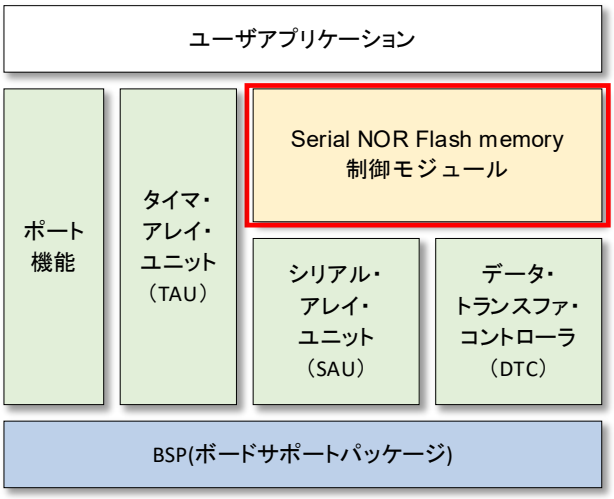
RL78 ファミリ

Serial NOR Flash Memory 制御モジュール Software Integration System

要旨

本アプリケーションノートは、Software Integration System (SIS)に準拠した Serial NOR Flash memory 制御モジュールについて説明します。本モジュールはルネサス エレクトロニクス製 RL78 ファミリ MCU 内蔵のシリアル・アレイ・ユニット（SAU）の簡易 SPI(CSI)を使用して、Serial NOR Flash memory を制御します。以降、本モジュールを Serial NOR Flash memory 制御モジュールと称します。

本モジュールの位置づけを以下に示します。



対象デバイス

・RL78/G15、RL78/G16、RL78/G22、RL78/G23、RL78/G24、RL78/L23

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

対象コンパイラ

・Renesas Electronics RL78 Family C Compiler Package
・LLVM for Renesas RL78
・IAR C/C++ Compiler for Renesas RL78

各コンパイラの動作確認内容については「7.1 動作確認環境」を参照してください。

関連ドキュメント

- RL78 ファミリ ボードサポートパッケージモジュール Software Integration System (R01AN5522)
- スマート・コンフィグレータ ユーザーズマニュアル RL78 API リファレンス編 (R20UT4852)

目次

1.　概要

1.1　Serial NOR Flash memory 制御モジュールとは

本モジュールはスマート・コンフィグレータのコード生成などルネサスが無償提供しているソフトウェアモジュール（図 1-1、表 1-1 参照）と組み合わせて使用することにより Renesas Electronics 社製、Macronix 社製の Serial NOR Flash memory の制御が可能になります。

制御対象の Serial NOR Flash memory は、「7.2 制御可能な Serial NOR Flash memory 製品」をご参照ください。

本モジュールは API として、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.11 SIS モジュールの追加方法」を参照してください。

Serial NOR Flash memory 制御モジュールを使用して Serial NOR Flash memory を制御する場合のアプリケーション構成例を図 1-1 に示します。

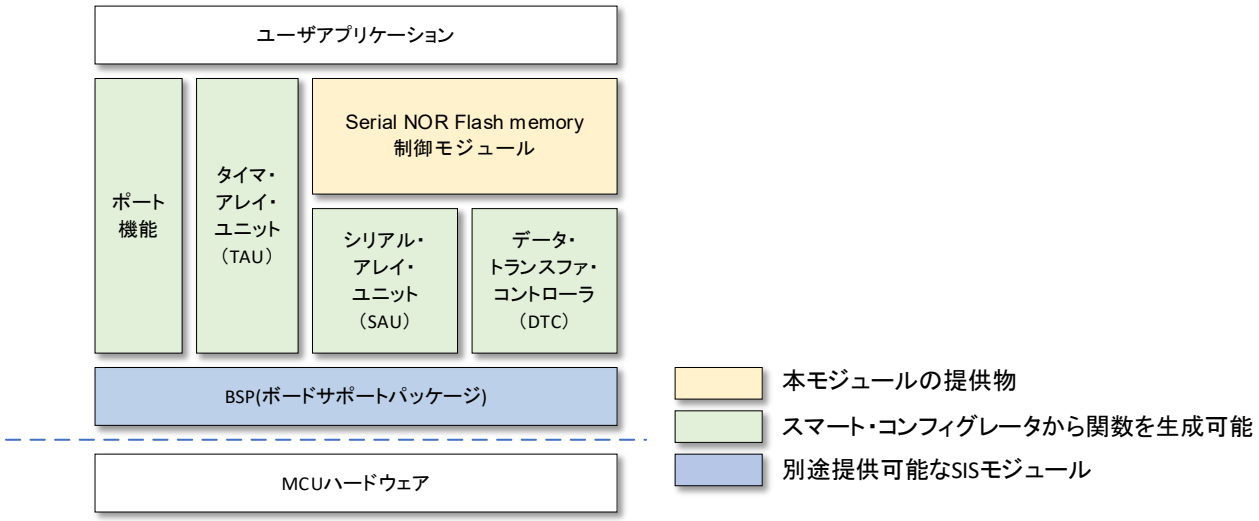


図 1-1　アプリケーション構成例

表 1-1 にアプリケーション構成で使用するモジュール一覧を示します。

表 1-1　アプリケーション構成で使用するモジュール一覧

名称	コンポーネント名称	コンポーネントタイプ
ボードサポートパッケージモジュール（BSP）	Board Support Packages	RL78 Software Integration System
シリアル・アレイ・ユニット（SAU）	SPI(CSI)通信	コード生成
データ・トランスファ・コントローラ（DTC）	データ・トランスファ・コントローラ	コード生成
タイマ・アレイ・ユニット（TAU）	インターバル・タイマ	コード生成
ポート機能	ポート	コード生成

(1)　ボードサポートパッケージモジュール（BSP）

クロックの設定など、初期設定を行います。
BSP については、BSP のドキュメントをご参照ください。
https://www.renesas.com/jp/ja/document/apn/rl78-family-board-support-package-module-using-software-integration-system

(2)　シリアル・アレイ・ユニット（SAU）

Serial NOR Flash memory と SPI 通信を行うためのコンポーネントです。

(3)　データ・トランスファ・コントローラ（DTC）

Serial NOR Flash memory と SPI 通信時に、CSI のレジスタへのリード、ライトを CPU を介さずに行うために使用するコンポーネントです。

(4)　タイマ・アレイ・ユニット (TAU)

DTC 転送を使用する場合、タイムアウトを検出するために使用します。

(5)　ポート機能

チップセレクト端子を選択して、H 出力に設定します。本 SIS モジュールを使用する前のチップセレクト端子の初期設定になります。

(6)　端子設定

スマート・コンフィグレータで SPI 通信を選択すると、シリアル・アレイ・ユニット（SAU）の端子も同時に設定されます。
SPI 通信で使用する端子を変更する場合は、スマート・コンフィグレータの端子ページから変更できます。

詳細はご使用の開発環境の「RL78 スマート・コンフィグレータ ユーザーガイド」をご参照ください。
RL78 スマート・コンフィグレータ | Renesas

1.2　API の概要

表 1-2 に本モジュールに含まれる API 関数を示します。

表 1-2　API 関数一覧

関数	関数説明
R_NOR_FLASH_Open()	本制御ソフトウェアの初期化処理
R_NOR_FLASH_Close()	本制御ソフトウェアの終了処理
R_NOR_FLASH_ReadStatus()	Status Register 1 読み出し処理
R_NOR_FLASH_ReadStatus2()	Status Register 2 読み出し処理
R_NOR_FLASH_ReadStatus3()	Status Register 3 読み出し処理
R_NOR_FLASH_WriteStatus()	Status Register 1 書き込み処理
R_NOR_FLASH_WriteStatus2()	Status Register 2 書き込み処理
R_NOR_FLASH_WriteStatus3()	Status Register 3 書き込み処理
R_NOR_FLASH_SetWriteProtect()	ライトプロテクト設定処理
R_NOR_FLASH_WriteDisable()	書き込み禁止コマンド処理
R_NOR_FLASH_ReadData()	データ読み出し処理
R_NOR_FLASH_WriteDataPage()	データ書き込み（1Page 書き込み用）処理
R_NOR_FLASH_Erase()	消去処理
R_NOR_FLASH_CheckBusy()	ビジーチェック処理
R_NOR_FLASH_ReadId()	ID 読み出し処理
R_NOR_FLASH_GetMemoryInfo()	メモリサイズ取得処理
R_NOR_FLASH_ReadConfiguration()	Configuration Register 読み出し処理
R_NOR_FLASH_WriteConfiguration()	Configuration Register 書き込み処理
R_NOR_FLASH_Set4byteAddressMode()	4 バイトアドレスモード設定処理
R_NOR_FLASH_ReadSecurity()	Security Register 読み出し処理
R_NOR_FLASH_ReadDataSecurityPage()	データセキュリティページ読み出し処理
R_NOR_FLASH_WriteDataSecurityPage()	データセキュリティページ書き込み処理
R_NOR_FLASH_GetVersion()	本制御ソフトウェアのバージョン情報取得処理
R_NOR_FLASH_Interval()　注 1	インターバルタイマカウンタ処理
R_NOR_FLASH_SendendNotification()	SPI 通信の送信完了コールバック処理

注 1：DTC 転送を使用する場合、タイムアウト検出のため、ハードウェアタイマやソフトウェアタイマ等を使って、1ms 間隔でコールする必要があります。

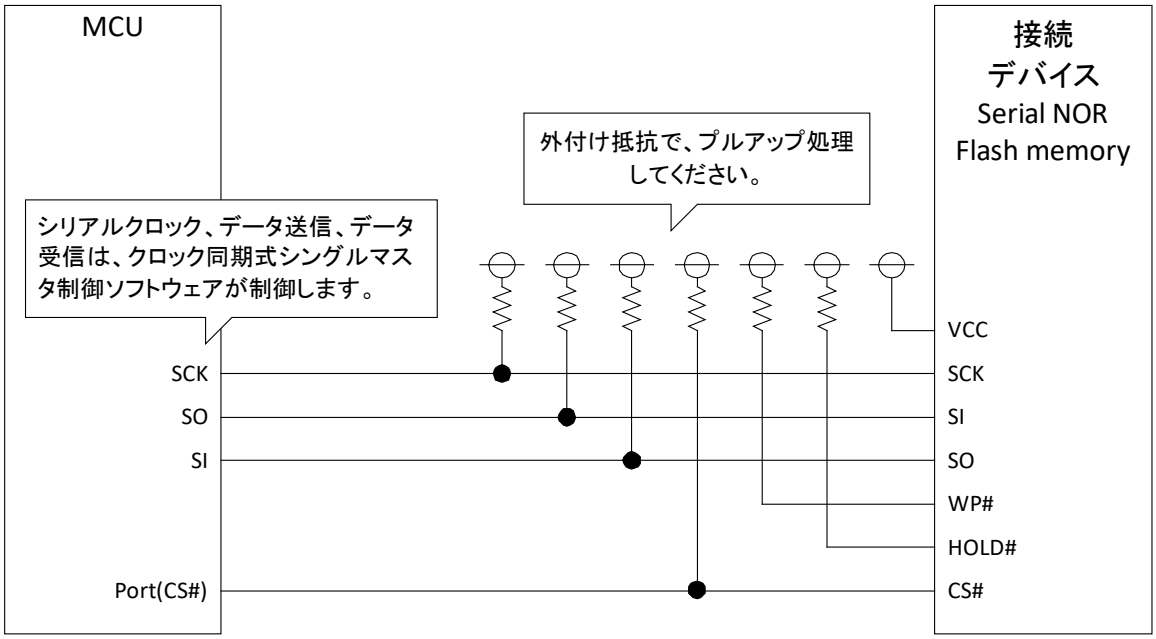
1.3　ハードウェア設定

1.3.1　ハードウェア構成例

図 **1-2** に端子接続図を示します。表 1-3 に示す使用端子と機能を参照し、使用する MCU の端子に割り当ててください。

なお、高速で動作させた場合を想定し、各信号ラインの回路的マッチングを取るためのダンピング抵抗やコンデンサの付加を検討してください。

本モジュールでは Single-SPI 制御のみサポートします。



・図中の端子名は Renesas Electronics 社製の Serial NOR Flash memory です。Serial NOR Flash memory の端子名は、製品によって異なります。
・WP#と HOLD#を使用しない場合の例です。WP#と HOLD#を使用する場合は接続デバイスの仕様書を確認してください。

