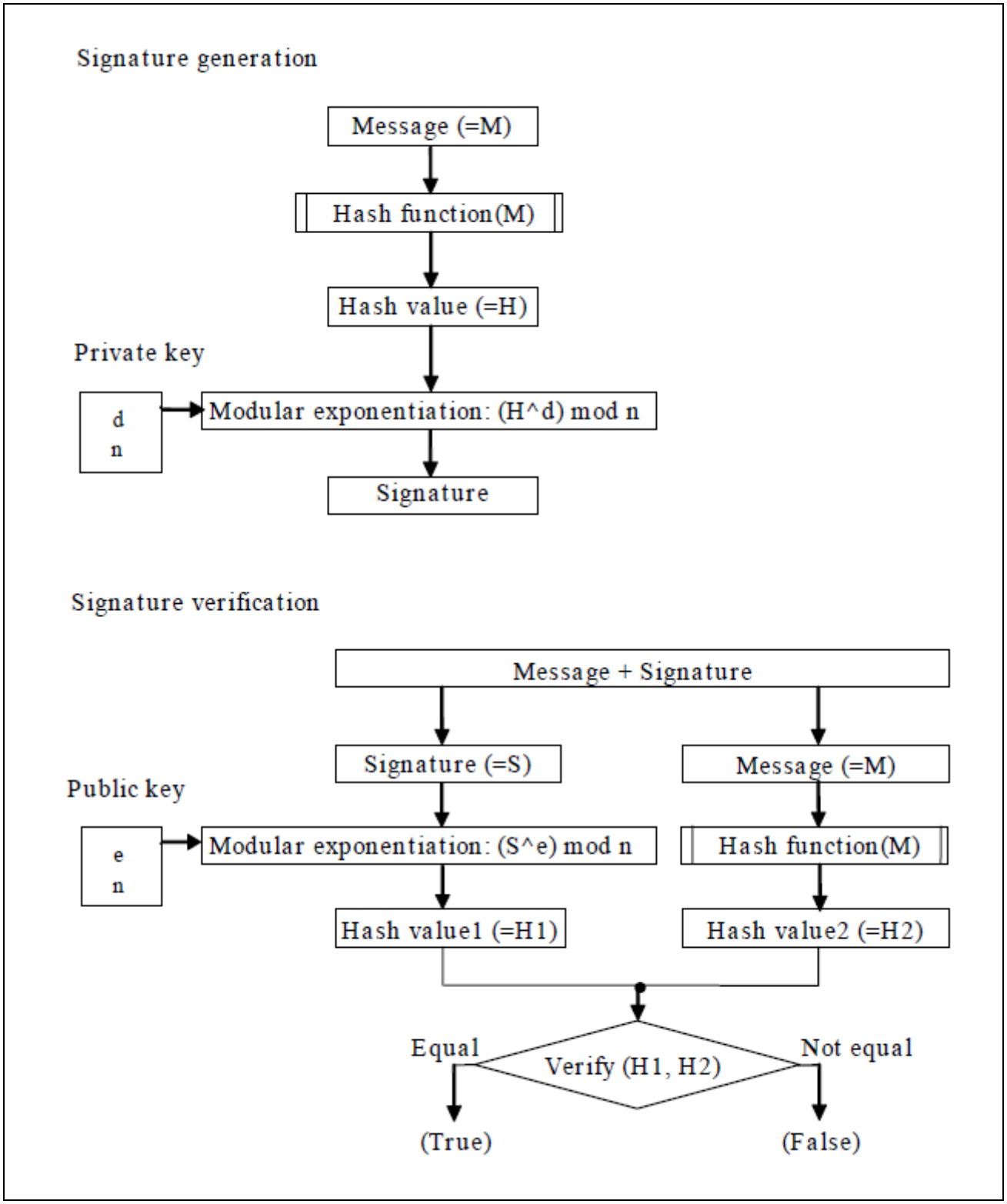

図 1.1 RSA 署名生成/検証イメージ

2. RSA ライブラリ概要

2.1 ソフトウェアスタック構成

RSA ライブラリは以下のソフトウェアスタック構成になっています。

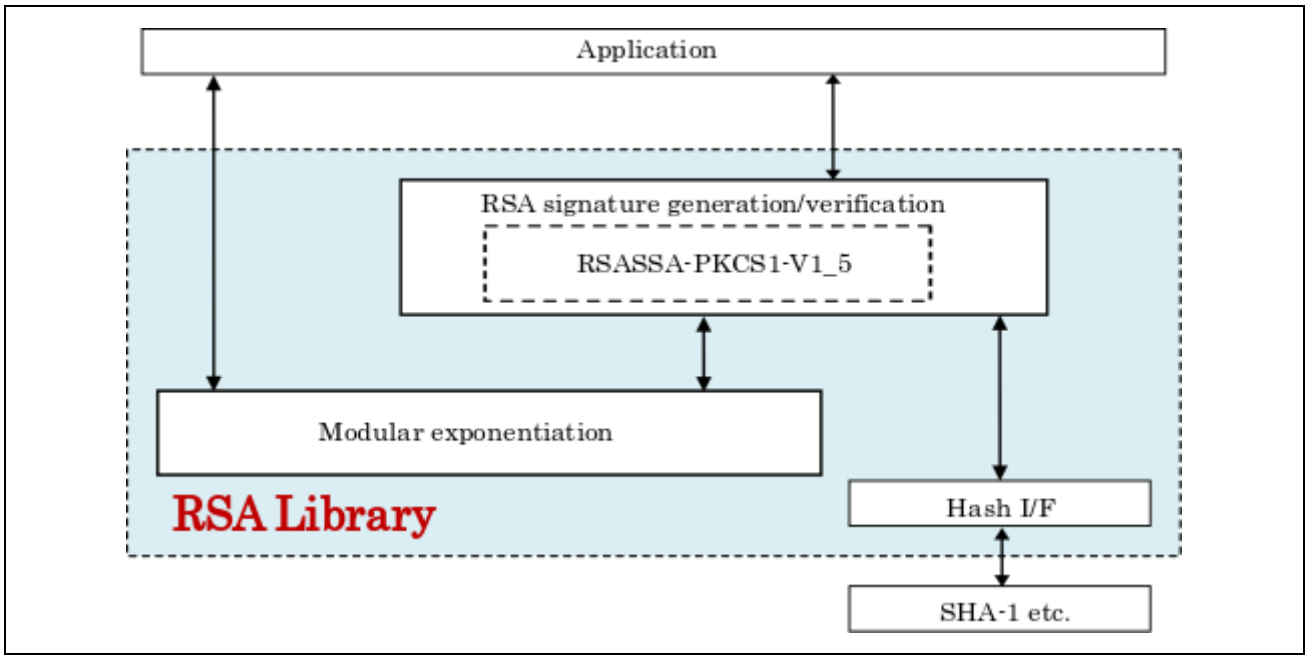

図 2.1　RSA ライブラリの構成

応用例

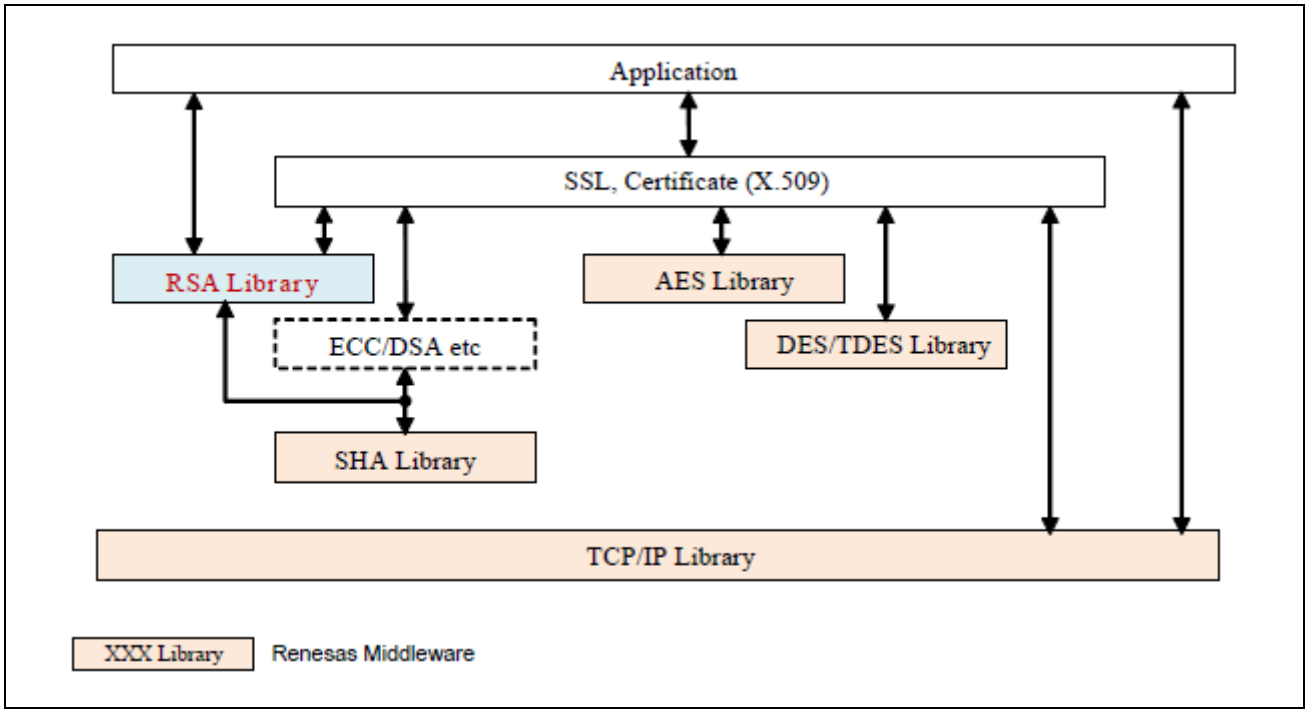

図 2.2　RSA ライブラリを用いたアプリケーション例

2.2　RSA ライブラリの仕様

RSA ライブラリの主な仕様を以下に示します。

表 2.1　RSA ライブラリの仕様

項目	仕様
署名生成方式	RSASSA-PKCS1-V1_5　*1
署名検証方式	RSASSA-PKCS1-V1_5　*1
べき乗剰余演算対応	対応
