RX ファミリ

SDHI モジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)を使用した SDHI 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 RX ファミリ MCU 内蔵 SD ホストインタフェース（SDHI）を制御するデバイスドライバです。以降、本モジュールを SDHI FIT モジュールと称します。

対象デバイス

RX231 グループ、RX23W グループ

RX64M グループ、RX65N グループ、RX651 グループ、RX66N グループ、RX671 グループ

RX71M グループ、RX72M グループ、RX72N グループ

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 C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については「6.1 動作確認環境」を参照してください。

関連ドキュメント

- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- SD モード SD メモリカードドライバ Firmware Integration Technology (R01AN4233)
- SD モード SDIO ドライバ・ソフトウェア RTM0RX0000DSDD2 d ブロード社製 eWBC 用 (R01UW0133)
- SD モード SDIO ドライバ・ソフトウェア RTM0RX0000DSDD3 村田製作所社製 TypeZX 用 (R01UW0134)
- RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology (R01AN2063)
- RX Family DTC モジュール Firmware Integration Technology (R01AN1819)
- RX ファミリ コンペアマッチタイマ（CMT）モジュール Firmware Integration Technology (R01AN1856)
- RX ファミリ ロングワード型キューバッファ（LONGQ）モジュール Firmware Integration Technology (R01AN1889)

目次

1. 概要

1.1 SDHI FIT モジュールとは

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、「2.12 FIT モジュールの追加方法」を参照してください。

1.2 SDHI FIT モジュールの概要

本モジュールが提供する API を組み合わせて使用することで、SD メモリカードや SDIO を SD モードで制御することができます。

なお、本モジュールの API を組み合わせて動作する SD メモリカードドライバや SDIO ドライバを別途提供しています。必要な場合は以下よりお問合せください。

RX ファミリ用 SD カードドライバ : https://www.renesas.com/driver/rtm0rx0000dsdd

1.2.1 機能概要

以下に機能を示します。

表 1-1 機能一覧

項目	機能
クロック供給	SDHI クロック供給／停止設定をサポート
SD バス	SD モード（1 ビット／4 ビット）設定をサポート
割り込み制御	SDHI で使用する割り込みの許可／禁止設定をサポート SDHI で使用する割り込みフラグのクリアをサポート
コールバック関数	以下の割り込み発生時、コールバック関数呼び出しをサポート ● カードアクセス割り込み（CACI） ● SDIO アクセス割り込み（SDACI） ● カード検出割り込み（CDETI） ● SD バッファアクセス割り込み（SBFAI）
SDHI レジスタ設定／取得	SDHI レジスタ設定／取得をサポート
エンディアン	リトルエンディアン／ビッグエンディアンでの動作をサポート
その他の機能	Firmware Integration Technology（FIT）に対応

1.3　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_SDHI_Open	本モジュールのオープン処理
R_SDHI_Close	本モジュールのクローズ処理
R_SDHI_IntHandler0	割り込みハンドラ
R_SDHI_IntCallback	カードアクセス割り込み（CACI）およびカード検出割り込み（CDETI）コールバック関数登録処理
R_SDHI_IntSDBuffCallBack	SD バッファアクセス割り込み（SBFAI）コールバック関数登録処理
R_SDHI_IntSdioCallback	SDIO アクセス割り込み（SDACI）コールバック関数登録処理
R_SDHI_EnableIcuInt	SDHI 用 ICU コントローラ割り込み有効処理
R_SDHI_DisableIcuInt	SDHI 用 ICU コントローラ割り込み無効処理
R_SDHI_SetIntMask	SD 割り込み有効処理
R_SDHI_ClearIntMask	SD 割り込み無効処理
R_SDHI_ClearSdstsReg	SD ステータスレジスタクリア処理
R_SDHI_SetSdioIntMask	SDIO 割り込みマスクレジスタセット処理
R_SDHI_ClearSdioIntMask	SDIO 割り込みマスクレジスタクリア処理
R_SDHI_ClearSdiostsReg	SDIO ステータスレジスタクリア処理
R_SDHI_SetClock	SD クロック供給／停止処理
R_SDHI_SetBus	SD バス設定処理
R_SDHI_GetResp	コマンドレスポンス取得処理
R_SDHI_OutReg	SDHI レジスタ設定処理
R_SDHI_InReg	SDHI レジスタ取得処理
R_SDHI_CDLayout	SDHI_CD 端子使用有無確認処理
R_SDHI_WPLayout	SDHI_WP 端子使用有無確認処理
R_SDHI_GetWP	SDHI_WP 端子状態取得処理
R_SDHI_GetSpeedType	スピードモード取得処理
R_SDHI_GetBuffRegAddress	SD バッファレジスタのアドレス取得処理
R_SDHI_GetVersion	本モジュールのバージョン情報取得処理
R_SDHI_SetLogHdlAddress	LONGQ モジュールのハンドラアドレス設定処理　注 1
R_SDHI_Log	エラーログ取得処理　注 1

注 1 ：別途 LONGQ FIT モジュールが必要です。

1.4　処理例

1.4.1　アプリケーション構成例

上位層に SD メモリカードドライバと FAT ファイルシステムを構築する場合のアプリケーション構成図を図 1-1 に示します。

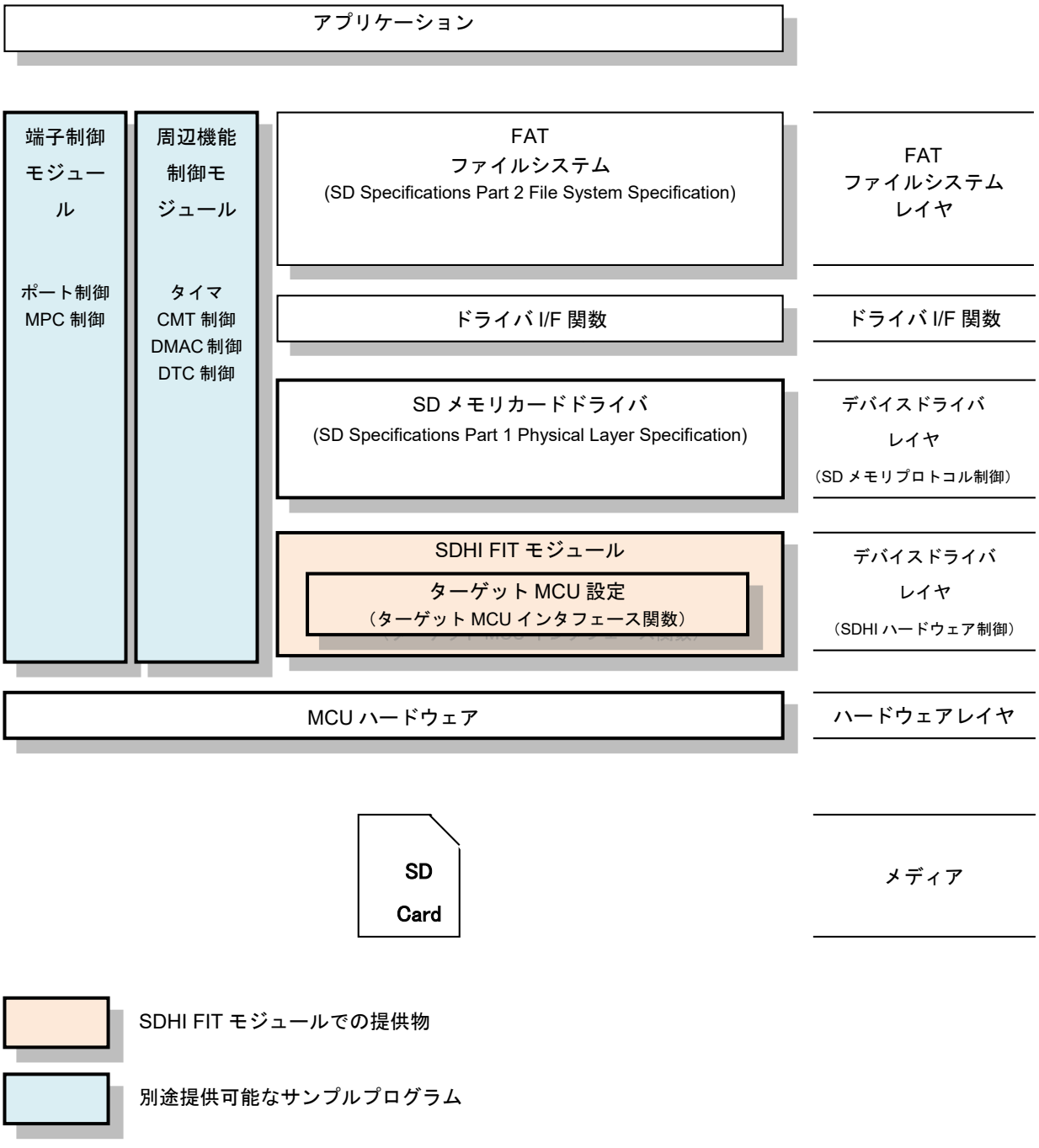


図 1-1　アプリケーション構成図

(1) FAT ファイルシステム

SD メモリをファイル管理する場合に使用するソフトウェアです。別途 FAT ファイルシステムが必要です。必要に応じて以下から入手してください。

オープンソース FAT ファイルシステム M3S-TFAT-Tiny：
https://www.renesas.com/software-tool/fat-file-system-m3s-tfat-tiny-rx-family

(2) ドライバ I/F 関数

ルネサス エレクトロニクス製 FAT ファイルシステム API と SD メモリカードドライバ API を接続するレイヤのソフトウェアです。必要に応じて、上記 M3S-TFAT-Tiny の Web ページから入手してください。

RX ファミリ M3S-TFAT-Tiny メモリドライバインタフェースモジュール Firmware Integration Technology

(3) SD メモリカードドライバ

SD Specifications Part 1 Physical Layer Specification の SD メモリプロトコル制御を行うソフトウェアです。

本モジュールの API を組み合わせて動作する SD メモリカードドライバや SDIO ドライバを別途提供しています。必要な場合は以下よりお問合せください。

RX ファミリ用 SD カードドライバ：https://www.renesas.com/driver/rtm0rx0000dsdd

(4) SDHI FIT モジュール

本モジュールです。また、MCU に依存するターゲット MCU インタフェース関数および割り込み設定ファイルが含まれます。

(5) 周辺機能制御モジュール（サンプルプログラム）

タイマ制御、DMAC 制御、DTC 制御を行うソフトウェアです。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。

(6) 端子制御モジュール（サンプルプログラム）

SDHI 制御のための端子制御用ソフトウェアです。使用する MCU リソースは、ポート制御（SDHI 機能制御と SD カード電源用ポート制御）、MPC 制御（SDHI 機能制御）です。

端子割り当てについては、使用端子が競合しないように、システムで一括して端子割り当てすることを推奨します。

2. API 情報

本 FIT 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。

- SDHI

2.2 ソフトウェアの要求

SDHI FIT モジュールは以下の FIT モジュールに依存しています。

- ボードサポートパッケージ (r_bsp) Rev.5.20 以上

2.3 サポートされているツールチェーン

本 FIT モジュールは「5.1 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

マクロ定義 SDHI_CFG_MODE_INT が SDHI_MODE_HWINT の時、SDHI 割り込みが有効になります。表 2-1 に本 FIT モジュールが使用する割り込みベクタを示します。