図 1-2 端子接続図（Single-SPI 構成例 ）

表 1-3 使用端子と機能

端子名	入出力	内容
SCK	出力	クロック出力
SO	出力	マスタデータ出力
SI	入力	マスタデータ入力
Port (図 **1-2** の Port(CS#))	出力	チップセレクト(CS#)出力

1.4　ソフトウェア説明

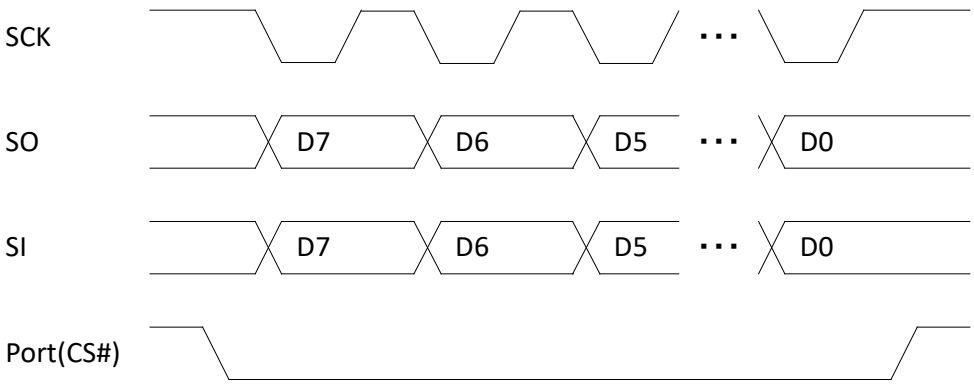
1.4.1　動作概要

MCU のクロック同期式シリアル通信機能を使って、内部クロックを使用したクロック同期式シングルマスタ制御を実現します。

なお、使用可能なシリアルクロック周波数は、MCU のユーザーズマニュアル ハードウェア編およびスレーブデバイスのデータシートで、確認してください。

本モジュールでは Single-SPI 制御のみサポートします。

図 **1-3** に示す SPI モード 3（CPOL=1、CPHA=1）で制御します。



・MCU->スレーブデバイスの送信時：転送クロックの立ち下がりで送信データ出力開始
　　転送クロックの立ち上がりで送信データ確定
・スレーブデバイス->MCU の受信時：転送クロックの立ち上がりで受信データの入力取り込み
・MSB ファーストでの転送
・転送を行っていないときの CLK 端子のレベル：”H”

図 1-3 制御可能なスレーブデバイスのタイミング

1.4.2　Serial NOR Flash memory のチップセレクト端子制御

Serial NOR Flash memory のチップセレクト端子を MCU の Port に接続し、MCU 汎用ポート出力で制御します。

Serial NOR Flash memory のチップセレクト（MCU の Port(CS#)）信号の立ち下がりから、Serial NOR Flash memory の Clock（MCU の SCK）信号の立ち下がりまでの時間は、Serial NOR Flash memory のチップセレクトセットアップ時間待ちのために、本モジュールがソフトウェア・ウェイトで制御します。

Serial NOR Flash memory の Clock（MCU の SCK）信号の立ち上がりから、Serial NOR Flash memory のチップセレクト（MCU の Port(CS#)）信号の立ち上がりまでの時間は、Serial NOR Flash memory のチップセレクトホールド時間待ちのために、本モジュールがソフトウェア・ウェイトで制御します。

本モジュールでは、チップセレクトセットアップ時間待ち、およびチップセレクトホールド時間待ちを約 1us としています。

2.　API 情報

本モジュールは、下記の条件で動作を確認しています。

2.1　ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- SPI（CSI）

2.2　ソフトウェアの要求

本モジュールは以下のパッケージに依存しています。

- r_bsp(Rev.1.62 以上)

また、以下のコード生成モジュールに依存しています。

- シリアル・アレイ・ユニット（SAU）
- データ・トランスファ・コントローラ（DTC）
- タイマ・アレイ・ユニット (TAU)
- ポート機能

2.3　サポートされているツールチェーン

本モジュールは「7.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4　使用する割り込みベクタ

なし

2.5　ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_nor_flash_rl78_if.h に記載しています。

ビルド毎の構成オプションは、r_nor_flash_rl78_config.h で選択します。

2.6　整数型

このドライバは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7　コンパイル時の設定

本モジュールのコンフィグレーションオプションの設定は、r_nor_flash_rl78_config.h で行います。

スマート・コンフィグレータ を使用して、ソフトウェアコンポーネント設定画面でコンフィグレーションオプションを設定できます。設定値はモジュールを追加する際に、自動的に r_nor_flash_rl78_config.h に反映されます。オプション名および設定値に関する説明を下表に示します。

Configuration options in r_nor_flash_rl78_config.h	
NOR_FLASH_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は“BSP_CFG_PARAM_CHECKING_ENABLE”	1：ビルド時にパラメータチェックの処理をコードに含めます。 0：ビルド時にパラメータチェックの処理をコードから省略します。 このオプションに BSP_CFG_PARAM_CHECKING_ENABLE を設定すると、システムのデフォルト設定が使用されます。
NOR_FLASH_CFG_WEL_CHK ※デフォルト値は“1（有効）”	Write Enable コマンド発行後に、WEL ビット確認を実行するかを選択できます。（1：有効、0：無効）
NOR_FLASH_CFG_DEVICE ※デフォルト値は NOR_FLASH_TYPE_AT25SF	制御する Serial NOR Flash memory を選択してください。 NOR_FLASH_TYPE_AT25SF：Renesas Electronics AT25SF NOR_FLASH_TYPE_AT25QF：Renesas Electronics AT25QF NOR_FLASH_TYPE_MX25L：Macronix MX25L NOR_FLASH_TYPE_MX66L：Macronix MX66L NOR_FLASH_TYPE_MX25R：Macronix MX25R
NOR_FLASH_CFG_SIZE ※デフォルト値は NOR_FLASH_SIZE_64M：64M-bit (8M Bytes)	制御する Serial NOR Flash memory の容量を選択してください。 NOR_FLASH_SIZE_4M：4M-bit　(512K Bytes) NOR_FLASH_SIZE_8M：8M-bit　(1M Bytes) NOR_FLASH_SIZE_16M：16M-bit　(2M Bytes) NOR_FLASH_SIZE_32M：32M-bit　(4M Bytes) NOR_FLASH_SIZE_64M：64M-bit　(8M Bytes) NOR_FLASH_SIZE_128M：128M-bit (16M Bytes) NOR_FLASH_SIZE_256M：256M-bit (32M Bytes) NOR_FLASH_SIZE_512M：512M-bit (64M Bytes) NOR_FLASH_SIZE_1G：1G-bit　(128M Bytes)
NOR_FLASH_CFG_CS_PORTNO ※デフォルト値は“0”	CS#を割り付けるポート番号を設定してください。 ポート番号を 0 - 15 で構成してください。
NOR_FLASH_CFG_CS_BITNO ※デフォルト値は“6”	CS#を割り付けるビット番号を設定してください。 ビット番号を 0 - 7 で構成してください。
NOR_FLASH_CFG_MODE_TRNS ※デフォルト値は NOR_FLASH_TRNS_CPU：CPU 転送（Software 転送）	NOR_FLASH_TRNS_CPU：CPU 転送（Software 転送） NOR_FLASH_TRNS_DTC：DTC 転送
NOR_FLASH_CFG_DRVR_CH ※デフォルト値は NOR_FLASH_DRVR_CH3：CSI11	SPI 通信に使用するチャネル番号を指定してください。 NOR_FLASH_DRVR_CH0：CSI00 NOR_FLASH_DRVR_CH1：CSI01 NOR_FLASH_DRVR_CH2：CSI10 NOR_FLASH_DRVR_CH3：CSI11 NOR_FLASH_DRVR_CH4：CSI20 NOR_FLASH_DRVR_CH5：CSI21 NOR_FLASH_DRVR_CH6：CSI30 NOR_FLASH_DRVR_CH7：CSI31
NOR_FLASH_CFG_DTCD_NO ※デフォルト値は“0”	DTC を使用する場合に、SPI 通信に使用する DTC コントロール・データ番号を設定してください。 DTC コントロール・データ番号は 0 - 22 で指定してください。

2.8　コードサイズ

本モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィギュレーションオプションによって決まります。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_nor_flash rev.1.00

コンパイラバージョン: Renesas Electronics RL78 Family C Compiler Package V1.13.00

(統合開発環境のデフォルト設定)

LLVM for Renesas RL78 10.0.0.202312

(統合開発環境のデフォルト設定)

IAR C/C++ Compiler for Renesas RL78 version 5.10.3

(統合開発環境のデフォルト設定)

コンフィグレーションオプション: デフォルト設定

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ							
		Renesas Compiler				LLVM Compiler		IAR Compiler	
		最適化レベル -Olite		最適化レベル -Odefault		Optmization level (-Og)		レベル 低	
		パラメータチェック		パラメータチェック		パラメータチェック		パラメータチェック	
		あり	なし	あり	なし	あり	なし	あり	なし
RL78/G23	ROM	2883	2608	2725	2511	3208	2926	2803	2531
	RAM	22		22		22		22	
	最大使用スタックサイズ	150		138		158		116	

2.9　引数

API 関数の引数である構造体を示します。この構造体は、API 関数のプロトタイプ宣言とともに r_nor_flash_rl78_if.h に記載されています。

(1)　st_nor_flash_info_t 構造体定義

```
/* Flash memory information */
typedef struct
{
    uint32_t        addr;           /* コマンドで指定する開始アドレス */
    uint16_t        cnt;            /* 読み出し／書き込みを行うバイト数 */
    uint16_t        data_cnt;       /* 本モジュールで使用するカウンタ */
    uint8_t       * p_data;         /* データ格納先ポインタ */
} st_nor_flash_info_t;              /* 10 bytes */
```

(2)　st_nor_flash_mem_info_t 構造体定義

```
/* Flash memory size information */
typedef struct
{
    uint32_t        mem_size;       /* 最大メモリサイズ */
    uint32_t        wpag_size;      /* 書き込みページサイズ */
} st_nor_flash_mem_info_t;          /* 8 bytes */
```

(3)　st_nor_flash_erase_info_t 構造体定義

```
/* Flash memory erase information */
typedef struct
{
    uint32_t        addr;           /* コマンドで指定する開始アドレス */
    e_nor_flash_erase_mode_t  mode; /* 消去モード */
} st_nor_flash_erase_info_t;        /* 6 bytes */
```

(4)　st_nor_flash_reg_info_t 構造体定義

```
/* Flash memory register information */
typedef struct
{
    uint8_t         status;         /* ステータスレジスタ */
    uint8_t         config1;        /* コンフィグレーションレジスタ-1 */
    uint8_t         config2;        /* コンフィグレーションレジスタ-2*/
    uint8_t         rsv[1];
} st_nor_flash_reg_info_t;          /* 4 bytes */
```

