RX ファミリ

SDSI モジュール Firmware Integration Technology

要旨

本アプリケーションノートでは、RX ファミリ SD スレーブインタフェース（以下、SDSI と略す）の制御モジュールとその使用方法を説明します。本モジュールは、Firmware Integration Technology （以下、FIT と略す）を使った SD スレーブ制御モジュールです。以降、本モジュールを SDSI FIT モジュールと称します。また、同様に FIT を使った他の機能制御モジュールを FIT モジュールもしくは“機能名”FIT モジュールと表します。

動作確認デバイス

RX65N グループ、RX651 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合わせて変更し、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。

FIT 関連ドキュメント

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- 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 SDSI 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 SDSI FIT モジュールの概要

RX ファミリマイコン内蔵の SDSI を使用し、SD スレーブ制御を行います。
表 1-1 に使用する周辺機器と用途を、に使用例を示します。

以下に、機能概略を示します。

- マスタデバイスを RX ファミリマイコンとする SDSI を使った SD スレーブ制御デバイスドライバ
- ハイスピードモード、デフォルトスピードモードに対応
- ブロック転送モード、バイト転送モードに対応
- ワイドバスモード（4-bit）、デフォルトバスモード（1-bit）から選択可能
- SD モード対応、SPI モードサポート対象外
- CCCR（Card Common Control Register）をベースにしたオペレーションに対応
- FBR（Function Basic Register）をベースにしたオペレーションに対応
- CIS（Card Information Structure）108 バイトへのアクセスに対応
- Function1 領域（Function Unique register space）へのアクセスに対応
- 専用 DMA バスを使用し、マイコン内蔵 RAM へのダイレクト転送に対応
- SDSI 割り込み検出時、コールバック関数の呼び出しが可能
- ユーザ設定した 1 チャネルもしくは複数チャネルの制御が可能
- 異なるチャネルからリエントラントが可能
- ビッグエンディアン／リトルエンディアンでの動作が可能

表 1.1 使用する周辺機器と用途

周辺機器	用途
SDSI	1 もしくは複数チャネル（必須）

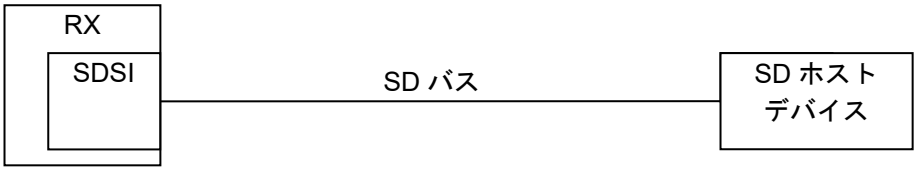


図 1-1　使用例

1.3 API の概要

表 1-2 に SDSI FIT モジュールに含まれる API 関数を示します。

表 1.2 API 関数

関数名	説明
R_SDSI_Open()	ドライバのオープン処理
R_SDSI_Close()	ドライバのクローズ処理
R_SDSI_Initialize()	初期化処理
R_SDSI_End()	終了処理
R_SDSI_CflagPolling()	C フラグポーリング処理
R_SDSI_WriteCisReg()	CIS データレジスタライト処理
R_SDSI_ReadCisReg()	CIS データレジスタリード処理
R_SDSI_WriteFuncReg()	FN1 データレジスタ n 書き込み処理（n=1,2,5）
R_SDSI_ReadFuncReg()	FN1 データレジスタ m 読み出し処理（m=1,3,5）
R_SDSI_WriteIntVectorReg()	FN1 割り込みベクタレジスタ書き込み処理
R_SDSI_ReadIntVectorReg()	FN1 割り込みベクタレジスタ読み出し処理
R_SDSI_ReadIntVectorClearReg()	FN1 割り込みクリアレジスタ読み出し処理
R_SDSI_EnableDirectTrans()	DMA 転送許可処理
R_SDSI_DisableDirectTrans()	DMA 転送禁止処理
R_SDSI_SetDirectTransAdr()	DMA 転送開始アドレス設定処理
R_SDSI_GetDirectTransAdr()	DMA 転送開始アドレス取得処理
R_SDSI_RegistIntCallback()	SDSI コマンド割り込みコールバック関数登録処理
R_SDSI_RegistCdIntCallback()	SDSI カード検出無効（Rise／Fall）割り込みコールバック関数登録処理
R_SDSI_RegistDtIntCallback()	SDSI DMA 転送終了割り込みコールバック関数登録処理
R_SDSI_GetVersion()	ドライバのバージョン情報取得処理
R_SDSI_IntHandler0()	割り込みハンドラ
R_SDSI_SetLogHdlAddress()	LONGQ モジュールのハンドラアドレス設定処理
R_SDSI_Log()	エラーログ取得処理

1.4　処理例

1.4.1　ハードウェア説明

図 1-2 に接続例を示します。マイコン内蔵 SDSI を使って、1-bit/4-bit バスの SD モード制御を行います。接続可能な SD ホストは、1 台／チャネルです。

SDIO モジュールの仕様書を参照し、システムに合った回路を検討してください。

プルアップ抵抗値は、SD Specifications Part 1 Physical Layer Specification を参照して決めてください。また、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。ただし、SDSI_CLK 端子のプルアップ処理は、SD Specifications Part 1 Physical Layer Specification に規定が無いため、記載していません。

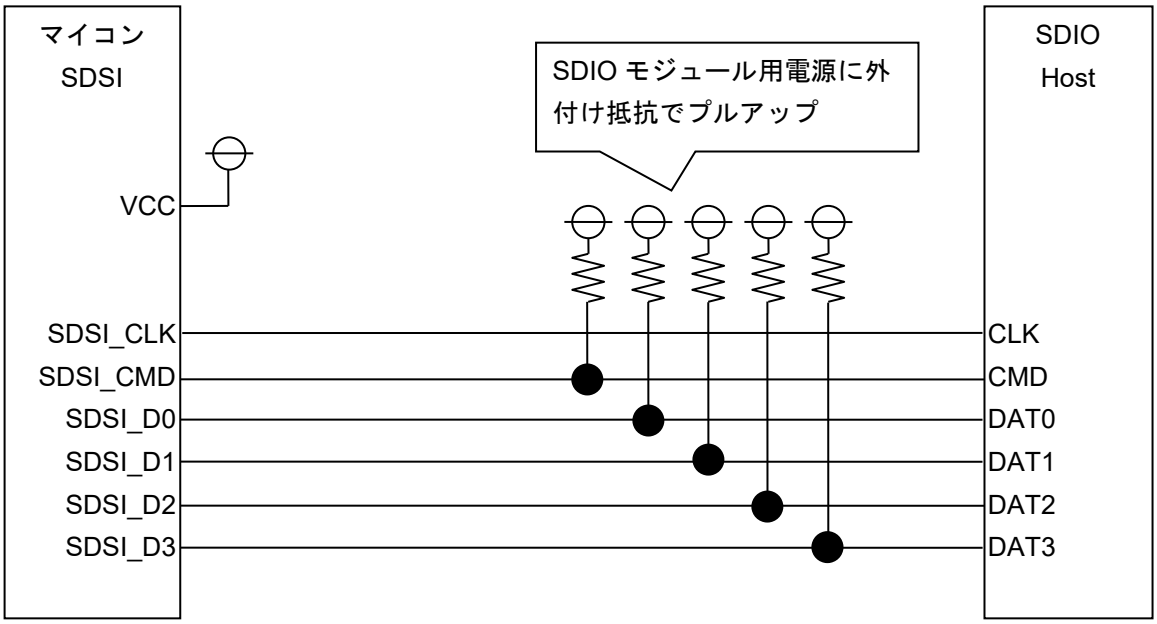


図 1-2　マイコンと SDIO モジュールの接続例

(1)　使用端子一覧

表 1-3 に、使用端子と機能を示します。

表 1.3 使用端子と機能

端子名	入出力	内容
SDSI_CLK	入力	SDSI クロック
SDSI_CMD	入出力	コマンドの入力、レスポンスの出力
SDSI_D3～SDSI_D0	入出力	SDSI データ