鍵長	最大 2048 ビット
鍵生成機能	なし

【注】　*1. ハッシュ関数はユーザにて作成する必要があります。

2.3　RSA 鍵データについて

RSA ライブラリは RSA 鍵を生成する機能を有しておりません。ユーザがあらかじめ用意する必要があります。

また、関数を実行する前に、鍵データを ROM または RAM 上に展開しておく必要があります。

RSA ライブラリは関数を終了した後、鍵データをクリアしませんので、ユーザが必要に応じてクリアしてください。

2.4　演算で使用するメモリの取り扱い

RSA ライブラリは、演算を行うときワーク領域に演算途中のデータを一時的に保存します。 鍵情報が演算途中のデータから推測される可能性があるため、RSA ライブラリは、関数を終了する前に使用したワーク領域をクリアします。

2.5　提供形態

表 2.2　RSA ライブラリの提供内容

種別	ファイル名	説明
Library	r_rsa.h r_mw_version.h r_stdint.h rsa_api.c mc_lib.c mc_lib.h rsa_internal_header.h	RSA の機能を提供するためのソフトウェアをソースコード提供します。
サンプルプログラム	対応マイコン毎のプロジェクトファイル（「導入ガイド」をご参照ください）	ルネサス統合開発環境で Application の動作確認が可能なプロジェクトファイル。また、これに付随するファイル・ディレクトリ。 署名生成/検証、べき乗剰余演算の基本動作を確認するためのソフトウェアをソースコード提供します。

2.6 プログラム開発手順

RSA ライブラリはソースコード形式で提供いたします。アプリケーションプログラムにライブラリ、「導入ガイド」を参照いただき、ライブラリのソースコードを加えてコンパイル・リンクして使用してください。

