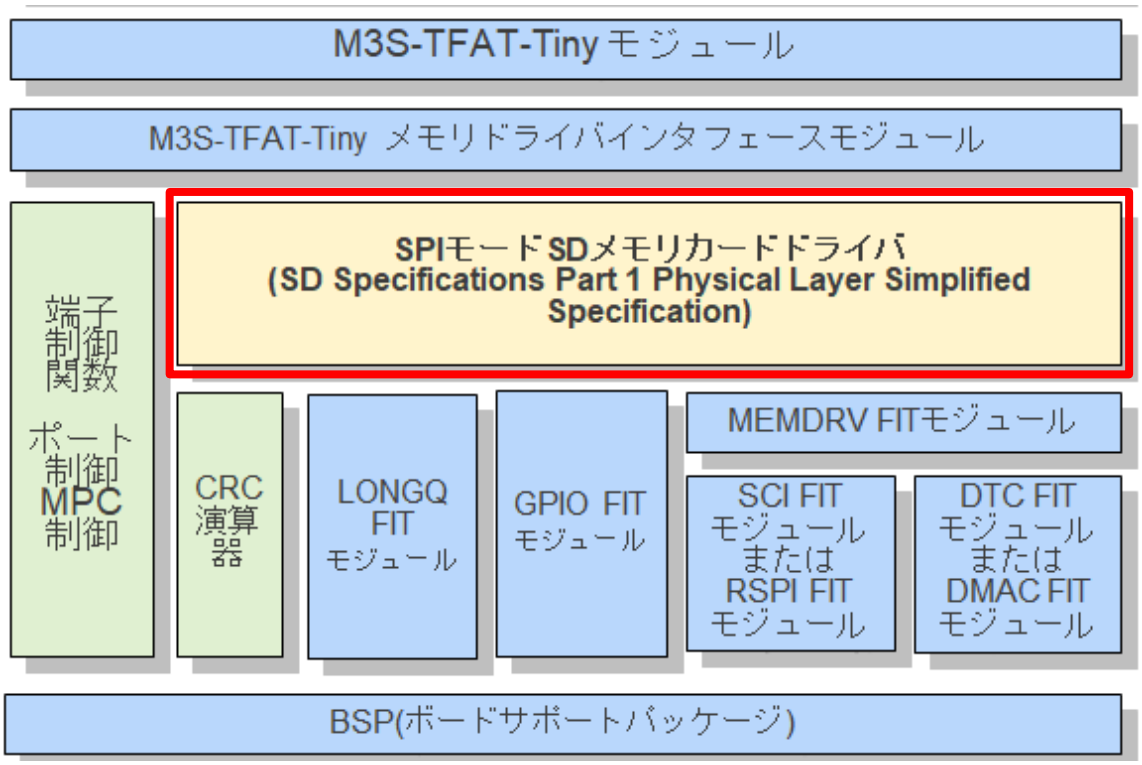
RX ファミリ

SPI モード SD メモリカードドライバ 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 SPI モード SD メモリカードドライバ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 内蔵のシリアルコミュニケーションインタフェース（SCI）又はシリアルペリフェラルインタフェース(RSPI)を使用して、SD メモリカードを SPI モードで制御します。以降、本モジュールを SPI モード SD メモリカードドライバと称します。

本モジュールの位置づけを以下に示します。



本モジュールは Simplified Specification に対応しています。

SD 規格に対応したホスト機器を開発するには、SD Card Association License Agreement(SDALA)及び SD Card Association Membership Agreement(SDAMA)の締結が必要です。

詳細は SD Association のサイトをご確認ください。
https://www.sdcard.org/

対象デバイス

・RX ファミリ

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

- RX ファミリ 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 DMAC モジュール Firmware Integration Technology (R01AN2063)
- RX ファミリ DTC モジュール Firmware Integration Technology (R01AN1819)
- 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)
- RX ファミリ SCI モジュール Firmware Integration Technology(R01AN1815)
- RX ファミリ RSPI モジュール Firmware Integration Technology(R01AN1827)
- RX ファミリ メモリアクセス用ドライバインタフェースモジュールモジュール Firmware Integration Technology(R01AN4548)
- RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール (R20AN0038)
- 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 (R20AN0335)
- RX ファミリ GPIO モジュール Firmware Integration Technology (R01AN1721)

目次

1. 概要

1.1 SPI モード SD メモリカードドライバとは

本モジュールはクロック同期式シリアルインタフェースなどルネサスが無償提供しているソフトウェアモジュール（次ページの図 1-1、表 1-1 参照）と組み合わせて使用することにより SD メモリカードの制御が可能になります。

別途提供している「RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール (以降、M3S-TFAT-Tiny モジュールとする。)」及び「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(以降、メモリドライバインタフェースモジュールとする。)」と組み合わせて使用することにより、SD メモリカードに対して FAT ファイルシステムによるファイルアクセスが可能になります。

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

SPI モード SD メモリカードドライバを使用して 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例を図 1-1 に示します。

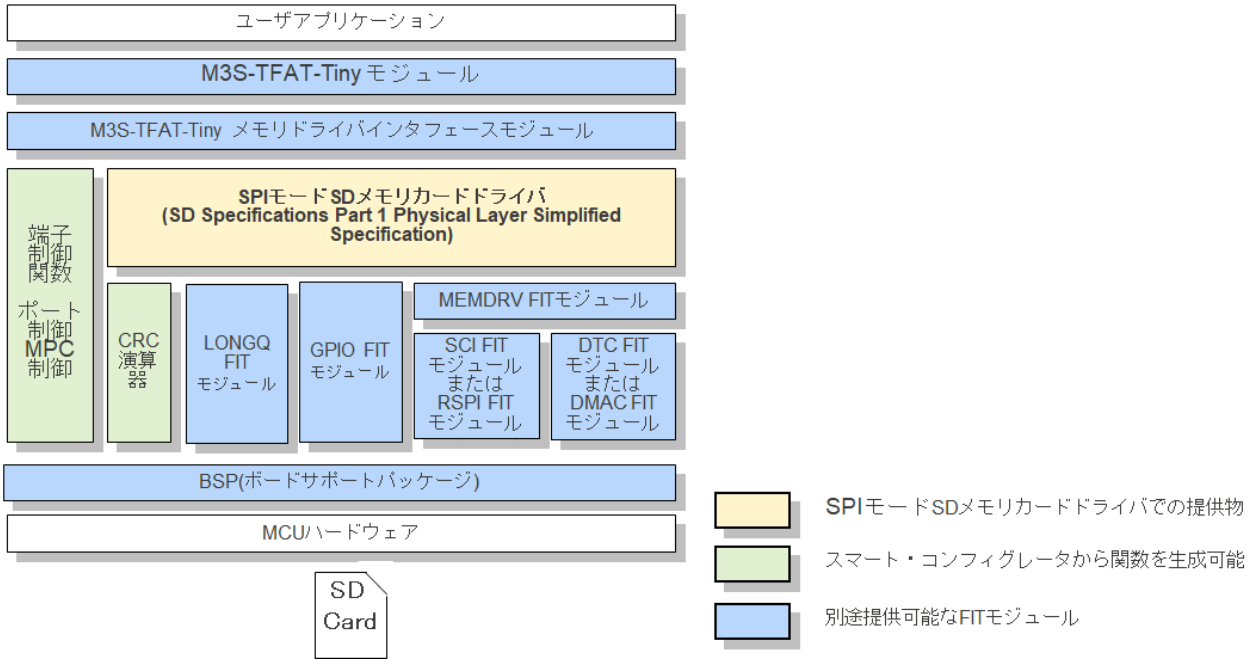


図 1-1　アプリケーション構成例

表 1-1 にアプリケーション構成で使用するモジュール一覧を示します。

表 1-1　アプリケーション構成で使用するモジュール一覧

項目	ドキュメント番号	備考
M3S-TFAT-Tiny モジュール	R20AN0038	
メモリドライバインタフェースモジュール	R20AN0335	
SPI モード SD メモリカードドライバモジュール	R01AN6908	
MEMDRV FIT モジュール	R01AN4548	
SCI FIT モジュール	R01AN1815	
RSPI FIT モジュール	R01AN1827	
DMAC FIT モジュール	R01AN2063	
DTC FIT モジュール	R01AN1819	
LONGQ FIT モジュール	R01AN1889	
Config_CRC(CRC 演算器)	-	スマート・コンフィグレータのコード生成から使用可能(注 1)
GPIO FIT モジュール	R01AN1721	
BSP(ボードサポートパッケージモジュール)	R01AN1685	
端子設定関数	-	スマート・コンフィグレータの端子設定機能から使用可能(注 2)

注 1: スマート・コンフィグレータのコンポーネントタブの＋アイコン（コンポーネントの追加）をクリックし、CRC 演算器を選択します。その後コードの生成ボタンを押してください。

注 2: スマート・コンフィグレータの端子タブで図 1-2 に対応する CD,CS,WP のポートと、SMOSI, SMISO, SCK（SPI の場合は MOSI, MISO, RSPCK)に対応する端子を選択後、コードの生成ボタンを押してください。

(1)　M3S-TFAT-Tiny モジュール

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。
RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny：https://www.renesas.com/mw/tfat-rx

(2)　メモリドライバインタフェースモジュール

ルネサス エレクトロニクス製 M3S-TFAT-Tiny モジュールと SPI モード SD メモリカードドライバ API を接続するソフトウェアです。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)　SPI モード SD メモリカードドライバモジュール

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

(4)　MEMDRV FIT モジュール

SPI モード SD メモリカードドライバと SCI FIT モジュール又は RSPI FIT モジュールのアダプターとなるソフトウェアです。

(5)　SCI FIT モジュール

SCI ハードウェア制御を行うソフトウェアです。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。

(6)　RSPI FIT モジュール

RSPI ハードウェア制御を行うソフトウェアです。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。

(7)　DTC FIT モジュール、DMAC FIT モジュール

DMAC 制御、DTC 制御を行うソフトウェアです。

(8)　GPIO FIT モジュール

汎用入出力ポートの制御行うソフトウェアです。

(9)　LONGQ FIT モジュール

エラーログ取得時に使用するロングワード型（uint32_t）のリングバッファを構成し管理するソフトウェアです。

(10) Config_CRC(CRC 演算器)

コマンド送受信時に使用する CRC を計算するソフトウェアです。

(11) BSP(ボードサポートパッケージモジュール)

FIT モジュールを使用するプロジェクトの基盤となるソフトウェアです。

(12) 端子設定関数

端子の割り当てを行う関数です。SCI 又は RSPI で使用するポートと SD カードの CD 端子、WP 端子、CS 端子に使用する端子のポート設定を行います。

本モジュール同梱のデモプロジェクトではスマート・コンフィグレータを使用し端子割り当てを行い、端子設定関数を生成しています。

1.2　SPI モード SD メモリカードドライバの概要

本モジュールは SPI を使用して SD メモリカードを制御します。

表 1-2、表 1-3 に本モジュールの機能を示します。

表 1-2　SPI モード SD メモリカードドライバ機能一覧

項目	機能
準拠した規格	SD Specifications Part 1 Physical Layer Simplified Specification Ver.9.00 準拠
SD ホストインタフェース 制御ドライバ	1 ブロック=512 バイトとするブロック型デバイスドライバ
SD カード動作電圧	2.7～3.6V 本ドライバでは 3.3V のみをサポート
SD カードインタフェース電圧	3.3V のみサポート
SD カード Bus インタフェース	SPI モード（1 ビット）をサポート
SD カード Speed mode	Default Speed mode をサポート
SD カードメモリ容量	標準容量 SD メモリカード（SDSC）と大容量 SD メモリカード（SDHC、SDXC）をサポート
SD カードメモリ制御対象	ユーザ領域のみをサポート プロテクト領域の制御はサポート対象外
SD カード検出機能	CD 端子による検出のみをサポート

表 1-3　MCU 機能一覧

項目	機能
対象 MCU	SCI または RSPI 搭載の RX ファミリ MCU
MCU 内データ転送方式	MEMDRV FIT モジュールにより、Software 転送／DMAC 転送／DTC 転送の選択が可能
エンディアン	ビッグエンディアン／リトルエンディアン対応

1.3　API の概要

表 1-4 に本モジュールに含まれる API 関数を示します。

表 1-4　API 関数一覧

関数	関数説明
R_SDC_SPI_Open()	ドライバのオープン処理
R_SDC_SPI_Close()	ドライバのクローズ処理
R_SDC_SPI_GetCardDetection()	挿入確認処理
R_SDC_SPI_Initialize()	初期化処理
R_SDC_SPI _End()	終了処理
R_SDC_SPI_Read()	リード処理
R_SDC_SPI_Write()	ライト処理
R_SDC_SPI_GetCardStatus()	カードステータス情報取得処理
R_SDC_SPI_GetCardInfo()	レジスタ情報取得処理
R_SDC_SPI_SetLogHdlAddress()	LONGQ FIT モジュールのハンドラアドレス登録処理
R_SDC_SPI_Log()	エラーログ取得処理(注 1)
R_SDC_SPI_GetVersion()	ドライバのバージョン情報取得処理

注 1：別途 LONGQ FIT モジュールが必要です。

1.4　ハードウェア設定

1.4.1　ハードウェア構成例

図 1-2 に端子接続図を示します。MCU のシリアルインタフェースにより端子名が異なります。使用端子と機能を参照し、使用する MCU の端子に割り当ててください。