1.4.2　ソフトウェア説明

(1)　ソフトウェア構成

図 1-3 にソフトウェア構成を示します。

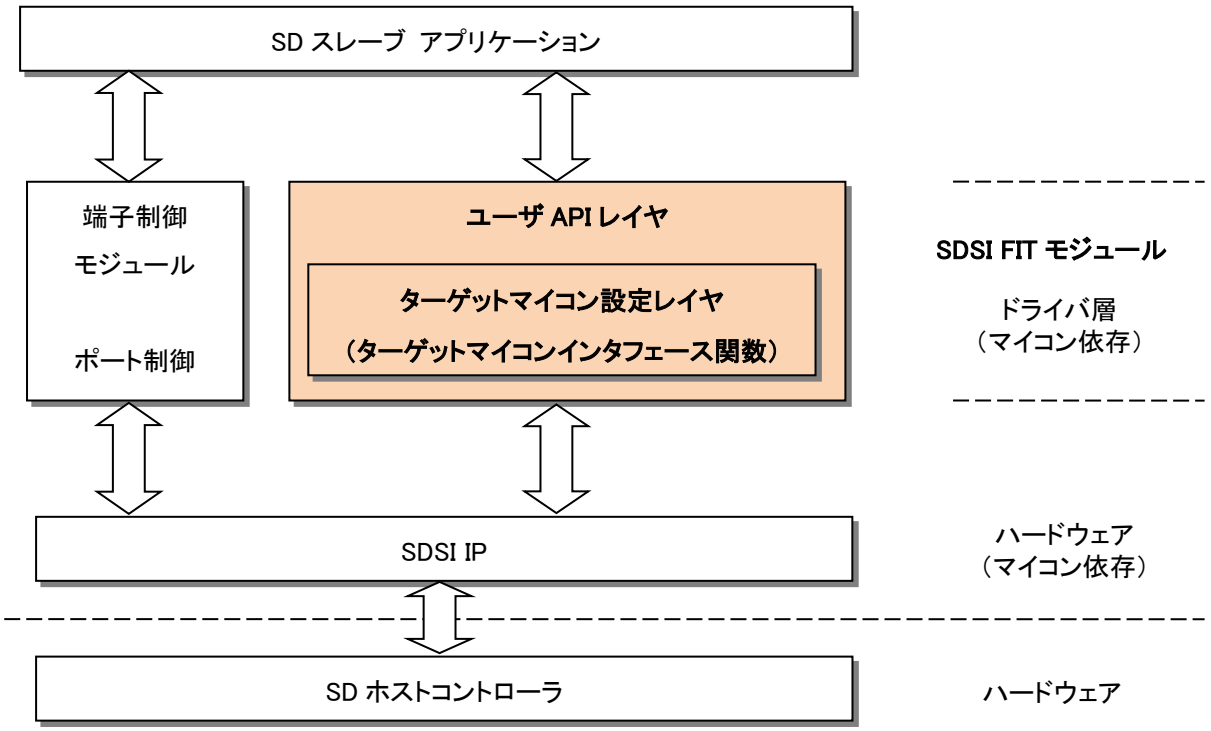


図 1-3 ソフトウェア構成

(a)　ユーザ API レイヤ（r_sdsi_rx.c）

SDSI FIT モジュールのユーザ API で、マイコンや SDSI 仕様に依存しない部分です。

(b)　ターゲットマイコン設定レイヤ（r_sdsi_dev.c）

SDSI レジスタへの読み出し、書き込みを行います。マイコンや SDSI 仕様に依存する部分です。
マイコン毎に処理を見直す必要があります。

(c)　SD スレーブ アプリケーション（r_sdsi_pxpd_rx.c）

SDSI FIT モジュールの制御例を同梱していますので、参照してください。

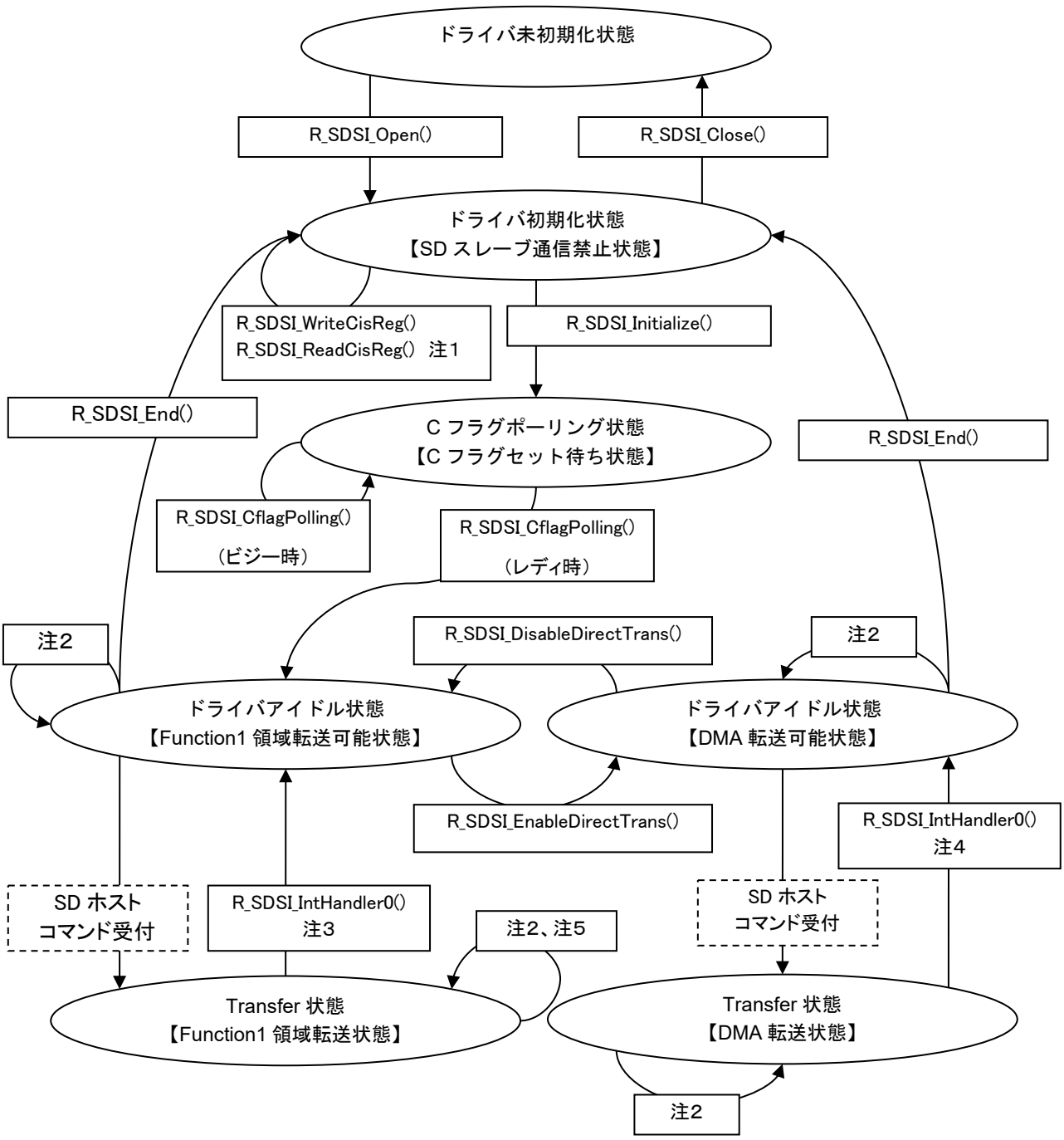
(2) CIS データレジスタ CISDATAR への 4 バイトアクセスとデータ配置

SDSI FIT モジュールは SDSI IP 仕様に基づき、CIS データレジスタ CISDATAR へ**4 バイトで**アクセスします。この際、RAM アドレスの先頭に配置されたデータから順番に、CIS データレジスタ CISDATAR の先頭アドレスへ書き込み／読み出しを行います。そのため、**4 バイトデータのデータ配置がエンディアンにより異なりますので、注意してください。**

```
typedef union
{
      uint32_t l;
      uint8_t c[4];
} sdsi_union_t;

sdsi_reg_t sdsi_reg;
sdsi_union_t io_buff = { 0 };

io_buff.c[0] = 0x01; /* RAM アドレス 0（先頭） */
io_buff.c[1] = 0x02; /* RAM アドレス 1 */
io_buff.c[2] = 0x03; /* RAM アドレス 2 */
io_buff.c[3] = 0x04; /* RAM アドレス 3 */

/*
   iobuff.l の値はエンディアンにより異なる。
   リトルエンディアンの場合：io_buff.l = 0x04030201
   ビッグエンディアンの場合：io_buff.l = 0x01020304
*/

sdsi_reg.offset = 0x0;
sdsi_reg.p_buff = &io_buff.l;
if (R_SDSI_WriteCisReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Error */
}
```

1.5　状態遷移図

図 1-4 に状態遷移図を示します。



注1：R_SDSI_WriteCisReg()はドライバ初期化状態のみコール可。
注2：R_SDSI_WriteCisReg()以外の R_SDSI_xxxxReg()のコールが可能。
注3：Function1 に対する CMD52_W or CMD53_W or CMD53_R 割り込み、カード検出無効（Rise/Fall）のいずれかの割り込み検出時。
注4：DMA 転送終了割り込み検出時。
注5：FN1 Data Register5 は SD ホスト/CPU 両側から同時アクセス禁止。

図 1-4 状態遷移図

2. API 情報

本制御ソフトウェアの API は、ルネサスの API 命名基準に従っています。

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。

- SDSI

2.2 ソフトウェアの要求

このドライバは以下の 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.5.00 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「6.1 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

R_SDSI_Initialize()関数を実行するとチャネルに対応した SDSI 割り込み [1]が有効になります。

「表 2-1 使用する割り込みベクター覧」に SDSI 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
RX65N	GROUPBL2 割り込み（ベクタ番号：107） • SDSI 割り込み[チャネル 0]（グループ割り込み要因番号：0）

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_sdsi_rx_if.h に記載しています。

ビルド毎の構成オプションは、r_sdsi_rx_config.h で選択します。以下の順番でインクルードしてください。

```
#include "r_sdsi_rx_if.h"
```

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。

[1] CMD53 リードコマンド割り込み、CMD53 ライトコマンド割り込み、CMD52 ライトコマンド割り込み、DMA 転送終了割り込み、カード検出無効（Rise/Fall）割り込み

2.7　コンパイル時の設定

本制御ソフトウェアのコンフィギュレーションオプションの設定は、r_sdsi_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_sdsi_rx_config.h*	
#define SDSI_CFG_USE_FIT ※デフォルト値は“有効”	SDSI FIT モジュールの BSP 環境での使用有無を選択できます。 無効にした場合、r_bsp 等の FIT モジュール制御を無効します。また、別途、処理を組み込む必要があります。 有効にした場合、r_bsp 等の FIT モジュール制御を有効にします。
#define SDSI_CFG_PARAM_CHECKING_ENABLE ※：デフォルトは “BSP_CFG_PARAM_CHECKING_ENABLE”	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 デフォルト設定は BSP_CFG_PARAM_CHECKING_ENABLE です。SDSI FIT モジュールのみチェック機能を有効にする場合は、本定義を”1”に、チェックを行わない場合は”0”に設定してください。
#define SDSI_CFG_CHx_INCLUDED ※チャネル 0 のデフォルト値は“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できます。 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。
#define SDSI_CFG_LONGQ_ENABLE ※デフォルトは“無効”	デバッグ用のエラーログ取得処理を使用するか選択できます。 無効にした場合、処理をコードから省略します。 有効にした場合、処理をコードに含めます。 使用するためには、別途 LONGQ FIT モジュールが必要です。
#define SDSI_CFG_CHx_INT_LEVEL ※チャネル 0 のデフォルト値は“5” ※“x”はチャネル番号	SDSI 割り込みの優先レベルを設定してください。 なお、マイコンによっては SDSI 割り込みがグループ割り込みに割り当てられている場合があります。その場合、グループ割り込みの優先レベルを定義してください。
#define SDSI_CFG_DISABLE_SYSTEM_INTERRUPT ※デフォルトは無効	R_SDSI_Open()処理中、全プロセッサ割り込みを禁止するか選択します。定義を有効にした場合、ドライバ未初期化状態（通信禁止状態）で発行された SDIO コマンドを検出し、レスポンスを返す可能性を低減します。 無効にした場合：全プロセッサ割り込みを禁止しません。 有効にした場合：全プロセッサ割り込みを禁止します。 なお、割り込みを禁止にする場合、CPU のプロセッサステータスワード（PSW）レジスタの I フラグをクリアします。そのため、**本定義を有効にする場合は CPU をスーパバイザモードに設定してください。ユーザモードの場合、割り込み禁止処理実行により、特権命令例外が発生します。**
#define SDSI_CFG_FBR_ADR_100H ※デフォルト値は“0x00”	FBR 0x100 Function1 Standard SDIO Function interface code を設定してください。なお、b7-b4 に設定した値は無視されます。
#define SDSI_CFG_FBR_ADR_101H ※デフォルト値は“0x01”	FBR 0x101 Function1 Extended standard SDIO Function interface code を設定してください。
#define SDSI_CFG_FBR_ADR_102H	FBR 0x102 は設定禁止です。設定した場合、設定値は無視されます。FBR 0x102 の SPS ビットの設定は、マクロ定義 SDSI_CFG_FBR_SPS_BIT にて行ってください。