図 2.3.「アプリケーションプログラムの開発フロー」 を示します。

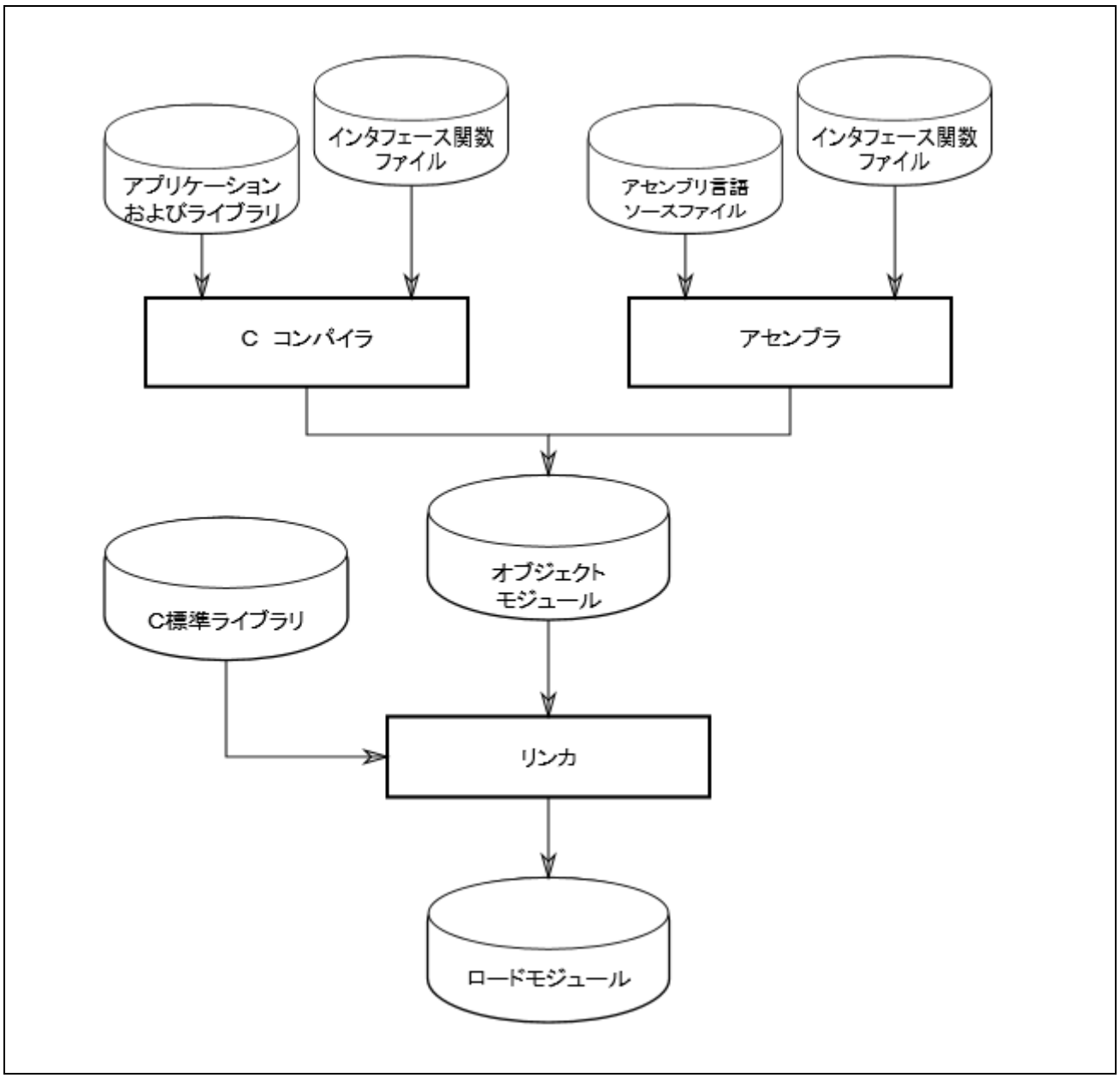

図 2.3 アプリケーションプログラムの開発フロー

3. ライブラリの型の定義

このセクションでは、RSA ライブラリで使用する型の定義について説明します。

3.1 基本データタイプ

RSA ライブラリは、ヘッダファイル"r_stdint.h"で定義される型を使用します。

表 3-1 RSA ライブラリの基本データ型

データ型	定義	バイト長
signed char	int8_t	1
signed short	int16_t	2
signed long	int32_t	4
unsigned char	uint8_t	1
unsigned short	uint16_t	2
unsigned long	uint32_t	4

4. ライブラリ構造体

このセクションでは、RSA ライブラリで使用する構造体について説明します。

4.1 R_RSA_WORK_t - ワークメモリ構造体

R_RSA_WORK_t 構造体は、RSA ライブラリが演算に使用するためのワーク領域を定義します。

アプリケーションはこの構造体を RAM 領域に割り当てる必要があります。 RSA ライブラリを並列に実行したい場合は、この構造体変数を複数用意して、API に各ワーク領域を渡してください。