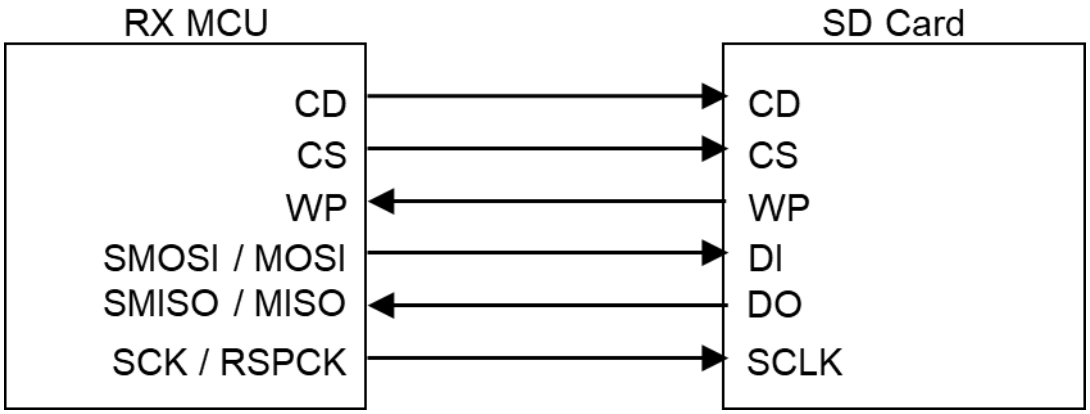
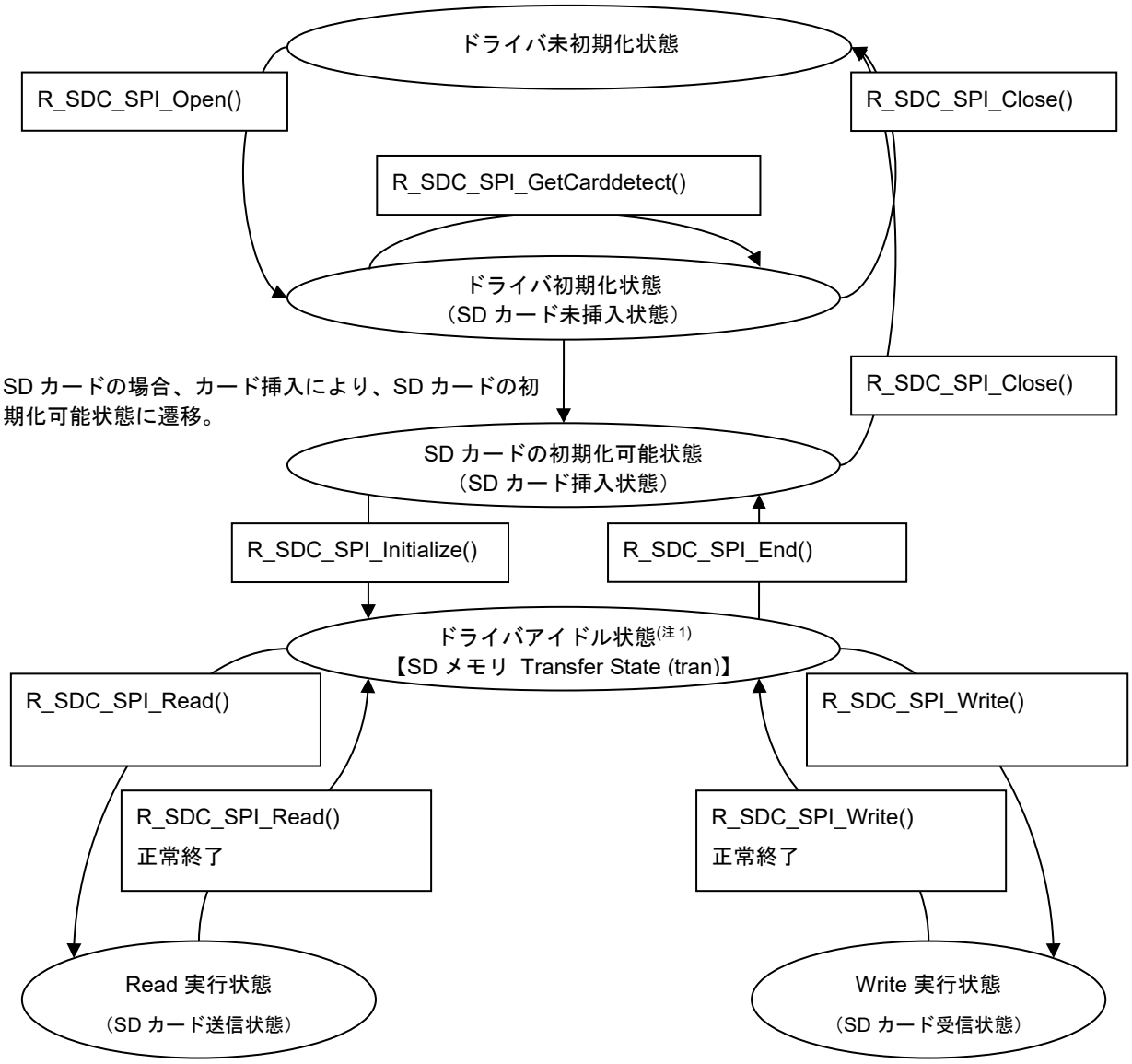


図 1-2 端子接続図

1.5　状態遷移図

図 1-3 に本ドライバの状態遷移図を示します。



注 1：【】は、SD Specifications Part 1 Physical Layer Simplified Specification 記載の SD メモリカードの状態を示す。

図 1-3　SD メモリカードドライバの状態遷移図

1.6　処理例

1.6.1　基本制御

(1)　サポートコマンドについて

SPI モード SD メモリカードドライバは、以下のコマンドを使用します。

以下の表は、SD カードコマンドと本 SPI モード SD メモリカードドライバのサポート状況を示したものです。

表 1-5　サポートコマンド一覧（○：サポート、×：未サポート）

コマンド	本モジュール	備考
CMD0	○	SD メモリ初期化で使用
CMD1	×	
CMD5	×	
CMD6	○	SD メモリ初期化で使用
CMD8	○	SD メモリ初期化で使用
CMD9	○	SD メモリ初期化で使用
CMD10	○	SD メモリ初期化で使用
CMD12	○	SD メモリのリード／ライト処理で使用
CMD13	○	SD メモリ初期化で使用
CMD16	×	
CMD17	○	SD メモリのリード処理で使用
CMD18	○	SD メモリのリード処理で使用
CMD24	○	SD メモリのライト処理で使用
CMD25	○	SD メモリのライト処理で使用
CMD27	×	
CMD28	×	
CMD29	×	
CMD30	×	
CMD32	×	
CMD33	×	
CMD38	×	
CMD42	×	
CMD55	○	SD メモリ初期化で使用
CMD56	×	
CMD58	○	SD メモリ初期化で使用
CMD59	○	SD メモリ初期化で使用
ACMD13	○	SD メモリ初期化で使用
ACMD18	×	
ACMD22	×	
ACMD23	○	SD メモリ初期化で使用
ACMD25	×	
ACMD26	×	
ACMD38	×	
ACMD41	○	SD メモリ初期化で使用
ACMD42	×	
ACMD51	○	SD メモリ初期化で使用

(2)　データバッファと SD カード上のデータとの関係

SD カードドライバは、送信／受信データポインタを引数として設定します。RAM 上のデータバッファのデータ並びと送信／受信順番の関係は図 1-4 に示すように、エンディアンに関係なく、送信データバッファの並びの順に送信し、また、受信の順に受信データバッファに書き込みます。

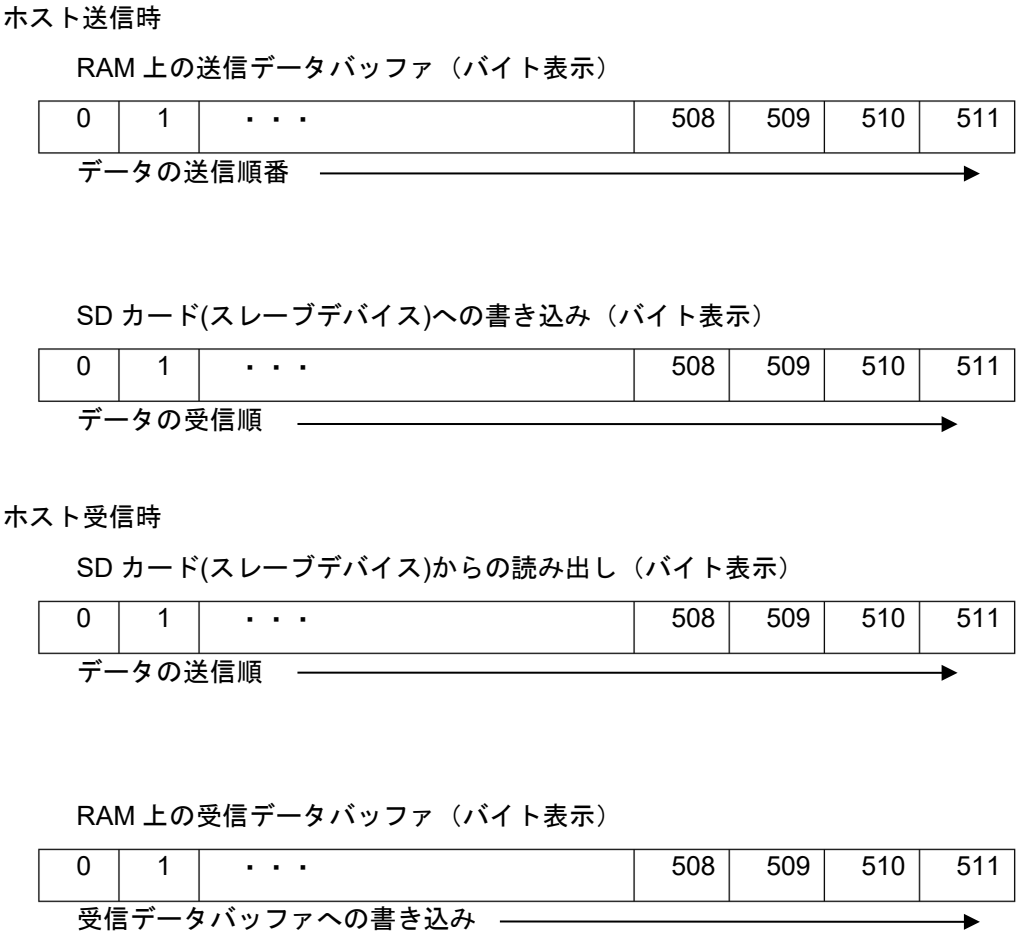


図 1-4　転送データの格納

(3)　動作電圧設定について

R_SDC_SPI_Initialize()関数の引数には SD カードの動作電圧を設定する必要があります。

2 回目以降初期化処理を行う場合、R_SDC_SPI_End()関数をコールし、本モジュールの状態を「SD カードの初期化可能状態」にした後、SD カードを抜去してください。その後、再度挿入し動作電圧を設定し直して、再度初期化処理を行ってください。

なお、本モジュールでサポートしている SD カード動作電圧及びインタフェース電圧は 3.3V です。このため R_SDC_SPI_Initialize()関数で動作電圧として 3.3V を設定してください。

(4)　SD カードステータス確認

SD カードの操作を行う上で、SD カードの挿抜検出を行う必要があります。
挿抜検出は表 1-6 に示す関数によるソフトウェアポーリングで検出することができます。

表 1-6　確認するステータス

分類	ステータス	備考
SD カード挿抜	SD カード 挿入／抜去状態	R_SDC_SPI_GetCardDetection()関数で検出可能

1.6.2　エラー時の制御

(1)　エラー発生時の処理方法

リード処理／ライト処理等でエラーが発生した場合、処理のリトライを推奨します。
処理のリトライにも関わらずエラーが発生する場合、SD カードの挿抜を実施し、SD カードを再初期化してください。SD カードの挿抜に関わる処理方法は、「4.1 SD カードの挿入と電源投入の手順」「4.2 SD カードの抜去と電源停止の手順」を参照してください。
また、SD カードドライバを FAT ファイルシステムと組み合わせて使用する場合、SD カードの挿抜処理の前に、事前にユーザアプリケーションでマウント、アンマウントなどの処理を実行してください。

(2)　Transfer State (tran)遷移後のエラー終了処理

Transfer State (tran)遷移後にエラーが発生した場合、データ転送の有無に関わらず、CMD12 を発行します。CMD12 の発行は、Transfer State (tran)状態に遷移させることを目的としています。但し、ライト処理中に CMD12 が発行された際、SD カードがビジー状態に遷移する場合があります。そのため、次の R_SDC_SPI_Read()関数／R_SDC_SPI_Write()関数コール時にエラーを返す場合があります。

1.7　制限事項

1.7.1　SD カードの電源供給の注意事項

SD カード挿入後、SD カードの仕様に基づいて、SD カード電源を供給する必要があります。SD Specifications Part 1 Physical Layer Simplified Specification の Power Scheme の章を参照してください。

特に、SD カードの抜去後の SD カードの再挿入制御、もしくは SD カードの電源の切断後の再投入制御を行う場合は、電圧値と電圧維持期間についての規定を参照し、システム側で回路や切断／再投入の制御タイミングを設けてください。正しい時間調整が必要です。