表 2-1 使用する割り込みベクター覧

デバイス	割り込みベクタ
RX64M RX65N RX66N RX671 RX71M RX72M RX72N	SD バッファアクセス割り込み（SBFAI）（ベクタ番号：44） GROUPBL1 割り込み（ベクタ番号: 111） ● カード検出割り込み（CDETI）（グループ割り込み要因番号：3） ● カードアクセス割り込み（CACI）（グループ割り込み要因番号：4） ● SDIO アクセス割り込み（SDACI）（グループ割り込み要因番号：5）
RX231 RX23W	SD バッファアクセス割り込み（SBFAI）（ベクタ番号：40） ● カード検出割り込み（CDETI）（ベクタ番号：41） ● カードアクセス割り込み（CACI）（ベクタ番号：42） ● SDIO アクセス割り込み（SDACI）（ベクタ番号：43）

2.5 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_sdhi_rx_if.h に記載しています。

2.6 整数型

SDHI FIT モジュールは 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ギュレーションオプションの設定は、r_sdhi_rx_config.h で行います。
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_sdhi_rx_config.h*	
#define SDHI_CFG_CHx_INCLUDED　(1) ※チャネル 0 のデフォルト値は“1（有効）” ※“x”はチャネル番号	該当チャネルを使用するかを選択してください。 無効にした場合、該当チャネルに関する処理をコードから省略します。 有効にした場合、該当チャネルに関する処理をコードに含めます。 複数チャネルをサポートする MCU の場合、多チャネルの定義を追加する必要があります。
#define SDHI_CFG_CHx_CD_ACTIVE　(1) ※デフォルト値は“1（有効）” ※“x”はチャネル番号	SDHI_CD 端子の割り当てが不要な場合、個別に SDHI FIT モジュールの制御対象から外すことができます。 端子を SDHI 機能に割り当てて制御する場合、(1)に設定してください。 制御対象から外す場合、(0)に設定してください。 制御対象から外した場合、その端子を他用途（汎用入出力ポート等）に利用できます。SDHI FIT モジュールは機能的に制御を外すものであり、使用端子設定機能を持っていません。そのため、他用途で利用する場合、別途、使用端子も併せて設定してください。使用チャネル毎に設定が必要です。
#define SDHI_CFG_CHx_WP_ACTIVE　(1) ※デフォルト値は“1（有効）” ※“x”はチャネル番号	SDHI_WP 端子の割り当てが不要な場合、個別に SDHI FIT モジュールの制御対象から外すことができます。 端子を SDHI 機能に割り当てて制御する場合、(1)に設定してください。 制御対象から外す場合、(0)に設定してください。 制御対象から外した場合、その端子を他用途（汎用入出力ポート等）に利用できます。SDHI FIT モジュールは機能的に制御を外すものであり、使用端子設定機能を持っていません。そのため、他用途で利用する場合、別途、使用端子も併せて設定してください。使用チャネル毎に設定が必要です。
#define SDHI_CFG_CHx_INT_LEVEL　(10)　/* SDHI channel x interrupt level */ ※デフォルト値は“(10)”	カード検出割り込み（CDETI）、カードアクセス割り込み（CACI）、SDIO アクセス割り込み（SDACI）のレベルを設定してください。
#define SDHI_CFG_CHx_INT_LEVEL_DMADTC (10)　/* SDHI channel x DMA/DTC interrupt level */ ※デフォルト値は“(10)”	SD バッファアクセス割り込み（SBFAI）レベルを設定してください。DMAC／DTC を使用して、SD バッファにデータを書き込む場合、または、SDHI バッファからデータを読み出す場合の割り込みレベルです。
#define SDHI_CFG_DIV_HIGH_SPEED SDHI_DIV_2 ※デフォルト値は“SDHI_DIV_2”（2 分周）　注 1	High-Speed mode のクロック周波数定義です。 SDHI クロック周波数選択ビット（CLKSEL[7:0]）に PCLKB の分周比を設定してください。設定値はユーザーズマニュアル ハードウェア編を参照し、SDHI_DIV_1～SDHI_DIV_512（1 分周～512 分周）としてください。 例えば、PCLKB=60MHz、High-Speed mode のクロック周波数=30MHz の場合、SDHI_DIV_2（2 分周）を設定してください。

#define SDHI_CFG_DIV_DEFAULT_SPEED SDHI_DIV_4 ※デフォルト値は“SDHI_DIV_4”（4 分周）　注 1	Default Speed mode のクロック周波数定義です。設定方法は、上記 High-Speed mode と同様です。 例えば、PCLKB=60MHz、Default Speed mode のクロック周波数=15MHz の場合、SDHI_DIV_4（4 分周）を設定してください。
#define SDHI_CFG_DIV_INIT_SPEED SDHI_DIV_1024 ※デフォルト値は“SDHI_DIV_256”（256 分周）注 1	カード認識モードのクロック周波数定義です。設定方法は、上記 High-Speed mode と同様です。 例えば、PCLKB=60MHz、カード認識モードのクロック周波数=234KHz の場合、SDHI_DIV_256（256 分周）を設定してください。
#define SDHI_CFG_SDOPT_CTOP　(0x000eul) /* CD time out count */ ※デフォルト値は“0x000eul”	カード検出時のタイムアウト設定です。 カードアクセスオプションレジスタ（SDOPT）のカード検出タイムアウトカウンタビット（CTOP）を bit 3-0 に設定してください。
#define SDHI_CFG_SDOPT_TOP　(0x00e0ul) /* response time out count */ ※デフォルト値は“0x00e0ul”	コマンドのレスポンスタイムアウト設定です。 カードアクセスオプションレジスタ（SDOPT）のタイムアウトカウンタビット（TOP）の SRBSYTO[3:0]のタイムアウト値を bit 7-4 に設定してください。
#define SDHI_CFG_PARAM_CHECKING_ENABLE　(1) ※デフォルト値は“1（有効）”	引数のチェックを有効または無効に設定します。 (0)：無効、(1)：有効
/* #define SDHI_CFG_LONGQ_ENABLE */ ※デフォルト値は“無効”	LONGQ FIT モジュールを使ったエラーログ取得機能を利用する場合に定義してください。 この機能を利用する場合、デバッグ用モジュール（この定義を有効にした作成した専用モジュール）を使用し、かつ LONGQ FIT モジュールを組み込む必要があります。

注 1：SDHI_DIV_n（n は、整数で分周比を示す。）は、SDHI の PCLK の分周比を示します。使用する MCU の電気的特性により、SD Specifications Part 1 Physical Layer Specification の最大転送周波数を設定できない場合があります。設定可能な最大転送周波数は、使用する MCU のユーザーズマニュアル ハードウェア編を参照してください。

2.8　コードサイズ

本モジュールのコードサイズを下表に示します。RX200 シリーズ、RX600 シリーズ、RX700 シリーズから代表して 1 デバイスずつ掲載しています。

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_sdhi_rx rev2.07

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 8.03.00.202002
(統合開発環境のデフォルト設定に” -std=gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.14.1
(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

表 2-2　コードサイズ

ROM、RAM およびスタックのコードサイズ（注１、注２）							
デバイス	分類	使用メモリ					
		Renesas Compiler		GCC		IAR Compiler	
		パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
RX231	ROM	1,793 バイト	1,508 バイト	4,036 バイト	3,388 バイト	3,048 バイト	2,620 バイト
	RAM	28 バイト		28 バイト		32 バイト	
	最大使用ユーザスタック	68 バイト		-		60 バイト	
	最大使用割り込みスタック	48 バイト		-		64 バイト	
RX65N	ROM	1,867 バイト	1,582 バイト	4,132 バイト	3,468 バイト	3,096 バイト	2,695 バイト
	RAM	28 バイト		28 バイト		32 バイト	
	最大使用ユーザスタック	88 バイト		-		68 バイト	
	最大使用割り込みスタック	36 バイト		-		68 バイト	
RX71M	ROM	1,864 バイト	1,579 バイト	4,132 バイト	3,468 バイト	3,104 バイト	2,680 バイト
	RAM	28 バイト		28 バイト		32 バイト	
	最大使用ユーザスタック	88 バイト		-		68 バイト	
	最大使用割り込みスタック	36 バイト		-		68 バイト	

注１：必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。
注２：リトルエンディアン時の値です。エンディアンにより、上記のメモリサイズは、異なります。

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_sdhi_rx_if.h に記載されています。

2.10 戻り値

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_sdhi_rx_if.h で記載されています。

2.11 コールバック関数

本モジュールでは、カード検出割り込み（CDETI）、カードアクセス割り込み（CACI）、SDIO アクセス割り込み（SDACI）が発生したタイミングで、ユーザが設定したコールバック関数を呼び出します。

コールバック関数の登録方法は「R_SDHI_IntCallback()」「R_SDHI_IntSDBuffCallback()」「R_SDHI_IntSdioCallback()」を参照してください。

2.12 FIT 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。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e^2 studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 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 FIT 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「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio の FIT 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 FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」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 FIT 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 FIT 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「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT 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「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT 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 FIT 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「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13 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等で 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は、「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。

```
while 文の例：
/* WAIT LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for 文の例：
/* Initialize reference counters to 0. */
/* WAIT LOOP */
for (i = 0; i < BSP REG PROTECT TOTAL ITEMS; i++)
{
    g protect counters[i] = 0;
}

do while 文の例：
/* Reset completion waiting */
do
{
    reg = phy read(ether channel, PHY REG CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API 関数

R_SDHI_Open()

SDHI FIT モジュールの API を使用する際、最初に使用する関数です。

Format

```
sdhi_status_t R_SDHI_Open(
    uint32_t channel,
)
```

Parameters

channel

チャネル番号	使用する SDHI チャネル番号（0 起算）

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

引数 channel で設定した SDHI チャネルリソースを取得し、SDHI FIT モジュールと SDHI チャネルを初期化します。また、その SDHI チャネルリソースを占有します。

Example

```
/* ==== Please add the processing to set the pins. ==== */

if (R_SDHI_Open(SDHI_CH0) != SDHI_SUCCESS)
{
    /* Error */
}
```

Special Notes

スワップコントロールレジスタ（SDSWAP）初期化後の値は、エンディアン設定により異なります。
リトルエンディアンの場合：0x00000000（スワップ書き込み／読み出し：無効）
ビッグエンディアンの場合：0x000000c0（スワップ書き込み／読み出し：有効）
本関数実行前に、端子設定が必要です。「4 端子設定」を参照してください。
本関数が正常終了しない場合、R_SDHI_GetVersion()関数、R_SDHI_Log()関数、R_SDHI_SetLogHdlAddress()関数以外のライブラリ関数が使用できません。
本関数実行前後で、端子の状態は変化しません。

R_SDHI_Close()

使用中の SDHI FIT モジュールのリソースを開放する関数です。

Format

```
sdhi_status_t R_SDHI_Close(
    uint32_t channel
)
```

Parameters

channel

チャネル番号	使用する SDHI チャネル番号（0 起算）

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDHI FIT モジュールの全ての処理を終了し、引数 channel で設定した SDHI チャネルのリソースを解放します。
その SDHI チャネルをモジュールストップ状態に設定します。
本関数実行後、挿抜割り込みは禁止状態になります。

Example

```
/* ==== Please add the processing to set the pins. ==== */

if (R_SDHI_Close(SDHI_CH0) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に、端子設定が必要です。「4 端子設定」を参照してください。また、本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数実行前後で、端子の状態は変化しません。