表 4-1 R_RSA_WORK_t のメンバ

データ型	メンバ名	説明
uint32_t	work[R_RSA_WORK_SIZE]	RSA ワーク領域　　3680 バイト必要

4.2 R_RSA_BYTEDATA_t - バイト列構造体

R_RSA_BYTEDATA_t 構造体は、RSA ライブラリの API を呼び出すための入出力情報を定義します。

アプリケーションはこの構造体を RAM 領域に割り当てる必要があります。

表 4-2 R_RSA_BYTEDATA_t のメンバ

データ型	メンバ名	説明
uint8_t *	p_adr	データの先頭アドレス
uint16_t	len	p_adr からの有効バイト数

4.3 R_RSA_KEY_t - RSA 鍵情報構造体

R_RSA_KEY_t 構造体は、RSA ライブラリが参照する鍵情報です。R_RSA_KEY_t 構造体は、RSA が鍵として使用する法と指数部を含みます。 法と指数部は、R_RSA_BYTEDATA_t 構造体として、データポインタとデータ長を含みます。

この時のデータポインタは、最上位バイトが 0x00 以外のデータを指します。 最上位バイトが 0x00 の場合は、各 API で正常に処理されず、戻り値に RSA_PARAM_ERR が返ります。

ユーザはこの構造体を変数で宣言します。ユーザはこの構造体の各メンバの値を設定します。p_adr には法または指数部の最上位バイトのアドレスを指定してください。 len には法または指数部の p_adr からの有効バイト数を指定してください。 以降、本マニュアルは法の p_adr からの有効バイト数を、"モジュラス n の有効バイト数"と定義して説明します。 また、指数部の p_adr からの有効バイト数を、"指数部の有効バイト数"と定義して説明します。

表 4-3 R_RSA_KEY_t のメンバ

データ型	メンバ名	説明
R_RSA_BYTEDATA_t	key_n	法(公開鍵(e,n)の n 、または秘密鍵(d,n)の n)
R_RSA_BYTEDATA_t	key_ed	指数部(公開鍵(e,n)の e 、または秘密鍵(d,n)の d)

公開鍵(e, n)の値が 65537(0x10001)の場合の設定値を以下に示します。(n の値は省略しています)

設定例

```
const uint8_t e[3] = {0x01,0x00,0x01};
const uint8_t n[256];
R_RSA_KEY_t rsa_key =
{
    n,      /* key_n.p_adr */
    256,    /* key_n.len */
    e,      /* key_ed.p_adr */
    3,      /* key_ed.len */
}
```

5. マクロ定義

このセクションでは、ライブラリで使用するマクロ定義について説明します。以下の定数は r_rsa.h に定義されています。

5.1 戻り値

RSA ライブラリの API は以下の表に示す戻り値を返します。

表 5.1 戻り値のマクロ

マクロ名	値	意味
RSA_OK	0	正常終了
RSA_PARAM_ERR	-1	パラメータ異常
RSA_USER_DEF_FUNC_ERR	-2	ユーザ定義関数異常終了
RSA_MOD_EXP_NG	-3	べき乗剰余演算 API の演算結果 NG
RSA_ENC_NG	-4	（未使用）
RSA_DEC_NG	-5	（未使用）
RSA_SIG_GEN_NG	-6	署名生成 API の署名生成失敗
RSA_SIG_VERIFY_NG	-7	署名検証 API の署名検証失敗

5.2 署名生成/検証の API に指定するハッシュの種類を定義するマクロ

以下のマクロは署名生成および検証で使用するハッシュの種類について定義します。

RSA ライブラリは署名生成/検証の API から、以下のマクロ値を引数としてユーザ定義関数を呼び出します。

ハッシュ関数はユーザで用意してください。

表 5.2 ハッシュ定義のマクロ

マクロ名	値	意味
RSA_HASH_MD2	0x00	MD2
RSA_HASH_MD5	0x01	MD5
RSA_HASH_SHA1	0x02	SHA-1
RSA_HASH_SHA256	0x03	SHA-256
RSA_HASH_SHA384	0x04	SHA-384
RSA_HASH_SHA512	0x05	SHA-512

