SH7753 グループ
SPI0 ドライバソフトウェア

R01AN1604JJ0100
Rev.1.00
2013.7.1

要旨

本仕様書は SH7753 グループの SPI0(Serial Peripheral Interface 0)ドライバについて説明します。

対象デバイス

SH7753

動作環境

本仕様書に示す SPI0 ドライバの動作環境を以下に示します。

- 評価ボード : SH7753 グループ EVB ボード
- SPI フラッシュ
  - : モデル - MX25L64
  - : サイズ - 8MB
  - : モード - CPOL = 0, CPHA = 0 / CPOL = 1, CPHA = 1
  - : クロック - 最大使用周波数 48Mhz
  - : 動作 - リード ID、リードステータス、ライトイネーブル/ディセーブル、ページライト、リード、ファストリード、デュアルアウトプットリード、イレース（セクタ（64KB）、バルク）
  - : モデル - S25FL512S
  - : サイズ - 64MB
  - : モード - CPOL = 0, CPHA = 0 / CPOL = 1, CPHA = 1
  - : クロック - 最大使用周波数 48Mhz
  - : 動作 - リード ID、リードステータス、ライトイネーブル/ディセーブル、ページライト、リード、ファストリード、デュアルアウトプットリード、イレース（セクタ（256KB）、バルク）
- ソフトウェア : High-Performance Embedded Workshop- Ver 4.09.01.007
  - : Toolchain - Ver 9.4.1.0
  - : OptLinker - Ver 10.01.00
  - : SH アセンブラ - Ver 7.01.02
  - : SH C/C++コンパイラ- Ver 9.04.01
  - : SH C/C++ライブラリジェネレータ - Ver 3.00.03
- エミュレータ : E10A エミュレータ Ver 3.03.00

目次

1. 機能概要

本ドライバは SH7753 に搭載されている SPI0 を使用し、SPI インタフェースを持つデバイス（主に SPI フラッシュメモリ）と通信する為の関数群で構成されています。

1.1 主な機能

SPI0 ドライバの主な機能を以下に示します。

・オペレーション設定

・ライトプロテクト領域設定／情報取得

・SPI フラッシュデータ書き込み（書き込み有効／無効、ページ書き込み、セクタ／バルク消去）

・SPI フラッシュデータ読み出し（ID 情報読み出し、データ読み出し（Normal／Fast／Dual Output））

・シリアルデータ通信（データ送受信）

1.2 関連ドキュメント

SH7753 グループ　ユーザーズマニュアル：ハードウェア編

2. ドライバ仕様

2.1 関数一覧

表 1に、SPI0 ドライバの関数一覧を示します。

表 1. SPI0 ドライバ関数一覧

分類	関数名	機能概要	スタック サイズ
設定関数	R_SPI0_SetOpr	オペレーション設定処理	8 バイト
	R_SPI0_SetWpMask	ライトプロテクトマスク設定処理	20 バイト
	R_SPI0_SetWpOpCode	ライトプロテクトオペコード設定処理	12 バイト
情報取得関数	R_SPI0_GetWpInfo	ライトプロテクト情報取得処理	0 バイト
	R_SPI0_GetStatus	ステータス情報取得処理	8 バイト
SPI フラッシュ アクセス関数	R_SPI0_SfReadId	リード ID 処理	16 バイト
	R_SPI0_SfReadStatusReg	リードステータスレジスタ処理	12 バイト
	R_SPI0_SfWriteEnableCmd	ライトイネーブルコマンド送信処理	8 バイト
	R_SPI0_SfWriteDisableCmd	ライトディスエイブルコマンド送信処理	8 バイト
	R_SPI0_SfWriteStatusRegCmd	ライトステータスレジスタコマンド送信処理	12 バイト
	R_SPI0_SfPageWriteCmd	ページライトコマンド送信処理	20 バイト
	R_SPI0_SfSectorEraseCmd	セクタイレースコマンド送信処理	20 バイト
	R_SPI0_SfBulkEraseCmd	バルクイレースコマンド送信処理	12 バイト
	R_SPI0_SfReadDataCmd	リードデータコマンド送信処理	16 バイト
データ通信関数	R_SPI0_ComDataTx	データ送信処理	32 バイト
	R_SPI0_ComDataRx	データ受信処理	8 バイト
	R_SPI0_ComDataRxDmac	データ受信（DMAC 連動）処理	16 バイト
	R_SPI0_ComStop	通信停止処理	8 バイト
割り込みハンドラ関数	R_SPI0_Interrupt	SPI0I 割り込みハンドラ	0 バイト