2.10　戻り値／エラーコード

API 関数の戻り値を示します。この列挙型は、API 関数のプロトタイプ宣言とともに r_nor_flash_rl78_if.h で記載されています。

本モジュールの API 関数は、その処理の途中でエラーが発生した場合、戻り値にエラーコードを返します。

表 2-1 にエラーコードを示します。なお、表にない値は、将来のための予約です。

表 2-1　エラーコード

マクロ定義	値	意味
NOR_FLASH_SUCCESS_BUSY	1	正常終了：Serial NOR Flash memory が busy 状態
NOR_FLASH_SUCCESS	0	正常終了
NOR_FLASH_ERR_PARAM	-1	パラメータエラー
NOR_FLASH_ERR_HARD	-2	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	-3	モジュール未 OPEN エラー
NOR_FLASH_ERR_ALREADY_OPEN	-4	モジュール OPEN 済エラー
NOR_FLASH_ERR_WEL_CHK	-5	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	-6	未サポート API

2.11　SIS モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。

(1) e² studio 上でスマート・コンフィグレータを使用して SIS モジュールを追加する場合
e² studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに SIS モジュールを追加します。詳細は、アプリケーションノート「RL78 スマート・コンフィグレータ ユーザーガイド e² studio 編 (R20AN0579)」を参照してください。

(2) IAREW 上でスマート・コンフィグレータを使用して SIS モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに SIS モジュールを追加します。詳細は、アプリケーションノート「RL78 スマート・コンフィグレータ ユーザーガイド IAR 編 (R20AN0581)」を参照してください。

(3) CS+上でスマート・コンフィグレータ を使用して SIS モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに SIS モジュールを追加します。詳細は、アプリケーションノート「RL78 スマート・コンフィグレータ ユーザーガイド (R20AN0580)」を参照してください。

2.12　for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理などで for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

```
for 文の例：
/* WAIT_LOOP */
for (w_count = 0U; w_count <= BSP_CFG_FIHWAITTIME; w_count++)
{
    BSP_NOP();
}

do while 文の例：
/* WAIT_LOOP */
do
{
    tmp stab wait  = OSTC;
    tmp stab wait &= tmp stab set;
}
while (tmp_stab_wait != tmp_stab_set);
```

3.　API 関数

API 関数は r_nor_flash_rl78_if.h にプロトタイプ宣言されています。

API 関数のコンセプト

1. 本モジュールを使用する際、必ず最初に R_NOR_FLASH_Open 関数を呼び出してください。
2. API 関数は全てブロッキング処理となっています。DTC 転送を使用した場合も同様です。
3. 必ず API 関数の処理を終えてから、次の API 関数を呼び出してください。
4. Serial NOR Flash memory への書き込みや消去を実行後は、R_NOR_FLASH_CheckBusy 関数を使用して書き込み、消去の完了を確認してください。
5. リエントラントには非対応です。

3.1　R_NOR_FLASH_Open()

Serial NOR Flash memory 制御ソフトウェアを使用する際に、必ず最初にコールする関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_Open(
void
)
```

Parameters

なし

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_ALREADY_OPEN	本モジュールが OPEN 済

Description

チップセレクト端子を初期化します。実行後は、ポート H 出力状態になります。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Open();
```

Special Notes

なし

3.2　R_NOR_FLASH_Close()

使用中の Serial NOR Flash memory 制御ソフトウェアを終了する際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_Close(
void
)
```

Parameters

なし

Return Values

NOR_FLASH_SUCCESS	正常終了

Description

チップセレクト端子を初期化します。実行後は、入力ポート状態になります。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Close();
```

Special Notes

なし

3.3　R_NOR_FLASH_ReadStatus()

Serial NOR Flash memory の Status Register 1 を読み出す際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus(
uint8_t * p_status
)
```

Parameters

* p_status

Status Register 1 格納バッファ（読み出しバッファとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

Serial NOR Flash memory の Status Register 1 を読み出し、p_status に格納します。

Status Register 1 については、使用する Serial NOR Flash memory のデータシートを参照してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus(&stat);
```

Special Notes

なし

3.4　R_NOR_FLASH_ReadStatus2()

Serial NOR Flash memory の Status Register 2 を読み出す際に使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus2(
uint8_t * p_status
)
```

Parameters

* p_status

Status Register 2 格納バッファ（読み出しバッファとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

Serial NOR Flash memory の Status Register 2 を読み出し、p_status に格納します。

Status Register 2 については、使用する Serial NOR Flash memory のデータシートを参照してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus2(&stat);
```

Special Notes

なし

3.5　R_NOR_FLASH_ReadStatus3()

Serial NOR Flash memory の Status Register 3 を読み出す際に使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus3(
uint8_t * p_status
)
```

Parameters

* p_status

Status Register 3 格納バッファ（読み出しバッファとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

Serial NOR Flash memory の Status Register 3 を読み出し、p_status に格納します。

Status Register 3 については、使用する Serial NOR Flash memory のデータシートを参照してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus3(&stat);
```

Special Notes

なし

3.6　R_NOR_FLASH_WriteStatus()

Serial NOR Flash memory の Status Register 1 への書き込みに使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus(
uint8_t * p_reg
)
```

Parameters

* p_reg

Status Register 1 設定データバッファ（設定データとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

p_reg に設定された値は Status Register 1 に書き込まれます。ただし、使用する Serial NOR Flash memory によっては、機能が割り当てられている場合や、Reserved bit の場合があります。

Status Register 1 については、使用する Serial NOR Flash memory のデータシートを参照してください。

このユーザ API 関数を呼び出す前に、Status Register 1 の値を読み取り、上書きが必要なビットのみ値を変更してください。 処理終了後、Status Register 1 を読み出し、書き込んだ値が正しいことを確認してください。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

3.7　R_NOR_FLASH_WriteStatus2()

Serial NOR Flash memory の Status Register 2 への書き込みに使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus2(
uint8_t * p_reg
)
```

Parameters

* p_reg

Status Register 2 設定データバッファ（設定データとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

p_reg に設定された値は Status Register 2 に書き込まれます。ただし、使用する Serial NOR Flash memory によっては、機能が割り当てられている場合や、Reserved bit の場合があります。

Status Register 2 については、使用する Serial NOR Flash memory のデータシートを参照してください。

このユーザ API 関数を呼び出す前に、Status Register 2 の値を読み取り、上書きが必要なビットのみ値を変更してください。 処理終了後、Status Register 2 を読み出し、書き込んだ値が正しいことを確認してください。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus2(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus2(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
/* Error */
}
```

Special Notes

なし

3.8　R_NOR_FLASH_WriteStatus3()

Serial NOR Flash memory の Status Register 3 への書き込みに使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus3(
uint8_t * p_reg
)
```

Parameters

* p_reg

Status Register 3 設定データバッファ（設定データとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

p_reg に設定された値は Status Register 3 に書き込まれます。ただし、使用する Serial NOR Flash memory によっては、機能が割り当てられている場合や、Reserved bit の場合があります。

Status Register 3 については、使用する Serial NOR Flash memory のデータシートを参照してください。

このユーザ API 関数を呼び出す前に、Status Register 3 の値を読み取り、上書きが必要なビットのみ値を変更してください。 処理終了後、Status Register 3 を読み出し、書き込んだ値が正しいことを確認してください。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus3(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus3(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

3.9　R_NOR_FLASH_SetWriteProtect()

Serial NOR Flash memory のライトプロテクトを設定する際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_SetWriteProtect(
uint8_t wpsts
)
```

Parameters

wpsts

ライトプロテクト設定データ

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー

Description

ライトプロテクトを設定します。

Macronix International Co., Ltd 社製の場合、SRWD ビットの値を 0 に設定します。

Renesas Electronics 社製の場合、SRP0 ビットの値は 0 を設定します。

ライトプロテクト設定データ(wpsts)は下記のように設定してください。

プロテクト領域とプロテクトビットの関係は、使用する Serial NOR Flash memory のデータシートを参照してください。

< Renesas Electronics 社製 AT25QF family Serial NOR Flash memory>

Status Register 1 の書き込み処理中に大小ブロックサイズ、トップ、ボトムの設定を行います。

Status Register 2 の書き込み処理中に補完設定を行います。

wpsts	BP2	BP1	BP0
0x00	0	0	0
0x01	0	0	1
0x02	0	1	0
0x03	0	1	1
0x04	1	0	0
0x05	1	0	1
0x06	1	1	0
0x07	1	1	1

< Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory >

Top/Bottom 設定は、Configuration Register 書き込み処理で設定してください。

wpsts	BP3	BP2	BP1	BP0
0x00	0	0	0	0
0x01	0	0	0	1
0x02	0	0	1	0
0x03	0	0	1	1
0x04	0	1	0	0
0x05	0	1	0	1
0x06	0	1	1	0
0x07	0	1	1	1
0x08	1	0	0	0
0x09	1	0	0	1
0x0a	1	0	1	0
0x0b	1	0	1	1
0x0c	1	1	0	0
0x0d	1	1	0	1
0x0e	1	1	1	0
0x0f	1	1	1	1

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

詳しくは、図 3-1 を参照してください。

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t    ret = NOR_FLASH_SUCCESS;
uint32_t                loop_cnt = 0;
e_nor_flash_check_busy_t      mode;

ret = R_NOR_FLASH_SetWriteProtect(0);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
        User application can perform other processing while flash is busy. */
    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

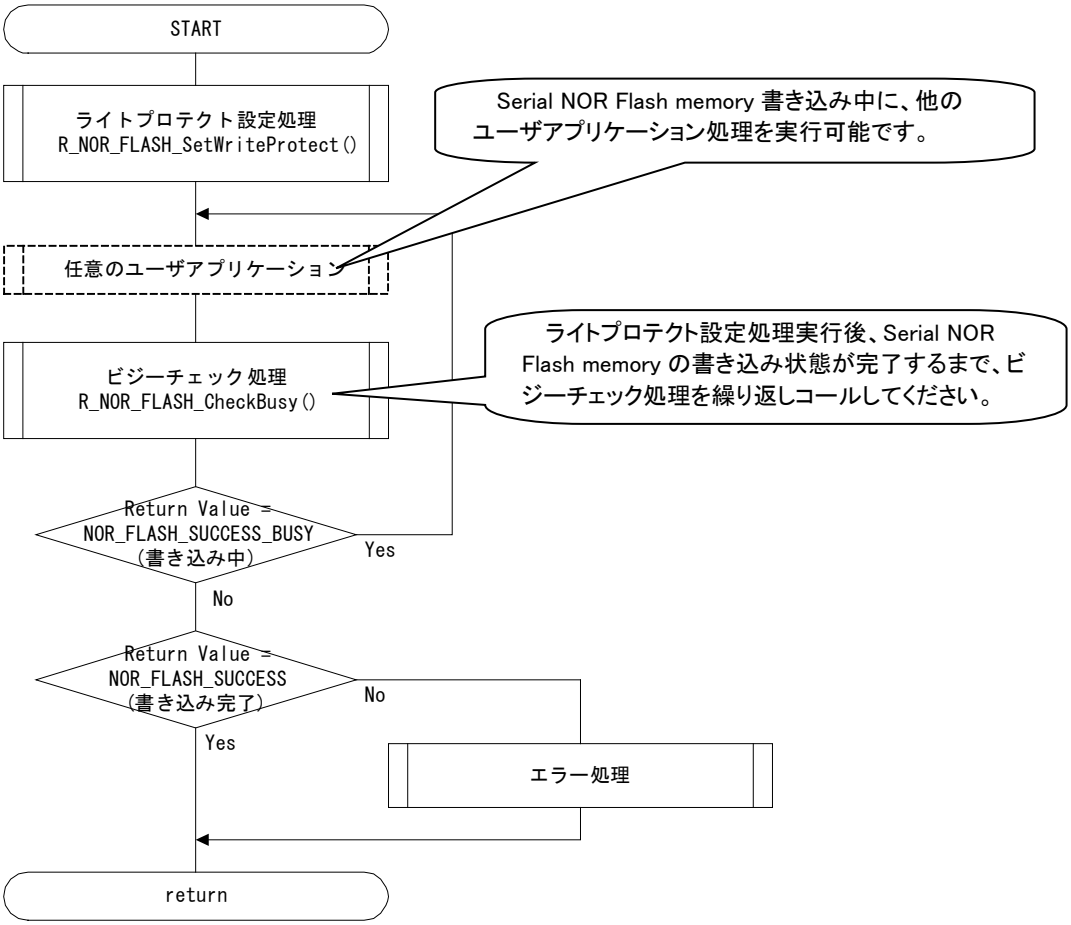


図 3-1　R_NOR_FLASH_SetWriteProtect()の処理例

3.10　R_NOR_FLASH_WriteDisable()

Serial NOR Flash memory の書き込みを禁止する際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDisable(
void
)
```