また、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時間待ち処理は、アプリケーションプログラムに設ける必要があります。

1.7.2　ソフトウェア・ライトプロテクト対応について

SPI モード SD メモリカードドライバは、ソフトウェアよるプロテクト状態制御機能をサポートしていません。

1.7.3　SDUC カードへの対応について

SDUC カードは SD Specifications Part 1 Physical Layer Simplified Specification の仕様上、SPI モードには対応しておりません。このため、SDUC カードを接続した場合の動作は保証できません。

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 が以下のいずれかの機能をサポートしている必要があります。

- SCI
- RSPI

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。

- r_bsp(Rev.5.20 以上)
- r_gpio_rx
- r_memdrv_rx
- r_sci_rx
- r_rspi_rx
- r_dmaca_rx
- r_dtc_rx
- r_longq_rx

また、以下のコード生成モジュールに依存しています。

- CRC 演算器

2.3 サポートされているツールチェーン

本モジュールは「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

なし

2.5 ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_sdc_spi_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_sdc_spi_rx_config.h で選択します。

2.6 整数型

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

2.7　コンパイル時の設定

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

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

Configuration options in r_sdc_spi_rx_config.h	
SDC_SPI_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は “BSP_CFG_PARAM_CHECKING_ENABLE”	1：ビルド時にパラメータチェックの処理をコードに含めます。 0：ビルド時にパラメータチェックの処理をコードから省略します。 このオプションに BSP_CFG_PARAM_CHECKING_ENABLE を設定すると、システムのデフォルト設定が使用されます。
SDC_SPI_CFG_ERROR_LOG_ACQUISITION ※デフォルト値は“0”	LONGQ FIT モジュールを使用するエラーログ取得機能を利用する場合に 1 を定義してください。 この機能を利用する場合、LONGQ FIT モジュールを組み込む必要があります。
SDC_SPI_CFG_CH0_CD_ENABLE ※デフォルト値は”1”	チャネル 0 の CD 端子を使用するか選択します。 0: CD 端子を使用しない 1: CD 端子を使用する
SDC_SPI_CFG_CH0_CD_PORT ※デフォルト値は”0”	チャネル 0 の CD 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号
SDC_SPI_CFG_CH0_CD_BIT ※デフォルト値は”0”	チャネル 0 の CD 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_CH0_CS_ENABLE ※デフォルト値は”1”	チャネル 0 の CS 端子を使用するか選択します。 0: CS 端子を使用しない 1: CS 端子を使用する
SDC_SPI_CFG_CH0_CS_PORT ※デフォルト値は”0”	チャネル 0 の CS 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号
SDC_SPI_CFG_CH0_CS_BIT ※デフォルト値は”0”	チャネル 0 の CS 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_CH0_WP_ENABLE ※デフォルト値は”1”	チャネル 0 の WP 端子を使用するか選択します。 0: WP 端子を使用しない 1: WP 端子を使用する
SDC_SPI_CFG_CH0_WP_PORT ※デフォルト値は”0”	チャネル 0 の WP 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号
SDC_SPI_CFG_CH0_WP_BIT ※デフォルト値は”0”	チャネル 0 の WP 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_CH1_CD_ENABLE ※デフォルト値は”1”	チャネル 1 の CD 端子を使用するか選択します。 0: CD 端子を使用しない 1: CD 端子を使用する
SDC_SPI_CFG_CH1_CD_PORT ※デフォルト値は”0”	チャネル 1 の CD 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号

Configuration options in r_sdc_spi_rx_config.h	
SDC_SPI_CFG_CH1_CD_BIT ※デフォルト値は”0”	チャネル 1 の CD 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_CH1_CS_ENABLE ※デフォルト値は”1”	チャネル 1 の CS 端子を使用するか選択します。 0: CS 端子を使用しない 1: CS 端子を使用する
SDC_SPI_CFG_CH1_CS_PORT ※デフォルト値は”0”	チャネル 1 の CS 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号
SDC_SPI_CFG_CH1_CS_BIT ※デフォルト値は”0”	チャネル 1 の CS 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_CH1_WP_ENABLE ※デフォルト値は”1”	チャネル 1 の WP 端子を使用するか選択します。 0: WP 端子を使用しない 1: WP 端子を使用する
SDC_SPI_CFG_CH1_WP_PORT ※デフォルト値は”0”	チャネル 1 の WP 端子で使用する I/O ポート番号を指定してください。 0x00～0x19: I/O ポート番号
SDC_SPI_CFG_CH1_WP_BIT ※デフォルト値は”0”	チャネル 1 の WP 端子で使用する I/O ポートのビット番号を指定してください。 0～7: 使用する I/O ポートのビット番号
SDC_SPI_CFG_SBLK_NUM ※デフォルト値は”1”	シングルブロックライトコマンドで書き込むブロック数の最大値を設定します。 1～255: 書き込むブロック数の最大値
SDC_SPI_CFG_USE_SC_CRC ※デフォルト値は “0”	スマート・コンフィグレータの CRC 演算器を使用して CRC 演算を行う場合に 1 を定義してください。 0: スマート・コンフィグレータの CRC 演算器を使用しない 1: スマート・コンフィグレータの CRC 演算器を使用する この機能を使用する場合、スマート・コンフィグレータの Config_CRC(CRC 演算器)を以下の設定で組み込む必要があります。 ・生成多項式: CRC_CCITT ・ビット順: MSB ・初期値: 0x0000

2.8　コードサイズ

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

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

下表の値は下記条件で確認しています。

モジュールリビジョン: r_sdc_spi_rx rev.1.00

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00
(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 8.3.0.202311
(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)
IAR C/C++ Compiler for Renesas RX version 5.10.1
(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

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ					
		Renesas Compiler		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
RX140	ROM	8085 バイト	7507 バイト	9722 バイト	8944 バイト	10371 バイト	10371 バイト
	RAM	15 バイト		0 バイト		16 バイト	
	最大使用スタックサイズ	176 バイト		-		108 バイト	

コードサイズは以下の条件で確認しています。

・SPI モード SD メモリカードドライバ FIT の設定

CH0 CD pin Enable : Enable

CH0 CS pin Enable : Enable

CH0 WP pin Enable : Enable

Use CRC smart configuration : Disable

・MEMDRV FIT の設定

Device 0 data transfer mode : CPU transfer

Device 0 drive : SCI clock synchronous control FIT module

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_sdc_spi_rx_if.h に記載されています。

(1)　sdc_spi_cfg_t 構造体定義

```
typedef struct
{
    uint32_t    mode;
    uint32_t    voltage;
}sdc_spi_cfg_t;
```

(2)　sdc_spi_access_t 構造体定義

```
typedef struct
{
    uint8_t     *p_buff;
    uint32_t    lbn;
    int32_t     cnt;
    uint32_t    write_mode;
}sdc_spi_access_t;
```

(3)　sdc_spi_card_status_t 構造体定義

```
typedef struct
{
    uint32_t card_sector_size;
    uint32_t max_block_number;
    uint8_t  write_protect;
    uint8_t  csd_structure;
}sdc_spi_card_status_t;
```

(4)　sdc_spi_card_reg_t 構造体定義

```
typedef struct
{
    uint32_t    ocr[1];
    uint32_t    cid[4];
    uint32_t    csd[4];
    uint32_t    scr[2];
    uint32_t    sdstatus[4];
}sdc_spi_card_reg_t;
```

2.10　戻り値／エラーコード

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

SD カードドライバの API 関数は、その処理の途中でエラーが発生した場合、戻り値にエラーコードを返します。

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

表 2-1　エラーコード

マクロ定義	値	意味	
SDC_SPI_SUCCESS	0	正常終了	・R_SDC_SPI_GetCardDetection()関数以外は正常終了 ・R_SDC_SPI_GetCardDetection()関数は以下の通りです 1)CD 端子を使用する場合に CD 端子に設定したポートレベルが Low の場合 2)CD 端子を使用しない場合
SDC_SPI_ERR	-1	一般エラー	R_SDC_SPI_Open()関数が実行されていない、パラメータのエラー等
SDC_SPI_ERR_WP	-2	ライトプロテクトエラー	ライトプロテクト状態の SD カードへのライト
SDC_SPI_ERR_CRC	-7	CRC エラー	CRC エラーを検出した
SDC_SPI_ERR_ILLEGALCMD	-83	異常コマンドエラー	R1 レスポンスのカードステータスエラー（ILLEGAL_COMMAND）
SDC_SPI_ERR_ADDRESS_BOUNDARY	-89	ワークアドレスエラー	引数のバッファアドレスエラー バッファアドレスに問題があります。
SDC_SPI_ERR_INTERNAL	-99	内部エラー	ドライバ内部で使用しているモジュールのエラー

2.11　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)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) e² 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
e² studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに 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) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT 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 e² 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「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。

(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。

2.12　for 文、while 文、do while 文について

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

以下に記述例を示します。

```
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

3.1 R_SDC_SPI_Open()

SD カードドライバの API を使用する際、最初に使用する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_Open(
uint32_t      card_no,
uint8_t       dev_no,
void           *p_sdc_spi_workarea
)
```

Parameters

card_no

SD カード番号　使用する SD カード番号（0 起算）

dev_no

チャネル番号　使用する MEMDRV FIT チャネル番号（0 起算）

*p_sdc_spi_workarea

4 バイト境界のワーク領域のポインタ

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_ADDRESS_BOUNDARY	ドライバで使用するメモリ領域アドレスに問題があります。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

引数 card_no で制御する汎用入出力ポートのリソース取得と SPI モード SD メモリカードドライバの初期化、引数 dev_no で指定した MEMDRV FIT モジュールを初期化します。

SPI モード SD メモリカードドライバのクローズ処理を終了させるまで、ワーク領域を保持し、その内容をアプリケーションプログラムで変更しないでください。

Example

```
uint32_t    g_sdc_spi_work[160/sizeof(uint32_t)];

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

if (R_SDC_SPI_Open(SDC_SPI_CARD0, MEMDRV_CH0, &g_sdc_spi_work) !=
SDC_SPI_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、SD カードの CD 端子、WP 端子、CS 端子を使用する場合、使用する端子に応じてコンフィグレーションオプションの設定を変更してください。

本関数が正常終了しない場合、R_SDC_SPI_GetVersion()関数、R_SDC_SPI_Log()関数以外のライブラリ関数が使用できません。

3.2　R_SDC_SPI_Close()

使用中の SD カードドライバのリソースを開放する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_Close(
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

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードドライバの全ての処理を終了し、引数 card_no で設定したチャネルのリソースを解放します。

R_SDC_SPI_Open()関数で設定したワーク領域は、本関数実行後は使用されません。他用途に使用できます。

Example

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

if (R_SDC_SPI_Close(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
      /* Error */
}
```

Special Notes

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

3.3　R_SDC_SPI_GetCardDetection()

SD カードの挿入状態を確認する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardDetection(
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

SDC_SPI_SUCCESS	CD 端子に設定したポートレベルは Low 又は CD 端子は未使用
SDC_SPI_ERR	パラメータ設定に問題がある 又は CD 端子に設定したポートレベルは High。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードの挿入状態を確認します。

コンパイル時の設定で指定した CD 端子のレベルが Low 又は CD 端子を未使用の場合、SDC_SPI_SUCCESS を返します。

コンパイル時の設定で指定した CD 端子のレベルが High 又はパラメータ設定に問題がある場合、SDC_SPI_ERR を返します。

Example

```
if (R_SDC_SPI_GetCardDetection(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
            /* Error */
}
```

Special Notes

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

本関数実行前に R_SDC_SPI_Open()関数によるドライバのオープン処理が必要です。

SD カード挿抜検出端子として、SD カードソケットの CD 端子に接続した汎用入出力ポートを使用します。

SD カード検出後、SD カードへの電源供給処理を実行してください。

3.4　R_SDC_SPI_Initialize()

SD カードを初期化し、「SD カードの初期化可能状態」から「ドライバアイドル状態」にする関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_Initialize(
uint32_t        card_no,
sdc_spi_cfg_t   *p_sdc_spi_config,
uint32_t        init_type
)
```

Parameters

card_no

SD カード番号　　使用する SD カード番号（0 起算）

*p_sdc_spi_config

動作設定情報構造体

mode　：動作モード

0x00000000 (固定値。ソフトウェア転送による設定に相当する値です。)

voltage　：電源電圧

0x00200000 (固定値。動作電圧設定の 3.3V に相当する値です。)

init_type　：初期化タイプ

初期化対象を指定してください。値は「表 3-1　SD カードドライバ　動作モード (mode) (mode)」のメディア対応方式のマクロ定義を使用してください。

表 3-1　SD カードドライバ　動作モード (mode)

種別	マクロ定義	値（ビット）	定義
メディア対応方式	SDC_SPI_MODE_MEM	0x00000000	SD メモリカード／SD メモリ

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_ILLEGALCMD	異常コマンドエラーが発生しました。
SDC_SPI_ERR_CRC	CRC エラーを検出しました。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードの初期化処理を行います。SD カードの検出後に、本関数を実行してください。

戻り値が、SDC_SPI_SUCCESS の場合、SD カードは Transfer State (tran)へ遷移し、SD カードのリード／ライトアクセスが可能になります。

Example

```
sdc_spi_cfg_t     sdc_spi_config;