2.2 コールバック関数仕様一覧

表 2に SPI0 ドライバのコールバック関数仕様一覧を示します。SPI0 ドライバのコールバック関数は各ドライバ関数の処理完了時に通知としてコールします。

表 2. SPI0 ドライバコールバック関数仕様一覧

関数名	コールバック名	仕様内容
R_SPI0_ComDataTx	void (*pv_data_tx_end_callback)(void)	データ送信完了時のコールバック。 （NULL の場合、コールバックなし）
R_SPI0_ComDataRx	void (*pv_data_rx_end_callback)(void)	データ受信完了時のコールバック。 （NULL の場合、コールバックなし）
R_SPI0_ComDataRxDmac	void (*pv_data_rx_dmac_end_callback)(void)	DMAC 連動データ受信完了時の コールバック。 （NULL の場合、コールバックなし）

2.3 エラーコード一覧

表 3に SPI0 ドライバのエラーコード一覧を示します。

表 3. SPI0 ドライバエラーコード一覧

値	エラーコード（マクロ定義）	エラー内容
0	RET_NORMAL	正常終了
-1	RET_ERR_PARAM1	第 1 引数不正
-2	RET_ERR_PARAM2	第 2 引数不正
-3	RET_ERR_PARAM3	第 3 引数不正
-4	RET_ERR_PARAM4	第 4 引数不正
-20	RET_ERR_SPI0_WPABORT	ライトプロテクト領域へのアクセスアボート