6. ライブラリ関数

このセクションでは、ライブラリの API について説明します。
各 API の詳細は、以下のフォーマットに沿っています。

＜ API 書式 ＞
関数の呼び出し形式を示します。#include "ヘッダファイル"で示すヘッダファイルは、この関数の実行に必要なヘッダファイルです。必ずインクルードしてください。

＜ 引数 ＞
関数の引数を示します。"I/O"には引数がそれぞれ入力値、出力値であることを示します。"説明"には"引数名"についての説明を示します。

＜ 戻り値 ＞
関数の戻り値を示します。"説明"には戻り値の値についての説明を示します。

＜ 説明 ＞
関数の仕様を示します。

＜ 注意事項 ＞
関数を使用する際の注意事項を示します。

＜ 使用例 ＞
関数の使用例を示します。

6.1　R_rsa_signature_generate_pkcs

署名生成 (RSASSA-PKCS1-V1_5)

＜ API 書式 ＞

```
#include "r_rsa.h"

int16_t R_rsa_signature_generate_pkcs (
                    R_RSA_BYTEDATA_t *p_mes,
                    R_RSA_BYTEDATA_t *p_sig,
                    R_RSA_KEY_t *p_key,
                    uint8_t hash_type,
                    R_RSA_WORK_t *p_wk);
```

＜ 引数 ＞

引数名	I/O	説明
p_mes	I	署名を付けるメッセージ情報
p_sig	I/O	署名文格納先情報
p_key	I	鍵情報
hash_type	I	ハッシュの種類
p_wk	I/O	ワーク領域

＜ 戻り値 ＞

戻り値	説明
RSA_OK	正常終了
RSA_PARAM_ERR	パラメータ異常
RSA_USER_DEF_FUNC_ERR	ユーザ定義関数異常終了
RSA_SIG_GEN_NG	署名生成失敗

＜ 説明 ＞

この API は、RSASSA-PKCS1-V1_5 に従った署名文を生成します。

モジュラス n の有効バイト数の下限値は、引数 hash_type に応じて異なります。ユーザは以下の表に示すサイズ以上の鍵データを用意してください。

署名を付けるメッセージ情報には 0 バイト以上のデータを設定してください。0 バイトを設定する場合は、p_adr には 0 番地以外のアドレスを設定してください。

署名文格納先情報には、モジュラス n の有効バイト数以上の領域を設定してください。

p_sig に指定する領域、p_sig の p_adr に指定する領域、および p_wk に指定する領域は、RAM 領域に配置してください。

ユーザは、ユーザ定義関数 R_rsa_if_hash() に引数 hash_type で指定したハッシュ計算処理を実装する必要があります。

不正なアドレス(=0 番地)を指定した場合、パラメータ異常を返します。

API が戻り値に RSA_OK を返した時、署名文は p_sig の p_adr に示す領域に出力され、出力したバイト数は p_sig の len に設定されます。出力バイト数はモジュラス n の有効バイト数と同じになります。

表 6.1　モジュラス n の有効バイト数

引数 hash_type の値	モジュラス n の有効バイト数の下限値
RSA_HASH_MD2	45
RSA_HASH_MD5	45
RSA_HASH_SHA1	46
RSA_HASH_SHA256	62
RSA_HASH_SHA384	78
RSA_HASH_SHA512	94

＜ 使用例 ＞

```
int16_t ret;
uint8_t mes_buff[5] = {'a','b','c','d','e'};
uint8_t sig_buff[256];
R_RSA_WORK_t rsa_work;
R_RSA_BYTEDATA_t sig_text;
R_RSA_BYTEDATA_t message;
extern R_RSA_KEY_t rsa_private_key;

message.p_adr = mes_buff;
message.len = 5;
sig_text.p_adr = sig_buff;
sig_text.len = 256;

ret = R_rsa_signature_generate_pkcs(
    &message,
    &sig_text,
    &rsa_private_key,
    RSA_HASH_SHA256,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Signature generation was successful. */
}
```

6.2　R_rsa_signature_verify_pkcs

署名検証 (RSASSA-PKCS1-V1_5)

＜ API 書式 ＞

```
#include "r_rsa.h"

int16_t R_rsa_signature_verify_pkcs (
                    R_RSA_BYTEDATA_t *p_sig,
                    R_RSA_BYTEDATA_t *p_mes,
                    R_RSA_KEY_t *p_key,
                    uint8_t hash_type,
                    R_RSA_WORK_t *p_wk);
```