Configuration options in *r_sdsi_rx_config.h*	
#define SDSI_CFG_FBR_ADR_103H ※デフォルト値は“0x00”	FBR 0x103 Function1 Standard iSDIO Function Interface Code を設定してください。
#define SDSI_CFG_FBR_ADR_104H ※デフォルト値は“0x00”	FBR 0x104 Function1 MID_MANF SDIO Card Manufacturer Code を設定してください。
#define SDSI_CFG_FBR_ADR_105H ※デフォルト値は“0x00”	FBR 0x105 Function1 MID_MANF SDIO Card Manufacturer Code を設定してください。
#define SDSI_CFG_FBR_ADR_106H ※デフォルト値は“0x00”	FBR 0x106 Function1 MID_CARD Manufacturer Information を設定してください。
#define SDSI_CFG_FBR_ADR_107H ※デフォルト値は“0x00”	FBR 0x107 Function1 MID_CARD Manufacturer Information を設定してください。
#define SDSI_CFG_FBR_ADR_108H ※デフォルト値は“0x00”	FBR 0x108 I/O block size for Function1 を設定してください。
#define SDSI_CFG_FBR_SPS_BIT ※デフォルト値は“0”	FBR 0x102 SPS ビットの値を設定してください（0 or 1）。
#define SDSI_CFG_CCCR_SMPC_BIT ※デフォルト値は“0”	CCCR 0x012 SMPC ビットの値を設定してください（0 or 1）。

2.8　コードサイズ

表 2-2 に最新バージョンのモジュールを使用した場合のコードサイズを示します。

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_sdsi_rx rev2.02

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(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 4.08.04.201803
(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 4.10.1
(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、注３）							
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
RX65N	ROM	**2349 バイト**	**1830 バイト**	**4692 バイト**	**3796 バイト**	**3864 バイト**	**3095 バイト**
	RAM	**8 バイト**		**4 バイト**		**12 バイト**	
	最大使用ユーザスタック	**56 バイト**		-		**108 バイト**	
	最大使用割り込みスタック	**0 バイト**		-		**68 バイト**	

注１：動作条件は以下のとおりです。
- r_sdsi_rx.c
- r_sdsi_dev.c
- r_sdsi_register.c

注２：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３：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_sdsi_rx_if.h で記載されています。

```
typedef struct
{
    uint32_t    reg_no;
    uint32_t    offset;
    uint32_t    * p_buff;
} sdsi_reg_t;

typedef struct
{
    uint32_t    adr;
    uint32_t    mode;
} sdsi_direct_trans_t;

typedef struct
{
    uint32_t    adr;
    uint16_t    blkcnt;
    uint16_t    bytcnt;
    uint8_t     sdcmdcr;
    uint8_t     cmd;
    uint8_t     rsv[2];
} sdsi_cmd_t;
```

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_sdsi_rx_if.h で記載されています。

```
typedef enum e_sdsi_status
{
    SDSI_SUCCESS                    =   0,
    SDSI_ERR                        =   -1,
    SDSI_ERR_BUSY                   =   -2,
    SDSI_ERR_ADDRESS_BOUNDARY       =   -3
} sdsi_status_t;
```

2.11 コールバック関数

表 2-3 に SDSI FIT モジュールのコールバック関数を示します。

表 2.3 コールバック関数

コールバック関数を登録する関数	コールバック関数が呼び出されるタイミング
R_SDSI_RegistIntCallback()	SDSI コマンド割り込み [2]検出時
R_SDSI_RegistCdCallback()	カード検出無効（Rise／Fall）割り込み検出時
R_SDSI_RegistDtCallback()	DMA 転送終了割り込み検出時

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、Smart Configurator を使用した(1)、(2)の追加方法を推奨しています。ただし、Smart Configurator 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(3)の方法を使用してください。

(1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) CS+上で Smart Configurator 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 Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(3) 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。

[2] SDSI コマンド割り込みとは SDSI の「CMD53 リードコマンド割り込み」「CMD53 ライトコマンド割り込み」「CMD52 ライトコマンド割り込み」を指します。

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_SDSI_Open()

SDSI FIT モジュールを初期化する関数です。他の API 関数を使用する前に実行してください。

Format

```
sdsi_status_t    R_SDSI_Open(
    uint32_t channel,
    void * p_sdsi_workarea
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_workarea*

4 バイト境界のワーク領域ポインタ（ワーク領域サイズ：28 バイト確保してください）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*
SDSI_ERR_ADDRESS_BOUNDARY	**p_sdsi_workarea の 4 バイト境界アドレスエラー*

Properties

ファイル r_sdsi_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 SDSI チャネルリソースを取得し、SDSI ドライバと SDSI チャネルを初期化します。また、その SDSI チャネルリソースを占有します。

SDSI ドライバのクローズ処理 R_SDSI_Close()をコールするまで、*p_sdsi_workarea で設定したワーク領域を保持し、その内容を変更しないでください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    g_sdsi_work[28/sizeof(uint32_t)];

if (R_SDSI_Open(SDSI_CH0, &g_sdsi_work[0]) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

SD ホストから SDIO コマンドが発行される前に本関数処理を終了させるため、システム電源投入後、直ちに本関数を実行してください。本関数実行中、SDSI のモジュールストップを解除してから、SDSI のソフトウェアリセット（SDSICR3.SRST）を行うまでの期間、IOR0=1（Function0 は有効/レディ状態）になります。この期間、SD スレーブは SDIO コマンドを検出しレスポンスを返します。#define SDSI_CFG_DISABLE_SYSTEM_INTERRUPT を有効にした場合、モジュールストップからソフトウェアリセットまでの期間は全プロセッサ割り込み要求を禁止するため、レディ期間を最小に抑えることが可能です。

本関数実行前後で端子の状態は変化しません。

本関数が正常終了しない場合、R_SDSI_GetVersion()関数、R_SDSI_Log()関数、R_SDSI_Set_LogHdlAddress()関数以外の API は使用できません。

R_SDSI_Close()

SDSI FIT モジュールのリソースを解放する関数です。

Format

```
sdsi_status_t     R_SDSI_Close(
      uint32_t channel
)
```

Parameters

channel

SDSI チャネル番号

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

SDSI FIT モジュールの全ての処理を終了し、引数 channel で設定した SDSI チャネルのリソースを解放します。その SDSI チャネルをモジュールストップ状態に設定します。

R_SDSI_Open()関数で設定したワーク領域を解放します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
if (R_SDSI_Close(SDSI_CH0) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

また、本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。なお、本関数実行前後で端子の状態は変化しません。

R_SDSI_Initialize()

SDSI IP の初期設定を行います。正常終了後、C フラグポーリング状態に遷移します。

Format

```
sdsi_status_t     R_SDSI_Initialize(
      uint32_t channel
)
```

Parameters

channel

SDSI チャネル番号

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

SDSI IP の初期設定を行います。正常終了後、C フラグポーリング状態に遷移します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_status_t ret = SDSI_SUCCESS;

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

if (R_SDSI_Initialize(SDSI_CH0) != SDSI_SUCCESS)
{
      /* Error */
}

/* ==== C flag polling ==== */
do
{
      ret = R_SDSI_CflagPolling(SDSI_CH0);
      if (SDSI_ERR == ret)
      {
            /* Error */
      }
}
while (SDSI_ERR_BUSY == ret);
```

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_SDSI_Open()関数によるオープン処理が必要です。

以下の設定を行います。

- Function1 の Register1～4 への CPU アクセスを有効にします。
- Function1 の Register5 の SD ホストアクセスを有効にします。
- FBR、FBR.SPS、CCCR.SMPC を設定します。
- SDSI 割り込みを許可します。
- CCCR.IOR1 を“1（レディ）”にします。
- I/O ファンクションレディ 0 ビット（SDSICR3.IOR0）に“1”を設定します。本ビットが“1”の時、SD ホストから発行される CMD5 を受付けると、R4 レスポンスの C フラグに“1”がセットされます。C フラグの状態は R_SDSI_CflagPolling()の戻り値で確認してください。

R_SDSI_End()

SDSI FIT モジュールをアイドル状態から初期化状態にする関数です。

Format

```
sdsi_status_t    R_SDSI_End(
    uint32_t channel
)
```

Parameters

channel

SDSI チャネル番号

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

SDSI の終了処理を行います。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
if (R_SDSI_End(SDSI_CH0) != SDSI_SUCCESS)
{
    /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。なお、本関数実行前後で端子の状態は変化しません。

R_SDSI_CflagPolling()

R4 レスポンスの C フラグ状態を取得する関数です。

R_SDSI_Initialize()関数による初期化処理実行後、本関数をコールして戻り値に SDSI_SUCCESS（C フラグは“1（レディ）”）になることを確認してください。

Format

```
sdsi_status_t    R_SDSI_CflagPolling(
    uint32_t channel
)
```

Parameters

channel

SDSI チャネル番号

Return Values

SDSI_SUCCESS	*C フラグは“1（レディ）”*
SDSI_ERR_BUSY	*C フラグは“0（ビジー）”*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

R4 レスポンスの C フラグ状態を取得します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_status_t ret = SDSI_SUCCESS;