2.4 SPI フラッシュアクセス用 DEFINE 定義一覧

表 4に各 SPI0 ドライバ関数で使用する SPI フラッシュアクセス用 DEFINE 定義一覧を示します。

※注意：接続する SPI フラッシュデバイスに対応した値に変更してください。

表 4. SPI0 ドライバ SPI フラッシュアクセス用 DEFINE 定義一覧

値[*1]	DEFINE 定義名[*2]	使用ドライバ関数名	内容
0x9F	SPI0_SF_CMD_RDID	R_SPI0_SfReadId	リード ID コマンド
0x05	SPI0_SF_CMD_RDSR	R_SPI0_SfReadStatusReg	リードステータスレジスタコマンド
0x06	SPI0_SF_CMD_WREN	R_SPI0_SfWriteEnableCmd	ライトイネーブルコマンド
0x04	SPI0_SF_CMD_WRDI	R_SPI0_SfWriteDisableCmd	ライトディスエイブルコマンド
0x01	SPI0_SF_CMD_WRSR	R_SPI0_SfWriteStatusRegCmd	ライトステータスレジスタコマンド
0x02	SPI0_SF_CMD_PGWR	R_SPI0_SfPageWriteCmd	ページライトコマンド
0xD8	SPI0_SF_CMD_SE	R_SPI0_SfSectorEraseCmd	セクタイレースコマンド
0xC7	SPI0_SF_CMD_BE	R_SPI0_SfBulkEraseCmd	バルクイレースコマンド
0x03	SPI0_SF_CMD_READ	R_SPI0_SfReadDataCmd	リードデータコマンド
0x0B	SPI0_SF_CMD_FREAD	R_SPI0_SfReadDataCmd	ファストリードデータコマンド
0x3B	SPI0_SF_CMD_DREAD	R_SPI0_SfReadDataCmd	デュアルアウトプットリードコマンド
0x12	SPI0_SF_CMD_4B_PGWR	R_SPI0_SfPageWriteCmd	4 バイトアドレスモード ページライトコマンド
0xDC	SPI0_SF_CMD_4B_SE	R_SPI0_SfSectorEraseCmd	4 バイトアドレスモード セクタイレースコマンド
0x13	SPI0_SF_CMD_4B_READ	R_SPI0_SfReadDataCmd	4 バイトアドレスモード リードデータコマンド
0x0C	SPI0_SF_CMD_4B_FREAD	R_SPI0_SfReadDataCmd	4 バイトアドレスモード ファストリードデータコマンド
0x3C	SPI0_SF_CMD_4B_DREAD	R_SPI0_SfReadDataCmd	4 バイトアドレスモード デュアルアウトプットリードコマンド
0x0001[*3]	SPI0_SF_SSNEG_WAITTIME	R_SPI0_SfReadId R_SPI0_SfReadStatusReg R_SPI0_SfWriteEnableCmd R_SPI0_SfWriteDisableCmd R_SPI0_SfWriteStatusRegCmd R_SPI0_SfSectorEraseCmd R_SPI0_SfBulkEraseCmd R_SPI0_ComStop	スレーブセレクト信号ネゲート後 必要 Wait 時間 [1 カウントの時間単位] [*4] 約 150ns（キャッシュ ON 時）／ 約 1.3μs（キャッシュ OFF 時）

[*1] “MX25L64”、“S5FL512S”の SPI フラッシュデバイス使用時。但し、4 バイトアドレスモードでは、“S5FL512S”のみ使用。

[*2] SPI0 ドライバのインタフェースヘッダ（r_spi0_if.h）ファイルで DEFINE 定義しています。

[*3] “MX25L64”、“S5FL512S”のスレーブセレクト信号ネゲート後の必要 Wait 時間。

[*4] SH-4A の 576Mhz オペレーティングモード使用時。

3. 関数

本章では、SPI0 ドライバの各関数仕様詳細を示します。各関数詳細の読み方は以下のとおりです。

関数名 機能概要	分類
書式	関数の呼び出し形式を示します。#include “ヘッダファイル”で示すヘッダファイルは、この関数の実行に必要な標準ヘッダファイルで、必ずインクルードします。
引数	I,O は、引数がそれぞれ入力データ、出力データであることを意味します。
戻り値	関数の戻り値を示します。
解説	関数の仕様について説明します。
注意	注意事項があればここに示します。

R_SPI0_SetOpr

SPI0 設定関数

SPI0 オペレーション設定処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SetOpr( spi0_opr_t *pst_opr );
```

引数

spi0_opr_t *pst_opr		I	SPI0 オペレーション処理管理構造体へのポインタ

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、SPI0 オペレーション処理管理構造体の情報に基づき、SPI0 のオペレーション（スレーブセレクト番号、クロック周波数、クロック Polarity/Phase モード、データ Endian）設定を行います。

【SPI0 オペレーション処理管理構造体】spi0_opr_t

uchar_t	uc_slave_select	I	スレーブセレクト番号（0～3）
uchar_t	uc_clock_speed	I	クロック周波数（3 : 48、4 : 38.4、5 : 32、6 : 27.4、7 : 24 Mhz）
uchar_t	uc_clock_polphase	I	クロック Polarity/Phase モード 0：モード 0（CPOL=0／CPHA=0） 1：モード 1（CPOL=0／CPHA=1） 2：モード 2（CPOL=1／CPHA=0） 3：モード 3（CPOL=1／CPHA=1）
uchar_t	uc_endian	I	データ Endian（0：MSB、1：LSB）

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

R_SPI0_SetWpMask

SPI0 設定関数

SPI0 ライトプロテクトマスク設定処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SetWpMask( spi0_wp_mask_t * pst_wpm );
```