＜ 引数 ＞

引数名	I/O	説明
p_sig	I	検証する署名文情報
p_mes	I	検証するメッセージ情報
p_key	I	鍵情報
hash_type	I	ハッシュの種類
p_wk	I/O	ワーク領域

＜ 戻り値 ＞

戻り値	説明
RSA_OK	正常終了
RSA_PARAM_ERR	パラメータ異常
RSA_USER_DEF_FUNC_ERR	ユーザ定義関数異常終了
RSA_SIG_VERIFY_NG	署名検証失敗

＜ 説明 ＞

この API は、RSASSA-PKCS1-V1_5 に従って署名文を検証します。

モジュラス n の有効バイト数の下限値は、引数 hash_type に応じて異なります。ユーザは R_rsa_signature_generate_pkcs()のページの表に示すサイズ以上の鍵データを用意してください。

検証する署名文情報には、モジュラス n の有効バイト数と同じサイズのデータを設定してください。

検証するメッセージ情報には、0 バイト以上のデータを設定してください。

p_wk に指定する領域は、RAM 領域を指定してください。

ユーザは、ユーザ定義関数 R_rsa_if_hash() に引数 hash_type で指定したハッシュ計算処理を実装する必要があります。

不正なアドレス(=0 番地)を指定した場合、パラメータ異常を返します。

＜ 使用例 ＞

```
int16_t ret;
extern uint8_t mes_buff[5];
extern uint8_t sig_buff[256];
R_RSA_WORK_t rsa_work;
R_RSA_BYTEDATA_t sig_text;
R_RSA_BYTEDATA_t message;
extern R_RSA_KEY_t rsa_public_key;

message.p_adr = mes_buff;
message.len = 5;
sig_text.p_adr = sig_buff;
sig_text.len = 256;

ret = R_rsa_signature_verify_pkcs(
    &sig_text,
    &message,
    &rsa_public_key,
    RSA_HASH_SHA256,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Signature Verify was successful. */
}
```

6.3　R_rsa_mod_exp

べき乗剰余演算

＜ API 書式 ＞

```
#include "r_rsa.h"

int16_t R_rsa_mod_exp (
                R_RSA_BYTEDATA_t *p_input,
                R_RSA_BYTEDATA_t *p_output,
                R_RSA_KEY_t *p_key,
                R_RSA_WORK_t *p_wk);
```

＜ 引数 ＞

引数名	I/O	説明
p_input	I	入力データ情報
p_output	I/O	出力データ格納先情報
p_key	I	べき乗剰余の指数部(e または d)と法(n)の情報
p_wk	I/O	ワーク領域

＜ 戻り値 ＞

戻り値	説明
RSA_OK	正常終了
RSA_PARAM_ERR	パラメータ異常
RSA_MOD_EXP_NG	演算結果 NG

＜ 説明 ＞

この API は、以下のべき乗剰余演算を行います。

p_output = (p_input ^ p_key->key_e) MOD p_key->key_n

入力データ情報には、(1～モジュラス n の有効バイト数)バイトのデータを設定してください。

出力データ格納先情報には、モジュラス n の有効バイト数以上の領域を設定してください。

p_output に指定するアドレス、p_output の p_adr に指定するアドレス、p_wk に指定するアドレスは、RAM 領域に配置してください。

不正なアドレス(=0 番地)を指定した場合、パラメータ異常を返します。

API が戻り値に RSA_OK を返した時、出力データは p_output の p_adr に示す領域に出力され、出力したバイト数は p_output の len に設定されます。出力バイト数はモジュラス n の有効バイト数と同じになります。

本 API は、署名生成／署名検証を行う API（R_rsa_signature_generate_pkcs、R_rsa_signature_verify_pkcs）から、内部で自動的に呼び出されます。したがって、上記の API 実行時に、ユーザが本 API を呼び出す必要はありません。

＜ 使用例 ＞

```
int16_t ret;
uint8_t inbuff[5] = {'a','b','c','d','e'};
extern uint8_t outbuff[256];
R_RSA_WORK_t rsa_work;
R_RSA_BYTEDATA_t output;
R_RSA_BYTEDATA_t input;
extern R_RSA_KEY_t rsa_key;

input.p_adr = inbuff;
input.len = 5;
output.p_adr = outbuff;
output.len = 256;

ret = R_rsa_mod_exp(
    &input,
    &output,
    &rsa_key,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Successful. */
}
```