R_SDHI_IntHandler0()

割り込みハンドラです。

Format

```
void R_SDHI_IntHandler0(
      void *vect
)
```

Parameters

**vect*

ベクタテーブル

Return Values

なし

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDHI FIT モジュールの割り込みハンドラです。
SDHI に対応する割り込み要因の処理ルーチンとしてシステムに組み込み済です。
カードアクセス割り込み（CACI）およびカード検出割り込み（CDETI）コールバック関数、SDIO アクセス割り込み（SDACI）コールバック関数を登録している場合は、本関数内からコールバック関数をコールします。

Example

システムに組み込み済であるため、設定不要です。

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_IntCallback()

カードアクセス割り込み（CACI）およびカード検出割り込み（CDETI）のコールバック関数を登録する関数です。

Format

```
sdhi_status_t R_SDHI_IntCallback(
    uint32_t channel,
    sdhi_status_t (*callback)(uint32_t, uint32_t)
)
```

Parameters

channel

チャネル番号　使用する SDHI チャネル番号（0 起算）

*(*callback)(uint32_t, uint32_t)　：登録するコールバック関数*

ヌルポインタを設定した場合、コールバック関数は登録されません。

第一引数（uint32_t）には SD ステータスレジスタ 1（SDSTS1）の値が格納されます。

第二引数(uint32_t)には SD ステータスレジスタ 2（SDSTS2）の値が格納されます。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

カードアクセス割り込み（CACI）およびカード検出割り込み（CDETI）のコールバック関数を登録します。本関数で登録したコールバック関数は、割り込みハンドラのサブルーチンとして、SD のプロトコルステータス変化による割り込み発生時にコールされます。

Example

```
/* Callback function */
sdhi_status_t r_sdhi_callback(uint32_t sdsts1, uint32_t sdsts2)
{
      /* Do nothing */

      return SDHI_SUCCESS;
}

if (R_SDHI_IntCallback(SDHI_CH0, r_sdhi_callback) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

本関数で登録するコールバック関数は、SD バッファアクセス割り込み（SBFAI）や SDIO アクセス割り込み（SDACI）のコールバック関数と異なります。そのため、これらの割り込みが発生した場合でも、本コールバック関数はコールされません。

R_SDHI_IntSDBuffCallback()

SD バッファアクセス割り込み（SBFAI）コールバック関数登録する関数です。

Format

```
sdhi_status_t R_SDHI_IntSDBuffCallback(
      uint32_t channel,
      sdhi_status_t (*callback)(void *)
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

*(*callback)(void *)　：登録するコールバック関数*

ヌルポインタを設定した場合、コールバック関数は登録されません。

(void *)には常に 0 が格納されます。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD バッファアクセス割り込み（SBFAI）のコールバック関数を登録します。本関数で登録したコールバック関数は、DTC によるデータ転送完了割り込みハンドラのサブルーチンとして、DTC 転送完了割り込み発生時にコールされます。

Example

```
/* Callback function */
sdhi_status_t r_sdhi_sdbuff_callback(void * vect)
{
      /* Do nothing */

      return SDHI_SUCCESS;
}

if (R_SDHI_IntSDBuffCallback(SDHI_CH0, r_sdhi_sdbuff_callback) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

本関数で登録するコールバック関数は、カードアクセス割り込（CACI）およびカード検出割り込（CDETI）や SDIO アクセス割り込（SDACI）のコールバック関数と異なります。そのため、これらの割り込みが発生した場合でも、本コールバック関数はコールされません。

R_SDHI_IntSdioCallback()

SDIO アクセス割り込み（SDACI）コールバック関数を登録する関数です。

Format

```
sdhi_status_t R_SDHI_IntSdioCallback(
      uint32_t channel,
      sdhi_status_t (*callback)(uint32_t)
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

*(*callback)(uint32_t)　：登録するコールバック関数*

ヌルポインタを設定した場合、コールバック関数は登録されません。

第一引数(uint32_t)には SDIO ステータスレジスタ（SDIOSTS）の値が格納されます。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDIO アクセス割り込み（SDACI）割り込みコールバック関数を登録します。
本関数で登録したコールバック関数は、割り込みハンドラのサブルーチンとして、SDHI の SDIO 割り込み発生時にコールされます。

Example

```
/* Callback function */
sdhi_status_t r_sdhi_sdio_callback(int32_t sdiosts)
{
      /* Do nothing */

      return SDHI_SUCCESS;
}

if (R_SDHI_IntSdioCallback(SDHI_CH0, r_sdhi_sdio_callback) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数で登録するコールバック関数は、カードアクセス割り込み（CACI）およびカード検出割り込み（CDETI）や SD バッファアクセス割り込み（SBFAI）のコールバック関数と異なります。そのため、これらの割り込みが発生した場合でも、本コールバック関数はコールされません。

R_SDHI_EnableIcuInt()

SDHI 用 ICU コントローラ割り込み [1]を有効にします。

Format

```
sdhi_status_t R_SDHI_EnableIcuInt(
    uint32_t channel,
    uint32_t select
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

select

下表のマクロ定義に示す値、もしくは、論理和で割り込み引数を設定してください。

マクロ定義	値（ビット）	処理内容
SDHI_HWINT_ACCESS_CD	0x0001	カード検出割り込み（CDETI）設定 カードアクセス割り込み（CACI）設定 SDIO アクセス割り込み（SDACI）設定
SDHI_HWINT_BUFFER	0x0010	SD バッファアクセス割り込み（SBFAI）設定

Return Values

SDHI_SUCCESS　　*正常終了*

SDHI_ERR　　*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

ICU コントローラのレジスタを設定します。

- SDHI 用の割り込み要因プライオリティレジスタ（IPR）を設定します。設定値は#define SDHI_CHx_INT_LEVEL および#define SDHI_CFG_CHx_INT_LEVEL_DMADTC で定義した値です。
- SDHI 用の割り込み要求許可レジスタ（IEN）を設定し、割り込みを有効にします。

Example

```
/* Enable all SDHI ICU interrupt */
R_SDHI_EnableIcuInt(SDHI_CH0, SDHI_HWINT_ACCESS_CD | SDHI_HWINT_BUFFER);

/* Enable only SD buffer access ICU interrupt */
R_SDHI_EnableIcuInt(SDHI_CH0, SDHI_HWINT_BUFFER);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

[1] 次の割り込みを有効にします。【SD バッファアクセス（SBFAI）割り込み、カード検出（CDETI）割り込み、カードアクセス（CACI）割り込み、SDIO アクセス（SDACI）割り込み】

R_SDHI_DisableIcuInt()

SDHI 用 ICU コントローラ割り込み [2]を無効にします。

Format

```
sdhi_status_t R_SDHI_DisableIcuInt(
    uint32_t channel,
    uint32_t select
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

select

下表のマクロ定義に示す値、もしくは、論理和で割り込み引数を設定してください。

マクロ定義	値（ビット）	処理内容
SDHI_HWINT_ACCESS_CD	0x00000001	カード検出割り込み（CDETI）設定 カードアクセス割り込み（CACI）設定 SDIO アクセス割り込み（SDACI）設定
SDHI_HWINT_BUFFER	0x00000010	SD バッファアクセス割り込み（SBFAI）設定

Return Values

SDHI_SUCCESS　　*正常終了*

SDHI_ERR　　*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

ICU コントローラのレジスタを設定します。

- SDHI 用の割り込み要因プライオリティレジスタ（IPR）を“0”にします。
- SDHI 用の割り込み要求許可レジスタ（IEN）を設定し、割り込みを無効にします。

Example

```
/* Disable all SDHI ICU interrupt */
R_SDHI_DisableIcuInt(SDHI_CH0, SDHI_HWINT_ACCESS_CD | SDHI_HWINT_BUFFER);

/* Disable only SD buffer access ICU interrupt */
R_SDHI_DisableIcuInt(SDHI_CH0, SDHI_HWINT_BUFFER);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

[2] 次の割り込みを無効にします。【SD バッファアクセス（SBFAI）割り込み、カード検出（CDETI）割り込み、カードアクセス（CACI）割り込み、SDIO アクセス（SDACI）割り込み】

R_SDHI_SetIntMask()

SD 割り込みマスクレジスタを制御し、SD 割り込みを有効にする関数です。

Format

```
sdhi_status_t R_SDHI_SetIntMask(
    uint32_t channel,
    uint32_t mask1,
    uint32_t mask2
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

mask1

SD 割り込みマスクレジスタ 1（SDIMSK1）制御

割り込みを有効にする場合、対象のビットを“1”にしてください。

割り込み設定を変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビットへの設定は無効です。

ビット	シンボル	ビット名	R/W
b0	RSPENDM	レスポンスエンド割り込み要求マスクビット	R/W
b1	-	予約ビット	R
b2	ACENDM	アクセスエンド割り込み要求マスクビット	R/W
b3	SDCDRMM	SDHI_CD 抜去割り込み要求マスクビット	R/W
b4	SDCDRMM	SDHI_CD 挿入割り込み要求マスクビット	R/W
b7-b5	-	予約ビット	R
b8	SDD3RMM	SDHI_D3 抜去割り込み要求マスクビット	R/W
b9	SDD3INM	SDHI_D3 挿入割り込み要求マスクビット	R/W
b31-b10	-	予約ビット	R

mask2

SD 割り込みマスクレジスタ 2（SDIMSK2）制御

割り込みを有効にする場合、対象のビットを“1”にしてください。

割り込み設定を変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビットへの設定は無効です。

ビット	シンボル	ビット名	R/W
b0	CMDEM	コマンドエラー割り込み要求マスクビット	R/W
b1	CRCEM	CRC エラー割り込み要求マスクビット	R/W
b2	ENDEM	エンドビットエラー割り込み要求マスクビット	R/W
b3	DTTOM	データタイムアウト割り込み要求マスクビット	R/W
b4	ILWM	SDBUFR 不正書き込み割り込み要求マスクビット	R/W
b5	ILRM	SDBUFR 不正読み出し割り込み要求マスクビット	R/W
b6	RSPTOM	レスポンスタイムアウト割り込み要求マスクビット	R/W
b7	-	予約ビット	R
b8	BREM	BRE 割り込み要求マスクビット	R/W
b9	BWEM	BWE 割り込み要求マスクビット	R/W
b14-b10	-	予約ビット	R
b15	ILAM	不正アクセスエラー割り込み要求マスクビット	R/W
b31-b16	-	予約ビット	R

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD 割り込みマスクレジスタ 1（SDIMSK1）と SD 割り込みマスクレジスタ 2（SDIMSK2）を制御し、SD 割り込みを有効にします。

Example

```
#define SDHI_SDIMSK1_DATA_TRNS      (0x0004u)   /* Command sequence end */
#define SDHI_SDIMSK2_BWE            (0x8a7fu)   /* Write enable and All errors*/

R_SDHI_SetIntMask(SDHI_CH0, SDHI_SDIMSK1_DATA_TRNS, SDHI_SDIMSK2_BWE);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数コール後、R_SDHI_EnableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを有効にしてください。有効にしない場合、SD 割り込みは発生しません。

R_SDHI_ClearIntMask()

SD 割り込みマスクレジスタを制御し、SD 割り込みを無効にする関数です。

Format