引数			
	spi0_wp_mask_t * pst_wpm	I	SPI0 ライトプロテクトマスク管理構造体へのポインタ

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、SPI0 ライトプロテクトマスク管理構造体の情報に基づき、ライトプロテクトマスク機能の設定を行います。

【SPI0 ライトプロテクトマスク管理構造体】spi0_wp_mask_t

uchar_t	uc_wp_mask_ctrl	I	ライトプロテクトマスク制御（1：有効、0：無効）
ulong_t	ul_wp_mask_base_addr	I	ライトプロテクトマスクベースアドレス
ulong_t	ul_wp_mask_size	I	ライトプロテクトマスクサイズ

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

SPI0 がライトプロテクト有効状態時、本関数をコールしないこと。

ライトプロテクトのマスクベースアドレスとマスクサイズの設定仕様については、「SH7753 グループ ユーザーズマニュアル：ハードウェア編 15 章 SPI0」を参照すること。

R_SPI0_SetWpOpCode

SPI0 設定関数

SPI0 ライトプロテクトオペコード設定処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SetWpOpCode( spi0_wp_opcode_t *pst_wp_opcode );
```

引数			
spi0_wp_opcode_t *pst_wp_opcode		I	SPI0 ライトプロテクトオペコード管理構造体へのポインタ

戻り値		
RET_NORMAL		正常終了

解説

本関数は、SPI0 ライトプロテクトオペコード管理構造体の情報に基づき、ライトプロテクト機能のオペコード設定を行います。

【SPI0 ライトプロテクトオペコード管理構造体】spi0_wp_opcode_t

uchar_t	uc_opcode_page_write	I	ページライトオペコード
uchar_t	uc_opcode_32kb_blk_erase	I	32KB ブロックイレースオペコード
uchar_t	uc_opcode_64kb_blk_erase	I	64KB ブロックイレースオペコード
uchar_t	uc_opcode_sector_erase	I	セクタイレースオペコード
uchar_t	uc_opcode_chip1_erase	I	チップ 1 イレースペコード
uchar_t	uc_opcode_chip2_erase	I	チップ 1 イレースオペコード
uchar_t	uc_opcode_read	I	リードオペコード
uchar_t	uc_opcode_fast_read	I	ファーストリードオペコード
uchar_t	uc_opcode_dual_output_read	I	デュアルアウトプットリードオペコード
uchar_t	uc_opt_opcode_list0[8]	I	任意オペコードリスト 0 （アドレス情報を必要とするオペコード）
uchar_t	uc_opt_opcode_list1[16]	I	任意オペコードリスト 1 （アドレス情報には必要のないオペコード）

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。
SPI0 がライトプロテクト有効状態時、本関数をコールしないこと。

R_SPI0_GetWpInfo

SPI0 情報取得関数

SPI0 ライトプロテクト情報取得処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_GetWpInfo( spi0_wp_info_t *pst_wp_info );
```

引数			
spi0_wp_info_t *pst_wp_info	**O**	SPI0 ライトプロテクト情報管理構造体へのポインタ

戻り値	
RET_NORMAL	正常終了

解説

本関数は、SPI0 ライトプロテクト情報管理構造体に対して、SPI0 のライトプロテクト機能に関する情報の取得を行います。

【SPI0 ライトプロテクト情報管理構造体】spi0_wp_info_t

uchar_t	uc_wp_state	O	ライトプロテクト状態（0：有効、1：無効）
uchar_t	uc_wp_size	O	ライトプロテクトサイズ 0：64 Kbytes 1：128 Kbytes 2：256 Kbytes 3：512 Kbytes
uchar_t	uc_sf_memory_size;	O	SPI フラッシュメモリサイズ 0：1 Mbyte 1：2 Mbytes 2：4 Mbytes 3：8 Mbytes 4：16 Mbyte 5：32 Mbytes 6：64 Mbytes 7：128 Mbytes