Parameters

なし

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

Write Disable コマンドを送信し、Serial NOR Flash memory の Status Register の WEL ビットをクリアします。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_WriteDisable();
```

Special Notes

なし

3.11　R_NOR_FLASH_ReadData()

Serial NOR Flash memory からデータを読み出す際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadData(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

* p_nor_flash_info

Serial NOR Flash memory 情報構造体。

addr

メモリの読み出し開始アドレスを設定してください。

cnt

読み出しバイト数を設定してください。設定可能範囲は 1～65,535 です。

0 を設定した場合、エラーを返します。

data_cnt

読み出しバイト数（本 API 関数で使用するため、ユーザ設定禁止）

*p_data

読み出しデータ格納バッファのアドレスを設定してください。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

Serial NOR Flash memory 上の指定アドレスから指定バイト数分、データを読み出し、p_data に格納します。

Serial NOR Flash memory の最終アドレスは、Serial NOR Flash memory 容量-1 です。

最終アドレスを超える読み出し指定はできません。最終アドレスの読み出し後、一度処理を完了させて、再度アドレスを設定し直してから、本ユーザ API をコールしてください。

読み出しバイト数 cnt と指定アドレス addr の合計値が最終アドレスを超えないように設定してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_R;
uint8_t buf2[128];

Flash_Info_R.addr = 0;
Flash_Info_R.cnt = 32;
Flash_Info_R.p_data = &buf2[0];
ret = R_NOR_FLASH_ReadData(&Flash_Info_R);
```

Special Notes

なし

3.12　R_NOR_FLASH_WriteDataPage()

Serial NOR Flash memory へ、1Page（最大 256byte）単位でデータを書き込む際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDataPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

* p_nor_flash_info

Serial NOR Flash memory 情報構造体。

addr

メモリの書き込み開始アドレスを設定してください。

cnt

書き込みバイト数を設定してください。設定可能範囲は 1～65,535 です。

0 を設定した場合、エラーを返します。

data_cnt

書き込みバイト数（本 API 関数で使用するため、ユーザ設定禁止）

*p_data

書き込みデータ格納バッファのアドレスを設定してください。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー

Description

p_data のデータを Serial NOR Flash memory 上の指定アドレスから指定バイト数分（最大 1 Page（256byte）サイズ）書き込みます。

1Page を超えるバイト数が設定されている場合でも、1Page 書き込み処理完了後、残バイト数と次アドレス情報が Serial NOR Flash memory 情報構造体 p_nor_flash_info に残ります。未変更のまま再びその p_nor_flash_info を本ユーザ API にセットすることで残バイト数の書き込みが可能です。

書き込みバイト数（cnt）に設定できる最大値は、Serial NOR Flash memory 容量の値です。

書き込みバイト数 cnt と指定アドレス addr の合計値が Serial NOR Flash memory の最終アドレスを超えないように設定してください。

Example

```
#define FLASH_PP_BUSY_WAIT (uint32_t)(3) /* 3 * 1ms = 3ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_W;
uint8_t buf1[512];
uint32_t loop_cnt = 0;

Flash_Info_W.addr = 0;
Flash_Info_W.cnt = 512;
Flash_Info_W.p_data = &buf1[0];

do
{
    ret = R_NOR_FLASH_WriteDataPage(&Flash_Info_W);
    if (NOR_FLASH_SUCCESS > ret)
    {
        /* Error */
    }

    loop_cnt = FLASH_PP_BUSY_WAIT;
    mode = NOR_FLASH_MODE_PROG_BUSY;
    do
    {
        /* FLASH is busy.
        User application can perform other processing while flash is busy. */
        ret = R_NOR_FLASH_CheckBusy(mode);
        if (NOR_FLASH_SUCCESS_BUSY != ret)
        {
            /* FLASH is ready or error. */
            break;
        }
        loop_cnt--;
        wait_timer(0, 1); /* 1ms */
    }
    while (0 != loop_cnt);
}
while (0 != Flash_Info_W.cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
/* Error */
}
```

Special Notes

Serial NOR Flash memory への書き込みは、ライトプロテクト解除状態の場合のみ可能です。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

詳しくは、図 3-2 を参照してください。

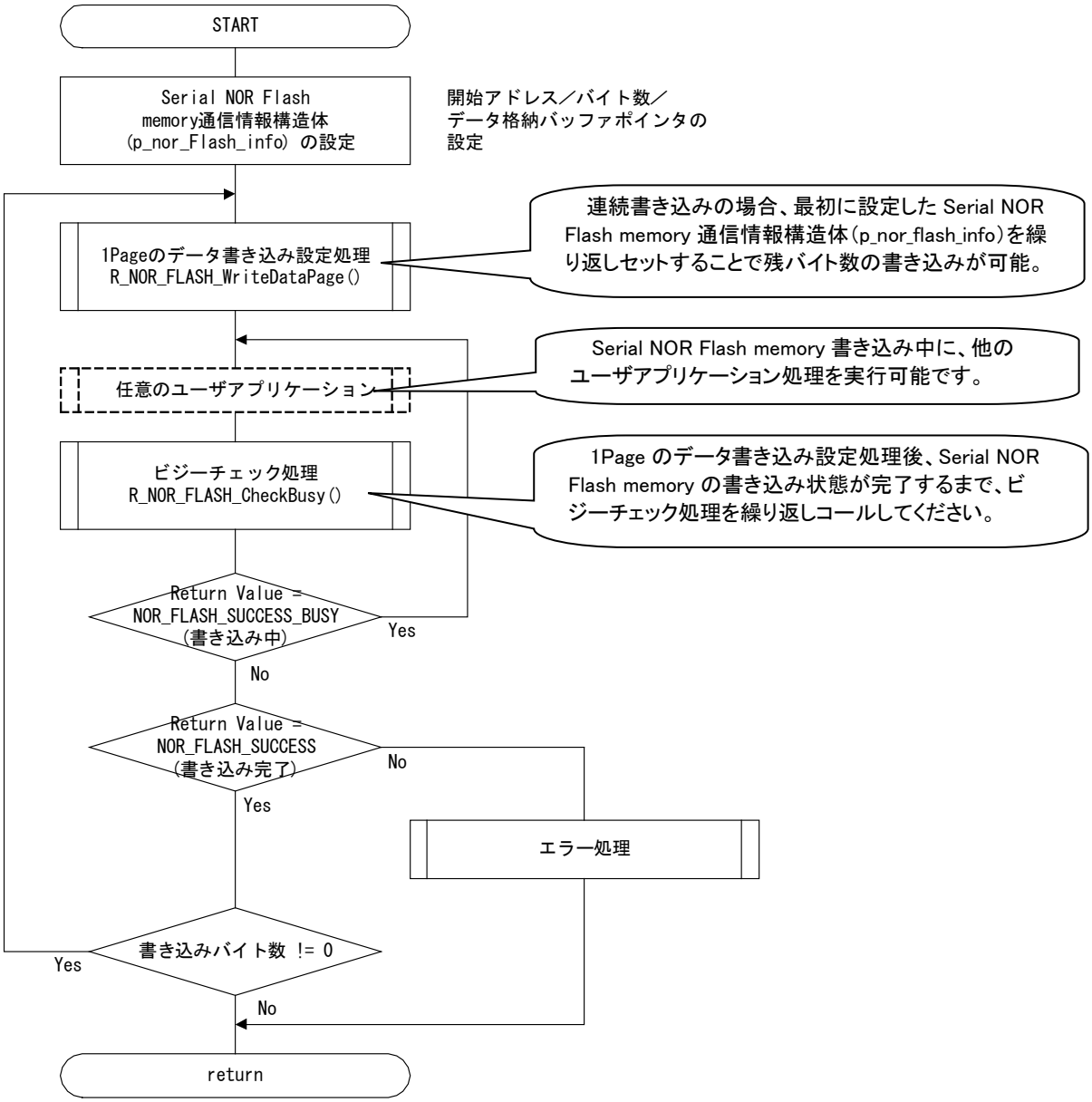


図 3-2　R_NOR_FLASH_WriteDataPage()の処理例

3.13　R_NOR_FLASH_Erase()

Serial NOR Flash memory に対し、指定したメモリを消去する際に使用する関数です。消去する対象は引数の mode で指定できます。

Format

```
e_nor_flash_status_t R_NOR_FLASH_Erase(
st_nor_flash_erase_info_t * p_nor_flash_erase_info
)
```

Parameters

* p_nor_flash_erase_info

Serial NOR Flash memory Erase 情報構造体。

addr

メモリの書き込み開始アドレスを設定してください。

mode

消去モード設定

以下から 1 つ選択してください。

＜Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory＞

```
NOR_FLASH_MODE_C_ERASE      /* チップの全データ消去（Chip Erase） */
NOR_FLASH_MODE_BK4_ERASE    /* ブロックの全データ消去（Block Erase：4KB） */
NOR_FLASH_MODE_B32K_ERASE   /* ブロックの全データ消去（Block Erase：32KB） */
NOR_FLASH_MODE_B64K_ERASE   /* ブロックの全データ消去（Block Erase：64KB） */
NOR_FLASH_MODE_SCUR_ERASE   /* Security Register ページを消去します */
```

＜Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory の場合＞

```
NOR_FLASH_MODE_C_ERASE      /* チップの全データ消去（Chip Erase） */
NOR_FLASH_MODE_S_ERASE      /* セクタの全データ消去（Sector Erase：4KB） */
NOR_FLASH_MODE_B32K_ERASE   /* ブロックの全データ消去（Block Erase：32KB） */
NOR_FLASH_MODE_B64K_ERASE   /* ブロックの全データ消去（Block Erase：64KB） */
```

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー

Description

Sector Erase の場合、addr にはセクタの先頭アドレスを設定してください。

Block Erase の場合、addr にはブロックの先頭アドレスを設定してください。

セキュリティ消去で、addr に Security Register ページの開始アドレスを指定します。

Chip Erase の場合、addr には 0x00000000 を設定してください。

Serial NOR Flash memory への消去は、ライトプロテクト解除状態の場合のみ可能です。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は消去状態に遷移しています。必ず、消去完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、消去状態中にユーザアプリケーションの他の処理を行うことができます。