```
sdhi_status_t R_SDHI_ClearIntMask(
    uint32_t channel,
    uint32_t mask1,
    uint32_t mask2
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

mask1

SD 割り込みマスクレジスタ 1（SDIMSK1）制御

割り込みを無効にする場合、対象のビットを“1”にしてください。
割り込み設定を変更しない場合、対象のビットを“0”にしてください。
但し、Read Only ビットへの設定は無効です。
SDIMSK1 レジスタの詳細は「R_SDHI_SetIntMask()」を参照してください。

mask2

SD 割り込みマスクレジスタ 2（SDIMSK2）制御

割り込みを無効にする場合、対象のビットを“1”にしてください。
割り込み設定を変更しない場合、対象のビットを“0”にしてください。
但し、Read Only ビットへの設定は無効です。
SDIMSK2 レジスタの詳細は「R_SDHI_SetIntMask()」を参照してください。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD 割り込みマスクレジスタ 1（SDIMSK1）と SD 割り込みマスクレジスタ 2（SDIMSK2）を制御し、SD 割り込みを無効にします。

Example

```
#define SDHI_SDIMSK1_DATA_TRNS      (0x0004u)   /* Command sequence end */
#define SDHI_SDIMSK2_BWE            (0x8a7fu)   /* Write enable and All errors*/

R_SDHI_ClearIntMask(SDHI_CH0, SDHI_SDIMSK1_DATA_TRNS, SDHI_SDIMSK2_BWE);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数コール前に、R_SDHI_DisableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを無効にしてください。無効にしない場合、意図しないタイミングで SD 割り込みが発生する可能性があります。

R_SDHI_ClearSdstsReg()

SD ステータスレジスタを制御し、割り込みフラグをクリアする関数です。

Format

```
sdhi_status_t R_SDHI_ClearSdstsReg(
    uint32_t channel,
    uint32_t clear_sdsts1,
    uint32_t clear_sdsts2
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

clear_sdsts1

SD ステータスレジスタ 1（SDSTS1）制御

割り込みフラグを 0 クリアする場合、対象のビットを“1”にしてください。

割り込みフラグを変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビットへの設定は無効です。

ビット	シンボル	ビット名	R/W
b0	RSPEND	レスポンスエンドフラグ	R/W
b1	-	予約ビット	R
b2	ACEND	アクセスエンドフラグ	R/W
b3	SDCDRM	SDHI_CD 抜去フラグ	R/W
b4	SDCDIN	SDHI_CD 挿入フラグ	R/W
b5	SDCDMON	SDHI_CD モニタフラグ	R
b6	-	予約ビット	R
b7	SDWPMON	SDHI_WP モニタフラグ	R
b8	SDD3RM	SDHI_D3 抜去フラグ	R/W
b9	SDD3IN	SDHI_D3 挿入フラグ	R/W
b10	SDD3MON	SDHI_D3 モニタフラグ	R
b31-b11	-	予約ビット	R

clear_sdsts2

SD ステータスレジスタ 2（SDSTS2）制御

割り込みフラグを 0 クリアする場合、対象のビットを“1”にしてください。

割り込みフラグを変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビット、及び、b12（予約ビット）への設定は無効です。

ビット	シンボル	ビット名	R/W
b0	CMDE	コマンドエラーフラグ	R/W
b1	CRCE	CRC エラーフラグ	R/W
b2	ENDE	エンドビットエラーフラグ	R/W
b3	DTO	データタイムアウトフラグ	R/W
b4	ILW	SDBUFR 不正書き込みフラグ	R/W
b5	ILR	SDBUFR 不正読み出しフラグ	R/W
b6	RSPTO	レスポンスタイムアウトフラグ	R/W
b7	SDD0MON	SDHI_D0 モニタフラグ	R
b8	BRE	SDBUFR 読み出し許可フラグ	R/W
b9	BWE	SDBUFR 書き込み許可フラグ	R/W
b10	-	予約ビット	R
b11	-	予約ビット（注 1）	R

b12	-	予約ビット	R
b13	SDCLKCREN	SDCLKCR 書き込み許可フラグ	R
b14	CBSY	コマンドシーケンスビジーフラグ	R
b15	ILA	不正アクセスエラーフラグ	R/W
b31-b16	-	予約ビット	R

注１：設定値に関係なく対象ビットに“1”を書き込みます。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD ステータスレジスタ 1（SDSTS1）と SD ステータスレジスタ 2（SDSTS2）の割り込みフラグをクリアします。

Example

```
#define SDHI_SDIMSK1_TRNS_RESP      (0x0005u)
                                    /* Command sequence end and Response end */
#define SDHI_SDIMSK2_CLEAR          (0x837fu)
                                    /* All initialization clear */

R_SDHI_ClearSdstsReg(SDHI_CH0, SDHI_SDIMSK1_TRNS_RESP, SDHI_SDIMSK2_CLEAR);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数コール前に、R_SDHI_DisableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを無効にしてください。無効にしない場合、意図しないタイミングで SD 割り込みが発生する可能性があります。

R_SDHI_SetSdioIntMask()

SDIO 割り込みマスクレジスタを制御し、SDIO 割り込みを有効にする関数です。

Format

```
sdhi_status_t R_SDHI_SetSdioIntMask(
    uint32_t channel,
    uint32_t mask
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

mask

SDIO 割り込みマスクレジスタ（SDIOIMSK）制御

割り込みを有効にする場合、対象のビットを“1”にしてください。

割り込み設定を変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビット、及び、b2-b1（予約ビット）への設定は無効です。

ビット	シンボル	ビット名	R/W
b0	IOIRQM	IOIRQ 割り込みマスクビット	R/W
b2-b1	-	予約ビット（注１）	R/W
b13-b3	-	予約ビット	R
b14	EXPUB52M	EXPUB52 割り込みマスクビット	R/W
b15	EXWTM	EXWT 割り込みマスクビット	R/W
b31-b16	-	予約ビット	R

注１：設定値に関係なく対象ビットに“1”を書き込みます。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDIO 割り込みマスクレジスタ（SDIOIMSK）を制御し、割り込みを有効にします。

Example

```
#define SDHI_SDIOIMSK_IOIRQ   (0x0001u)   /* Interrupt from IO Card */

R_SDHI_SetSdioIntMask(SDHI_CH0, SDHI_SDIOIMSK_IOIRQ);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

本関数コール後、R_SDHI_EnableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを有効にしてください。有効にしない場合、SDIO 割り込みは発生しません。

R_SDHI_ClearSdioIntMask()

SDIO 割り込みマスクレジスタを制御し、SDIO 割り込みを無効にする関数です。

Format

```
sdhi_status_t R_SDHI_ClearSdioIntMask(
    uint32_t channel,
    uint32_t mask
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

mask

SDIO 割り込みマスクレジスタ（SDIOIMSK）制御

割り込みを無効にする場合、対象のビットを“1”にしてください。
割り込み設定を変更しない場合、対象のビットを“0”にしてください。
但し、Read Only ビット、及び、b2-b1（予約ビット）への設定は無効です。
SDIOIMSK レジスタの詳細は「R_SDHI_SetSdioIntMask()」を参照してください。

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDIO 割り込みマスクレジスタ（SDIOIMSK）を制御し、割り込みを無効にします。

Example

```
#define SDHI_SDIOIMSK_IOIRQ   (0x0001u)   /* Interrupt from IO Card */

R_SDHI_ClearSdioIntMask(SDHI_CH0, SDHI_SDIOIMSK_IOIRQ);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
本関数コール前に、R_SDHI_DisableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを無効にしてください。無効にしない場合、意図しないタイミングで SD 割り込みが発生する可能性があります。

R_SDHI_ClearSdiostsReg()

SDIO ステータスレジスタを制御し、割り込みフラグをクリアする関数です。

Format

```
sdhi_status_t R_SDHI_ClearSdiostsReg(
    uint32_t channel,
    uint32_t clear
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

clear

SDIO ステータスレジスタ（SDIOSTS）制御

割り込みフラグをクリアする場合、対象のビットを“1”にしてください。

割り込みフラグを変更しない場合、対象のビットを“0”にしてください。

但し、Read Only ビット、及び、b2-b1（予約ビット）への設定は無効です。

ビット	シンボル	ビット名	R/W
b0	IOIRQ	SDIO 割り込みフラグ	R/W
b2-b1	-	予約ビット（注１）	R/W
b13-b3	-	予約ビット	R
b14	EXPUB52	EXPUB52 ステータスフラグ	R/W
b15	EXWT	EXWT ステータスフラグ	R/W
b31-b16	-	予約ビット	R

注１：設定値に関係なく対象ビットに“1”を書き込みます。

Return Values

SDHI_SUCCESS　　*正常終了*

SDHI_ERR　　*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDIO ステータスレジスタ（SDIOSTS）の割り込みフラグをクリアします。

Example

```
#define SDHI_SDIOIMSK_IOIRQ   (0x0001u)   /* Interrupt from IO Card */

R_SDHI_ClearSdiostsReg(SDHI_CH0, SDHI_SDIOIMSK_IOIRQ);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

本関数コール前に、R_SDHI_DisableIcuInt()関数をコールして、SDHI 用 ICU コントローラ割り込みを無効にしてください。無効にしない場合、意図しないタイミングで SD 割り込みが発生する可能性があります。

R_SDHI_SetClock()

SD クロックの供給／停止を行う関数です。

Format

```
sdhi_status_t R_SDHI_SetClock(
      uint32_t channel,
      uint32_t div,
      int32_t enable
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

div

以下の値を設定してください。

ハイスピードモード：SDHI_CFG_DIV_HIGH_SPEED

デフォルトスピードモード：SDHI_CFG_DIV_DEFAULT_SPEED

カード認識モード：SDHI_CFG_DIV_INIT_SPEED

なお、上記の定義は「2.7 コンパイル時の設定」を参照して設定してください。

enable

以下の値を設定してください。

クロック停止：SDHI_CLOCK_DISABLE

クロック供給：SDHI_CLOCK_ENABLE

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD クロックの供給／停止を行います。

Example

```
/* Supply the clock */
if (R_SDHI_SetClock(SDHI_CH0, SDHI_CFG_DIV_INIT_SPEED, SDHI_CLOCK_ENABLE) !=
SDHI_SUCCESS)
{
      /* Error */
}

/* Stop the clock */
if (R_SDHI_SetClock(SDHI_CH0, 0, SDHI_CLOCK_DISABLE) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_SetBus()

SD バスを設定する関数です。

Format

```
sdhi_status_t R_SDHI_SetBus(
    uint32_t channel,
    int32_t width
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

width

以下の値を設定してください。

1 ビットバス：SDHI_PORT_1BIT

4 ビットバス：SDHI_PORT_4BIT

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD Bus 幅選択ビット（SDOPT.WIDTH）を制御し、SD バスを 1 ビットまたは 4 ビットに設定します。

Example

```
R_SDHI_SetBus(SDHI_CD0, SDHI_PORT_1BIT);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
コマンドシーケンス実行中（SDSTS2.CBSY=1）の場合、本関数はコールしないでください。

R_SDHI_GetResp()

SD カードからのレスポンスを取得する関数です。

Format