注意 なし

R_SPI0_GetStatus

SPI0 情報取得関数

SPI0 ステータス情報取得処理

書式

```
#include "r_common.h"
#include "iodefine.h"
#include "r_spi0_if.h"
char_t R_SPI0_GetStatus( void );
```

引数　なし

戻り値	0～3	ステータス情報

解説　本関数は、SPI0 のステータス情報の取得処理を行います。
本関数をコールすることにより SPI0 のライトプロテクト状態及び、SPI0 がビジー状態（SPI フラッシュアクセス中、データ通信中）、またはアイドル状態かを確認できます。

【ステータス情報詳細】

SPI0 ビジー状態	:	bit[0] = 0
SPI0 アイドル状態	:	bit[0] = 1
SPI0 ライトプロテクト有効状態	:	bit[1] = 0
SPI0 ライトプロテクト無効状態	:	bit[1] = 1

注意　なし

R_SPI0_SfReadId

SPI フラッシュデバイスアクセス関数

SPI フラッシュリード ID 処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
long_t R_SPI0_SfReadId( void );
```

引数

なし

戻り値

>= 0	メーカーID 情報（bit [23 :16]）、デバイス ID 情報（bit [15 :0]）

解説

本関数は、SPI フラッシュデバイスのリード ID（メーカーID、デバイス ID）処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、リード ID コマンド（SPI0_SF_CMD_RDID）を送信します。そして、SPI フラッシュの ID 情報を受信した後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

R_SPI0_SfReadStatusReg

SPI フラッシュデバイスアクセス関数

SPI フラッシュリードステータスレジスタ処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfReadStatusReg( void ) ;
```

引数

なし

戻り値

>= 0	SPI フラッシュステータスレジスタ情報

解説

本関数は、SPI フラッシュデバイスのリードステータスレジスタ処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、リードステータスレジスタコマンド（SPI0_SF_CMD_RDSR）を送信します。そして、SPI フラッシュのステータスレジスタ情報を受信した後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

R_SPI0_SfWriteEnableCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュライトイネーブルコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfWriteEnableCmd( void ) ;
```

引数

なし

戻り値

RET_NORMAL	正常終了

解説

本関数は、SPI フラッシュデバイスへのライトイネーブルコマンドの送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、ライトイネーブルコマンド（SPI0_SF_CMD_WREN）を送信します。そして、送信完了後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数の処理完了後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのライトイネーブル処理完了を確認すること。

R_SPI0_SfWriteDisableCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュライトディスエイブルコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfWriteDisableCmd( void ) ;
```

引数

なし

戻り値

RET_NORMAL	正常終了

解説

本関数は、SPI フラッシュデバイスへのライトディスエイブルコマンドの送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、ライトディスエイブルコマンド（SPI0_SF_CMD_WRDI）を送信します。そして、送信完了後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数の処理完了後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのライトディスエイブル処理完了を確認すること。

R_SPI0_SfWriteStatusRegCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュライトステータスレジスタコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfWriteStatusRegCmd( uchar_t uc_write_data ) ;
```

引数

uchar_t uc_write_data	I	ライトデータ

戻り値

RET_NORMAL	正常終了

解説

本関数は、SPI フラッシュデバイスへのライトステータスレジスタコマンドの送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、ライトステータスレジスタコマンド（SPI0_SF_CMD_WRSR）と引数”uc_write_data”に指定したライトデータを送信します。そして、送信完了後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数の処理完了後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのライトステータスレジスタ処理完了を確認すること。

R_SPI0_SfPageWriteCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュページライトコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfPageWriteCmd( ulong_t ul_sf_page_addr );
```