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

if (R_SDSI_Initialize(SDSI_CH0) != SDSI_SUCCESS)
{
      /* Error */
}

/* ==== C flag polling ==== */
do
{
      ret = R_SDSI_CflagPolling(SDSI_CH0);
      if (SDSI_ERR == ret)
      {
            /* Error */
      }
}
while (SDSI_ERR_BUSY == ret);
```

Special Notes

本関数実行前に R_SDSI_Initialize()関数による初期化処理が必要です。

IOR0=1 の時、SD ホストから発行される CMD5 を受付けると、R4 レスポンスの C フラグに“1”がセットされます。

R_SDSI_WriteCisReg()

CIS レジスタへ値を書き込みます。

Format

```
sdsi_status_t    R_SDSI_WriteCisReg(
    uint32_t channel,
    sdsi_reg_t * p_sdsi_reg
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_reg*

reg_no *レジスタ No.（設定不要）*

offset *CIS レジスタオフセット（4 の倍数：0,4,8,12…100,104）*

** p_buff* *書き込みバッファポインタ（4 バイト）*

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

CIS レジスタに値を書き込みます。CIS レジスタには 4 バイトでアクセスします。ドライバ初期化状態のみコール可能です。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
typedef union
{
      uint32_t l;
      uint8_t c[4];
} sdsi_union_t;

sdsi_reg_t sdsi_reg;
sdsi_union_t io_buff = { 0 };

io_buff.c[0] = 0x20;
io_buff.c[1] = 0x04;
io_buff.c[2] = 0x00;
io_buff.c[3] = 0x20;

sdsi_reg.offset = 0;
sdsi_reg.p_buff = &io_buff.l;
if (R_SDSI_WriteCisReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

R_SDSI_ReadCisReg()

CIS レジスタから値を読み出します。

Format

```
sdsi_status_t    R_SDSI_ReadCisReg(
    uint32_t channel,
    sdsi_reg_t * p_sdsi_reg
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_reg*

reg_no	*レジスタ No.（設定不要）*
offset	*CIS レジスタオフセット（4 の倍数：0,4,8,12…100,104）*
** p_buff*	*読み出しバッファポインタ（4 バイト）*

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

CIS レジスタから値を読み出します。CIS レジスタには 4 バイトでアクセスします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
typedef union
{
      uint32_t l;
      uint8_t c[4];
} sdsi_union_t;

sdsi_reg_t sdsi_reg;
sdsi_union_t io_buff = { 0 };

io_buff.l = 0;

sdsi_reg.offset = 0;
sdsi_reg.p_buff = &io_buff.l;
if (R_SDSI_ReadCisReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

R_SDSI_WriteFuncReg()

FN1 データレジスタ n へ値を書き込みます（n=1,2,5）。Function1 領域にデータを書き込むための処理です。

Format

```
sdsi_status_t    R_SDSI_WriteFuncReg(
    uint32_t channel,
    sdsi_reg_t * p_sdsi_reg
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_reg*

reg_no　　*レジスタ No.（1 or 2 or 5）*

offset　　*FN1 データレジスタ n（n=1 or 2 or 5）　オフセット*

＜設定可能な値＞

FN1 データレジスタ 1（Function1 Register1）：0,1,2,3…255

FN1 データレジスタ 2（Function1 Register2）：0,1,2,3…255

FN1 データレジスタ 5（Function1 Register5）：0, 1,2,3…1023

** p_buff*　　*書き込みバッファポインタ（1 バイト）*

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

FN1 データレジスタ n へ値を書き込みます。FN1 データレジスタ n には 1 バイトでアクセスします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_reg_t sdsi_reg;
uint8_t io_buff = 0;

sdsi_reg.reg_no = SDSI_FUNC1_REG1;
sdsi_reg.offset = 0x0;
sdsi_reg.p_buff = &io_buff;
if (R_SDSI_WriteFuncReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

FN1 データレジスタ 5 は、SD ホスト／CPU 両側から同時アクセスすることができません。そのため、以下の手順でアクセスします。

1. FN1 データレジスタ 5 を CPU からアクセス可能状態にする（SDSICR2.REG5EN = 1）。
   SD ホストコントローラからのアクセスは禁止。
2. FN1 データレジスタ 5 にアクセスする。
3. FN1 データレジスタ 5 を SD ホストコントローラからアクセス可能状態にする（SDSICR2.REG5EN = 0）。
   SD ホストコントローラからのアクセスは可能。

上記 2 の期間、SD ホストコントローラから FN1 データレジスタ 5 へアクセスした場合、書き込んだ値は無視され、読み出した値は不定値となります。

R_SDSI_ReadFuncReg()

FN1 データレジスタ m から値を読み出します（m=1,3,5）。Function1 領域からデータを読み出すための処理です。

Format

```
sdsi_status_t    R_SDSI_ReadFuncReg(
    uint32_t channel,
    sdsi_reg_t * p_sdsi_reg
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_reg*

reg_no *レジスタ No.（1 or 3 or 5）*

offset *FN1 データレジスタ m（m=1 or 3 or 5） オフセット*

＜設定可能な値＞

FN1 データレジスタ 1（Function1 Register1）：0,1,2,3…255

FN1 データレジスタ 3（Function1 Register3）：0,1,2,3…255

FN1 データレジスタ 5（Function1 Register5）：0, 1,2,3…1023

** p_buff* *読み出しバッファポインタ（1 バイト）*

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

FN1 データレジスタ m から値を読み出します。FN1 データレジスタ m には 1 バイトでアクセスします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_reg_t sdsi_reg;
uint8_t io_buff = 0

sdsi_reg.reg_no = SDSI_FUNC1_REG1;
sdsi_reg.offset = 0x0;
sdsi_reg.p_buff = &io_buff;
if (R_SDSI_ReadFuncReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

FN1 データレジスタ 5 は、SD ホスト／CPU 両側から同時アクセスすることができません。そのため、以下の手順でアクセスします。

1. FN1 データレジスタ 5 を CPU からアクセス可能状態にする（SDSICR2.REG5EN = 1）。
   SD ホストコントローラからのアクセスは禁止。
2. FN1 データレジスタ 5 に CPU からアクセスする。
3. FN1 データレジスタ 5 を SD ホストコントローラからアクセス可能状態にする（SDSICR2.REG5EN = 0）。
   SD ホストコントローラからのアクセスは可能。

上記 2 の期間、SD ホストコントローラから FN1 データレジスタ 5 へアクセスした場合、書き込んだ値は無視され、読み出した値は不定値となります。

R_SDSI_WriteIntVectorReg()

FN1 割り込みベクタレジスタ（FN1INTVECR）へ値を書き込みます。

Format

```
sdsi_status_t     R_SDSI_WriteIntVectorReg(
        uint32_t channel,
        uint8_t vector
)
```

Parameters

channel

SDSI チャネル番号

vector

FN1 割り込みベクタ値（1 バイト）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

FN1 割り込みベクタレジスタへ値を書き込みます。CCCR の IEN1 が“1”に設定されている場合、SDSI_D1 を使用した SDIO 割り込みが発生します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
if (R_SDSI_WriteIntVectorReg(SDSI_CH0, 0xff) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。SDIO 割り込みが発生した場合、SDSI_D1 は H→L に遷移します。

SDIO 割り込みの発生タイミングは SD バスが 1-bit と 4-bit の場合で異なります。

- 1-bit バス（CCCR 0x07 Bus Width＝00b）

  SD クロックに同期せず、非同期なタイミングで SDIO 割り込みが発生します。

- 4-bit バス（CCCR 0x07 Bus Width＝10b）

  SD クロックに同期して SDIO 割り込みが発生します。SD クロック停止中に本 API を実行しても SDIO 割り込みは発生せず、SD クロックが供給されたタイミングで SDIO 割り込みが発生します。

  非同期なタイミングで SDIO 割り込みを発生させたい場合には、バス設定を 1-bit バスに切り替えてから本 API をコールしてください。

R_SDSI_ReadIntVectorReg()

FN1 割り込みベクタレジスタ（FN1INTVECR）から値を読み出します。

Format

```
sdsi_status_t     R_SDSI_ReadIntVectorReg(
      uint32_t channel,
      uint8_t * p_vector
)
```

Parameters

channel

SDSI チャネル番号

** p_vector*

読み出しバッファポインタ（1 バイト）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

FN1 割り込みベクタレジスタから値を読み出します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint8_t     vector = 0;

if (R_SDSI_ReadIntVectorReg(SDSI_CH0, &vector) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

R_SDSI_ReadIntClearReg()

FN1 割り込みクリアレジスタから値を読み出します。

Format

```
sdsi_status_t     R_SDSI_ReadIntClearReg(
      uint32_t channel,
      uint8_t * p_vector
)
```

Parameters

channel

SDSI チャネル番号

** p_vector*

読み出しバッファポインタ（1 バイト）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

FN1 割り込みクリアレジスタから値を読み出します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint8_t     vector = 0;

if (R_SDSI_ReadIntClearReg(SDSI_CH0, &vector) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

R_SDSI_EnableDirectTrans()

DMA 転送許可設定を行います。

Format