```
sdhi_status_t R_SDHI_GetResp(
    uint32_t channel,
    sdhi_get_resp_t * p_resp_reg
)
```

Parameters

channel

チャネル番号　使用する SDHI チャネル番号（0 起算）

*p_resp_reg

レスポンスレジスタ情報構造体

sdrsp10：レスポンスレジスタ 10 格納用変数

sdrsp32：レスポンスレジスタ 32 格納用変数

sdrsp54：レスポンスレジスタ 54 格納用変数

sdrsp76：レスポンスレジスタ 76 格納用変数

Return Values

SDHI_SUCCESS　*正常終了*

SDHI_ERR　*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

レスポンスレジスタ（SDRSP10、SDRSP32、SDRSP54、SDRSP76）に格納される値をレスポンスレジスタ情報構造体に格納します。レスポンスの種類により、レスポンスの内容を sdrsp10、sdrsp32、sdrsp54、sdrsp76 レジスタに分割して格納します。表 3-1 にレスポンスレジスタ情報構造体とレスポンスの格納先の対応を示します。

表 3-1　レスポンスレジスタ情報構造体とレスポンスの格納先の対応

レスポンスタイプ	sdrsp76	sdrsp54	sdrsp32	sdrsp10
R1	-	[39:8]　注 1	-	[39:8]
R1b	-	[39:8]　注 1	-	[39:8]
R2	[127:104]	[103:72]	[71:40]	[39:8]
R3	-	-	-	[39:8]
R4	-	-	-	[39:8]
R5	-	-	-	[39:8]
R6	-	-	-	[39:8]
R7	-	-	-	[39:8]

注 1：CMD18 または CMD25 に対するレスポンスは、SDRSP10 レジスタと SDRSP54 レジスタの両方に格納されます。このため、自動送信された CMD12 のレスポンスが SDRSP10 レジスタに上書きされた場合でも、SDRSP54 レジスタに格納された値を参照することで、CMD18 または CMD25 に対するレスポンスを確認できます。

Example

```
sdhi_get_resp_t   resp_reg;

R_SDHI_GetResp(channel, &resp_reg);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_OutReg()

SDHI レジスタを設定する関数です。

Format

```
sdhi_status_t R_SDHI_OutReg(
    uint32_t channel,
    uint32_t reg,
    uint32_t data
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

reg

SDHI ベースレジスタオフセット値。下表を参照しマクロ定義を設定してください。

レジスタ名称	オフセット	マクロ定義
コマンドレジスタ（SDCMD）	0x00u	SDHI_SDCMD
アーギュメントレジスタ（SDARG）	0x08u	SDHI_SDARG
データストップレジスタ（SDSTOP）	0x10u	SDHI_SDSTOP
ブロックカウントレジスタ（SDBLKCNT）	0x14u	SDHI_SDBLKCNT
レスポンスレジスタ 10（SDRESP10）	0x18u	SDHI_SDRESP10
レスポンスレジスタ 32（SDRESP32）	0x20u	SDHI_SDRESP32
レスポンスレジスタ 54（SDRESP54）	0x28u	SDHI_SDRESP54
レスポンスレジスタ 76（SDRESP76）	0x30u	SDHI_SDRESP76
SD ステータスレジスタ 1（SDSTS1）	0x38u	SDHI_SDSTS1
SD ステータスレジスタ 2（SDSTS2）	0x3cu	SDHI_SDSTS2
SD 割り込みマスクレジスタ 1（SDIMSK1）	0x40u	SDHI_SDIMSK1
SD 割り込みマスクレジスタ 2（SDIMSK2）	0x44u	SDHI_SDIMSK2
SDHI クロックコントロールレジスタ（SDCLKCR）	0x48u	SDHI_SDCLKCR
転送データサイズレジスタ（SDSIZE）	0x4cu	SDHI_SDSIZE
カードアクセスオプションレジスタ（SDOPT）	0x50u	SDHI_SDOPT
SD エラーステータスレジスタ 1（SDERSTS1）	0x58u	SDHI_SDERSTS1
SD エラーステータスレジスタ 2（SDERSTS2）	0x5cu	SDHI_SDERSTS2
SD バッファレジスタ（SDBUFR）	0x60u	SDHI_SDBUFR
SDIO モードコントロールレジスタ（SDIOMD）	0x68u	SDHI_SDIOMD
SDIO ステータスレジスタ（SDIOSTS）	0x6cu	SDHI_SDIOSTS
SDIO 割り込みマスクレジスタ（SDIOIMSK）	0x70u	SDHI_SDIOIMSK
DMA 転送許可レジスタ（SDDMAEN）	0x1b0u	SDHI_SDDMAEN
SDHI ソフトウェアリセットレジスタ（SDRST）	0x1c0u	SDHI_SDRST
バージョンレジスタ（SDVER）	0x1c4u	SDHI_SDVER
スワップコントロールレジスタ（SDSWAP）	0x1e0u	SDHI_SDSWAP

data

レジスタ設定値

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDHI レジスタを設定します。

Example

```
R_SDHI_OutReg(SDHI_CD0, SDHI_SDCMD, cmd);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_InReg()

SDHI レジスタの値を取得する関数です。

Format

```
sdhi_status_t R_SDHI_InReg(
      uint32_t channel,
      uint32_t reg,
      uint32_t * p_data
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

reg

SDHI ベースレジスタオフセット値。「R_SDHI_OutReg()」の表を参照しマクロ定義を設定してください。

** p_data*

取得したレジスタ値の格納先ポインタ

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDHI レジスタを設定します。

Example

```
R_SDHI_InReg(SDHI_CD0, SDHI_SDSTS1, &sdstas1);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_CDLayout()

SDHI_CD（SD カード検出）端子の使用有無を確認する関数です。

Format

```
sdhi_status_t R_SDHI_CDLayout(
    uint32_t channel
)
```

Parameters

channel

チャネル番号	使用する SDHI チャネル番号（0 起算）

Return Values

SDHI_SUCCESS	*CD 端子を使用する*
SDHI_ERR	*CD 端子を使用しない*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

CD 端子の使用有無を確認します。

Example

```
if (R_SDHI_CDLayout(SDHI_CD0) == SDHI_SUCCESS)
{
      /* User setting */
}
```

Special Notes

なし

R_SDHI_WPLayout()

SDHI_WP（SD ライトプロテクト）端子の使用有無を確認する関数です。

Format

```
sdhi_status_t R_SDHI_WPLayout(
      uint32_t channel
)
```

Parameters

channel

チャネル番号	使用する SDHI チャネル番号（0 起算）

Return Values

SDHI_SUCCESS	*WP 端子を使用する*
SDHI_ERR	*WP 端子を使用しない*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

WP 端子の使用有無を確認します。

Example

```
if (R_SDHI_WPLayout(SDHI_CD0) == SDHI_SUCCESS)
{
      /* User setting */
}
```

Special Notes

なし

R_SDHI_GetWP()

SDHI_WP（SD ライトプロテクト）端子の状態を取得する関数です。

Format

```
sdhi_status_t R_SDHI_GetWP(
    uint32_t channel,
    uint32_t * p_wp
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

** p_wp*

SDHI_WP 端子状態格納先バッファポインタ

0：SDHI_WP 端子のレベルは High

1：SDHI_WP 端子のレベルは Low

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SDHI_WP 端子の状態を取得します。

Example

```
R_SDHI_GetWP(SDHI_CH0, &wp);
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。
この機能を実行するには、SDHI_WP 端子のピン設定処理が必要です。詳しくは「4 端子設定」を参照してください。

R_SDHI_GetSpeedType()

対象 MCU が対応しているスピードモード情報を取得する関数です。

Format

```
sdhi_status_t R_SDHI_GetSpeedType(
    uint32_t channel
)
```

Parameters

channel

チャネル番号	使用する SDHI チャネル番号（0 起算）

Return Values

SDHI_SUCCESS	*デフォルトスピード、ハイスピードモード対応*
SDHI_ERR	*デフォルトスピードモード対応*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

対象 MCU が対応しているスピードモード情報を取得します。

Example

```
if (R_SDHI_GetSpeedType(SDHI_CH0) == SDHI_SUCCESS)
{
      /* User setting */
}
```

Special Notes

なし

R_SDHI_GetBuffRegAddress()

SD バッファレジスタのアドレスを取得する関数です。

Format

```
sdhi_status_t R_SDHI_GetBuffRegAddress(
      uint32_t channel,
      uint32_t *p_reg_buff
)
```

Parameters

channel

チャネル番号　　使用する SDHI チャネル番号（0 起算）

**p_reg_buff*

SD バッファレジスタアドレスポインタ

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

SD バッファレジスタのアドレスを取得し、バッファに格納します。
DMAC 転送／DTC 転送使用時のデータレジスタアドレスを設定する場合等に使用します。

Example

```
uint32_t    reg_buff = 0;

if (R_SDHI_Get_BuffRegAddress(SDHI_CH0, &reg_buff) != SDHI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDHI_Open()関数による初期化処理が必要です。

R_SDHI_GetVersion()

ドライバのバージョン情報を取得する関数です。

Format

```
uint32_t R_SDHI_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）*

Properties

r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

ドライバのバージョン情報を返します。

Example

```
uint32_t version;
version = R_SDHI_GetVersion();
```

Special Notes

なし

R_SDHI_SetLogHdlAddress()

LONGQ FIT モジュールのハンドラアドレスを設定する関数です。

Format

```
sdhi_status_t R_SDHI_SetLogHdlAddress(
        uint32_t user_long_que
)
```

Parameters

user_long_que
LONGQ FIT モジュールのハンドラアドレス

Return Values

SDHI_SUCCESS	*正常終了*

Properties

ファイル r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

LONGQ FIT モジュールのハンドラアドレスを SDHI FIT モジュールに設定します。

Example

```
#define ERR_LOG_SIZE (16)
#define SDHI_USER_LONGQ_IGN_OVERFLOW    (1)

sdhi_status_t     ret = SDHI_SUCCESS;
uint32_t          MtlLogTbl[ERR_LOG_SIZE];
longq_err_t       err;
longq_hdl_t       p_sdhi_user_long_que;
uint32_t          long_que_hndl_address;


/* Open LONGQ module. */
err = R_LONGQ_Open(&MtlLogTbl[0],
                   ERR_LOG_SIZE,
                   SDHI_USER_LONGQ_IGN_OVERFLOW,
                   &p_sdhi_user_long_que
);

long_que_hndl_address = (uint32_t)p_sdhi_user_long_que;
ret = R_SDHI_SetLogHdlAddress(long_que_hndl_address);
```

Special Notes

LONGQ FIT モジュールを使用し、エラーログを取得するための準備処理です。R_SDHI_Open()関数をコールする前に処理を実行してください。
別途 LONGQ FIT モジュールを組み込んでください。
SDHI_CFG_LONGQ_ENABLE が無効のときにこの関数が呼び出された場合、この関数はなにもしません。

R_SDHI_Log()

エラーログを取得する関数です。

Format

```
uint32_t   R_SDHI_Log(
        uint32_t flg,
        uint32_t fid,
        uint32_t line
)
```

Parameters

flg

0x00000001（固定値）

fid

0x0000003f（固定値）

line

0x00001fff（固定値）

Return Values

0	*正常終了*

Properties

ファイル r_sdhi_rx_if.h にプロトタイプ宣言されています。

Description

エラーログを取得します。
エラーログ取得を終了する場合、コールしてください。

Example

```
#define USER_DRIVER_ID          (1)
#define USER_LOG_MAX            (63)
#define USER_LOG_ADR_MAX        (0x00001fff)

if (R_SDHI_Open(SDHI_CH0) != SDHI_SUCCESS)
{
      /* Error */
      R_SDHI_Log(USER_DRIVER_ID, USER_LOG_MAX, USER_LOG_ADR_MAX);
}
```

Special Notes:

デバッグ用モジュールを使用してください。
別途 LONGQ FIT モジュールを組み込んでください。
SDHI_CFG_LONGQ_ENABLE が無効のときにこの関数が呼び出された場合、この関数はなにもしません。

4.　端子設定

SDHI FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。

e^2 studio の場合はスマート・コンフィグレータの端子設定機能を使用することができます。スマート・コンフィグレータの端子設定機能を使用すると、端子設定画面で選択したオプションに応じて、ソースファイルが出力されます。そのソースファイルで定義された関数を呼び出すことにより端子を設定できます。詳細は表 4-1 を参照してください。

端子設定の制御手順は「4.1 SD カードの挿入と電源投入タイミング」と「4.2 SD カードの抜去と電源停止タイミング」を参照してください。

表 4-1　スマート・コンフィグレータが出力する関数一覧

出力される関数名	機能
R_SDHI_PinSetInit()	SDHI 端子の初期設定を行います。 実行後は SDHI_CD 端子と SDHI_WP 端子のみ有効です。
R_SDHI_PinSetTransfer()	SDHI 端子を SD コマンド発行可能状態にします。 実行後は全 SDHI 端子が有効です。
R_SDHI_PinSetDetection()	SDHI 端子を SD コマンド発行禁止状態にします。 実行後は SDHI_CD 端子と SDHI_WP 端子のみ有効です。
R_SDHI_PinSetEnd()	SDHI 制御無効状態にします。 実行後は全 SDHI 端子が無効です。

4.1　SD カードの挿入と電源投入タイミング

制御手順を図 4-1、表 4-2 に示します。SD カードの挿入は、R_SDHI_Open()関数の正常終了後、SD カードへの電源電圧供給停止状態、かつ SDHI 出力端子を L 出力状態で行ってください。

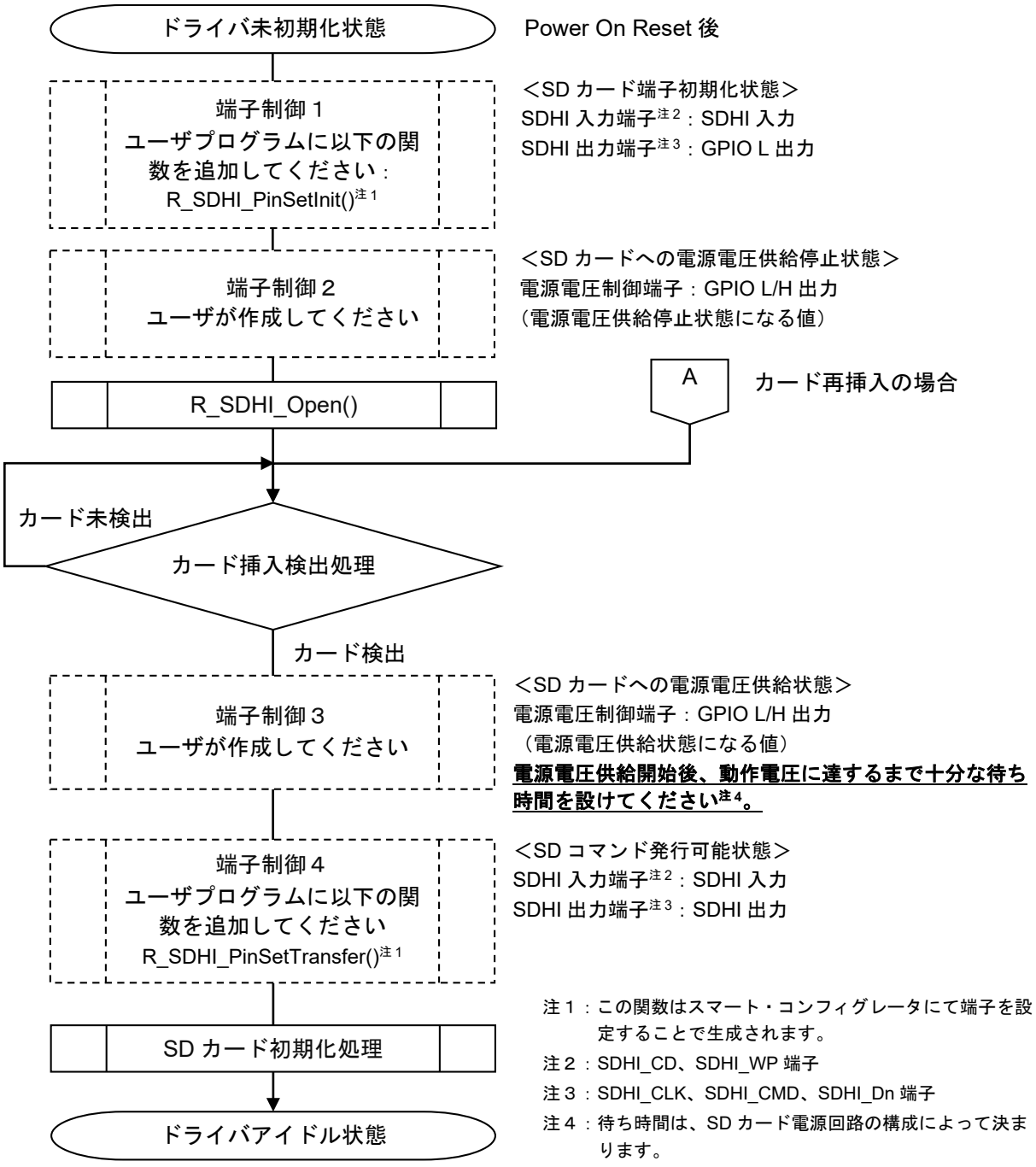


図 4-1　SD カードの挿入と電源投入タイミング

表 4-2　SD カード挿入時のユーザ設定方法

処理	対象端子	端子設定	実行後の端子状態
端子制御 1	SDHI 入力端子 注 1	PMR 設定：汎用入出力ポート PCR 設定：入力プルアップ抵抗無効 注 3 PDR 設定：入力 MPC 設定：SDHI PMR 設定：周辺モジュール	SDHI 入力 （SD カード検出可能状態）
	SDHI 出力端子 注 2	PMR 設定：汎用入出力ポート DSCR 設定：高駆動出力 PCR 設定：入力プルアップ抵抗無効 注 3 PODR 設定：L 出力 PDR 設定：出力 MPC 設定：Hi-z	GPIO L 出力
端子制御 2	電源電圧制御端子	PMR 設定：汎用入出力 PCR 設定：入力プルアップ抵抗無効 注 4 PODR 設定：L 出力／H 出力（電源電圧供給停止状態になる値を出力） PDR 設定：出力	GPIO L/H 出力 （電源電圧供給停止状態）
端子制御 3	電源電圧制御端子	PODR 設定：L 出力／H 出力（電源電圧供給状態になる値を出力）	GPIO L/H 出力 （電源電圧供給状態）
端子制御 4	SDHI 入力端子 注 1	MPC 設定：SDHI PMR 設定：周辺モジュール	SDHI 入力
	SDHI 出力端子 注 2	MPC 設定：SDHI PMR 設定：周辺モジュール	SDHI 出力 （SD コマンド発行可能状態）

注 1：SDHI_CD、SDHI_WP 端子

注 2：SDHI_CLK、SDHI_CMD、SDHI_Dn 端子

注 3：MCU 外部でプルアップされることを想定しているため、MCU 内蔵プルアップは無効にしてください。

注 4：システムに合わせて設定を見直してください。

4.2　SD カードの抜去と電源停止タイミング

制御手順を図 4-2、表 4-3 に示します。SD カードの抜去は、SD カードへの電源電圧供給停止状態で行ってください。また、意図せず SD カードが抜去された場合でも、同様の手順で電源電圧供給を停止してください。

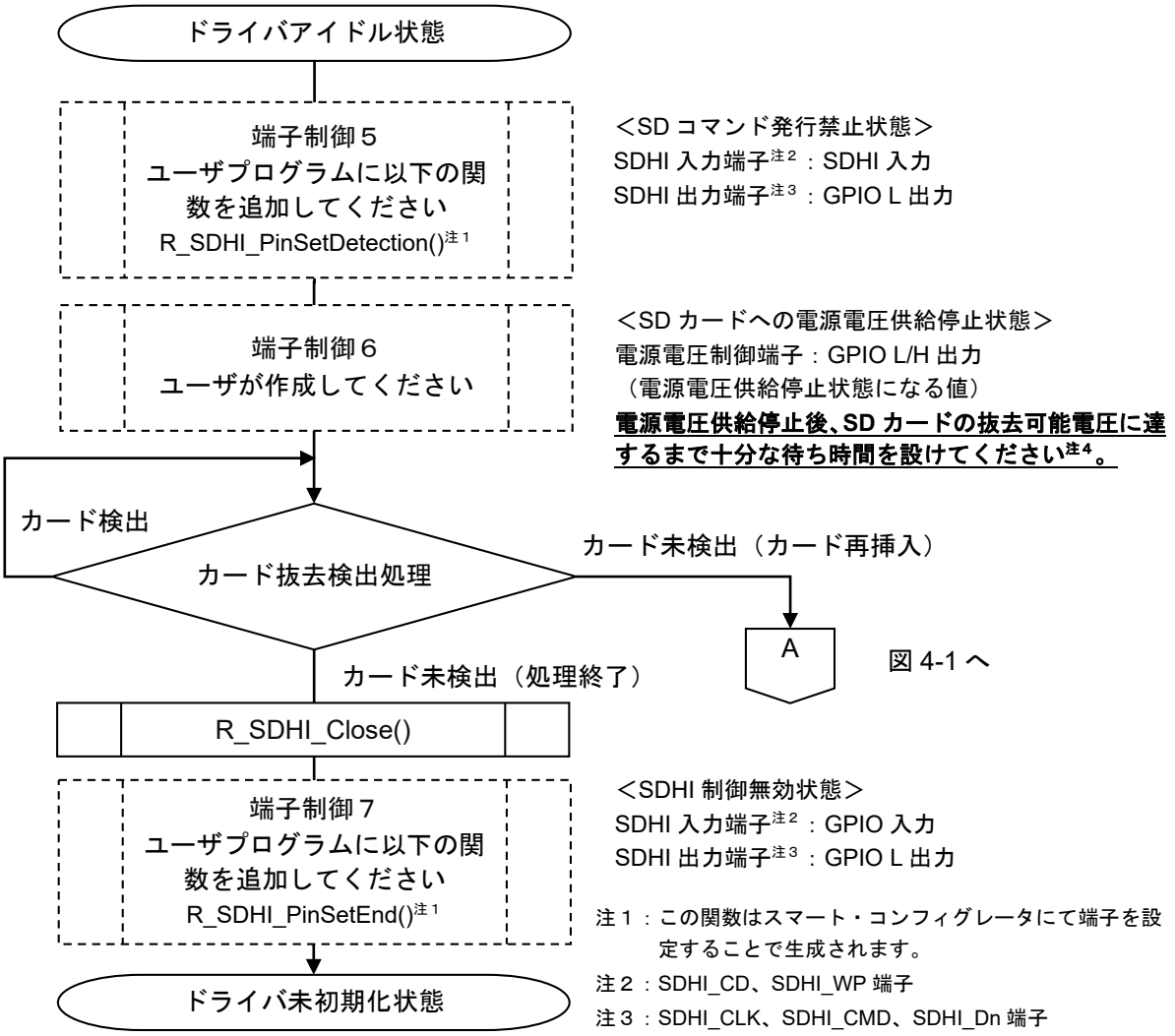


図 4-2　SD カードの抜去と電源停止タイミング

表 4-3　SD カード抜去時のユーザ設定方法

処理	対象端子	端子設定	実行後の端子状態
端子制御 5	SDHI 入力端子　注 1	MPC 設定：SDHI PMR 設定：周辺モジュール	SDHI 入力
	SDHI 出力端子　注 2	PMR 設定：汎用入出力ポート MPC 設定：Hi-z	GPIO L 出力
端子制御 6	電源電圧制御端子	PODR 設定：L 出力／H 出力（電源電圧供給停止状態になる値を出力）	GPIO L/H 出力 （電源電圧供給停止状態）
端子制御 7	SDHI 入力端子　注 1	PMR 設定：汎用入出力ポート MPC 設定：Hi-z	GPIO 入力
	SDHI 出力端子　注 2	PMR 設定：汎用入出力ポート MPC 設定：Hi-z	GPIO L 出力