引数

ulong_t ul_sf_page_addr	I	SPI フラッシュページアドレス（ページ境界）

戻り値

RET_NORMAL	正常終了
RET_ERR_SPI0_WPABORT	ライトプロテクト領域へのアクセスアボート

解説

本関数は、SPI フラッシュデバイスへのページライトコマンド送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、ページライトコマン（SPI0_SF_CMD_PGWR／SPI0_SF_CMD_4B_PGWR）と引数” ul_sf_page_addr”に指定したページアドレス（3 バイトアドレスモード時は bit[23:0]、4 バイトアドレスモード時は bit[31:0]を使用）を送信します。本関数コール後、R_SPI0_ComDataTx関数をコールすることにより、指定ページへのライトデータ送信ができます。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数コール後、全ライトデータの送信完了後に、R_SPI0_ComStop関数をコールし、SPI フラッシュアクセスを停止すること（スレーブセレクト信号のネゲート）。

ページライト送信完了（R_SPI0_SfPageWriteCmd →R_SPI0_ComDataTx → R_SPI0_ComStop）後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのページライト処理完了を確認すること。

4 バイトアドレスモードを使用する場合、BOOTSZ[2 :0]を 32MB 以上（32,64,128MB）に設定してください。

R_SPI0_SfSectorEraseCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュセクタイレースコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfSectorEraseCmd( ulong_t ul_sf_sector_addr ) ;
```

引数

ulong_t ul_sf_sector_addr	I	SPI フラッシュセクタアドレス （セクタアドレス境界）

戻り値

RET_NORMAL	正常終了
RET_ERR_SPI0_WPABORT	ライトプロテクト領域へのアクセスアボート

解説

本関数は、SPI フラッシュデバイスへのセクタイレースコマンド送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、セクタイレースコマンド（SPI0_SF_CMD_SE／SPI0_SF_CMD_4B_SE）と引数” ul_sf_sector_addr”に指定したセクタアドレス（3 バイトアドレスモード時は bit[23:0]、4 バイトアドレスモード時は bit[31:0]を使用）を送信します。そして、送信完了後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数コール後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのセクタイレース処理完了を確認すること。

4 バイトアドレスモードを使用する場合、BOOTSZ[2 :0]を 32MB 以上（32,64,128MB）に設定してください。

R_SPI0_SfBulkEraseCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュバルクイレースコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfBulkEraseCmd( void );
```

引数

なし

戻り値

RET_NORMAL	正常終了
RET_ERR_SPI0_WPABORT	ライトプロテクト領域へのアクセスアボート

解説

本関数は、SPI フラッシュデバイスへのバルクイレースコマンド送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、バルクイレースコマンド（SPI0_SF_CMD_BE）を送信します。そして、送信完了後、SPI フラッシュアクセスを停止します（スレーブセレクト信号のネゲート）。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数コール後、R_SPI0_SfReadStatusReg関数をコールし、SPI フラッシュデバイスのバルクイレース処理完了を確認すること。

R_SPI0_SfReadDataCmd

SPI フラッシュデバイスアクセス関数

SPI フラッシュリードデータコマンド送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_SfReadDataCmd( uchar_t uc_mode, ulong_t ul_sf_read_addr );
```

引数

uchar_t uc_mode	I	SPI フラッシュリードデータモード （0：Normal、1：Fast Read、2 ：Dual Output）
ulong_t ul_sf_read_addr	I	SPI フラッシュリード先頭アドレス

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、SPI フラッシュデバイスへのリードデータコマンドの送信処理を行います。

本関数は、SPI0 と接続 SPI フラッシュへのアクセスを開始（スレーブセレクト信号のアサート）し、リードデータコマンド（SPI0_SF_CMD_READ／SPI0_SF_CMD_FREAD／SPI0_SF_CMD_DREAD／SPI0_SF_CMD_4B_READ ／ SPI0_SF_CMD_4B_FREAD ／ SPI0_SF_CMD_4B_DREAD ） と 引 数 ”ul_sf_read_addr”に指定したリードアドレス（3 バイトアドレスモード時は bit[23:0]、4 バイトアドレスモード時は bit[31:0]を使用）を送信します。本関数コール後、R_SPI0_ComDataRx関数、またはR_SPI0_ComDataRxDmac関数をコールすることにより、リードデータの受信ができます。

注意

SPI0 が SPI フラッシュアクセス中、またはデータ通信中に、本関数をコールしないこと。

本関数コール後、全リードデータの受信完了後に、R_SPI0_ComStop関数をコールし、SPI フラッシュアクセスを停止（スレーブセレクト信号のネゲート）すること。

接続 SPI フラッシュがファストリードまたはデュアルアウトプットリード時に、Dummy データの送信が必要の場合、R_SPI0_ComDataTx関数を使用してください。

4 バイトアドレスモードを使用する場合、BOOTSZ[2 :0]を 32MB 以上（32,64,128MB）に設定してください。

R_SPI0_ComDataTx

データ通信関数

データ送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_ComDataTx( ulong_t ul_src_addr, uchar_t uc_tx_data_size
void (*pv_data_tx_end_callback )( void ) );
```