詳しくは、図 3-3 を参照してください。

Example

```
#define FLASH_SE_BUSY_WAIT (uint32_t)(200) /* 200 * 1ms = 200ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_erase_info_t Flash_Info_E;
uint32_t loop_cnt = 0;

Flash_Info_E.addr = 0;
Flash_Info_E.mode = NOR_FLASH_MODE_S_ERASE;

ret = R_NOR_FLASH_Erase(&Flash_Info_E);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_SE_BUSY_WAIT;
mode = NOR_FLASH_MODE_ERASE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

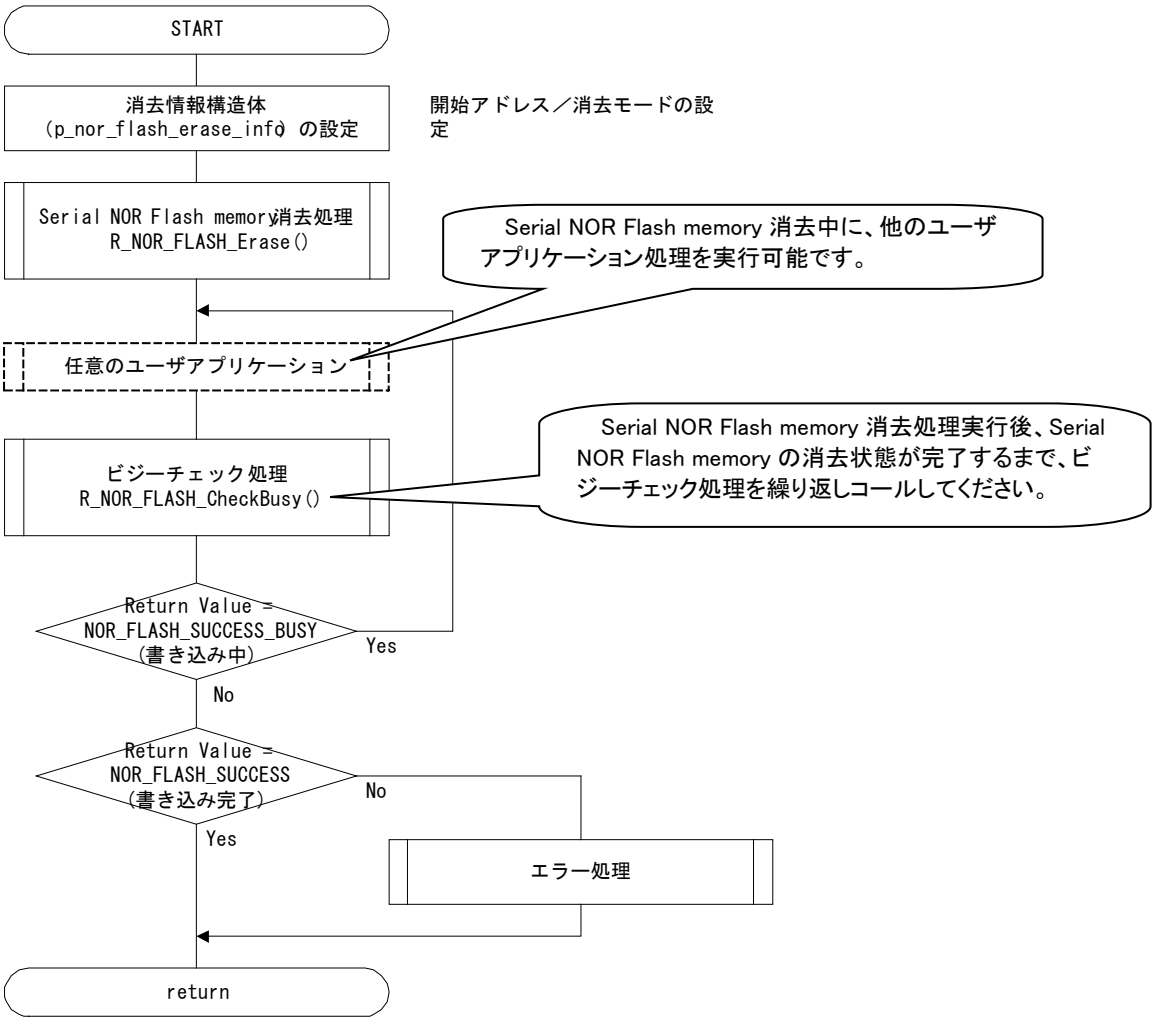


図 3-3　R_NOR_FLASH_Erase()の処理例

3.14　R_NOR_FLASH_CheckBusy()

Serial NOR Flash memory への書き込み、または消去完了を確認する際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_CheckBusy(
e_nor_flash_check_busy_t mode
)
```

Parameters

mode

完了確認処理設定

< Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory >

NOR_FLASH_MODE_REG_WRITE_BUSY　/* レジスタ書き込み完了確認 */

< Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory の場合 >

以下から 1 つ選択してください。

NOR_FLASH_MODE_REG_WRITE_BUSY　/* レジスタ書き込み完了確認 */

NOR_FLASH_MODE_PROG_BUSY　/* データ書き込み完了確認 */

NOR_FLASH_MODE_ERASE_BUSY　/* 消去完了確認 */

Return Values

NOR_FLASH_SUCCESS	正常終了、かつ、書き込み完了
NOR_FLASH_SUCCESS_BUSY	正常終了、かつ、書き込み中
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

書き込み、または消去状態の完了判定を行います。

Example

図 3-1 もしくは図 3-2 をご参照ください。

Special Notes

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

3.15　R_NOR_FLASH_ReadId()

Serial NOR Flash memory の ID 情報を読み出す際に使用する関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadId(
uint8_t * p_data
)
```

Parameters

* p_data

ID 情報格納バッファ。

Manufacture ID、及び Device ID を読み出します。読み出しバッファとして、3 バイトを設定してください。

(1) Manufacturer ID（1 バイト）

(2) Device ID（2 バイト）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

Serial NOR Flash memory の ID 情報を p_data に格納します。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t gID[4];

ret = R_NOR_FLASH_ReadId(&gID[0]);
```

Special Notes

なし

3.16　R_NOR_FLASH_GetMemoryInfo()

Serial NOR Flash memory のサイズ情報を戻り値として返す関数です。

Format

e_nor_flash_status_t R_NOR_FLASH_GetMemoryInfo(
st_nor_flash_mem_info_t * p_nor_flash_mem_info
)

Parameters

* p_nor_flash_mem_info

Serial NOR Flash memory サイズ情報構造体。

mem_size

最大メモリサイズ

wpag_size

ページサイズ

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常

Description

Serial NOR Flash memory サイズ情報を取得します。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_mem_info_t Flash_MemInfo;

ret = R_NOR_FLASH_GetMemoryInfo(&Flash_MemInfo);
```

Special Notes

指定した Serial NOR Flash memory のサイズ情報を戻り値として返す関数です。Serial NOR Flash memory との通信はせず、本 API が定義した固定値を返します。

3.17　R_NOR_FLASH_ReadConfiguration()

Serial NOR Flash memory の Configuration Register を読み出す際に使用する関数です。Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory 専用の API です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadConfiguration(
uint8_t * p_config
)
```

Parameters

* p_config

Configuration Register 格納バッファ。使用する Serial NOR Flash memory によりサイズが異なります。下記を参照し、読み出しバッファを確保してください。

< Macronix International Co., Ltd 社製 MX25L/MX66L family serial NOR Flash memory の場合 >

Configuration Register を読み出します。読み出しバッファとして、1 バイトを設定してください。

< Macronix International Co., Ltd 社製 MX25R family serial NOR Flash memory の場合 >

Configuration Register-1 と Configuration Register-2 を読み出します。読み出しバッファとして、2 バイトを設定してください。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

Serial NOR Flash memory の Configuration Register を読み出し、p_config に格納します。

Configuration Register については、使用する Serial NOR Flash memory のデータシートを参照してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t gConfig[2];

ret = R_NOR_FLASH_ReadConfiguration(&gConfig[0]);
```

Special Notes

なし

3.18　R_NOR_FLASH_WriteConfiguration()

Serial NOR Flash memory の Configuration Register への書き込みに使用する関数です。Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory 専用の API です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteConfiguration(
st_nor_flash_reg_info_t * p_reg
)
```

Parameters

* p_reg

レジスタ情報構造体。

status

Status Register（本 API 関数で使用するため、ユーザ設定禁止）

config1

Configuration Register 設定データ

config2

Configuration Register-2 設定データ

なお、使用する Serial NOR Flash memory により構造体の構成が異なります。下記を参照し、値を設定してください。また、設定値は Description を参照してください。

< Macronix International Co., Ltd 社製 MX25L/MX66L family serial NOR Flash memory の場合 >

p_reg->config1 に設定した値を Configuration Register に書き込みます。

p_reg->config2 の設定は無効です。

< Macronix International Co., Ltd 社製 MX25R family serial NOR Flash memory の場合 >

p_reg->config1 に設定した値を Configuration Register-1 に書き込みます。

p_reg->config2 に設定した値を Configuration Register-2 に書き込みます。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

p_reg->config1 及び p_reg->config2 に設定した値を Configuration Register に書き込みます。

Configuration Register については、使用する Serial NOR Flash memory のデータシートを参照してください。

本ユーザ API をコールする場合、事前に Configuration Register の値を読み出し、書き換えたい bit 値のみ変更し、p_reg->config1 及び p_reg->config2 に設定してください。

処理終了後、Configuration Register を読み出して、書き込み値を確認してください。

4BYTE bit は読み出し専用であるため、設定は無視されます。

本ユーザ API が正常終了した場合、Serial NOR Flash memory は書き込み状態に遷移しています。必ず、書き込み完了を R_NOR_FLASH_CheckBusy()で確認してください。

R_NOR_FLASH_CheckBusy()はユーザの任意のタイミングでコールすることができます。そのため、書き込み状態中にユーザアプリケーションの他の処理を行うことができます。

詳しくは、図 3-4 を参照してください。

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
st_nor_flash_reg_info_t Reg;
uint8_t gConfig[2];

ret = R_NOR_FLASH_ReadConfiguration(&gConfig[0]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg.config1 = (gConfig[0] | 0x10); /* Set Preamble bit Enable */
ret = R_NOR_FLASH_WriteConfiguration(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
    /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