注 1：SDHI_CD、SDHI_WP 端子

注 2：SDHI_CLK、SDHI_CMD、SDHI_Dn 端子

5. デモプロジェクト

5.1 概要

FITDemos にサンプルプログラムを同梱しています。本サンプルプログラムでは、「4.1 SD カードの挿入と電源投入タイミング」、「4.2 SD カードの抜去と電源停止タイミング」、SD カードへの読み出し／書き込みの処理を行います。

5.2 状態遷移図

図 5-1 に状態遷移図を示します。

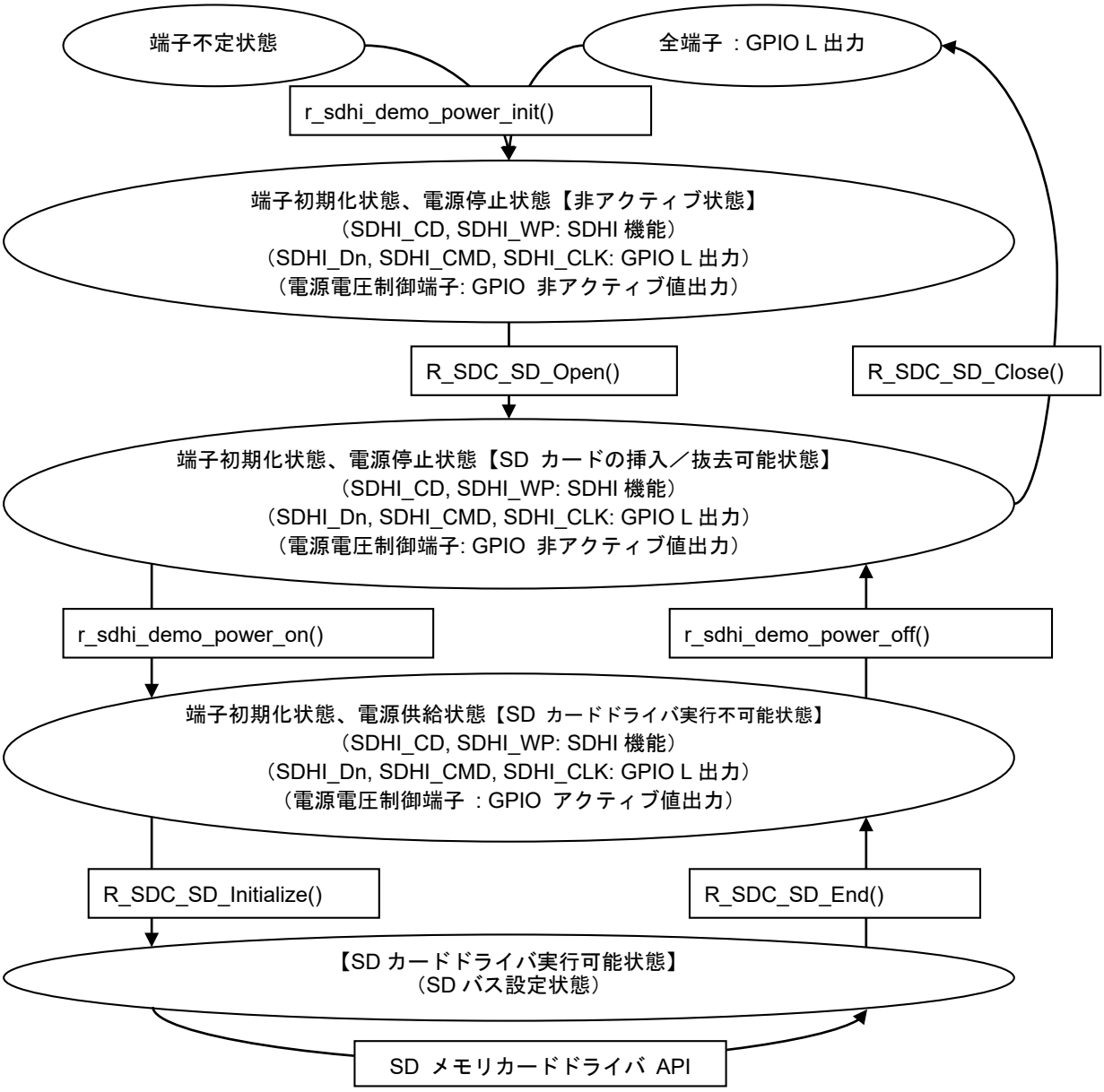


図 5-1　状態遷移図

5.3　コンパイル時の設定

サンプルプログラムのコンフィグレーションオプションの設定は、r_sdhi_rx_demo_pin_config.h で行います。

RX64M RSK、RX65N-2M RSK または RX72N を使用する場合の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_sdhi_rx_demo_pin_config.h*	
#define SDHI_CFG_POWER_PORT_NONE ※デフォルト値は“無効”	SD カードを使用する場合の定義です。 SD カード電源制御が不要な場合、定義を有効にしてください。 SD カード電源制御が必要な場合、定義を無効にしてください。
#define SDHI_CFG_POWER_HIGH_ACTIVE (1) ※ デフォルト値は“1（High を供給）”	SD カードを使用し、かつ、SD カード電源制御が必要な場合に設定する定義です。 “1”の場合、SD カード電源回路を有効にするために、SD カード電源回路を制御しているポートに High を供給します。 “0”の場合、SD カード電源回路を有効にするために、SD カード電源回路を制御しているポートに Low を供給します
#define SDHI_CFG_POWER_ON_WAIT (100) ※デフォルト値は“100（100ms ウェイト）”	SD カードを使用する場合の定義です。 SD カード用電源回路に電源供給開始後、動作電圧に達するまでのウェイト時間を設定してください。1 カウントあたり、1ms のウェイトを行います。 システムに合わせて設定してください。
#define SDHI_CFG_POWER_OFF_WAIT (100) ※デフォルト値は“100（100ms ウェイト）”	SD カードを使用する場合の定義です。 SD カード用電源回路に電源供給停止後、SD カードの抜去可能電圧に達するまでのウェイト時間を設定してください。1 カウントあたり、1ms のウェイトを行います。 システムに合わせて設定してください。
#define R_SDHI_CFG_POWER_CARDx_PORT ※CARDx の“x”は SD カード番号（x=0）	SD カード番号 x 用の電源電圧制御端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「'」「'」をつけてください。
#define R_SDHI_CFG_POWER_CARDx_BIT ※CARDx の“x”は SD カード番号（x=0）	SD カード番号 x 用の電源電圧制御端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「'」「'」をつけてください。

5.4 API 関数

サンプルプログラム内 API 関数を以下に示します。必要に応じて、関数の追加／修正してください。

表 5-1 API 関数一覧

関数名	機能概要
r_sdhi_demo_power_init()	電源電圧制御端子設定の初期化処理
r_sdhi_demo_power_on()	電源電圧の供給開始処理
r_sdhi_demo_power_off()	電源電圧の供給停止処理
r_sdhi_demo_softwaredelay()	時間待ち処理

(1) r_sdhi_demo_power_init()

SD メモリカードドライバで使用する SD カードの電源電圧制御端子の設定を初期化する関数です。

Format

```
sdhi_status_t r_sdhi_demo_power_init(
    uint32_t card_no
)
```

Parameters

card_no

SD カード番号	使用する SD カード番号（0 起算）

Return Values

SDHI_SUCCESS	*正常終了*

Description

SD カードの電源電圧制御端子の設定を初期化します。

Special Notes

電源電圧制御端子について、以下のとおり設定します。
・ポートモードレジスタ（PMR）を汎用入出力ポートに設定します。
・プルアップ制御レジスタ（PCR）を入力プルアップ抵抗無効に設定します。
・端子出力を非アクティブ状態に設定します。

(2) r_sdhi_demo_power_on()

SD カードの電源電圧制御端子を制御し、電源供給を開始する関数です。

Format

```
sdhi_status_t r_sdhi_demo_power_on(
    uint32_t card_no
)
```

Parameters

card_no

SD カード番号	使用する SD カード番号（0 起算）

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Description

SD カードの電源電圧制御端子を制御し、電源供給を開始します。その後、r_sdhi_rx_demo_pin_config.h の SDHI_CFG_POWER_ON_WAIT で設定された時間経過後に結果を返します。

Special Notes

必要に応じて修正してください。
電源電圧供給開始後、動作電圧に達するまでの時間待ちのため、r_sdhi_demo_softwaredelay()関数を実行します。待ち時間は「5.3 コンパイル時の設定」の「SDHI_CFG_POWER_ON_WAIT」で設定してください。
本関数実行前に r_sdhi_demo_power_init()関数による初期化処理が必要です。

(3) r_sdhi_demo_power_off()

SD カードの電源電圧制御端子を制御し、電源供給を停止する関数です。

Format

```
sdhi_status_t r_sdhi_demo_power_off(
    uint32_t card_no
)
```

Parameters

card_no

SD カード番号	使用する SD カード番号（0 起算）

Return Values

SDHI_SUCCESS	*正常終了*
SDHI_ERR	*一般エラー*

Description

SD カードの電源電圧制御端子を制御し、電源供給を停止します。その後、r_sdhi_rx_demo_pin_config.h の SDHI_CFG_POWER_OFF_WAIT で設定された時間経過後に結果を返します。

Special Notes

電源電圧供給停止後、抜去可能電圧に達する動作電圧に達するまでの時間待ちのため、r_sdhi_demo_softwaredelay()関数を実行します。待ち時間は「5.3 コンパイル時の設定」の「SDHI_CFG_POWER_OFF_WAIT」で設定してください。
本関数実行前に r_sdhi_demo_power_init()関数による初期化処理が必要です。

(4) r_sdhi_demo_softwaredelay()

時間待ちを行う際に使用する関数です。

Format

```
bool r_sdhi_demo_softwaredelay(
    uint32_t delay,
    sdhi_delay_units_t units
)
```

Parameters

delay

タイムアウト時間（単位：units で設定）

units

マイクロ秒：SDHI_DELAY_MICROSECS
ミリ秒：SDHI_DELAY_MILLISECS
秒：SDHI_DELAY_SECS

Return Values

true	*正常終了*
false	*パラメータエラー*

Description

時間待ち処理を行います。
タイムアウト時間 delay になると、true を返します。

Special Notes

表 5-2 に時間待ち処理を示します。本関数は、設定時間を待つ機能のみのため、OS の自タスク遅延処理（例：μITRON の dly_tsk()）等に置き換えることが可能です。