引数

ulong_t ul_src_addr	I	送信データ格納領域先頭アドレス
uchar_t uc_tx_data_size	I	送信データサイズ（1～32 バイト）
void (*pv_data_tx_end_callback)(void)	I	データ送信完了時のコールバックアドレス（NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、引数 “uc_tx_data_size” に指定した送信データサイズまで、SPI0 と接続デバイスへのデータ送信処理を行います。データ送信前に、SPI0 がアイドル状態の場合、通信開始処理（スレーブセレクト信号のアサート）を行います。

本関数は SPI0 仕様により、DMAC を使用できません。

注意

R_SPI0_SfPageWriteCmd関数コールを未実行、かつ SPI0 ライトプロテクト有効状態で、本関数を使用しないこと。

R_SPI0_ComDataRx

データ通信関数

データ受信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_ComDataRx( ulong_t ul_dest_addr, uchar_t uc_rx_data_size,
void (*pv_data_rx_end_callback ) ( void ) );
```

引数

ulong_t ul_dest_addr	I	受信データを格納する領域への先頭アドレス
uchar_t uc_rx_data_size	I	受信データサイズ（1～32 バイト）
void (*pv_data_rx_end_callback)(void)	I	データ受信完了時のコールバックアドレス（NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、引数“uc_rx_data_size”に指定した受信データサイズまで、SPI0 と接続デバイスからのデータ受信処理を行います。

本関数は SPI0 の内蔵 DMAC は使用しない。なお、SPI0 内蔵 DMAC 連動のデータ受信を使用したい場合、R_SPI0_ComDataRxDmac関数を使用してください。

注意

本関数をコールする前に、R_SPI0_ComDataTx関数、R_SPI0_SfPageWriteCmd関数、または R_SPI0_SfReadDataCmd関数をコールし、SPI0 をデータ通信可能状態にしておくこと。

R_SPI0_ComDataRxDmac

データ通信関数

データ受信（DMAC 連動）処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_ComDataRxDmac( ulong_t ul_dest_addr, ulong_t ul_rx_data_size,
void (*pv_data_rx_dmac_end_callback ) ( void ) );
```

引数

ulong_t ul_dest_addr	I	受信データを格納する領域への先頭アドレス（32 バイト境界）
ulong_t ul_rx_data_size	I	受信データサイズ（32 バイト単位）
void (*pv_data_rx_dmac_end_callback) (void)	I	DMAC 連動データ受信完了時のコールバックアドレス（NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、SPI0 の内蔵 DMAC を使用し、引数 “ul_rx_data_size” に指定した受信データサイズまで、SPI0 と接続デバイスからのデータ受信処理を行います。

注意

本関数をコールする前に、R_SPI0_SfReadDataCmd関数、またはR_SPI0_ComDataTx関数をコールし、SPI0 をデータ受信可能状態にしておくこと。