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

sdc_spi_config.mode = 0x00000000;
sdc_spi_config.voltage = 0x00200000;
if (R_SDC_SPI_Initialize(SDC_SPI_CARD0, &sdc_spi_config, SDC_SPI_MODE_MEM) !=
SDC_SPI_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_SDC_SPI_Open()関数によるドライバのオープン処理が必要です。

エラー終了の場合、R_SDC_SPI_End()関数をコールし、「SD カードの初期化可能状態」にした後、再度初期化処理を行ってください。

初期化正常終了後、2 回目以降初期化処理を行う前に R_SDC_SPI_End()関数をコールし終了処理を行ってください。

3.5　R_SDC_SPI_End()

ワーク領域の値をクリアし、「ドライバアイドル状態」から「SD カードの初期化可能状態」にする関数です。本関数を実行した場合でも SD カードの状態は変化しません。

Format

```
sdc_spi_status_t R_SDC_SPI_End(
uint32_t      card_no,
uint32_t      end_type
)
```

Parameters

card_no

SD カード番号	使用する SD カード番号（0 起算）

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードの終了処理を行い、SD カードを取り外し可能な状態にします。

Example

```
if (R_SDC_SPI_End(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
      /* Error */
}

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

Special Notes

本関数実行後、SD カードを抜去する場合、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_SDC_SPI_Open()関数によるドライバのオープン処理を行ってください。

3.6　R_SDC_SPI_Read()

リード処理を実行する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_Read(
uint32_t                card_no,
sdc_spi_access_t        *p_sdc_spi_access
)
```

Parameters

card_no

SD カード番号　　使用する SD カード番号（0 起算）

*p_sdc_spi_access

アクセス情報構造体

*p_buff　：読み出しバッファポインタ

4 バイト境界のアドレスを設定してください。

lbn　：読み出し開始ブロック番号

cnt　：ブロック数

設定できる最大値は、65,535 です。

write_mode　：書き込みモード（設定不要）

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_ILLEGALCMD	異常コマンドエラーが発生しました。
SDC_SPI_ERR_CRC	CRC エラーを検出しました。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

引数 p_sdc_spi_access の lbn で設定したブロックから引数 p_sdc_spi_access の cnt ブロック分のデータを読み出し、引数 p_sdc_spi_access の p_buff に格納します。

Example

```
#define TEST_BLOCK_CNT  (4)
#define BLOCK_NUM       (512)

sdc_spi_access_t        sdc_spi_access;
uint32_t    g_test_r_buff[(TEST_BLOCK_CNT*BLOCK_NUM)/sizeof(uint32_t)];

sdc_spi_access.p_buff   = (uint8_t *)&g_test_r_buff[0];
sdc_spi_access.lbn      = 0x10000000;
sdc_spi_access.cnt      = TEST_BLOCK_CNT;

if(R_SDC_SPI_Read(SDC_SPI_CARD0, &sdc_spi_access) != SDC_SPI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDC_SPI_Open()関数によるドライバのオープン処理と R_SDC_SPI_Initialize()関数による初期化処理が必要です。

リードのエラー終了の場合、再度リード処理を行うことを推奨します。

1 ブロックのサイズは、512 バイトです。

3.7　R_SDC_SPI_Write()

ライト処理を実行する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_Write(
uint32_t                card_no,
sdc_spi_access_t        *p_sdc_spi_access
)
```

Parameters

card_no

SD カード番号　　使用する SD カード番号（0 起算）

*p_sdc_spi_access

アクセス情報構造体

*p_buff　：書き込みバッファポインタ

4 バイト境界のアドレスを設定してください。

lbn　：書き込み開始ブロック番号

cnt　：ブロック数

設定できる最大値は、65,535 です。

write_mode　：書き込みモード

設定値は「表 3-2　SD カードドライバ　書き込みモード (write_mode)」のマクロ定義に示す種別から 1 つ設定してください。

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。
SDC_SPI_ERR_WP	ライトプロテクト状態の SD カードへのデータ書き込みが行われました。
SDC_SPI_ERR_ILLEGALCMD	異常コマンドエラーが発生しました。
SDC_SPI_ERR_CRC	CRC エラーを検出しました。
SDC_SPI_ERR_INTERNAL	ドライバ内部で使用しているモジュールでエラーが発生しました。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

引数 p_sdc_spi_access の lbn で設定したブロックから引数 p_sdc_spi_access の cnt ブロック分の領域に引数 p_sdc_spi_access の p_buff のデータを書き込みます。

表 3-2　SD カードドライバ　書き込みモード (write_mode)

種別	マクロ定義	値（ビット）
プレイレーズを伴う書き込み	SDC_SPI_WRITE_WITH_PREERASE	0x00000000
通常の書き込み	SDC_SPI_WRITE_OVERWRITE	0x00000001

Example

```
#define TEST_BLOCK_CNT  (4)
#define BLOCK_NUM       (512)

sdc_spi_access_t  sdc_spi_access;
uint32_t          g_test_w_buff[(TEST_BLOCK_CNT*BLOCK_NUM)/sizeof(uint32_t)];

sdc_spi_access.p_buff   = (uint8_t *)&g_test_w_buff[0];
sdc_spi_access.lbn      = 0x10000000;
sdc_spi_access.cnt      = TEST_BLOCK_CNT;
sdc_spi_access.write_mode= SDC_SPI_WRITE_WITH_PREERASE;

if(R_SDC_SPI_Write(SDC_SPI_CARD0, &sdc_spi_access) != SDC_SPI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDC_SPI_Open()関数によるドライバのオープン処理と R_SDC_SPI_Initialize()関数による初期化処理が必要です。

ライトのエラー終了の場合、再度ライト処理を行うことを推奨します。

転送ブロック数が 65,535 を超える場合は、分割して、コールしてください。M3S-TFAT-Tiny モジュール等、上位アプリケーションプログラムからコールする場合に注意してください。

1 ブロックのサイズは、512 バイトです。

3.8　R_SDC_SPI_GetCardStatus()

カードステータス情報を取得する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardStatus(
uint32_t                    card_no,
sdc_spi_card_status_t       *p_sdc_spi_card_status
)
```

Parameters

card_no

SD カード番号　　使用する SD カード番号（0 起算）

*p_sdc_spi_card_status

カードステータス情報構造体ポインタ

card_sector_size　：ユーザ領域ブロック数

max_block_number　：最大ブロック数

write_protect　：ライトプロテクト情報（「表 3-3　ライトプロテクト情報 (write_protect)を参照」

csd_structure　：CSD 情報

0 : Standard Capacity カード（SDSC）

1 : High Capacity カード（SDHC, SDXC）

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードのカードステータス情報を取得し、カードステータス情報構造体に格納します。

表 3-3　ライトプロテクト情報 (write_protect)

マクロ定義	値（ビット）	定義
SDC_SPI_WP_OFF	0x00	ライトプロテクト解除状態
SDC_SPI_WP_HW	0x01	ハードウェア・ライトプロテクト状態
SDC_SPI_WP_TEMP	0x02	CSD レジスタ TEMP_WRITE_PROTECT ビット ON
SDC_SPI_WP_PERM	0x04	CSD レジスタ PERM_WRITE_PROTECT ビット ON
SDC_SPI_WP_ROM	0x10	SD ROM

Example

```
sdc_spi_card_status_t   sdc_spi_card_status;

if (R_SDC_SPI_GetCardStatus(SDC_SPI_CARD0, &sdc_spi_card_status) !=
SDC_SPI_SUCCESS)
{
      /* Error */
}
```

Special Notes

本関数実行前に R_SDC_SPI_Open()関数によるドライバのオープン処理と R_SDC_SPI_Initialize()関数による初期化処理が必要です。

3.9　R_SDC_SPI_GetCardInfo()

SD カードレジスタ情報を取得する関数です。

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardInfo(
uint32_t                card_no,
sdc_spi_card_reg_t      *p_sdc_spi_card_reg
)
```

Parameters

card_no

SD カード番号　　使用する SD カード番号（0 起算）

*p_sdc_spi_card_reg

SD カードのレジスタ情報構造体ポインタ

ocr[1] ：SD メモリ OCR 情報

cid[4] ：SD メモリ CID 情報

csd[4] ：SD メモリ CSD 情報

scr[2] ：SD メモリ SCR 情報

sdstatus[4] ：SD メモリ SD Status 情報

Return Values

SDC_SPI_SUCCESS	正常終了
SDC_SPI_ERR	パラメータ設定に問題があります。

Properties

r_sdc_spi_rx_if.h にプロトタイプ宣言されています。

Description

SD カードレジスタ情報を取得し、SD カードのレジスタ情報構造体に格納します。

Example

```
sdc_spi_card_reg_t      sdc_spi_card_reg;

if (R_SDC_SPI_GetCardInfo(SDC_SPI_CARD0, &sdc_spi_card_reg) != SDC_SPI_SUCCESS)
{
            /* Error */
}
```

Special Notes

本関数実行前に R_SDC_SPI_Open()関数によるドライバのオープン処理と R_SDC_SPI_Initialize()関数による初期化処理が必要です。

3.10　R_SDC_SPI_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

sdc_spi_status_t R_SDC_SPI_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

SDC_SPI_SUCCESS	正常終了

Properties

r_sdc_spi_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 SD カードドライバに設定します。

Example

```
#define SDC_SPI_USER_LONGQ_MAX                          (8)
#define SDC_SPI_USER_LONGQ_BUFSIZE                (SDC_SPI_USER_LONGQ_MAX * 4)
#define SDC_SPI_USER_LONGQ_IGN_OVERFLOW   (1)

uint32_t    g_sdc_spi_user_longq_buf[SDC_SPI_USER_LONGQ_BUFSIZE];
static longq_hdl_t      sdc_spi_user_long_que;
longq_err_t             err = LONGQ_SUCCESS;
uint32_t                user_long_que = 0;

err = R_LONGQ_Open(g_sdc_spi_user_longq_buf,
                   SDC_SPI_USER_LONGQ_BUFSIZE,
                   SDC_SPI_USER_LONGQ_IGN_OVERFLOW,
                   &sdc_spi_user_long_que);
if (LONGQ_SUCCESS != err)
{
            /* Error */
}
user_long_que = (uint32_t)sdc_spi_user_long_que;
if (R_SDC_SPI_SetLogHdlAddress(user_long_que) != SDC_SPI_SUCCESS)
{
            /* Error */
}
```

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_SDC_SPI_Open()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。

3.11　R_SDC_SPI_Log()

エラーログを取得する関数です。

Format

```
uint32_t R_SDC_SPI_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

r_sdc_spi_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)

sdc_spi_cfg_t                 sdc_spi_config;

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

sdc_spi_config.mode = 0x0000;
sdc_spi_config.voltage = 0x00200000;
if (R_SDC_SPI_Initialize(SDC_SPI_CARD0, &sdc_spi_config, SDC_SPI_MODE_MEM) !=
SDC_SPI_SUCCESS)
{
      /* Error */
      R_SDC_SPI_Log(USER_DRIVER_ID, USER_LOG_MAX, USER_LOG_ADR_MAX);
}
```

Special Notes

別途 LONGQ FIT モジュールを組み込んでください。

3.12　R_SDC_SPI_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_SDC_SPI_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_sdc_spi_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_SDC_SPI_GetVersion();
```

Special Notes

なし

4. 端子設定

本 SPI モード SD メモリカードドライバモジュールを使用し SD カードを制御するためには、「1.4 ハードウェア設定」に示す端子を使用する必要があります。

各端子は、SPI モード SD メモリカードドライバ、「1.1 SPI モード SD メモリカードドライバとは」に示す MEMDRV FIT モジュール、および SCI FIT モジュールまたは RSPI FIT モジュールが使用します。各モジュールが端子を使用できるようにするためには、端子への周辺機能の割り付けが必要です。

端子への機能割り付けは、統合開発環境 e^2 studio のスマート・コンフィグレータで設定することができます。表 4-1 を参照し、各端子を設定してください。設定に応じ、端子を制御する関数が記述されたソースファイル「Pin.c」、「r_sci_rx_pinset.c」、「r_rspi_rx_pinset.c」が生成されます。生成された関数を表 4-1 に示す呼び出し元関数からコールすることで端子が設定され、SD カードを制御できるようになります。

表 4-1　SPI モードで SD カードを制御するための端子設定

端子	周辺機能割り付け	端子設定関数を呼び出すモジュール		端子設定関数 (スマート・コンフィグレータ生成)
		FIT モジュール	関数	
CDn(注 1、注 5)	GPIO	SPI モード SD メモリカードドライバ	• R_SDC_SPI_Open()	• GPIO FIT モジュール(注 5) R_GPIO_PinDirectionSet ()
CSn(注 1、注 5)				
WPn(注 1、注 5)				
SCKm(注 2)	SCI(注 3)	MEMDRV FIT モジュール	• r_memdrv_sci.c sci_init_ports()	• r_sci_pinset.c R_SCI_PinSet_SCIm()(注 2)
SMOSIm(注 2)				
SMISOm(注 2)				
RSPCKm(注 2)	RSPI(注 4)	MEMDRV FIT モジュール	• r_memdrv_rspi.c rspi_init_ports()	• r_rspi_pinset.c R_RSPI_PinSet_RSPIm()(注 2)
MOSIm(注 2)				
MISOm(注 2)				