7. ユーザ定義関数

このセクションでは、RSA ライブラリが呼び出すユーザ定義関数について説明します。これらの関数のプロトタイプと各関数の実装で必要な処理について説明します。これらの関数の実装は、ユーザが作成する必要があります。

7.1　R_rsa_if_hash

ハッシュ計算

＜ API 書式 ＞

```
#include "r_rsa.h"

int16_t R_rsa_if_hash (
                        uint8_t * p_mes ,
                        uint8_t * p_hash ,
                        uint16_t mes_len ,
                        uint8_t hash_type );
```

＜ 引数 ＞

引数名	I/O	説明
p_mes	I	メッセージの先頭アドレス
p_hash	O	ハッシュ計算結果格納アドレス
mes_len	I	メッセージの有効バイト数
hash_type	I	ハッシュの種類

＜ 戻り値 ＞

戻り値	説明
0	ハッシュ値格納成功
0 以外	ハッシュ値格納失敗

＜ 説明 ＞

この関数の実装は、R_rsa_signature_generate()、R_rsa_signature_verify()を実行する場合に必要です。

引数 p_mes で指定されたアドレスから引数 mes_len バイトまでの領域についてハッシュ計算を行います。

ハッシュ方式は引数 hash_type で指定されます。

計算結果は、引数 p_hash に指定されたアドレスに格納してください。

計算結果を格納するサイズはハッシュ方式により異なります。

表 7.1　p_hash に格納するハッシュ値のバイト数

引数 hash_type の値	p_hash に格納するハッシュ値のバイト数
RSA_HASH_MD2	16
RSA_HASH_MD5	16
RSA_HASH_SHA1	20
RSA_HASH_SHA256	32
RSA_HASH_SHA384	48
RSA_HASH_SHA512	64

＜ 作成例 ＞

ルネサス製 SHA ライブラリを使用した場合

```
#include "r_sha.h"

int16_t R_rsa_if_hash(
    uint8_t *p_mes,
    uint8_t *p_hash,
    uint16_t mes_len,
    uint8_t hash_type)
{
    R_sha_handle sha_work;
    int8_t hash_ret; /* SHA Library return type */
    int16_t ret;

    ret = -1;
    switch(hash_type)
    {
        case RSA_HASH_SHA1:
            /* SHA-1 calculation */
            hash_ret = R_Sha1_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
                (void*)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
            {
                ret = 0;
            }
            break;
        case RSA_HASH_SHA256:
            /* SHA-256 calculation */
            hash_ret = R_Sha256_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
                (void *)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
            {
                ret = 0;
            }
            break;
        case RSA_HASH_MD2:
            /* MD2 calculation */
            break;
        case RSA_HASH_MD5:
            /* MD5 calculation */
            break;
        case RSA_HASH_SHA384:
            /* SHA-384 calculation */
#if !(defined(__RL78__) || defined(__ICCRL78__) || defined(__CCRL__))
            hash_ret = R_Sha384_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
                (void *)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
```

```
            {
                ret = 0;
            }
#endif
            break;
        case RSA_HASH_SHA512:
            /* SHA-512 calculation */
            break;
        default:
            break;
    }
    return ret;
}
```

注）添付のサンプルコード r_sample_rsa_if.c では、以下のオプションで Hash Digest の生成方法を選択できます。

定義	説明
#define USE_SHA_LIB (0) ※デフォルト値は「”0” : Hash Digest (固定値)」が設定されます	Hash Digest の生成方法の選択 0 : Hash Digest (固定値) を使用 1 : SHA ライブラリで Hash Digest を生成

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.03.31	ー	初版発行
1.01	2016.09.01	ー	営業お問合せ窓口住所変更 改定履歴追加
2.00	2021.04.23	ー	RSAES-PKCS1-V1_5 のサポートを終了しました。 それに伴い、R_rsa_encrypt_pkcs、R_rsa_decrypt_pkcs 、R_rsa_if_rand の関数仕様を削除しました。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 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。
6. 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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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。
13. 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。
14. 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.5.0-1　2020.10)

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。