```
sdsi_status_t    R_SDSI_EnableDirectTrans(
    uint32_t channel,
    sdsi_direct_trans_t * p_sdsi_direct_trans
)
```

Parameters

channel

SDSI チャネル番号

** p_sdsi_direct_trans*

adr *DMA 転送開始アドレス（設定可能範囲：内蔵 RAM アドレス）*

アドレスは 4 バイト境界に配置してください。

mode *DMA 転送モード*

＜アドレス設定：下記から 1 つ設定してください＞

SDSI_MODE_DIRECT_ADDR_FIXED ：DMA 転送アドレス固定

SDSI_MODE_DIRECT_ADDR_INC ：DMA 転送アドレスインクリメント

DMA 転送終了割り込み検出時に、次の DMA 転送アドレスを設定します。

＜バス設定：下記から 1 つ選択してください＞

SDSI_MODE_DIRECT_BUS_LOCK ：DMA 転送で使用するバスをロックする

SDSI_MODE_DIRECT_BUS_UNLOCK ：DMA 転送で使用するバスをロックしない

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*
SDSI_ERR_ADDRESS_BOUNDARY	*adr の 4 バイト境界アドレスエラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

DMA 転送許可設定を行います。正常終了後、SD ホストコントローラから CMD53（Function1 指定）が発行された場合、SDSI IP は内蔵 RAM に対してデータ転送を行います。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    g_io_buff[512/sizeof(uint32_t)];
sdsi_direct_trans_t     sdsi_direct_trans;

sdsi_direct_trans.adr = &g_io_buff[0];
sdsi_direct_trans.mode = (SDSI_MODE_DIRECT_ADDR_INC |
SDSI_MODE_DIRECT_BUS_UNLOCK);
if (R_SDSI_EnableDirectTrans(SDSI_CH0, &sdsi_direct_trans) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

本関数実行が正常終了した場合、再び本関数をコールしないでください。コールした場合、エラーを返します。再度本関数を実行する場合は、事前に DMA 転送禁止処理 R_SDSI_DisableDirectTrans()を実行してください。

SDSI は DMA バスを使用し DMA 転送を行います。RX65N の場合、SDSI とイーサネットコントローラ（ETHERC）で使用する DMA バスを共有しているため、排他制御が必要です。本関数が正常動作する前提条件は以下のとおりです。前提条件を満たさない限り、戻り値に *SDSI_ERR* を返します。

1. ETHER、EDMAC がモジュールストップ状態であること。
2. Ethernet FIT モジュールの初期設定は行われていないこと。
   （ETHERC と EDMAC のハードウェアリソースが開放されていること。）

また、ETHERC はその通信方式上、動的に DMA バスを解放することができません。したがって、ユーザシステムにて SDSI と ETHERC を併用する場合、SDSI の DMA 転送は行わないでください。また、以下の手順で設定してください。（ETHERC の制御に Ethernet FIT モジュールを使用することを前提）

1. Ethernet FIT モジュールの初期設定を行う。
   （ETHERC と EDMAC のハードウェアリソースロック、モジュールストップ解除を行う。）
2. SDSI FIT モジュールの初期設定を行う。
3. 以降、R_SDSI_EnableDirectTrans()のコールは禁止。

上記 3 の状態で本関数をコールした場合、戻り値に *SDSI_ERR* を返します。

R_SDSI_DisableDirectTrans()

DMA 転送禁止設定を行います。

Format

```
sdsi_status_t     R_SDSI_DisableDirectTrans(
      uint32_t channel
)
```

Parameters

channel

SDSI チャネル番号

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

DMA 転送禁止設定を行います。正常終了後、SD ホストコントローラから CMD53（Function1 指定）が発行された場合、SDSI IP は Function1 領域に対してデータ転送を行います。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
if (R_SDSI_DisableDirectTrans(SDSI_CH0) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

R_SDSI_SetDirectTransAdr()

DMA 転送アドレスを設定します。

Format

```
sdsi_status_t    R_SDSI_SetDirectTransAdr(
    uint32_t channel,
    uint32_t adr
)
```

Parameters

channel

SDSI チャネル番号

adr

DMA 転送開始アドレス（設定可能範囲：内蔵 RAM アドレス）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

DMA 転送アドレスを設定します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    g_io_buff[512/sizeof(uint32_t)];

if (R_SDSI_SetDirectTransAdr(SDSI_CH0, &g_io_buff[0]) != SDSI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。また、本関数は DMA 転送が開始される前にコールしてください。

R_SDSI_GetDirectTransAdr()

DMA 転送アドレスを取得します。

Format

```
sdsi_status_t     R_SDSI_GetDirectTransAdr(
      uint32_t channel,
      uint32_t * p_adr
)
```

Parameters

channel

SDSI チャネル番号

** p_adr*

DMA 転送開始アドレスバッファ（4 バイト）

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

DMA 転送アドレスを取得します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    adr = 0;

if (R_SDSI_GetDirectTransAdr(SDSI_CH0, &adr))
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。また、本関数は DMA 転送が開始される前にコールしてください。

R_SDSI_RegistIntCallback()

SDSI コマンド割り込み [3]コールバック関数を登録する関数です。

Format

```
sdsi_status_t     R_SDSI_RegistIntCallback(
      uint32_t channel,
      sdsi_status_t (* callback)(sdsi_cmd_t *)
)
```

Parameters

channel

SDSI チャネル番号

(callback)(sdsi_cmd_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。

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

SDSI コマンド割り込みコールバック関数を登録します。本関数は、R_SDSI_Initialize()関数実行前にコールしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_cmd_t g_sdsi_cmd;
sdsi_status_t r_sdsi_callback(sdsi_cmd_t * p_cmd);

if (R_SDSI_RegistIntCallback(SDSI_CH0, r_sdsi_callback) != SDSI_SUCCESS)
{
      /* Error */
}

static sdsi_status_t r_sdsi_callback(sdsi_cmd_t * p_cmd)
{
      g_sdsi_cmd.adr    = p_cmd->adr;
      g_sdsi_cmd.blkcnt  = p_cmd->blkcnt;
      g_sdsi_cmd.bytcnt  = p_cmd->bytcnt;
      g_sdsi_cmd.sdcmdcr = p_cmd->sdcmdcr;
      g_sdsi_cmd.cmd    = p_cmd->cmd;

      return SDSI_SUCCESS;
}
```

[3] SDSI コマンド割り込みとは「SDSI の CMD53 リードコマンド割り込み」「CMD53 ライトコマンド割り込み」「CMD52 ライトコマンド割り込み」を指します。

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

コールバック関数の引数(sdsi_cmd_t *)に格納される情報を表 3-1 に示します。本情報は SD ホストコントローラのコマンド発行により上書きされます。SD ホストが次のコマンド発行する前に読み出してください。

表 3.1　SDSI コマンド割り込みコールバック関数　引数情報

<table>
<tr><th rowspan="2">型</th><th rowspan="2">メンバ名</th><th rowspan="2">名称</th><th colspan="3">内容</th></tr>
<tr><th>CMD52 ライト</th><th>CMD53 ライト</th><th>CMD53 リード</th></tr>
<tr><td>uint32_t</td><td>adr</td><td>SD コマンドアクセスアドレス</td><td colspan="3">I/O レジスタの読み出し／書き込み開始アドレス
【有効データ：下位 17 ビット b16-b0】</td></tr>
<tr><td>uint16_t</td><td>blkcnt</td><td>ブロックカウンタ</td><td>“0”固定</td><td colspan="2">0：無限
1：1 ブロック
～
511：511 ブロック
【有効データ：下位 9 ビット b8-b0】</td></tr>
<tr><td>uint16_t</td><td>bytcnt</td><td>バイトカウンタ</td><td>“0”固定</td><td colspan="2">0：0 バイト
1：1 バイト
～
2048：2048 バイト
【有効データ：下位 12 ビット b11-b0】

■バイトモードの場合
CMD53 のアーギュメントに含まれるバイトカウントが格納されます。バイトカウントが“0”の場合、512 が格納されます。
■ブロックモードの場合
CMD53 のアーギュメントに含まれるファンクションナンバに対応するブロックサイズが格納されます。</td></tr>
<tr><td>uint8_t</td><td>sdcmdcr</td><td>SD コマンド制御情報</td><td colspan="3">SD コマンド制御レジスタ（SDCMDCR）の値を格納します。
<table>
<tr><th>ビット</th><th>ビット名</th><th>機能</th></tr>
<tr><td>b0</td><td>SD コマンドインデックス</td><td>0：コマンド発行なし、または、CMD52 ライトコマンド発行
1：CMD53 コマンド発行</td></tr>
<tr><td>b1</td><td>転送方向</td><td>0：本モジュールから SD ホストコントローラへリード
1：SD ホストコントローラから本モジュールへライト</td></tr>
<tr><td>b2</td><td>SD コマンドライト後リード</td><td>0：リードデータはライトデータと同じ（注 1）
1：リードデータはライト後に読み出されたデータ</td></tr>
<tr><td>b3</td><td>SD コマンドバイト／ブロックモード</td><td>0：バイトモード（注２）
1：ブロックモード</td></tr>
<tr><td>b4</td><td>SD コマンド CMD53 アドレスモード</td><td>0：固定転送アドレス
1：インクリメンタル転送アドレス（注２）</td></tr>
<tr><td>b7-b5</td><td>予約ビット</td><td>“0”を格納します。</td></tr>
</table>
注１：CMD53 時は不定。
注２：CMD52 時は不定。</td></tr>
<tr><td>uint8_t</td><td>cmd</td><td>コマンド情報</td><td>SDSI_CMD52_W（0x01）</td><td>SDSI_CMD53_W（0x02）</td><td>SDSI_CMD53_R（0x04）</td></tr>
</table>

R_SDSI_RegistCdIntCallback()

SDSI カード検出無効（Rise／Fall）割り込みコールバック関数を登録する関数です。

Format

```
sdsi_status_t    R_SDSI_RegistCdIntCallback(
    uint32_t channel,
    sdsi_status_t (* callback)(uint32_t)
)
```

Parameters

channel

SDSI チャネル番号

(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。

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

SDSI カード検出無効（Rise／Fall）割り込みコールバック関数を登録します。本関数は、R_SDSI_Initialize()関数実行前にコールしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_status_t r_sdsi_cd_callback(uint32_t cd);

if (R_SDSI_RegistCdIntCallback(SDSI_CH0, r_sdsi_cd_callback) != SDSI_SUCCESS)
{
      /* Error */
}

static sdsi_status_t r_sdsi_cd_callback(uint32_t cd)
{
      if (SDSI_CD_RISE == cd)
      {
            /* Card detection disable (rise) interrupt. */
            R_BSP_NOP();
      }
      else
      {
            /* Card detection disable (fall) interrupt. */
            R_BSP_NOP();
      }
      return SDSI_SUCCESS;
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

コールバック関数の引数(uint32_t)に格納される情報は以下のとおりです。

SDSI_CD_RISE：カード検出無効（Rise）割り込み検出時
SDSI_CD_FALL：カード検出無効（Fall）割り込み検出時

R_SDSI_RegistDtIntCallback()

SDSI DMA 転送終了割り込みコールバック関数を登録する関数です。

Format

```
sdsi_status_t    R_SDSI_RegistDtIntCallback(
    uint32_t channel,
    sdsi_status_t (* callback)(sdsi_cmd_t *)
)
```