注 1　n は、カード番号を示します。

注 2　m は、チャネル番号を示します。使用するチャネルにより番号は変化します。

注 3　RSPI を使用する場合、SCI の設定は不要です。

注 4　SCI を使用する場合、RSPI の設定は不要です。

注 5　スマート・コンフィグレータ による端子設定関数は生成せず、GPIO FIT モジュールを使用して制御します。

端子設定の手順は「4.1 SD カードの挿入と電源投入の手順」と「4.2 SD カードの抜去と電源停止の手順」を参照してください。

具体的なコーディング例は、本アプリケーションノート同梱のデモプロジェクトを参照してください。

4.1　SD カードの挿入と電源投入の手順

SD カードの挿入と電源投入の手順を図 4-1、表 4-2 に示します。SD カードの挿入は、R_SDC_SPI_Open() 関数の正常終了後、SD カードへの電源供給停止状態、かつ SCI 又は RSPI の出力端子が L 出力の状態で行ってください。

ここでの電源とは SD カード用電源回路、電源制御端子とは SD カード用電源回路の出力を制御するために割り当てられた RX マイコンの端子とします。

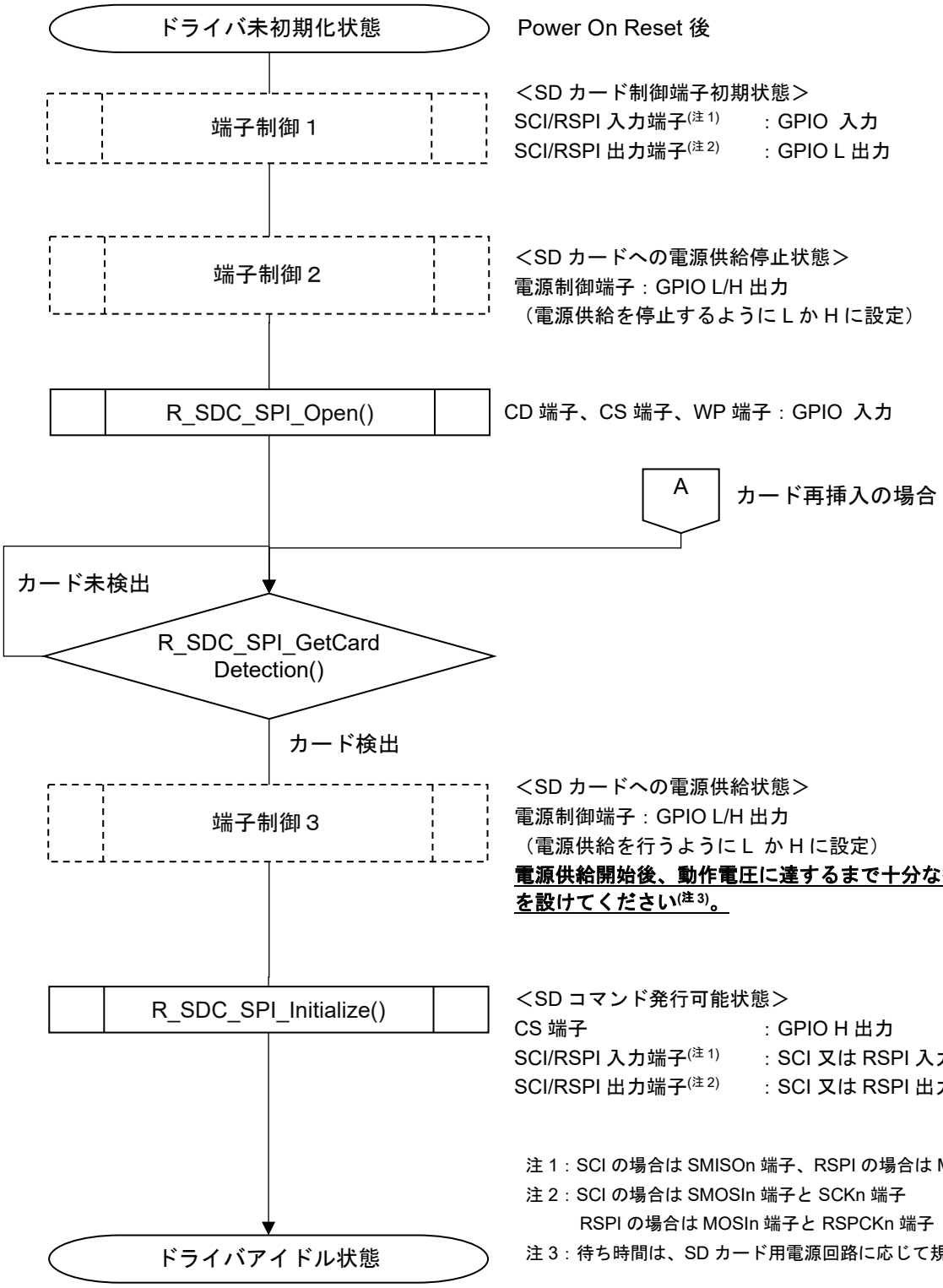


図 4-1　SD カードの挿入と電源投入の手順

表 4-2　SD カード挿入時の端子制御

処理	対象端子	端子設定	実行後の端子状態
端子制御 1	SCI 又は RSPI 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：汎用入出力ポート PMR 設定：周辺モジュール	GPIO 入力
	SCI 又は RSPI 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 (注 5)	電源制御端子	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（電源供給を停止する値を設定） PDR 設定：出力	GPIO L/H 出力 （電源供給停止状態）
端子制御 3 (注 6)	電源制御端子	PODR 設定：L 出力／H 出力（電源供給を行う値を設定）	GPIO L/H 出力 （電源供給状態）

注 1：SCI の場合は SMISO 端子、RSPI の場合は MISO 端子に設定してください。

注 2：SCI の場合は SMOSI 端子と SCK 端子、RSPI の場合は MOSI 端子と RSPCK 端子に設定してください。

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：本ドライバを応用するシステムに応じ、MCU 内蔵入力プルアップ抵抗を有効または無効に設定してください。

注 5：デモプロジェクトでは、r_sdc_spi_demo_power_init() で制御します。

注 6：デモプロジェクトでは、r_sdc_spi_demo_power_on() で制御します。

4.2　SD カードの抜去と電源停止の手順

SD カードの抜去と電源停止の手順を図 4-2、表 4-3 に示します。SD カードの抜去は、ドライバアイドル状態での R_SDC_SPI_End()関数の正常終了後、SD カードへの電源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、同様の手順で電源供給を停止してください。

ここでの電源とは SD カード用電源回路、電源制御端子とは SD カード用電源回路の出力を制御するために割り当てられた RX マイコンの端子とします。

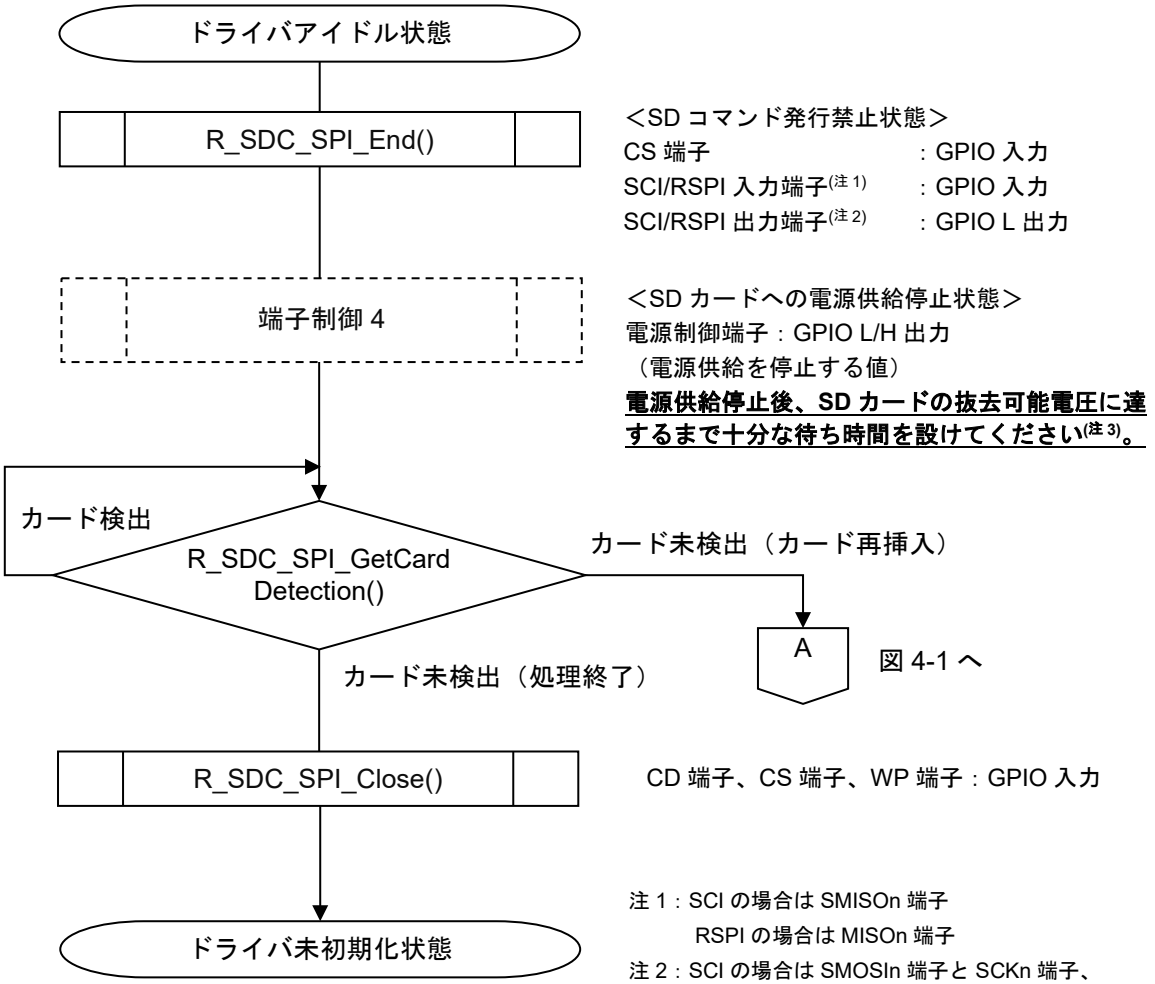


図 4-2　SD カードの抜去と電源停止の手順

表 4-3　SD カード抜去時の端子制御

処理	対象端子	端子設定	実行後の端子状態
端子制御 4(注 1)	電源制御端子	PODR 設定：L 出力／H 出力（電源供給を停止する値に設定）	GPIO L/H 出力 （電源供給停止状態）

注 1：デモプロジェクトでは、r_sdc_spi_demo_power_off() で制御します。

5.　デモプロジェクト

5.1　概要

本アプリケーションノートには、SPI モード SD メモリカードドライバの使用方法を説明するためのデモプロジェクトを同梱しています。

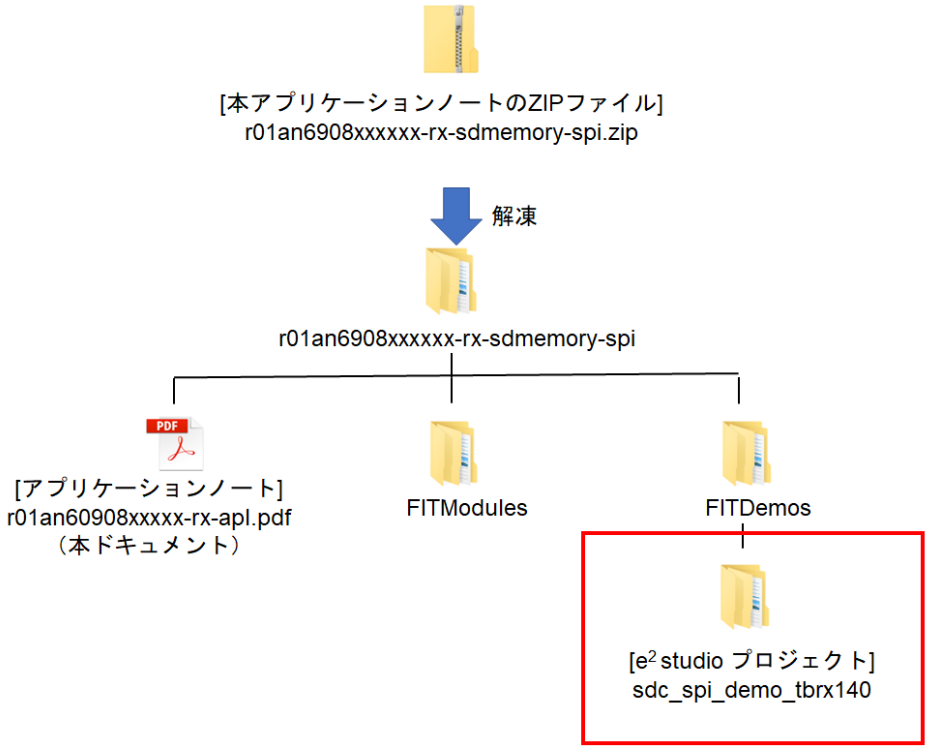


図 5-1　アプリケーションノートの構成

デモプロジェクトは、以下の処理を順次行います。

- SD カードの挿入と電源投入
- 本ドライバを使用した SD カードへのリード／ライト
- SD カードの抜去と電源停止

デモプロジェクトは、1 個の SD カードに対しリード／ライトの処理を行います。
デモプロジェクトは、SCI モジュールを使用、CPU 転送でリード／ライト処理を行います。
処理の詳細は、「5.4 デモプロジェクトのフローチャート」をご参照ください。