表 5-2　時間待ち処理

分類	内容　< >中の値は提供時の設定値を示す
SD カード電源 On 時の電圧安定待ち時間	SD カード用電源回路に電源供給開始後、動作電圧に達するまでの待ち時間＜100ms＞ ※待ち時間は SDHI_CFG_POWER_ON_WAIT で変更可能。
SD カード電源 Off 時の電圧安定待ち時	SD カード用電源回路に電源供給停止後、 SD カードの抜去可能電圧に達するまでの待ち時間＜100ms＞ ※待ち時間は SDHI_CFG_POWER_OFF_WAIT で変更可能。

5.5　待ち処理の OS 処理への置き換え方法

サンプルプログラムで発生する時間待ち処理 r_sdhi_demo_softwaredelay()を OS の自タスク遅延処理（例：μITRON の dly_tsk()）に置き換えることができます。

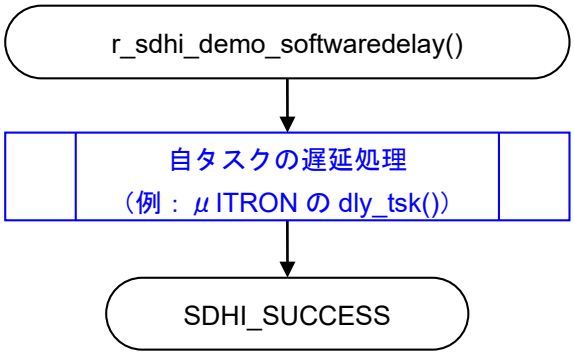


図 5-2　OS の自タスク遅延処理を使った時間待ち例

5.6 sdhi_demo_rskrx64m, sdhi_demo_rskrx65n_2m, sdhi_demo_rskrx72n, sdhi_demo_rskrx64m_gcc, sdhi_demo_rskrx65n_2m_gcc, sdhi_demo_rskrx72n_gcc

コードをコンパイルし、ターゲットボードにダウンロードし、実行すると、初期化後に LED0 が点灯します。SDHI モジュールが正しくオープンされると、LED1 が点灯します。SD カードにデータが正しく書き込まれると、LED2 が点灯します。SD カードからデータが正しく読みだされると、LED3 が点灯します。SDHI モジュールが正しくクローズされると、全ての LED が消灯します。

セットアップと実行

1. r_sdhi_rx_config.h でチャネル 0 のドライバサポートを有効にします。

```
#define SDHI_CFG_CH0_INCLUDED (1)
```

2. データ転送モジュールの選択

DMAC 転送モードを使用する場合、Smart Configurator/Components/r_sdc_sdmem_rx のデータ転送タイプを DMAC 転送に設定してください。

DTC 転送モードを使用する場合、Smart Configurator/Components/r_sdc_sdmem_rx のデータ転送タイプを DTC 転送に設定してください。

デフォルトでは、転送モードはソフトウェア転送になっています。

3. RSK ボードを PC に接続します（Renesas E1 エミュレータを使用）。外部電源として DC 5V 3A の電源アダプタを RSK ボードの電源ジャック(PWR)に接続する必要があります。本サンプルアプリケーションをビルドし、ボードにダウンロードします。

4. ルネサス e2 studio IDE の Renesas Views tab をクリック -> デバッグの項目内にある Renesas Debug Virtual Console を選択してください。

5. ログと LED をチェックすることで、SD カードへの 3 セクタ(512 バイト/セクタ)の書き込みと読み出しを確認してください。

サポートされるボード

RSKRX64M, RSKRX65N-2M, RSKRX72N

5.7 ワークスペースにデモを追加する

デモプロジェクトは、本アプリケーションノートで提供されるファイルの FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」 >> 「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「終了」をクリックします。

5.8 デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 付録

6.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 6-1 動作確認環境 (Ver.2.02)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.1.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.00.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	リトルエンディアン
モジュールのバージョン	Ver.2.02
使用ボード	Renesas Starter Kit for RX65N（型名：RTK500565NSxxxxxxx）

表 6-2 動作確認環境 (Ver.2.03)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	リトルエンディアン
モジュールのバージョン	Ver.2.03

表 6-3 動作確認環境 (Ver.2.04)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201803 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.2.04
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565Nxxxxxx）

表 6-4　動作確認環境 (Ver.2.05)

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 V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.2.05
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6-5　動作確認環境 (Ver.2.06)

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 V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.2.06
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxxx）

表 6-6　動作確認環境 (Ver.2.07)

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 2021-01 (21.1.0) IAR Embedded Workbench for Renesas RX 4.14.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.03.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.03.00.202002 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.14.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.2.07
使用ボード	Renesas Starter Kit+ for RX671 (型名：RTK55671xxxxxxxxxx)

表 6-7　動作確認環境 (Rev.2. 10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2022-10 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.04.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0.202202 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、統合開発環境のデフォルト設定に以下のオプションを追加 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.10
使用ボード	Renesas Starter Kit+ for RX64M (型名：R0K50564Mxxxxxx) Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565N2Cxxxxxxx) Renesas Starter Kit+ for RX72N (型名：RTK5572NNDCxxxxxx)

6.2 トラブルシューティング

(1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_sdhi_rx module.」エラーが発生します。

A : 追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

6.3　RSK の SD カードソケットを使った SD カード評価方法

Renesas Starter Kits（以降、RSK とする）を用いて、SD カードへの読み出し／書き込み処理を行う手順を以下に示します。

6.3.1　ハードウェア設定

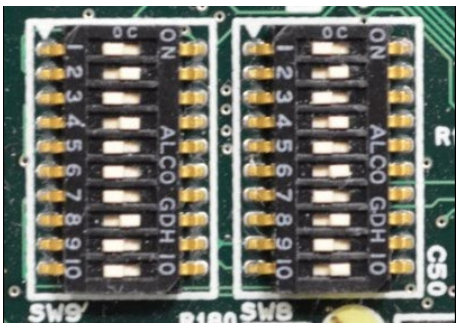
ターゲット MCU の RSK 毎に設定が必要です。

なお、RSK for RX231 の場合、PMOD SD Card 変換基板を別途入手してください。

(a)　RSK for RX64M／RX71M

SD カードソケットを有効にするため、以下のとおり設定してください。

SW9		SW8	
ピン番号	設定	ピン番号	設定
Pin 1	OFF	Pin 1	OFF
Pin 2	ON	Pin 2	ON
Pin 3	OFF	Pin 3	OFF
Pin 4	ON	Pin 4	ON
Pin 5	OFF	Pin 5	OFF
Pin 6	OFF	Pin 6	ON
Pin 7	OFF	Pin 7	OFF
Pin 8	ON	Pin 8	ON
Pin 9	OFF	Pin 9	OFF
Pin 10	OFF	Pin 10	ON



(b)　RSK for RX65N

SD カードソケットを有効にするため、以下のとおり設定してください。

SW7		SW8	
ピン番号	設定	ピン番号	設定
Pin 1	OFF	Pin 1	OFF
Pin 2	ON	Pin 2	ON
Pin 3	OFF	Pin 3	OFF
Pin 4	ON	Pin 4	ON
Pin 5	OFF	Pin 5	OFF
Pin 6	ON	Pin 6	ON
Pin 7	OFF	Pin 7	OFF
Pin 8	ON	Pin 8	ON
Pin 9	OFF	Pin 9	OFF
Pin 10	ON	Pin 10	OFF

(c) RSK for RX65N-2MB

設定不要です。

(d) RSK for RX231

PMOD SD Card 変換基板を RSK for RX231 上の PMOD2 に装着してください。

(e) RSK for RX72M

設定不要です。

(f) RSK for RX72N

設定不要です。

(g) RSK for RX671

設定不要です。

7. 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
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（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。）

8. テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- TN-RX*-A195A/J
- TN-RX*-A196A/J
- TN-RX*-A197A/J

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
2.00	2017.07.31	-	初版発行
2.01	2017.12.31	43-44	4.1 SD カードの挿入と電源投入タイミング　図 4-1、表 4-1 において、内容を更新した。
		45	4.2 SD カードの抜去と電源停止タイミング　図 4-2、表 4-2 において、内容を更新した。
		46	5.1 動作確認環境　表 5-1　において、以下の内容を更新した。 ・統合開発環境のバージョン ・モジュールのバージョン
2.02	2018.11.30	-	xml ファイル更新のため、バージョンアップ 7.テクニカルアップデートの対応について　更新
2.03	2019.02.01	48	5.1 動作確認環境　に、表 5 2 動作確認環境 (Ver.2.03)　を追加
		-	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 ■内容 GUI によるコンフィグオプション設定機能に対応するため、設定ファイルを追加。
2.04	2019.05.20	—	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	関連ドキュメント R01AN1723 と R01AN1826 を削除 SD モード SD メモリカードドライバ・ソフトウェア RTM0RX0000DSDD0 (R01UW0135)を削除 SD モード SD メモリカードドライバ Firmware Integration Technology (R01AN4233)を追加
		1	「対象コンパイラ」を追加
		8	「2.2 ソフトウェアの要求」 依存する r_bsp モジュールのリビジョンを追加
		11	「2.8 コードサイズ」を更新
		48	5.1 動作確認環境　に、表 5.3 動作確認環境 (Ver.2.04)　を追加
2.05	2019.07.30	-	RX23W、RX72M に関連した変更
		1	関連ドキュメント 更新
		8	「2.4 仕様する割り込みベクタ」に、RX23W と RX72M の割り込みベクタ情報を追加
		11	「2.8 コードサイズ」を更新
		14	2.13「for 文、while 文、do while 文について」を追加
		15-44	API 説明ページの「Reentrant」項目を削除
		51	5.1 動作確認環境　に、表 5.4 動作確認環境 (Ver.2.05)　を追加
		53	RX72M のハードウェア設定を追加
2.06	2019.11.22	-	RX66N、RX72N に関連した変更
		1	「対象コンパイラ」を追加

Rev.	発行日	改訂内容	
		ページ	ポイント
2.06	2019.11.22	8	「2.4 仕様する割り込みベクタ」に、RX66N と RX72N の割り込みベクタ情報を追加
		11	「2.8 コードサイズ」を更新
		43-44	「3.26 R_SDHI_SetLogHdlAddress」と「3.27 R_SDHI_Log」の「Special Notes」を修正
		51	5.1 動作確認環境　に、表 5.5 動作確認環境 (Ver.2.06)　を追加
		54	RX72N のハードウェア設定を追加
2.07	2021.06.30	-	RX671 に関連した変更
		1	「対象コンパイラ」を追加
		8	「2.4 仕様する割り込みベクタ」に、RX671 の割り込みベクタ情報を追加
		11	「2.8 コードサイズ」 を更新
		13	「2.12 FIT モジュールの追加方法」を更新
		51	5.1 動作確認環境 に、 表 5.6 動作確認環境 (Ver.2.07) を追加
		54	RX671 のハードウェア設定を追加
2.10	2022.12.27	50	「5. デモプロジェクト」を追加。
		50	「5. デモプロジェクト」に RSK RX64M、RSK RX65N-2M、RSKRX72N を追加。
		60	「6.1 動作確認環境」： Rev.2.10 に対応する表を追加。
		プログラム	デモプロジェクトの追加。 Linux 対応のため、インクルードヘッダファイルパスの形式を更新しました。

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。