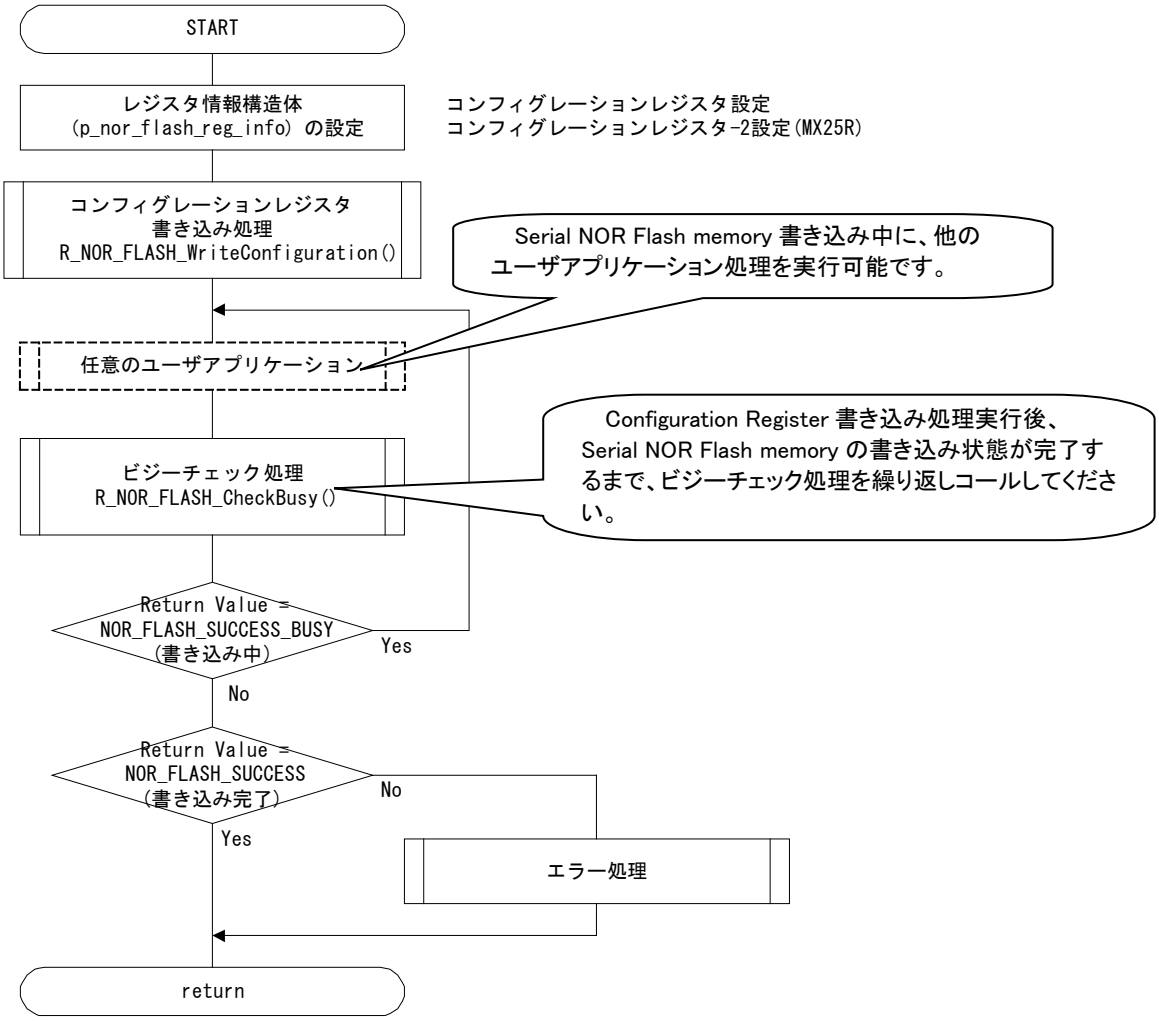


図 3-4　R_NOR_FLASH_WriteConfiguration()の処理例

3.19　R_NOR_FLASH_Set4byteAddressMode()

Serial NOR Flash memory のアドレスモードを 4Byte アドレスモードに設定する際に使用する関数です。Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory 専用の API です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_Set4byteAddressMode(
void
)
```

Parameters

なし

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

< Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory の場合 >

EN4B（0xb7）コマンドを発行し、Configuration Register の 4BYTE bit に 1 を設定します。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Set4byteAddressMode();
```

Special Notes

なし

3.20　R_NOR_FLASH_ReadSecurity()

Serial NOR Flash memory の Security Register を読み出す際に使用する関数です。Macronix International Co., Ltd 社製 MX25L/MX66L/MX25R family serial NOR Flash memory 専用の API です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadSecurity(
uint8_t * p_scur
)
```

Parameters

* p_scur

Security Register 格納バッファ（読み出しバッファとして、1 バイトを設定してください。）

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない

Description

Security Register を読み出し、p_scur に格納します。

Security Register については、使用する Serial NOR Flash memory のデータシートを参照してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t scur = 0;

ret = R_NOR_FLASH_ReadSecurity(&scur);
```

Special Notes

なし

3.21　R_NOR_FLASH_ReadDataSecurityPage()

Serial NOR Flash memory の Security Register からデータを読み出す際に使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadDataSecurityPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

* p_nor_flash_info

Serial NOR Flash memory 情報構造体。

addr

メモリの読み出し開始アドレスを設定してください。

cnt

読み出しバイト数を設定してください。設定可能範囲は 1～256 です。

0 を設定した場合、エラーを返します。

data_cnt

読み出しバイト数（本制御ソフトウェアで使用するため、設定禁止）

*p_data

読み出しデータ格納バッファのアドレスを設定してください。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

Security Register の指定されたアドレスから指定されたバイト数のデータを読み出し、そのデータを p_data に格納します。

Security Register の最終アドレスは、ページサイズ - 1 です。

最終アドレスを超える読み出し指定はできません。最終アドレスの読み出し後、一度処理を完了させて、再度アドレスを設定し直してから、本ユーザ API をコールしてください。

読み出しバイト数 cnt と指定アドレス addr の合計値が最終アドレスを超えないように設定してください。

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_R;
uint8_t buf2[128];

Flash_Info_R.addr = 0x1000;/* Security Register 1 Address */
Flash_Info_R.cnt = 32;
Flash_Info_R.p_data = &buf2[0];
ret = R_NOR_FLASH_ReadDataSecurityPage(&Flash_Info_R);
```

Special Notes

なし

3.22　R_NOR_FLASH_WriteDataSecurityPage()

Serial NOR Flash memory の Security Register ページに 1 ページ単位でデータを書き込む際に使用する関数です。Renesas Electronics 社製 AT25QF/AT25SF family Serial NOR Flash memory 用の専用 API 関数です。

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDataSecurityPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

* p_nor_flash_info

Serial NOR Flash memory 情報構造体。

addr

メモリの書き込み開始アドレスを設定してください。

cnt

書き込みバイト数を設定してください。設定可能範囲は 1～65,535 です。

0 を設定した場合、エラーを返します。

data_cnt

書き込みバイト数（本制御ソフトウェアで使用するため、設定禁止）

*p_data

書き込みデータ格納バッファのアドレスを設定してください。

Return Values

NOR_FLASH_SUCCESS	正常終了
NOR_FLASH_ERR_PARAM	パラメータ異常
NOR_FLASH_ERR_HARD	ハードウェアエラー
NOR_FLASH_ERR_NOT_OPEN	本モジュールの R_NOR_FLASH_Open 関数が実行されていない
NOR_FLASH_ERR_WEL_CHK	Write Enable チェックエラー
NOR_FLASH_ERR_NON_SUPPORTED_API	未サポート API

Description

指定されたアドレスから開始して、p_data 内の指定されたバイト数 (最大サイズ 1 ページまで) のデータを Security Register ページに書き込みます。

大容量のデータ書き込みの際、Page 単位に通信を分割するため、通信中に他の処理ができなくなることを防ぐことができます。

Security Register ページへの書き込みは、ロックされていない場合にのみ可能です。

書き込みバイト数 (cnt) に設定できる最大値は、Security Register のページサイズの容量です。

書き込みバイト数 cnt と指定アドレス addr の合計値が Security Register の最終アドレスを超えないように設定してください。

Example

```
#define FLASH_PP_BUSY_WAIT (uint32_t)(3) /* 3 * 1ms = 3ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_W;
uint8_t buf1[128];
uint32_t loop_cnt = 0;

Flash_Info_W.addr = 0;
Flash_Info_W.cnt = 128;
Flash_Info_W.p_data = &buf1[0];

do
{
    ret = R_NOR_FLASH_WriteDataSecurityPage(&Flash_Info_W);
    if (NOR_FLASH_SUCCESS > ret)
    {
        /* Error */
    }

    loop_cnt = FLASH_PP_BUSY_WAIT;
    mode = NOR_FLASH_MODE_PROG_BUSY;
    do
    {
        /* FLASH is busy.
        User application can perform other processing while flash is busy. */

        ret = R_NOR_FLASH_CheckBusy(mode);
        if (NOR_FLASH_SUCCESS_BUSY != ret)
        {
            /* FLASH is ready or error. */
            break;
        }
        loop_cnt--;
        wait_timer(0, 1); /* 1ms */
    }
    while (0 != loop_cnt);
}
while (0 != Flash_Info_W.cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

なし

3.23　R_NOR_FLASH_GetVersion()

Serial NOR Flash memory 制御ソフトウェアのバージョン情報を取得する際に使用する関数です。

Format

uint32_t R_NOR_FLASH_GetVersion(
void
)

Parameters

なし

Return Values

上位 2 バイト	メジャーバージョン（10 進表示）
下位 2 バイト	マイナーバージョン（10 進表示）

Description

バージョン情報を返します。

Example

```
uint32_t version;
version = R_NOR_FLASH_GetVersion();
```

Special Notes

なし

3.24　R_NOR_FLASH_Interval()

DTC 転送時にデータ送信が終わらない状態などになってしまった場合に、タイムアウトを検出するために使用する関数です。

DTC を使用する場合、タイマを使用して 1ms 毎に本関数をコールしてください。

Format

```
void R_NOR_FLASH_Interval(
void
)
```

Parameters

なし

Return Values

なし

Description

DTC 転送完了待ち時に内部タイマカウンタをインクリメントします。

Example

```
static void __near r_Config_TAU0_1_higher8bits_interrupt(void)
{
    /* Start user code for r Config TAU0 1 higher8bits interrupt. Do not edit comment generated here
*/
    R_NOR_FLASH_Interval();
    /* End user code. Do not edit comment generated here */
}
```

Special Notes

タイマ等を使用して本関数を 1ms 毎にコールしてください。

上記 Example は、1ms 毎に発生する割り込み処理で本関数をコールする例です。

他の API のように main からコールする API ではありません。

3.25　R_NOR_FLASH_SendendNotification()

SPI 通信の送信完了通知処理です。SPI(CSI)通信の送信完了コールバック関数から本関数をコールしてください。

Format

void R_NOR_FLASH_SendendNotification(
void
)

Parameters

なし

Return Values

なし

Description

SPI 通信の通信中フラグをクリアします。

Example

```
/* Start user code for include. Do not edit comment generated here */
#include "r_nor_flash_rl78_if.h"
/* End user code. Do not edit comment generated here */


void r_Config_CSI11_callback_sendend(void)
{
    /* Start user code for r_Config_CSI11_callback_sendend. Do not edit comment generated here */
    R_NOR_FLASH_SendendNotification();
    /* End user code. Do not edit comment generated here */
}
```

Special Notes

本モジュールに使用する SPI(CSI)通信の送信完了コールバック関数から本関数をコールしてください。
上記 Example は、生成した SPI(CSI)通信の送信完了コールバック関数で本関数をコールする例です。
他の API のように main からコールする API ではありません。

4.　端子設定

本 SIS モジュールは CS 端子を使用します。

CS 端子は Open 関数と Close 関数で下表のように変化します。

表 4-1 Serial NOR Flash memory を制御するための端子設定

関数名	チップセレクト端子（注１）
R_NOR_FLASH_Open()前	ユーザ設定に依存
R_NOR_FLASH_Open()後	H 出力状態
R_NOR_FLASH_Close()後	入力状態

注１　チップセレクト端子は、外付け抵抗でプルアップ処理してください。

5.　本モジュール追加後の各種設定

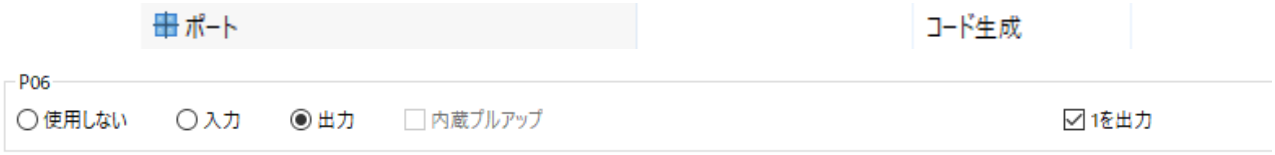
本モジュールをプロジェクトに追加後、使用する環境に合わせて各コンポーネントの追加と設定、コンフィグレーションオプションを設定する必要があります。