5.2　動作確認環境

表 5-1 にデモプロジェクトを動作させるためのハードウェアや設定などの条件を示します。
図 5-2 に評価ボードと SD カードなどハードウェアの構成を示します。
図 5-3 に RX140 の端子と SD カードの端子の結線を示します。
図 5-4 に評価ボードの追加工を示します。
これらの表と図を参照し、デモプロジェクトの動作環境を構成してください。

表 5-1 デモプロジェクトの動作条件

項目	説明
MCU	R5F51403ADFM（RX140 グループ）
動作周波数	・ HOCO: 48MHz ・ システムクロック（ICLK）: 48 MHz（HOCO 出力を 1 分周する） ・ 周辺モジュールクロック（PCLKB）: 24 MHz（HOCO 出力を 2 分周する）
動作電圧	3.3V(USB 経由で供給)
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
統合開発環境	ルネサス エレクトロニクス e² studio 2024-07
C コンパイラ	ルネサス エレクトロニクス C/C++ Compiler for RX Family V3.06.00
	コンパイルオプション ・ -lang = c99
エンディアン	・ データ・エンディアン：リトルエンディアン ・ デバッグツール設定：リトルエンディアン
デモプロジェクト	Version 1.00
評価ボード	Target Board for RX140 (製品型名: RTK5RX1400CxxxxxBJ) ボード設定：図 **5-4** に示す追加工を行う
Emulator	E2 エミュレータ Lite（オンボード）
Pmod™ SD カードコネクタ	Digilent® PmodSD™ Pmod™ 及び PmodSD ™ は米国 Digilent Inc.の商標です。
SD カード	SDHC メモリカード

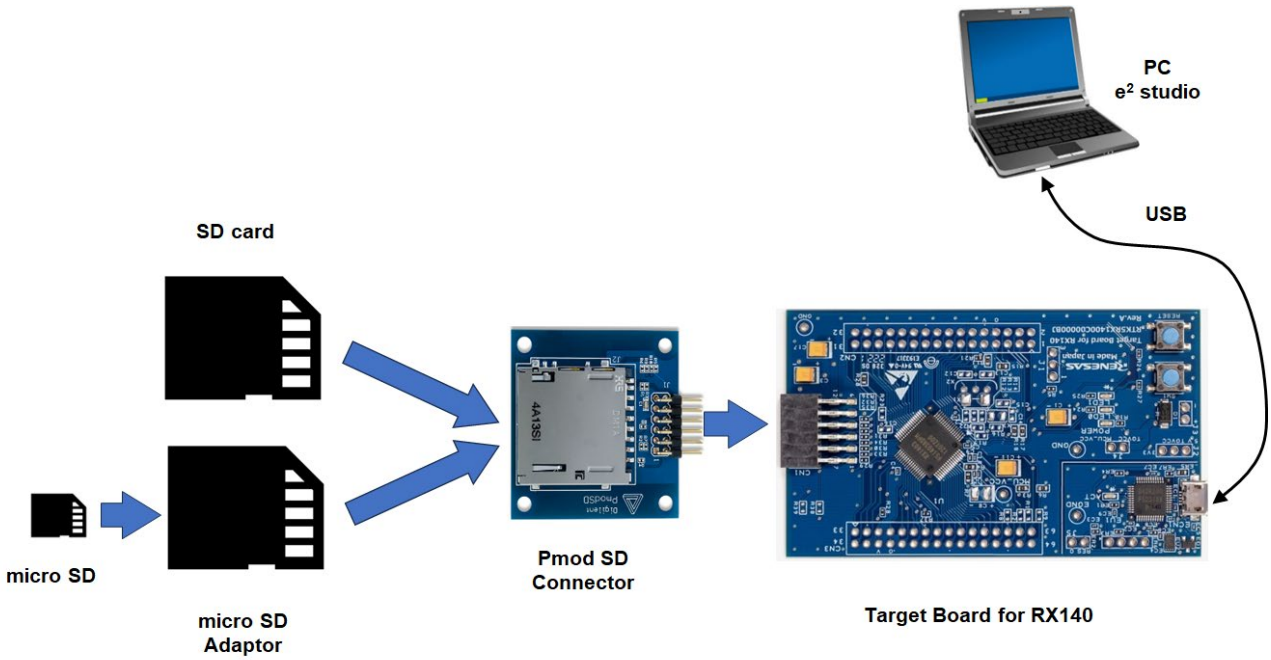


図 5-2 デモプロジェクトの動作環境

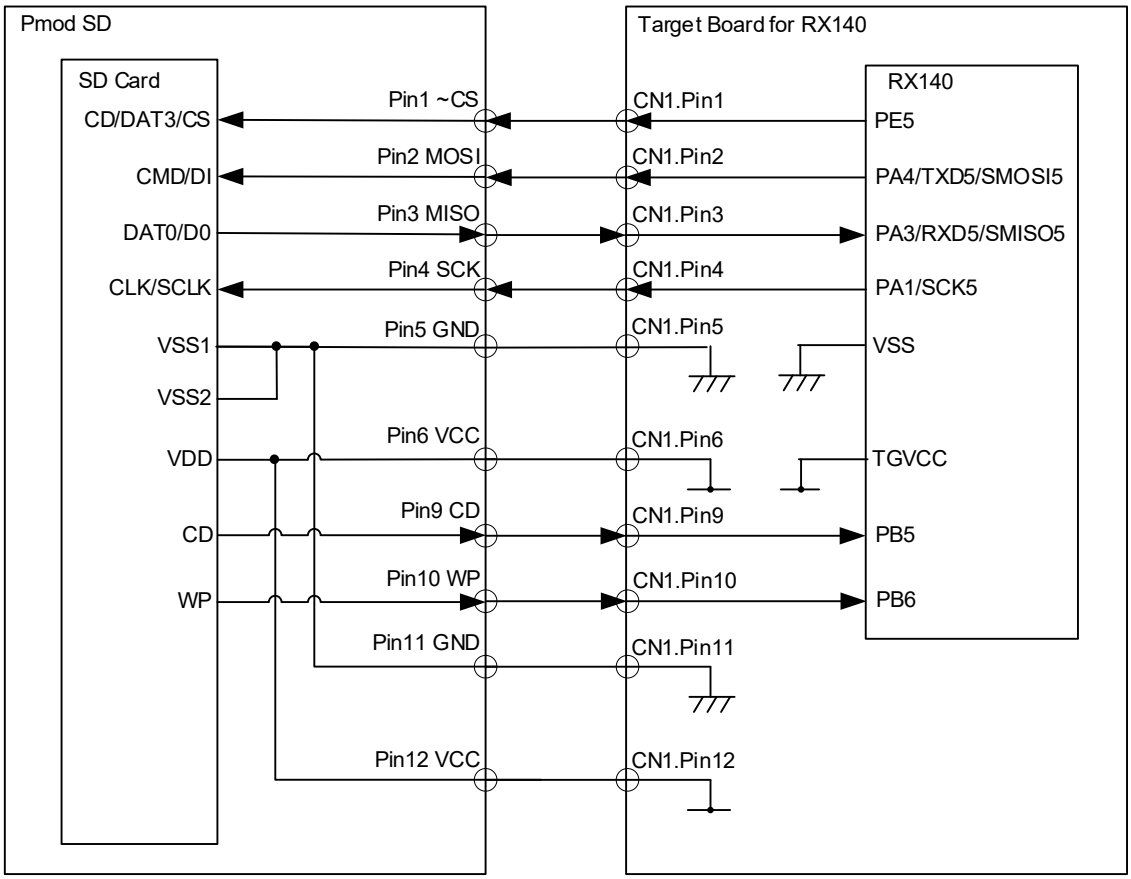


図 5-3　Target Board for RX140 と SD カードの端子結線図

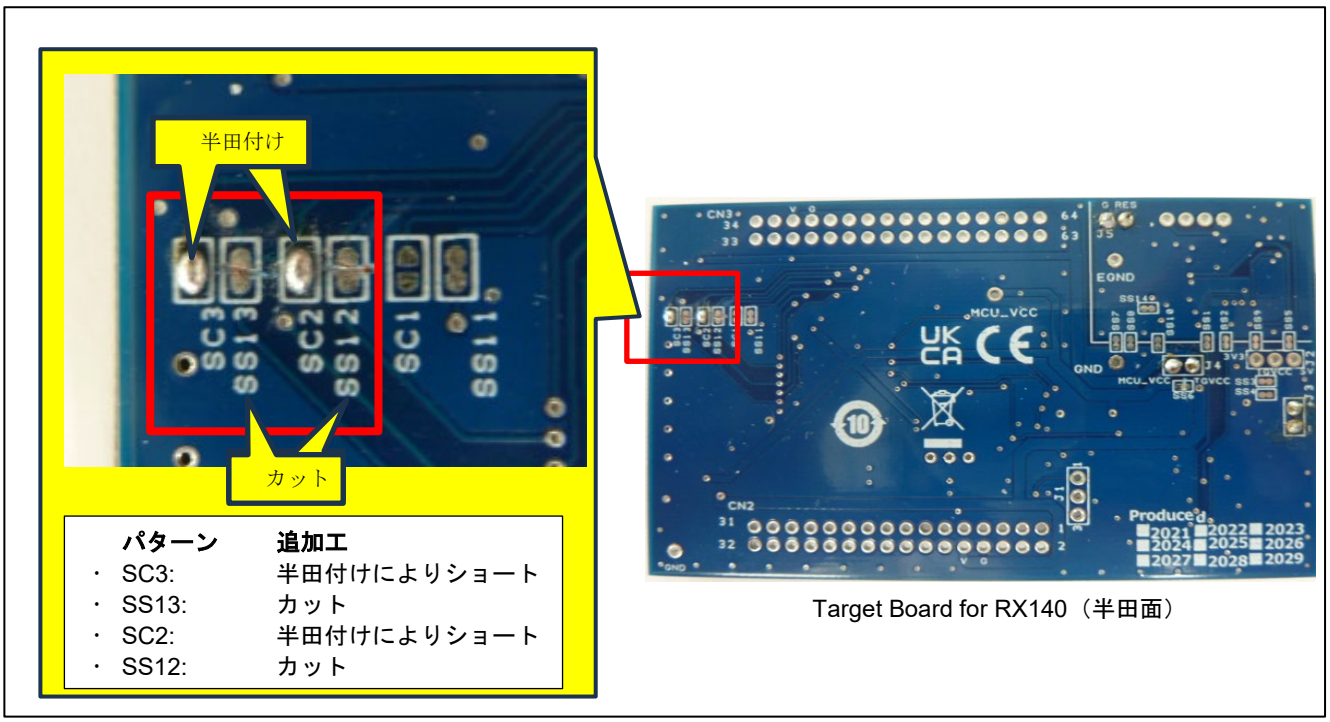


図 5-4 Target Board for RX140 の追加工

5.3　コンパイル時の設定

デモプロジェクトのコンフィグレーションオプションの設定は、r_sdc_spi_rx_demo_config.h で行います。

Configuration options in r_sdc_spi_rx_demo_config.h	
#define SDC_SPI_DEMO_CFG_POWER_CONTROL (0) ※デフォルト値は“0”	SD カードの電源制御に汎用入出力ポートを使用する場合の定義です。 “1”の場合: SD カード電源制御が行われます。 “0”の場合: SD カード電源制御が行われなくなります。
#define SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE (1) ※デフォルト値は“1（High を供給）”	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 SDC_SPI_DEMO_CFG_POWER_ON_WAIT (100) ※デフォルト値は“100（100ms ウェイト）”	SD カード用電源回路の電源供給を開始してから動作電圧に達するまでの時間を設定してください。 1 を設定すると、1ms のウェイトを行います。SD カードの電源回路に合わせて設定してください。
#define SDC_SPI_DEMO_CFG_POWER_OFF_WAIT (100) ※デフォルト値は“100（100ms ウェイト）”	SD カード用電源回路の電源供給を停止してから SD カードの抜去可能電圧に達するまでの時間を設定してください。 1 を設定すると、1ms のウェイトを行います。SD カードの電源回路に合わせて設定してください。
#define SDC_SPI_DEMO_CFG_POWER_CARDx_PORT (‘2’) ※デフォルト値は“ ’2’ ” ※CARDx の“x”は SD カード番号（x=0）	SD カード番号 x 用の電源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 SDC_SPI_DEMO_CFG_POWER_CARDx_BIT (‘0’) ※デフォルト値は“ ‘0’ ” ※CARDx の“x”は SD カード番号（x=0）	SD カード番号 x 用の電源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　デモプロジェクトのフローチャート

デモプロジェクトは、RX140 に接続された SD カードに対し表 5-2 に示す処理を順次行います。
図 5-5 及び図 5-6 にデモプロジェクトのフローチャートを示します。