Parameters

channel

SDSI チャネル番号

(callback)(sdsi_cmd_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。

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_rx_if.h にプロトタイプ宣言されています。

Description

DMA 転送終了割り込みコールバック関数を登録します。本関数は、R_SDSI_Initialize()関数実行前にコールしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
sdsi_cmd_t g_sdsi_cmd;
sdsi_status_t r_sdsi_dt_callback(sdsi_cmd_t * p_cmd);

if (R_SDSI_RegistDtIntCallback(SDSI_CH0, r_sdsi_dt_callback) != SDSI_SUCCESS)
{
      /* Error */
}

static sdsi_status_t r_sdsi_dt_callback(sdsi_cmd_t * p_cmd)
{
      g_sdsi_cmd.adr    = p_cmd->adr;
      g_sdsi_cmd.blkcnt  = p_cmd->blkcnt;
      g_sdsi_cmd.bytcnt  = p_cmd->bytcnt;
      g_sdsi_cmd.sdcmdcr = p_cmd->sdcmdcr;
      g_sdsi_cmd.cmd    = p_cmd->cmd;

      return SDSI_SUCCESS;
}
```

Special Notes

本関数実行前に R_SDSI_Open()関数によるオープン処理が必要です。

コールバック関数の引数(sdsi_cmd_t *)に格納される情報は、SDSI コマンド割り込みと同様です。詳細は表 3-1 を参照してください。本情報は SD ホストコントローラのコマンド発行により上書きされます。SD ホストが次のコマンド発行する前に読み出してください。

R_SDSI_GetVersion()

SDSI FIT モジュールのバージョン情報を取得する関数です。

Format

```
uint32_t    R_SDSI_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

ファイル r_sdsi_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。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    version = 0;

version = R_SDSI_GetVersion();
```

Special Notes

なし

R_SDSI_SetLogHdlAddress()

LONG 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

```
sdsi_status_t    R_SDSI_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

SDSI_SUCCESS	*正常終了*

Properties

ファイル r_sdsi_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 SDSI FIT モジュールに設定します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

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

sdsi_status_t     ret = SDSI_SUCCESS;
uint32_t          MtlLogTbl[ERR_LOG_SIZE];
longq_err_t       err;
longq_hdl_t       p_sdsi_user_long_que;
uint32_t          long_que_hndl_address;


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

long_que_hndl_address = (uint32_t)p_sdsi_user_long_que;
ret = R_SDSI_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_SDSI_Open()関数をコールする前に処理を実行してください。

別途 LONG FIT モジュールを組み込んでください。

R_SDSI_Log()

エラーログを取得する関数です。

Format

```
uint32_t      R_SDSI_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_sdsi_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。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
#define     USER_DRIVER_ID    (0x00000001)
#define     USER_LOG_MAX      (0x0000003f)
#define     USER_LOG_ADR_MAX  (0x00001fff)

sdsi_reg_t sdsi_reg;
sdsi_union_t io_buff = 0;

sdsi_reg.reg_no = SDSI_FUNC1_REG1;
sdsi_reg.offset = 0x0;
sdsi_reg.p_buff = &io_buff;
if (R_SDSI_WriteFuncReg(SDSI_CH0, &sdsi_reg) != SDSI_SUCCESS)
{
      /* Set last error log to buffer. */
      R_SDSI_Log(
            USER_DRIVER_ID,
            USER_LOG_MAX,
            USER_LOG_ADR_MAX
      );
}
```

Special Notes

別途 LONG FIT モジュールを組み込んでください。

4. 端子設定

SDSI 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。端子設定は、R_SDSI_Open()関数を呼び出す前に行ってください。

e² studio で端子設定を行う場合、スマート・コンフィグレータの端子設定機能を利用できます。端子設定機能を使用する場合、スマート・コンフィグレータの端子設定ウィンドウで選択したオプションに従ってソースファイルが出力されます。端子は、ソースファイルで定義された関数を呼び出すことによって設定されます。詳細については、表 4.1 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

選択したオプション	出力される関数名	備考
チャネル 0	R_SDSI_PinSet()	

4.1 駆動能力設定

SDSI_CMD と SDSI_D0～SDSI_D3 は入力ピンであり、応答とデータ出力を行います。

MCU を実装する回路に合わせて設定を見直してください。

I/O ポートの設定は、ドライブ容量制御レジスタ(DSCR)またはドライブ容量制御レジスタ 2(DSCR2)を使用して変更できます。

「スマート・コンフィグレータ」の端子設定機能を使用する場合は、SDSI_CMD 端子、SDSI_D0～SDSI_D3 端子を DSCR=1(高駆動出力)に設定してください。

表 4.2 を参照し、必要に応じて設定を確認してください。

表 4.1 駆動能力設定

DSCR2	DSCR	駆動能力	サポート MCU のデフォルト設定
0	0	通常駆動出力	-
0	1	高駆動出力	RX65N
1	Don’t care	高速インタフェース用高駆動出力	-

5. デモプログラム

5.1 デモプログラムの概要

本デモプログラムは、SD ホストが発行する SD コマンドを受信し、SD コマンドに応じた周辺機能制御を実現します。これにより、SD ホストは SD スレーブの周辺機能を自身の周辺機能のように扱うことができます。なお、本サンプルプログラムは SD スレーブの処理例であり、SD ホストの処理は別途必要です。

本デモプログラムでは以下を実現します。

- 特定の SD コマンドを受信した場合、GPIO FIT モジュールを制御して RSK ボード上の LED を点灯

5.2 API の概要

表 5-1 にデモプログラムに含まれる API 関数を示します。

表 5.1 デモプログラム API 関数一覧

関数名	説明
R_SDSI_PXPD_Open()	SDSI FIT モジュールの初期化処理
R_SDSI_PXPD_ReadCmd()	SD コマンドの読み出し処理
R_SDSI_PXPD_WriteResp()	SD レスポンスの書き込み処理
R_SDSI_PXPD_SetSDIOInt()	SDIO 割り込み発行処理
R_SDSI_PXPD_GetSDIOInt()	SDIO 割り込みベクタ読み出し処理

5.3 動作説明

5.3.1 ハードウェア説明

図 5-1 にブロック図を示します。

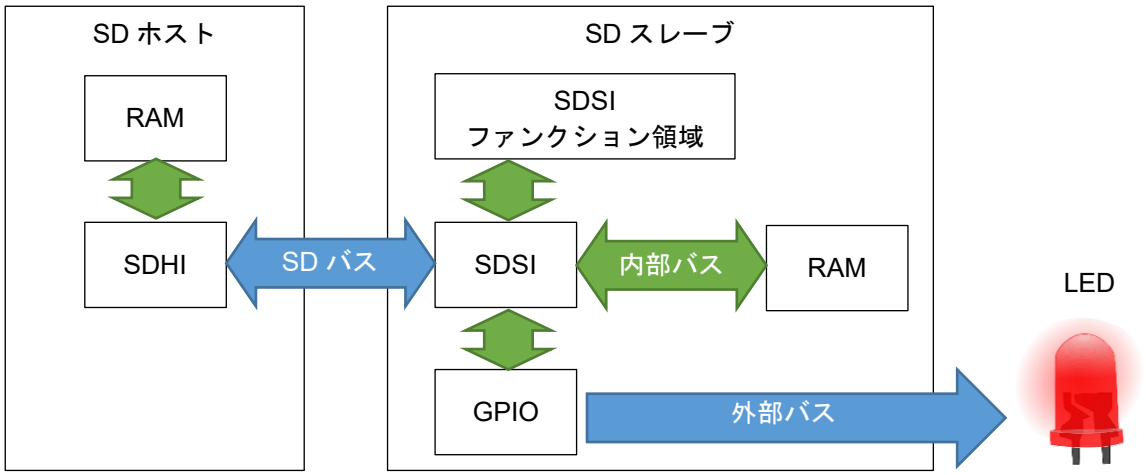


図 5-1 ブロック図

5.3.2 ソフトウェア説明

(1) 状態遷移図

図 5-2 に状態遷移図を示します。

初期化後、特定の SD コマンドを受信した場合、GPIO FIT モジュールを制御して RSK 上の LED を点灯します。点灯処理が完了したら、SDIO 割り込みを発行して SD ホストに処理が完了したことを通知します。通知を受け取った SD ホストが SD スレーブの SDIO 割り込みを解除します。それにより、SD スレーブが SDIO 割り込み解除待ち状態から、アイドル状態に遷移します。