(1)　wait に使用する関数の有効化

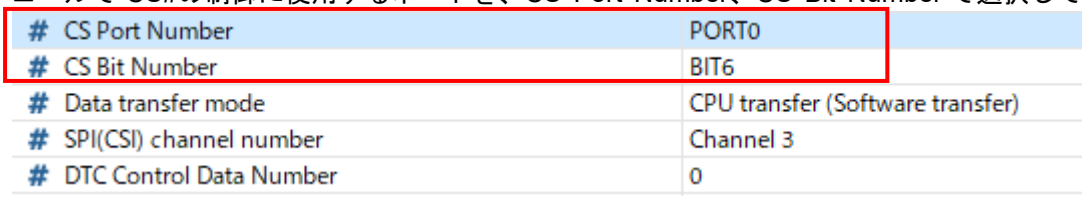
本モジュールで wait に使用する R_BSP_SoftwareDelay を Enable にしてください。

(2)　CS#に使用するポート

コード生成のポートを追加して、本モジュールで CS#の制御に使用するポートを H 出力に設定してください。

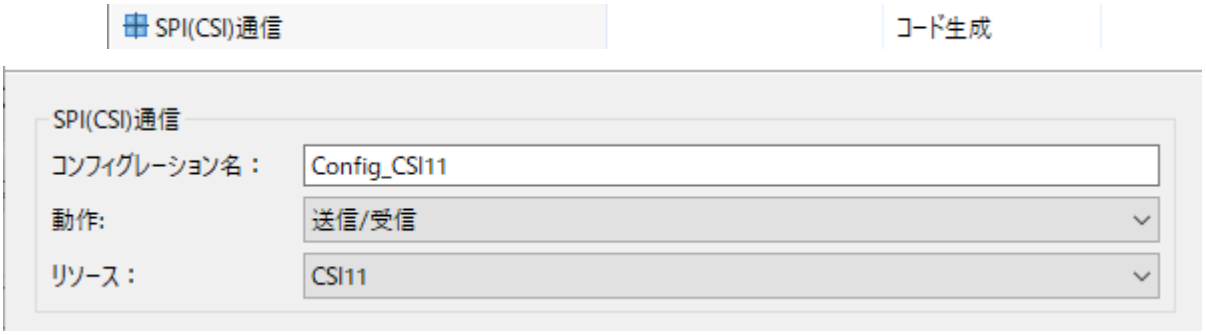


本モジュールで CS#の制御に使用するポートを、CS Port Number、CS Bit Number で選択してください。



(3)　SPI 通信に使用するチャネル

コード生成の SPI(CSI)通信で、SPI 通信に使用するチャネルを追加してください。



生成した SPI(CSI)通信コードの送信完了 callback 内で本モジュールの下記 callback 関数を呼び出してください。

void R_NOR_FLASH_SendendNotification(void);

Example

```
/* Start user code for include. Do not edit comment generated here */
#include "r_nor_flash_rl78_if.h"
/* End user code. Do not edit comment generated here */


void r_Config_CSI11_callback_sendend(void)
{
    /* Start user code for r_Config_CSI11_callback_sendend. Do not edit comment generated here */
    R_NOR_FLASH_SendendNotification();
    /* End user code. Do not edit comment generated here */
}
```

本モジュールで SPI 通信に使用するチャネルを SPI(CSI) channel number で選択してください。

(4)　DTC 転送を使用する場合

コード生成のデータ・トランスファ・コントローラを追加して、SPI 通信に使用するチャネルに合わせて設定してください。

Data transfer mode で DTC transfer を選択してください。

# CS Port Number	PORT0
# CS Bit Number	BIT6
# Data transfer mode	DTC transfer
# SPI(CSI) channel number	Channel 3
# DTC Control Data Number	0

使用するコントロール・データ番号を DTC Control Data Number に入力してください。

# CS Port Number	PORT0
# CS Bit Number	BIT6
# Data transfer mode	DTC transfer
# SPI(CSI) channel number	Channel 3
# DTC Control Data Number	0

6.　デモプロジェクト

6.1　概要

本アプリケーションノートには、Serial NOR Flash memory 制御モジュールの使用方法を説明するためのデモプロジェクトを同梱しています。

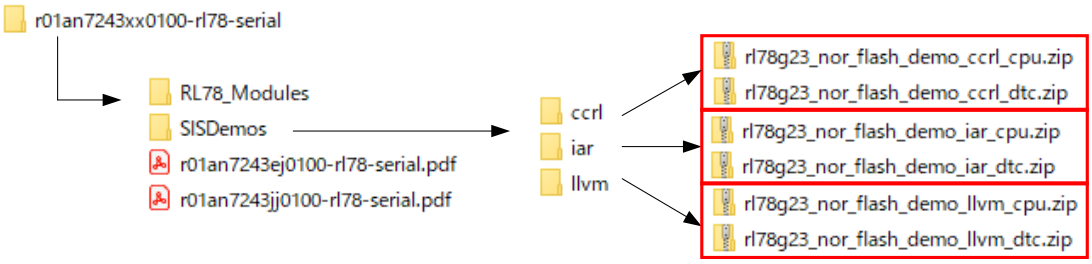


表 6-1 デモプロジェクト一覧

コンパイラ	デモプロジェクト名	転送方法
Renesas Compiler [CC-RL]	rl78g23_nor_flash_demo_ccrl_cpu	CPU 転送
	rl78g23_nor_flash_demo_ccrl_dtc	DTC 転送
LLVM	rl78g23_nor_flash_demo_llvm_cpu	CPU 転送
	rl78g23_nor_flash_demo_llvm_dtc	DTC 転送
IAR	rl78g23_nor_flash_demo_iar_cpu	CPU 転送
	rl78g23_nor_flash_demo_iar_dtc	DTC 転送

デモプロジェクトは、以下の処理を順次行います。

- 本モジュールの OPEN 処理、Serial NOR Flash memory の busy チェック
- ライトプロテクトのクリア
- Serial NOR Flash memory の消去、書き込み、読み出し処理
- Serial NOR Flash memory のレジスタ制御
- Serial NOR Flash memory 情報の取得
- 本モジュールのバージョン情報取得
- 本モジュールの CLOSE 処理

6.2　動作確認環境

表 6-2 にデモプロジェクトを動作させるためのハードウェアや設定などの条件を示します。
図 6-1 に評価ボードと Serial NOR Flash memory などハードウェアの構成を示します。
図 6-2 に RL78/G23 の端子と Serial NOR Flash memory の端子の結線を示します。

表 6-2 デモプロジェクトの動作条件

項目	説明
MCU	R7F100GLGxFB（RL78/G23）
動作周波数	32MHz
統合開発環境	ルネサス エレクトロニクス e^2 studio 2024-01 IAR Embedded Workbench for Renesas RL78 5.10.3
C コンパイラ	ルネサス エレクトロニクス RL78 Family C Compiler Package V1.13.00 LLVM for Renesas RL78 10.0.0.202312 IAR C/C++ Compiler for Renesas RL78 version 5.10.3
デモプロジェクト	Version 1.00
評価ボード	RL78/G23-64p Fast Prototyping Board (型名: RTK7RLG230CLG000BJ)
動作電圧	3.3V(USB 経由で供給)
Emulator	COM Port デバッグ
Serial NOR Flash memory	AT25QF641B

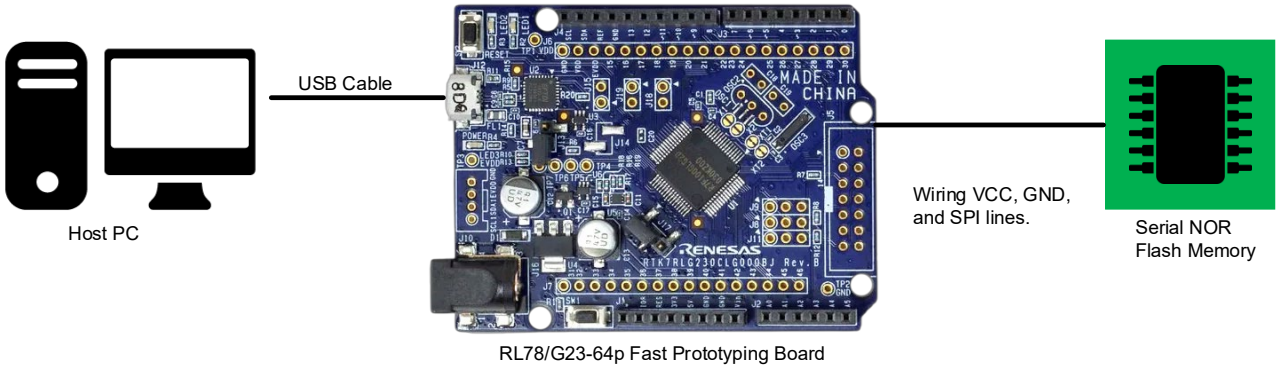


図 6-1 デモプロジェクトの動作環境

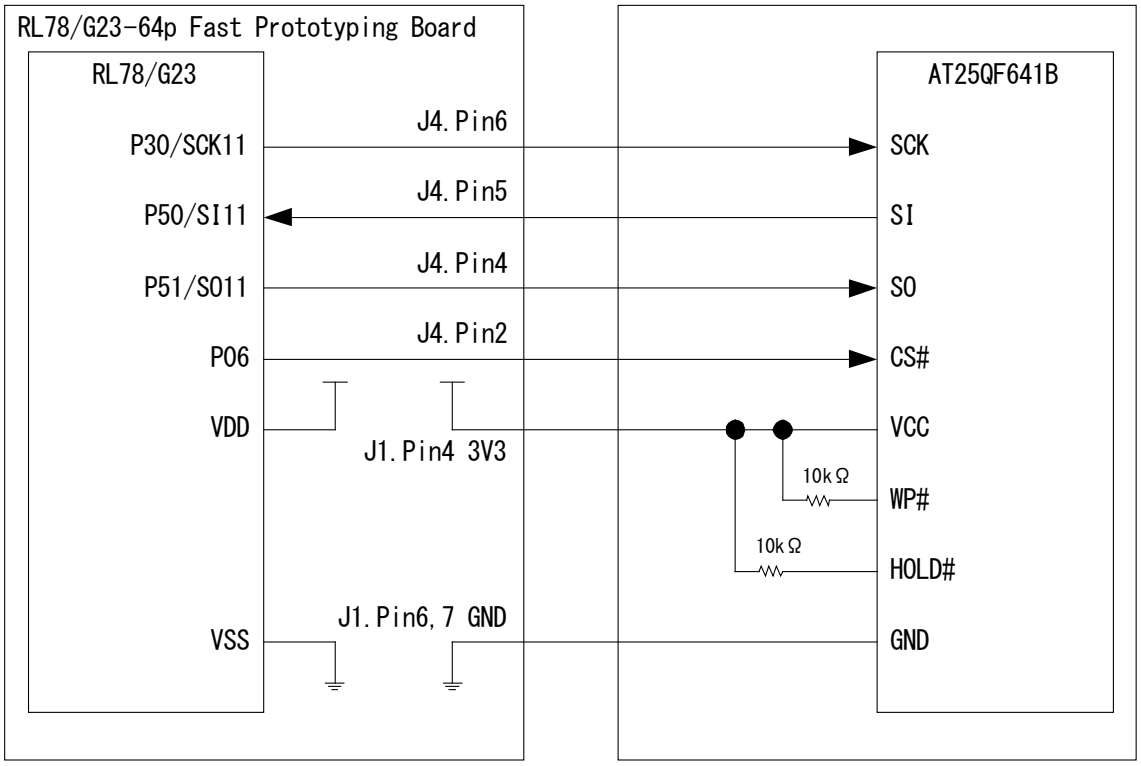


図 6-2　RL78/G23-64p Fast Prototyping Board と Serial NOR Flash memory の端子結線図

6.3　ファイル構成

デモプロジェクトのソースファイルを表 6-3 に示します。

表 6-3　デモプロジェクトのファイル一覧

ファイル名	説明
rl78g23_nor_flash_demo.c	デモプロジェクトの main 関数を含む C ソースファイル。

6.4　関数

デモプロジェクトは、main 関数と内部関数で構成されます。これら関数は rl78g23_nor_flash_demo.c に記述されています。

表 6-4　関数一覧

関数名	説明
main()	デモプロジェクトの main 関数。 ・以下 7 関数を呼び出します。
demo_nor_flash_open()	本モジュールの OPEN、Serial NOR Flash memory の busy チェックを行います。
demo_nor_flash_set_write_protect()	ライトプロテクトのクリアを行います。
demo_nor_flash_erase_write_read()	Serial NOR Flash memory の消去、書き込み、読み出し、結果確認を行います。
demo_nor_flash_status()	Serial NOR Flash memory のレジスタ制御を行います。
demo_nor_flash_getmemoryinfo	Serial NOR Flash memory の情報取得を行います。
demo_nor_flash_getversion	本モジュールのバージョン情報取得を行います。
demo_nor_flash_close	Serial NOR Flash memory の busy チェック、本モジュールの CLOSE 処理を行います。