表 5-2　デモプロジェクトの処理

	処理	説明
1)	SD カードの挿入と電源投入	・SD カードの挿入を検出後、SD カードに電源供給を開始します。(注 1) ・電源供給開始後、動作電圧に達するまで待機してから以降の処理を行います。 ・待機時間は SDC_SPI_CFG_POWER_ON_WAIT で指定されます。
2)	本ドライバを使用した SD カードへの読み出し／書き込み	・4 ブロック(2048 バイト)分の書き込み及び読出しを行い、正常に書き込まれたことを確認します。
3)	SD カードの抜去と電源停止	・SD カードへの電源供給を停止します。(注 1) ・電源供給停止後、SD カードの抜去可能電圧に達するまで待機してから SD カードの抜去が行えます。 ・待機時間は SDC_SPI_CFG_POWER_OFF_WAIT で指定されます。

注 1：デモプロジェクトでは SDC_SPI_DEMO_CFG_POWER_CONTROL を(0)に設定し、電源制御端子の設定処理を無効にしています。デモプロジェクトの動作環境である Target Board for RX140 は、SD カード用電源回路を実装していないためです。このためボードに電源を供給されると、SD カードにも電源が供給されます。
SD カード用電源回路が実装されたハードウェアの場合、
SDC_SPI_DEMO_CFG_POWER_CONTROL を(1)に設定することで、SD カードの挿入検出後、電源供給を開始するための処理を有効にすることができます。

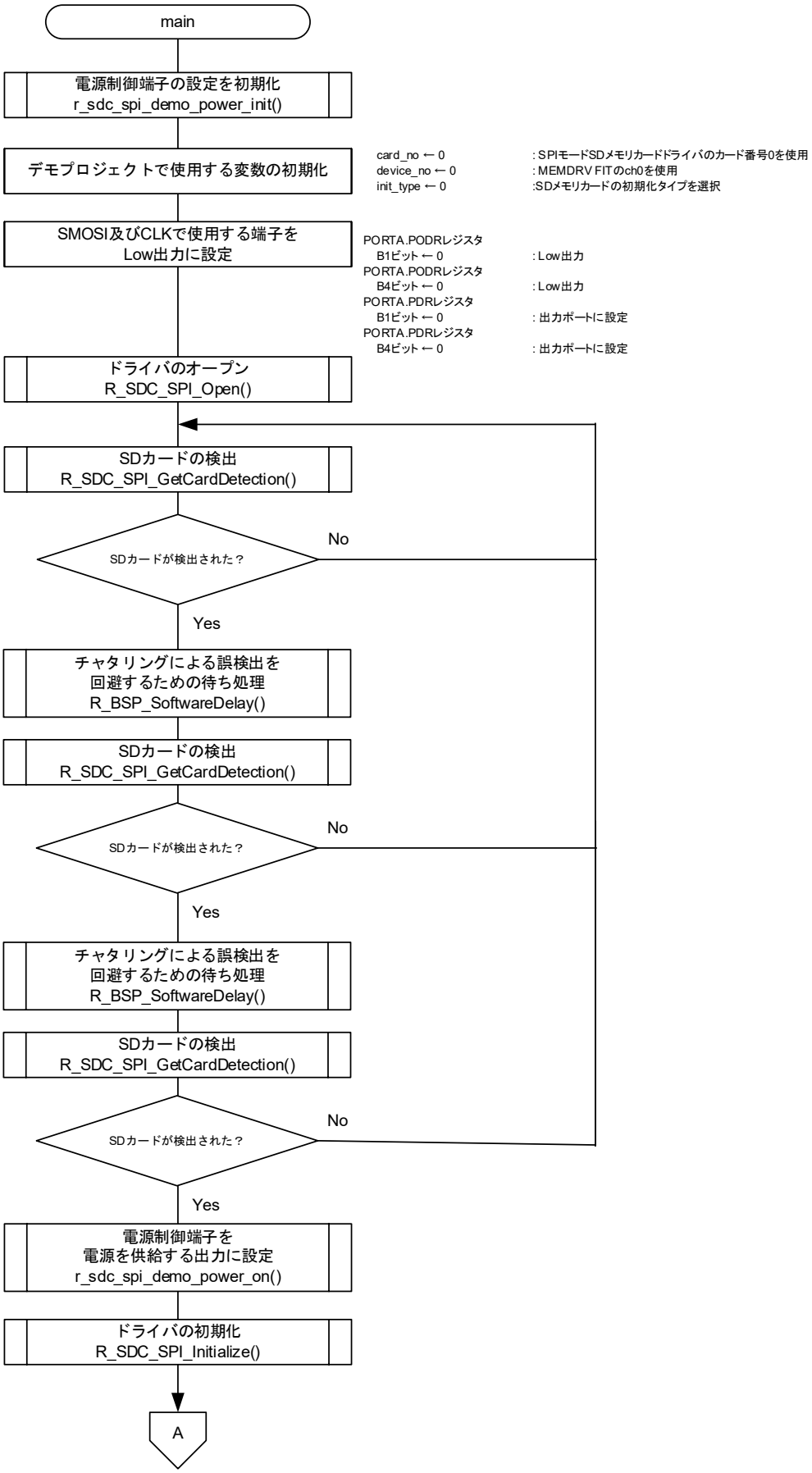


図 5-5　デモプロジェクトのフローチャート(1/2)

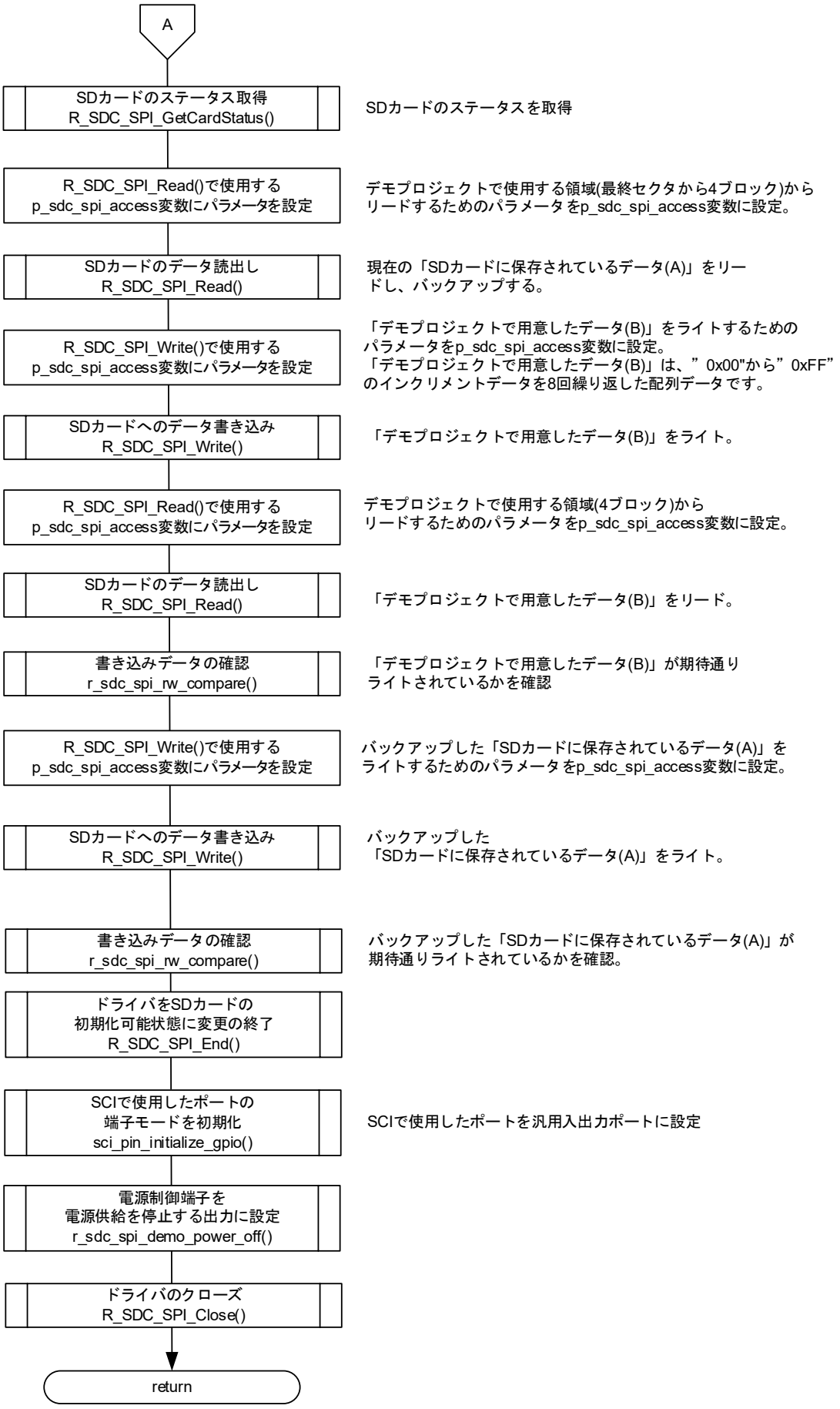
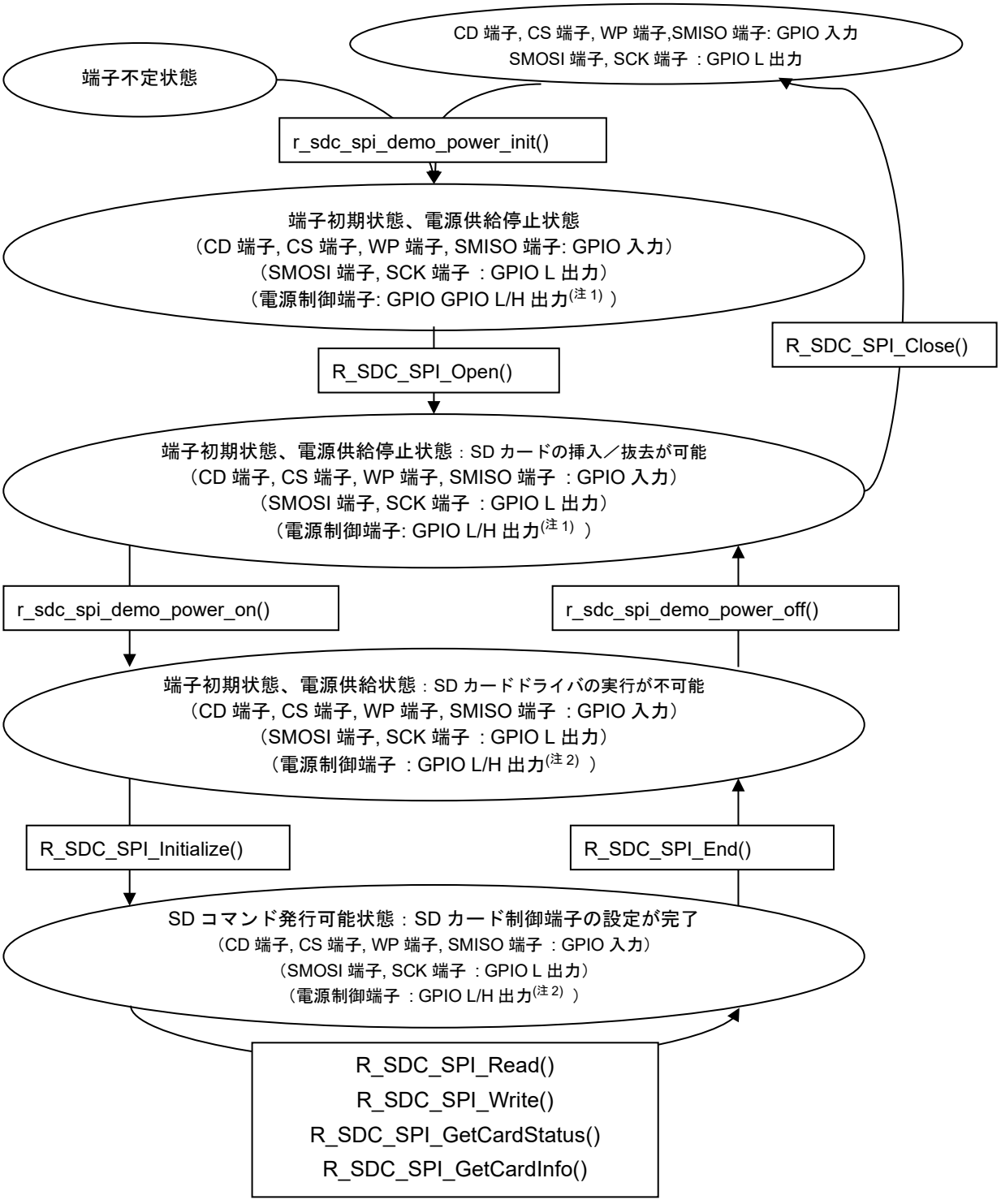


図 5-6　デモプロジェクトのフローチャート(2/2)

5.5　端子状態の遷移

図 5-7 に SD カードの端子が接続される RX140 の端子の状態の遷移を示します。



注 1：電源供給を停止するよう L か H に設定
注 2：電源供給を行うよう L か H に設定

図 5-7　端子の状態の遷移

5.6　ファイル構成

デモプロジェクトのソースファイルを表 5-3 に示します。

表 5-3　デモプロジェクトのファイル一覧

ファイル名	説明
main.c	デモプロジェクトの main 関数を含む C ソースファイル。 main.c に記述された関数が、SPI モード SD メモリカードドライバなど FIT モジュールの API を使用して「5.4 デモプロジェクトのフローチャート」に示す動作を行います。
r_sdc_spi_rx_demo.c	デモプロジェクトで使用する関数を含む C ソースファイル。
r_sdc_spi_rx_demo_config.h	デモプロジェクトのコンフィグレーションオプションのヘッダファイル。