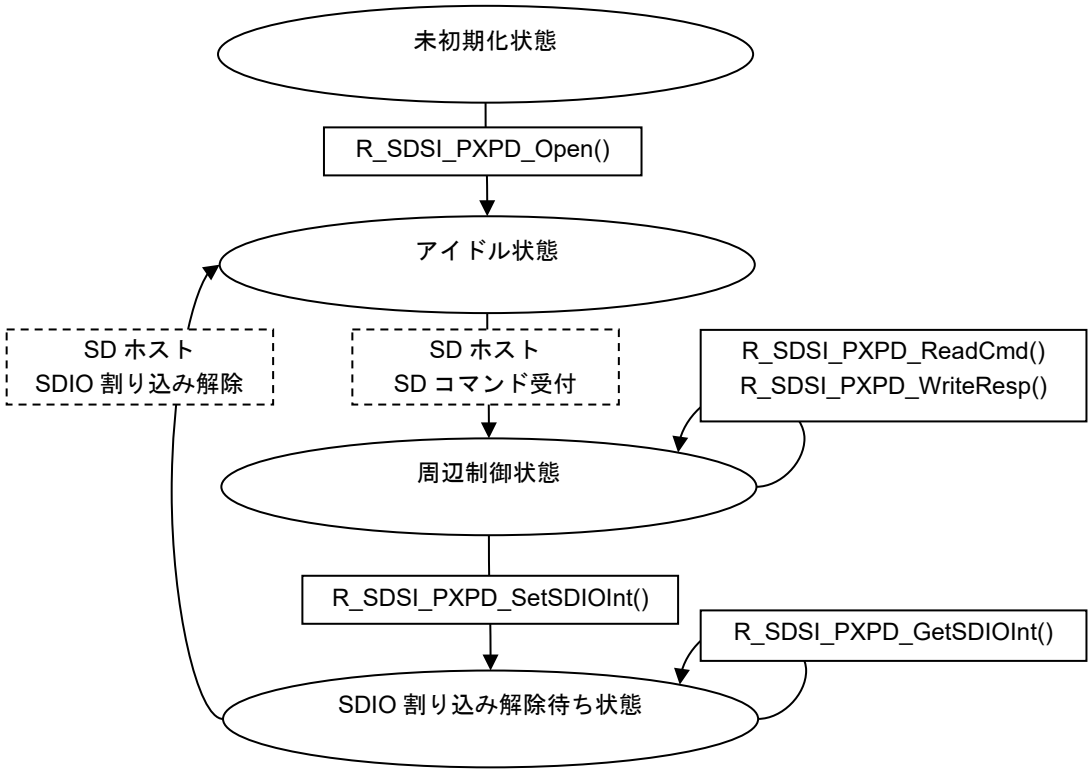


図 5-2 状態遷移図

(2)　シーケンス図

図 5-3 にシーケンス図を示します。SD ホストから SD スレーブにアクセスし、GPIO を制御して LED 点灯を行います。以下に詳細を示します。

① SD ホストは、SD スレーブで制御する GPIO FIT モジュール情報をパケット化し、SD スレーブに送信します。

② SD スレーブは、SD ホストから受信したパケット情報を取り出し、GPIO FIT モジュールの API を実行します。

③ SD スレーブは、API の実行結果を SD ホストに通知するため、SDIO 割り込みを発行します。

④ SDIO 割り込みを検出した SD ホストは、API の実行結果確認し、SDIO 割り込みを解除します。

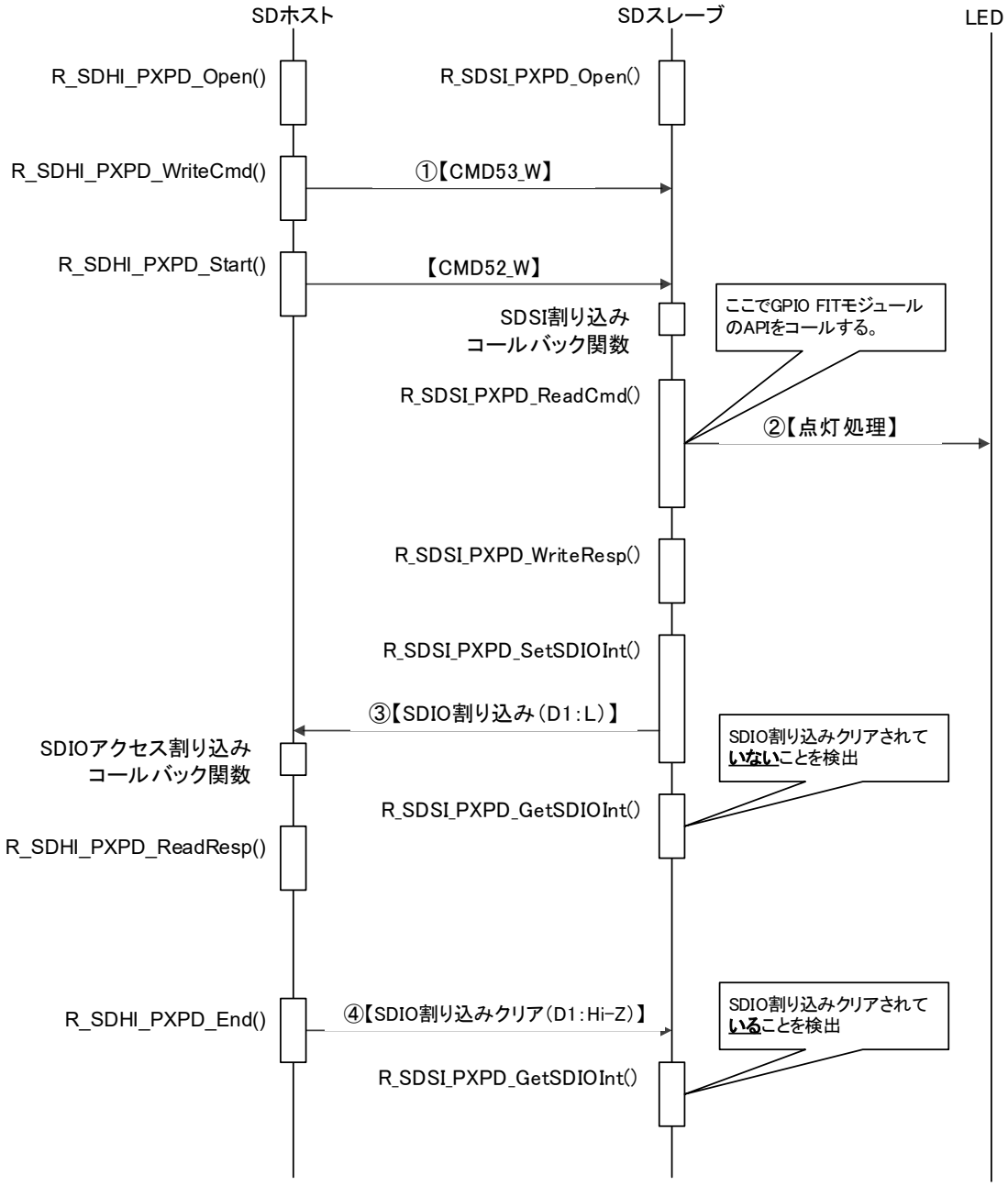


図 5-3 GPIO 制御による LED 点灯シーケンス図

(3)　r_sdsi_pxpd_rx.c の説明

表 5-2 に Function1 Register1 領域（以降、Func1 Reg1）のデータ割り当てを示します。r_sdsi_pxpd_rx.c では、SD ホストからアクセスされる Func1 Reg1 のデータを参照し、SD スレーブの LED を点灯させます。以下に詳細を説明します。

- R_SDSI_PXPD_Open()を実行し、Func1 Reg1 のオフセットアドレス“0x00（Status）”の状態が”0xD0”に変化するまで待ちます。
- “0xD0”を検出したら R_SDSI_PXPD_ReadCmd()をコールし、表 5-2 に示す情報を読み出します。サンプルプログラムでは“0x0C（関数番号）”から”0x03”を読み出し、GPIO FIT モジュールの R_GPIO_PinWrite()を実行します。これにより、RSK ボード上の LED を点灯させます。
- R_SDSI_PXPD_WriteResp()をコールし、オフセットアドレス“0x18（FIT モジュール戻り値）”を更新します。
- 処理の完了を SD ホストに通知するため、R_SDSI_PXPD_SetSDIOInt()をコールし、SDIO 割り込みを発生させます。
- SD ホストが SDIO 割り込み解除し、処理が終了します。

表 5.2 Register1 割り当て

オフセットアドレス	内容	設定値
0x00	Status	0x00：アイドル 0x01：転送情報セット ホストが Register1 へ書き込み 0x02：転送結果セット スレーブが Register1 に書き込み 0xD0：実行 FIT モジュール実行
0x04	関数の引数情報の有効データサイズ（4 の倍数）	0x00：初期値 0x04 – 0xE0：設定可（4 の倍数のみ） 上記以外：設定禁止
0x08	FIT モジュール番号	0x00：使用しない 0x01：GPIO 0x02 – 0xFF：設定無効（拡張用）
0x0C	関数番号	FIT モジュール番号：0x00（GPIO）の場合 0x00：R_GPIO_PortWrite() 0x01：R_GPIO_PortRead() 0x02：R_GPIO_PortDirectionSet() 0x03：R_GPIO_PinWrite() 0x04：R_GPIO_PinRead() 0x05：R_GPIO_PinDirectionSet() 0x06：R_GPIO_PinControl() 0x07：R_GPIO_GetVersion() 0x08 – 0xFFFFFFFF：設定無効
0x10	コントロール （完了時の SDIO 割り込み要求 bit）	0x00：完了時、SDIO 割り込み要求を発行する、かつ、レスポンスフォーマットに書き込みは行わない。 0x01：完了時、SDIO 割り込み要求を発行する、かつ、レスポンスフォーマットに書き込みは行う。 0x02 – 0xFF：設定無効（拡張用）

0x14	SDIO 割り込みベクタ番号	0x00：初期値 0x01 -0x FF：設定可能
0x18 – 0x1B	FIT モジュール戻り値	FIT モジュール番号：0x00000001（GPIO）の場合 0x01000000：SDSI_PXPD_FIT_GPIO_VOID 0x01000001：SDSI_PXPD_FIT_GPIO_SUCCESS 0x01000002： SDSI_PXPD_FIT_GPIO_ERR_INVALID_MODE 0x01000003： SDSI_PXPD_FIT_GPIO_ERR_INVALID_CMD その他 0x00000000：SDSI_PXPD_VOID 0x00000001：SDSI_PXPD_SUCCESS 0xFFFFFFFF：SDSI_PXPD_ERR_FUNCTION
0x1C – 0x1F	予約領域	0x00000000 – 0xFFFFFFFF：設定無効
0x20 – 0xFF	関数の引数情報（最大：224 バイト）	0x00000000 – 0xFFFFFFFF：設定可 有効データの最終オフセットアドレスを超える値は設定無効

(4) r_sdsi_pxpd_rx_config.h の説明

デモプログラムの初期設定は RSKRX65N-2MB の GPIO_PORT_7_PIN_3 を使用して、LED を点灯させる設定になっています。r_sdsi_pxpd_rx_config.h をカスタマイズすることで、お客様 MCU と LED を接続し、デモプログラムで点灯させることができます。

1．お使いになる MCU のハードウェアマニュアルを用意ください。

2．LED（点灯させるもの）と接続されている MCU ポートをご確認ください。

3．当該 MCU ポートのポート出力データレジスタ（PODR）をヘッダファイルの LED0_PODR に設定して下さい。

4．当該 MCU ポートのボート方向レジスタ（PDR）をヘッダファイルの LED0_PDR に設定して下さい。

5.4　組み込みからビルドまでの手順

FIT モジュールをプロジェクトに組み込み、ビルドするまでの手順を説明します。なお、以下の手順は SD スレーブに対するものであり、SD ホストの環境は別途構築する必要があります。