(1)　demo_nor_flash_open()

Serial NOR Flash Memory 制御モジュールの R_NOR_FLASH_Open 関数実行後、Serial NOR Flash memory の busy チェックを行う関数です。

Format

```
void demo_nor_flash_open(
void
)
```

Parameters

なし

Return Values

なし

Description

・R_NOR_FLASH_Open を実行します。

・busy チェックを行います。

Special Notes

なし

(2)　demo_nor_flash_set_write_protect()

Serial NOR Flash memory にライトプロテクトのクリア処理を行う関数です。

Format

```
void demo_nor_flash_set_write_protect(
void
)
```

Parameters

なし

Return Values

なし

Description

・R_NOR_FLASH_SetWriteProtect(0)を実行します。

・R_NOR_FLASH_ReadStatus を実行して Status Register を確認します。

Special Notes

なし

(3)　demo_nor_flash_erase_write_read()

Serial NOR Flash memory の消去、書き込み、読み出し、結果の確認を行う関数です。

Format

```
void demo_nor_flash_erase_write_read(
void
)
```

Parameters

なし

Return Values

なし

Description

・Block Erase (4KB)を実行します。
・Serial NOR Flash memory に 256 バイトの書き込みを行います。
・Serial NOR Flash memory から 256 バイトの読み出しを行います。
・結果を確認します。
・消去、1 バイトから 511 バイトまで書き込み、読み出し、結果確認を順次実行します。
・Chip Erase を実行します。
・Block Erase (32KB)を実行します。
・Block Erase (64KB)を実行します。

Special Notes

なし

(4)　demo_nor_flash_status()

Serial NOR Flash memory のレジスタ制御を行う関数です。

Format

```
void demo_nor_flash_status(
void
)
```

Parameters

なし

Return Values

なし

Description

・Status Register 1 の SEC ビットをセットし、結果を確認します。
・Status Register 1 の SEC ビットをクリアし、結果を確認します。
・Status Register 2 の CMP ビットをセットし、結果を確認します。
・Status Register 2 の CMP ビットをクリアし、結果を確認します。
・R_NOR_FLASH_ReadId を実行します。
・R_NOR_FLASH_WriteDisable を実行します。

Special Notes

なし

(5)　demo_nor_flash_getmemoryinfo()

Serial NOR Flash memory の情報取得を行う関数です。

Format

```
void demo_nor_flash_getmemoryinfo(
void
)
```

Parameters

なし

Return Values

なし

Description

・R_NOR_FLASH_GetMemoryInfo を実行します。

Special Notes

なし

(6)　demo_nor_flash_getversion()

Serial NOR Flash Memory 制御モジュールのバージョン情報取得を行う関数です。

Format

```
void demo_nor_flash_getversion(
void
)
```

Parameters

なし

Return Values

なし

Description

・R_NOR_FLASH_GetVersion を実行します。

Special Notes

なし

(7)　demo_nor_flash_close()

Serial NOR Flash Memory 制御モジュールの CLOSE を実行する関数です。

Format

```
void demo_nor_flash_close(
void
)
```

Parameters

なし

Return Values

なし

Description

・R_NOR_FLASH_Close を実行します。

Special Notes

なし

(8)　busy_check()

Serial NOR Flash memory の busy チェックを実行する関数です。

Format

```
e_nor_flash_status_t busy_check(
e_nor_flash_check_busy_t mode,
uint32_t loop_cnt
)
```

Parameters

mode

R_NOR_FLASH_CheckBusy に与える mode

loop_cnt

busy チェックを実行する時間（ms）

Return Values

なし

Description

・R_NOR_FLASH_CheckBusy を実行します。

・指定時間の busy チェックを行うか、Serial NOR Flash memory が busy でなくなれば処理を終了します。

Special Notes

なし

6.5　プロジェクトをインポートする方法

デモプロジェクトは e² studio のプロジェクト形式で提供しています。本章では、 e² studio へプロジェクトをインポートする方法を示します。インポート完了後、ビルドおよびデバッグの設定を確認してください。

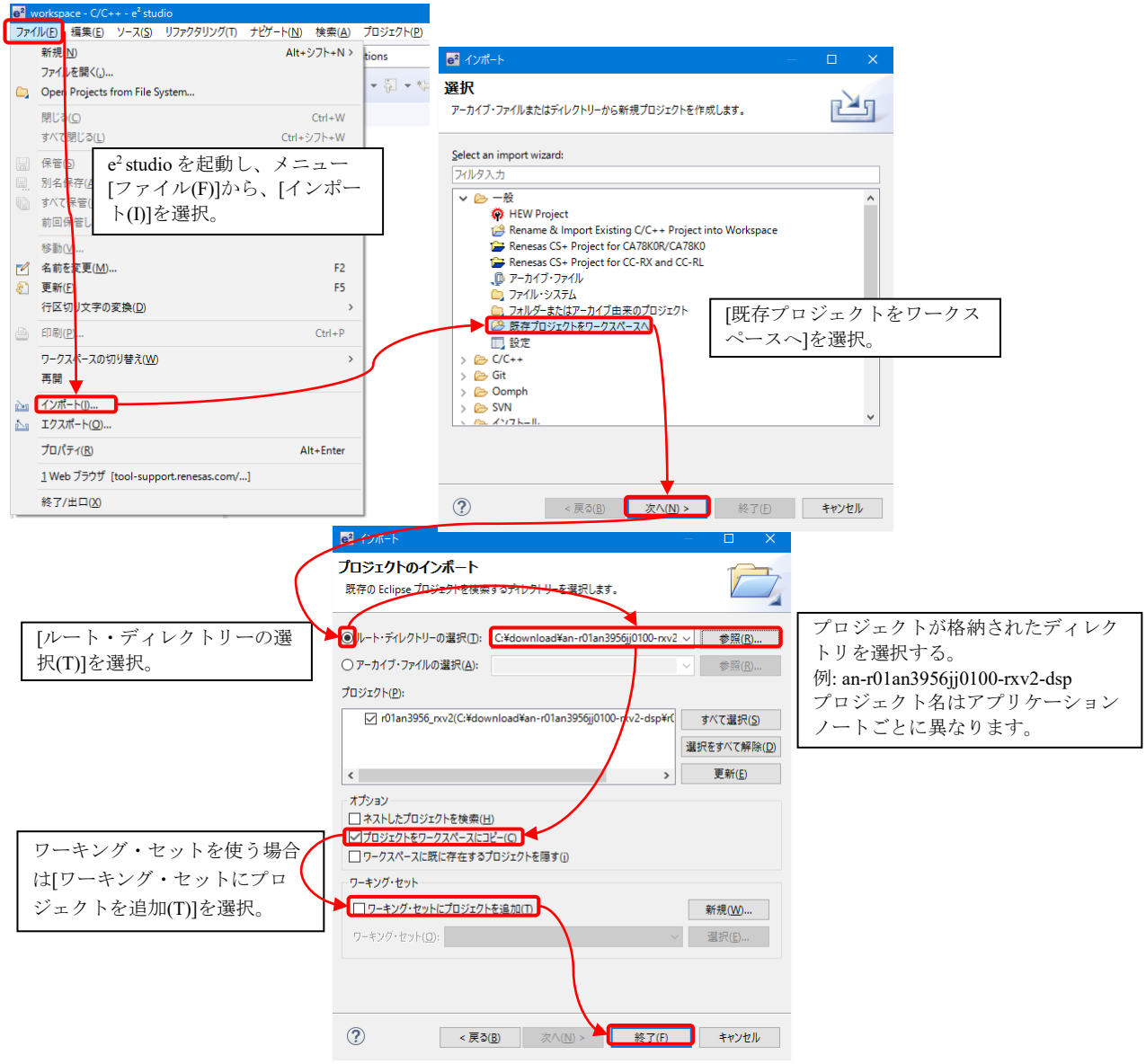


図 6-3　プロジェクトを e² studio にインポートする方法

7.　付録

7.1　動作確認環境

本モジュールの動作確認環境を以下に示します。

表 7-1　動作確認環境(Rev. 1.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V2024-01 IAR Embedded Workbench for Renesas RL78 5.10.3
C コンパイラ	ルネサスエレクトロニクス製 RL78 Family C Compiler Package V1.13.00 コンパイルオプション：統合開発環境のデフォルト設定
	LLVM for Renesas RL78 10.0.0.202312 コンパイルオプション：統合開発環境のデフォルト設定
	IAR C/C++ Compiler for Renesas RL78 version 5.10.3 コンパイルオプション：統合開発環境のデフォルト設定
モジュールのバージョン	Ver.1.00
使用ボード	RL78/G23-64p Fast Prototyping Board (型名: RTK7RLG230CLG000BJ)

表 7-2　動作確認環境(Rev. 1.01)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V2025-04.1 IAR Embedded Workbench for Renesas RL78 5.20.1
C コンパイラ	ルネサスエレクトロニクス製 RL78 Family C Compiler Package V1.15.00 コンパイルオプション：統合開発環境のデフォルト設定
	LLVM for Renesas RL78 17.0.0.202506 コンパイルオプション：統合開発環境のデフォルト設定
	IAR C/C++ Compiler for Renesas RL78 version 5.20.1 コンパイルオプション：統合開発環境のデフォルト設定
モジュールのバージョン	Ver.1.01
使用ボード	RL78/L23 Fast Prototyping Board (型名: RTK7RLL230S00001BJ)

7.2　制御可能な Serial NOR Flash memory 製品

本モジュールが制御可能な Serial NOR Flash memory 製品を以下に示します。

表 7-2　制御可能な Serial NOR Flash memory 製品

シリーズ	型番	メモリ
AT25SF	AT25SF041B	4Mbit
	AT25SF081B	8Mbit
	AT25SF161B	16Mbit
	AT25SF321B	32Mbit
	AT25SF641B	64Mbit
AT25QF	AT25QF641B	64Mbit
MX25L	MX25L3233F	32Mbit
	MX25L6433F	64Mbit
	MX25L12833F	128Mbit
	MX25L25645G	256Mbit
	MX25L51245	512Mbit
MX66L	MX66L1G45	1Gbit
MX25R	MX25R6435F	64Mbit

8. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境
RL78 ファミリ CC-RL コンパイラ ユーザーズマニュアル（R20UT3123）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2024.3.22	-	初版発行。
1.01	2025.7.1	-	RL78/L23 グループのサポートを追加。
		76	表 7-2 動作確認環境(Rev1.01)を追加

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。