5.7　関数

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

表 5-4　関数一覧

関数名	説明
main()	デモプロジェクトの main 関数。 ・以下 4 つの関数を使用して「5.4 デモプロジェクトのフローチャート」に示す動作を行います。 ・SPI モード SD メモリカードドライバの API 関数をコールしてリードとライトの処理を行います。
r_sdc_spi_demo_power_init()	電源制御端子(注 1)の設定を初期化します。
r_sdc_spi_demo_power_on()	SD カードへの電源供給を開始するよう、電源制御端子(注 1)を設定します。
r_sdc_spi_demo_power_off()	SD カードへの電源供給を停止するよう、電源制御端子(注 1)を設定します。
r_sdc_spi_rw_compare()	リードデータとライトデータの比較を行います。

注 1：ここでの電源とは SD カード用電源回路、電源制御端子とは SD カード用電源回路の出力を制御するために割り当てられた RX マイコンの端子とします。 なおデモプロジェクトの動作環境である Target Board for RX140 は、SD カード用電源回路を実装していません。SD カード用電源回路を実装したハードウェアを想定しデモプロジェクトを設計しています。

(1)　r_sdc_spi_demo_power_init()

電源制御端子の設定を初期化する関数です。

Format

```
void r_sdc_spi_demo_power_init(
uint32_t card_no
)
```

Parameters

card_no

SD カード番号	制御対象の SD カード番号（0 起算）

Return Values

なし

Description

SD カード用電源回路を制御する電源制御端子の設定を初期化する関数です。

電源制御端子は、config.h で指定される RX マイコンの端子です。

以下のように電源制御端子を設定します。

- ポートモードレジスタ（PMR）を汎用入出力ポートに設定。
- 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。
- ポート出力データレジスタ（PODR）で端子出力を L か H のいずれかに設定。
  電源回路に応じ、電源供給を停止するよう L か H を指定。
  （SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE での設定を参照）
- PDR を出力に設定します。

Special Notes

- 本関数は、SD カード制御システムのハードウェアの SD カード用電源を制御するために設計したものです。
  SD カードへの電源供給は「4.1 SD カードの挿入と電源投入の手順」に従う必要があります。
  手順に示された「端子制御 2」、すなわち電源制御端子の初期化をこの関数で行います。
  なお、デモプロジェクトの動作環境である Target Board for RX140 は、SD カード用電源回路を実装していません。このため r_sdc_spi_rx_demo_config.h で SDC_SPI_DEMO_CFG_POWER_CONTROL を(0)に設定し、本関数内の電源制御端子の初期化処理を無効にしています。
  電源回路が実装されたハードウェアの場合、SDC_SPI_DEMO_CFG_POWER_CONTROL を(1)に設定することで、電源制御端子の初期化処理を有効にすることができます。

(2)　r_sdc_spi_demo_power_on()

SD カードへの電源供給を開始するよう、電源制御端子を設定する関数です。

Format

```
bool r_sdc_spi_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

true	正常終了
false	エラー

Description

SD カードへの電源供給を開始するため、電源制御端子の出力レベルを設定します。

- ポート出力データレジスタ（PODR）で端子出力を L か H のいずれかに設定。
  電源回路に応じ、電源供給を開始するよう L か H を指定。
  （SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE での設定を参照）
- r_sdc_spi_rx_demo_config.h の SDC_SPI_CFG_POWER_ON_WAIT で設定された時間経過後、本関数の実行結果を戻り値として返します。

Special Notes

- 電源供給開始後、動作電圧に達するまでの時間待ちのため、R_BSP_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 SDC_SPI_CFG_POWER_ON_WAIT で設定してください。
- 本関数実行前に r_sdc_spi_demo_power_init() 関数による初期化処理が必要です。
- 本関数は、SD カード制御システムのハードウェアの SD カード用電源を制御するために設計したものです。
  電源供給は「4.1 SD カードの挿入と電源投入の手順」に従う必要があります。
  手順に示された「端子制御 3」、すなわち電源供給の開始ための端子設定をこの関数で行います。
  なお、デモプロジェクトの動作環境である Target Board for RX140 は、SD カード用電源回路を実装していなません。このため r_sdc_spi_rx_demo_config.h で
  SDC_SPI_DEMO_CFG_POWER_CONTROL を(0)に設定し、本関数内の電源制御端子の設定処理を無効にしています。
  電源回路が実装されたハードウェアの場合、SDC_SPI_DEMO_CFG_POWER_CONTROL を(1)に設定することで、本関数内の電源制御端子の設定処理を有効にすることができます。

(3)　r_sdc_spi_demo_power_off()

SD カードへの電源供給を停止するよう、電源制御端子を設定する関数です。

Format

```
bool r_sdc_spi_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

true	正常終了
false	エラー

Description

SD カードへの電源供給を停止するため、電源制御端子の出力レベルを設定します。

- PODR で L か H のいずれかに設定。
  電源回路に応じ、電源供給を停止するよう H か L に設定。
  （SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE での設定に従う）
- r_sdc_spi_rx_demo_config.h の SDC_SPI_CFG_POWER_OFF_WAIT で設定された時間経過後、本関数の実行結果を戻り値として返します。

Special Notes

- 電源供給停止後、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_BSP_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。」を参照し、SDC_SPI_CFG_POWER_OFF_WAIT で設定してください。
- 本関数実行前に r_sdc_spi_demo_power_init()関数による電源制御端子の初期化が必要です。
- 本関数は、SD カード制御システムのハードウェアの SD カード用電源を制御するために設計したものです。
  電源供給の停止は「4.2 SD カードの抜去と電源停止の手順」に従う必要があります。
  手順に示された「端子制御 4」、すなわち電源供給の開始ための端子設定をこの関数で行います。
  なお、デモプロジェクトの動作環境である Target Board for RX140 は、SD カード用電源回路を実装していなません。このため r_sdc_spi_rx_demo_config.h で SDC_SPI_DEMO_CFG_POWER_CONTROL を(0)に設定し、本関数内の電源制御端子の設定処理を無効にしています。
  電源回路が実装されたハードウェアの場合、SDC_SPI_DEMO_CFG_POWER_CONTROL を(1)に設定することで、本関数内の電源制御端子の設定処理を有効にすることができます。

(4)　r_sdc_spi_rw_compare()

書き込みデータバッファと読出しデータバッファに格納されたデータを比較し、比較結果を返す関数です。

Format

```
bool r_sdc_spi_rw_compare(
uint32_t * p_buf_w
uint32_t * p_buf_r
uint32_t cnt
)
```

Parameters

* p_buf_w

書き込みデータバッファポインタ

512 バイト単位のバッファを設定してください。

* p_buf_r

読出しデータバッファポインタ

512 バイト単位のバッファを設定してください。

cnt

比較するブロック数

Return Values

true	リードデータバッファのデータとライトデータバッファのデータは一致
false	リードデータバッファのデータとライトデータバッファのデータは不一致

Description

引数で指定した書き込みデータバッファと読出しデータバッファに格納されたデータを比較し、比較結果を返します。

Special Notes

なし

5.8　デモのダウンロード方法

本 FIT モジュールを RX Driver Package から入手した場合の注意を示します。

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。

ダウンロードした zip ファイル内の「FITDemos」にあるデモプロジェクトをインポートしてください。

5.9　プロジェクトをインポートする方法

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

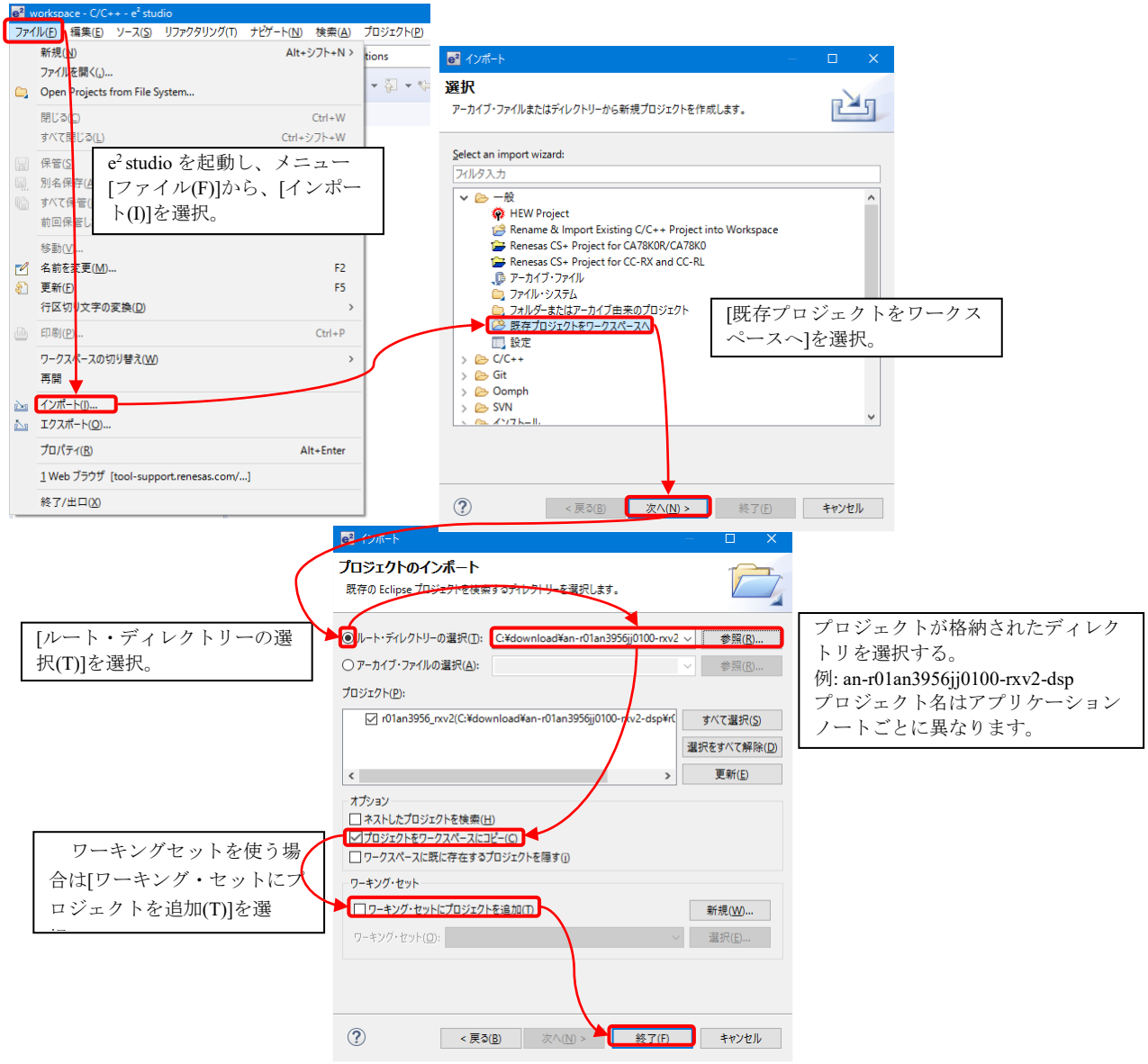


図 5-8　プロジェクトを e² studio にインポートする方法

5.10　デモの注意事項

デモプロジェクトは、「5.2 動作確認環境」のように Target Board for RX140 の Pmod SD に Pmod SD カードコネクタを接続した条件で動作します。このため Target Board for RX140 へ電源が供給されると Pmod SD カードコネクタの電源端子にも電源供給が開始されます。

なお SD カードへの電源供給は、SD カードが接続されている状態で行われる必要があります。

このため、SD カードを接続してから Target Board for RX140 へ電源供給を開始してください。

また、Target Board for RX140 への電源供給が停止してから SD カードを抜去してください。

6. 付録

6.1 動作確認環境

本モジュールの動作確認環境を以下に示します。

表 6-1　動作確認環境(Rev. 1.11)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V2025-01 IAR Embedded Workbench for Renesas RX 5.10.1
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 V3.07.00 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
	GCC for Renesas RX 8.03.00.202411 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 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのバージョン	Ver.1.11
使用ボード	—

表 6-2　動作確認環境(Rev. 1.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V2024-07 IAR Embedded Workbench for Renesas RX 5.10.1
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 V3.06.00 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 8.03.00.202405 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 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのバージョン	Ver.1.10
使用ボード	Target Board for RX140 (製品型名: RTK5RX1400CxxxxxBJ)

表 6-3　動作確認環境(Rev. 1.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V2023-10 IAR Embedded Workbench for Renesas RX 5.10.1
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 V3.05.00 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 8.03.00.202311 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 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのバージョン	Ver.1.00
使用ボード	Target Board for RX140 (製品型名: RTK5RX1400CxxxxxBJ)

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_sdc_spi_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
（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。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2023.12.28	-	初版発行。
1.10	2024.9.16		6.1 動作確認環境に Ver1.10 を追加
		プログラム	#include "r_gpio_rx_if.h"を削除
		プログラム	CSD レジスタ Ver2 のサイズ計算式を修正
1.11	2025.3.20	62	表 6 1　動作確認環境(Rev. 1.11)を追加
		プログラム	プログラムの免責事項を変更

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。