- SD ホストと SD スレーブの端子を以下の通りに接続してください。
  - ・SDHI_CLK←→SDSI_CLK
  - ・SDHI_CMD←→SDSI_CMD
  - ・SDHI_D0←→SDSI_D0
  - ・SDHI_D1←→SDSI_D1
  - ・SDHI_D2←→SDSI_D2
  - ・SDHI_D3←→SDSI_D3
- e² studio で新規プロジェクトを作成してください。
- 「2.12 FIT モジュールの追加方法」を参考にして、以下の FIT モジュールをプロジェクトに追加してください。
  - ・r_bsp
  - ・r_gpio_rx
  - ・r_sdsi_rx
- 「5.5 デモのダウンロード方法」を参考にして製品パッケージを入手してください。
- 製品パッケージの FITDemos フォルダにある以下のデモプログラムをプロジェクトに追加してください。
  - ・r_sdsi_pxpd_rx.c
  - ・r_sdsi_pxpd_rx.h
  - ・r_sdsi_pxpd_rx_config.h
- 設定が終わったら次の手順でビルドしてください。
  メニュー[プロジェクト]>[プロジェクトのビルド]
- 正常にビルドできない場合、「6.2 トラブルシューティング」または、「7 参考ドキュメント」をご覧ください。

5.5　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。

5.6 API 関数

5.6.1 R_SDSI_PXPD_Open()

SDSI FIT モジュールを初期化する関数です。他の API 関数を使用する前に実行してください。

Format

```
sdsi_pxpd_status_t      R_SDSI_PXPD_Open(
    sdsi_pxpd_int_callback_info_t * p_int_callback_info
)
```

Parameters

** p_int_callback_info*

コールバック関数用の構造体へのポインタ

(* callback)(sdsi_cmd_t *)

R_SDSI_RegistIntCallback ()用コールバック関数

(* callback_dt)(sdsi_cmd_t *)

R_SDSI_RegistDtIntCallback()用コールバック関数

Return Values

SDSI_PXPD_SUCCESS	*正常終了*
SDSI_PXPD_ERR	*一般エラー*

Properties

ファイル r_sdsi_pxpd_rx.h にプロトタイプ宣言されています。

Description

次の順序で関数をコールし、SDSI FIT モジュールの初期化を行います。

1. R_SDSI_Open()をコールし、SDSI FIT モジュールを初期化します。
2. R_SDSI_RegistIntCallback()をコールし、SDSI コマンド割り込みコールバック関数を登録します。
3. R_SDSI_RegistDtIntCallback()をコールし、SDSI DMA 転送終了割り込みコールバック関数を登録します。
4. R_SDSI_WriteCisReg()、R_SDSI_ReadCisReg()の順でコールし、CIS レジスタにアクセスします。
5. R_SDSI_PinSet()をコールし、端子にポートを割り当てます。
6. R_SDSI_Initialize()をコールし、SDSI IP の初期設定を行います。正常終了後、C フラグポーリング状態に遷移します。
7. R_SDSI_CflagPolling()をコールし、R4 レスポンスの C フラグ状態を取得します。R_SDSI_Initialize()による初期化処理実行後、本関数をコールし、戻り値に SDSI_SUCCESS（C フラグは“（レディ）”)になることを確認します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
ret = R_SDSI_PXPD_Open(&call);
if (SDSI_PXPD_SUCCESS != ret)
{
       trap();
}
```

Special Notes

各関数の詳細は「3 API 関数」を参照下さい。

5.6.2 R_SDSI_PXPD_ReadCmd()

SD コマンドの読み出し処理です。

Format

```
sdsi_pxpd_status_t      R_SDSI_PXPD_ReadCmd(
    uint32_t * p_result,
    uint8_t * p_arg_data
)
```

Parameters

** p_result*

FIT GPIO 戻り値

** p_arg_data*

読み出しバッファポインタ(1 バイト)

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_pxpd_rx.h にプロトタイプ宣言されています。

Description

次の順序で関数をコールし、SD コマンドの読み出し処理を行います。

1. R_SDSI_ReadFuncReg()をコールし、FN1 データレジスタ m から値を読み出します（m=1,3,5）。
2. オフセット 0x00 の値により、動作が分岐します。

・SDSI_PXPD_FIT_GPIO の時、オフセット 0x08 の値が SDSI_PXPD_FIT_GPIO であれば、r_sdsi_pxpd_fit_gpio()をコールします。

・SDSI_PXPD_STATUS_DO_ENABLE_DIRECT の時、SDSI ダイレクト転送 ON で r_sdsi_pxpd_direct()をコールします。

・SDSI_PXPD_STATUS_DO_DISABLE_DIRECT の時、SDSI ダイレクト転送 OFF で r_sdsi_pxpd_direct()をコールします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
ret = R_SDSI_PXPD_ReadCmd(&io_buff.l, &g_sdsi_pxpd_buff[0]);
if (SDSI_PXPD_SUCCESS != ret)
{
trap();
}
```

Special Notes

なし

5.6.3 R_SDSI_PXPD_WriteResp()

SD レスポンスの書き込み処理です。

Format

```
sdsi_pxpd_status_t R_SDSI_PXPD_WriteResp(
    uint32_t result
)
```

Parameters

result

書き込みバッファ

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_pxpd_rx.h にプロトタイプ宣言されています。

Description

次の順序で関数をコールし、SD レスポンスの書き込み処理を行います。

1. R_SDSI_ReadFuncReg()をコールし、FN1 データレジスタ 1 のオフセット 0x10 から値を読み出し、実行コマンドを確認します。
2. 読み込んだコマンドが、SDSI_PXPD_CTRL_SDIO_INT_WRITE であれば、R_SDSI_WriteFuncReg()をコールし、FN1 データレジスタ 1 のオフセット 0x18 に result をセットします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
ret = R_SDSI_PXPD_WriteResp(io_buff.l);
if (SDSI_PXPD_SUCCESS != ret)
{
       trap();
}
```

Special Notes

なし

5.6.4 R_SDSI_PXPD_SetSDIOInt()

SDIO 割り込み発行処理。

Format

```
sdsi_pxpd_status_t R_SDSI_PXPD_SetSDIOInt(
            void
)
```

Parameters

void

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_pxpd_rx.h にプロトタイプ宣言されています。

Description

次の順序で関数をコールし、SD コマンドの読み出し処理を行います。

1. R_SDSI_WriteFuncReg()をコールし、FN1 データレジスタ 1 のオフセット 0x00 を 0 で初期化します。
2. R_SDSI_ReadFuncReg()をコールし、FN1 データレジスタ 1 のオフセット 0x14 から値を読み出し、SDIO 割り込みベクタ番号を取得します。
3. R_SDSI_WriteIntVectorReg()をコールし、SDIO 割り込みを発行します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
ret = R_SDSI_PXPD_SetSDIOInt();
if (SDSI_PXPD_SUCCESS != ret)
{
       trap();
}
```

Special Notes

なし

5.6.5 R_SDSI_PXPD_GetSDIOInt()

SDIO 割り込みベクタ読み出し処理。

Format

```
sdsi_pxpd_status_t R_SDSI_PXPD_GetSDIOInt(
            uint8_t * p_vector
)
```

Parameters

** p_vector*
SDIO 割り込みベクタバッファ

Return Values

SDSI_SUCCESS	*正常終了*
SDSI_ERR	*一般エラー*

Properties

ファイル r_sdsi_pxpd_rx.h にプロトタイプ宣言されています。

Description

次の順序で関数をコールし、SDIO 割り込みベクタ読み出し処理を行います。

1. R_SDSI_ReadIntVectorReg()

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
ret = R_SDSI_PXPD_GetSDIOInt(&io_buff.c[0]);
if (SDSI_PXPD_SUCCESS != ret)
{
    trap();
}
```

Special Notes

なし

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

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 V6.0.0
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 V.2.07.00
	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	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	Rev.2.00
使用ボード	Renesas Starter Kit for RX65N (型名：RTK500565NSxxxxxxx) Renesas Starter Kit for RX65N-2MB(型名：RTK50565N2Sxxxxxxx)

表 6.2 動作確認環境

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 V7.3.0 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	Rev.2.02
使用ボード	Renesas Starter Kit+ for RX65N (型名：RTK500565Nxxxxxxxx)

表 6.3 動作確認環境

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 V.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 Package 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	Rev.2.03

表 6.4 動作確認環境

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 V.2023-10 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 Package for RX Family V3.05.00 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 8.3.0.202305 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	Rev.2.04

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_sdsi_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。

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
テクニカルアップデート／テクニカルニュース
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。）

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*-A176A/J

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.9.30	-	新規発行
2.00	2017.7.31	-	ドキュメント内の「ダイレクト転送」を「DMA 転送」に変更
		1	対象デバイスに、RX651 を追加。
		8	1.4.2 ソフトウェア説明 (2)　新規に追加した。
		11	2.7 コンパイル時の設定 「SDSI_CFG_PARAM_CHECKING_ENABLE」 「SDSI_CFG_FBR_ADR_102H」の説明を追加した。また、 「SDSI_CFG_DISABLE_SYSTEM_INTERRUPT」の内容を見直した。
		14	2.12 FIT モジュールの追加方法　内容を見直した。
		23	3.8 R_SDSI_WriteFuncReg()　データの書き込みサイズを 4 バイトから 1 バイトに仕様変更した。
		25	3.9 R_SDSI_ReadFuncReg()　データの読み出しサイズを 4 バイトから 1 バイトに仕様変更した。
		27	3.10 R_SDSI_WriteIntVectorReg() Special Notes　内容を追加した。
		36	3.17 R_SDSI_RegistIntCallback() Special Notes　内容を追加した。
		45	4. 端子設定　内容を見直した。
		46-56	5. デモプログラム　新規に追加した。
		57	6. 付録　新規に追加した。
2.02	2019.5.20	—	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」に、R01AN1723 と R01AN1826 を削除
		1	「対象コンパイラ」を追加
		10	「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
		13	「2.8 コードサイズ」を更新
		38	3.18 R_SDSI_RegistCdIntCallback の Example に、nop の処理を BSP の組み込み関数に置き換え
		58	「6.1 動作確認環境」に、表 6-2 動作確認環境を追加
2.03	2022.12.27	55 プログラム	「6.3 動作確認環境」：。 Rev.2.03 に対応する表を追加。。 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。
2.04	2023.12.13	15, 43 16 55 プログラム	「2.12 FIT モジュールの追加方法」、「4 端子設定」から FIT Configurator の説明を削除した。 2.13「for 文、while 文、do while 文について」を追加。 「6.4 動作確認環境」：。 Rev.2.04 に対応する表を追加。。 WAIT_LOOP コメントを追加。

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。