R_SPI0_ComStop

データ通信関数

データ通信停止処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_spi0_if.h”
char_t R_SPI0_ComStop( void );
```

引数

なし

戻り値

RET_NORMAL	正常終了

解説

本関数は、SPI0 と接続デバイスへの通信停止処理を行います。
本関数は、送受信 FIFO をリセットし、SPI0 のスレーブセレクト信号をネゲートします。
本関数をコールすることにより、SPI0 が通信中及び、データ送受信中に停止することができます。

注意

SPI0 が通信状態時に、本関数をコールすること。
SPI0 初期設定の最後に、本関数をコールすること。

R_SPI0_Interrupt

割り込みハンドラ関数

SPI0 割り込みハンドラ

書式

```
#include “r_common.h”
#include “iodefine.h”
#include "r_spi0_if.h"
void R_SPI0_Interrupt( void );
```

引数

なし

戻り値

なし

解説

本関数は、SPI0I 割り込み内での処理を行います。

“SPI0I”割り込みハンドラから、本割り込みハンドラ関数をコールすることにより、データ送受信関数（R_SPI0_ComDataTx、R_SPI0_ComDataRx、R_SPI0_ComDataRxDmac）の継続処理及び完了確認を行います。

本割り込みハンドラ関数で、各データ送受信関数の処理完了を確認した場合、該当コールバック関数をコールします。

注意

本関数は、SPI0I 割り込みのハンドラ内からコールすること。

4. 動作条件

4.1 ポートについて

本デバイスドライバ使用時は、各ポートの有効状態で使用すること。表 5に有効の設定が必要ポートを示します。

表 5. ポート設定一覧

GPIO レジスタ	ビット	設定
PDCR	PD6MD	0 : SP0-MISO 機能を設定します
	PD4MD	0 : SP0-SCK_FB 機能を設定します
	PD2MD	0 : $\overline{\text{SP0-SS1}}$ 機能を設定します
	PD1MD	0 : $\overline{\text{SP0-SS2}}$ 機能を設定します
	PD0MD	0 : $\overline{\text{SP0-SS3}}$ 機能を設定します
PSEL1	PD2SEL	0 : $\overline{\text{SP0-SS1}}$ 機能選択を設定します
	PD1SEL	0 : $\overline{\text{SP0-SS2}}$ 機能選択を設定します
	PD0SEL	0 : $\overline{\text{SP0-SS3}}$ 機能選択を設定します

4.2 割り込みについて

本デバイスドライバ使用時は、各割り込み要因許可状態で使用すること。また、割り込みベクタテーブルに各割り込みハンドラ関数を登録すること。表 1に登録が必要な割り込みハンドラ関数を示します。

表 6. 割り込みベクタテーブル設定関数一覧

割り込み要因番号	割り込み要因	割り込みハンドラ関数
0xCC0	SPI0I	R_SPI0_Interrupt

4.3 使用メモリ領域について

本デバイスドライバで使用するメモリ領域を表 7に示します。また各ドライバ関数使用時に確保必要な構造体領域を表 8に示します。

表 7. メモリ使用量一覧表

内容	セクション	属性	バイト数
プログラムコード	P	code, align=4	5736 バイト
定数データ	C	data, align=4	0 バイト
初期値ありデータ	D	data, align=4	0 バイト
初期値なしデータ	B	data, align=4	28 バイト

表 8. 使用構造体領域一覧表

構造体 typedef 名*	内容	バイト数	使用ドライバ関数
spi0_wp_info_t	SPI0 ライトプロテクト情報管理構造体	4 バイト	R_SPI0_GetWpInfo

* SPI0 ドライバのインタフェースヘッダ（r_spi0_if.h）ファイルで TYPEDEF 定義しています。

5. 注意事項

・下記の SPI フラッシュデバイス仕様に対して、本デバイスドライバでは非サポートであります。

- Dual インプットモード
- Quad インプット／アウトプットモード

6. 付録

6.1 ドライバ関数の使用例

SPI0 ドライバの SPI フラッシュアクセス及び、SPI デバイスデータ通信時の使用例を図 1～図 6に示します。

6.1.1 SPI フラッシュリードデータアクセス時の使用例

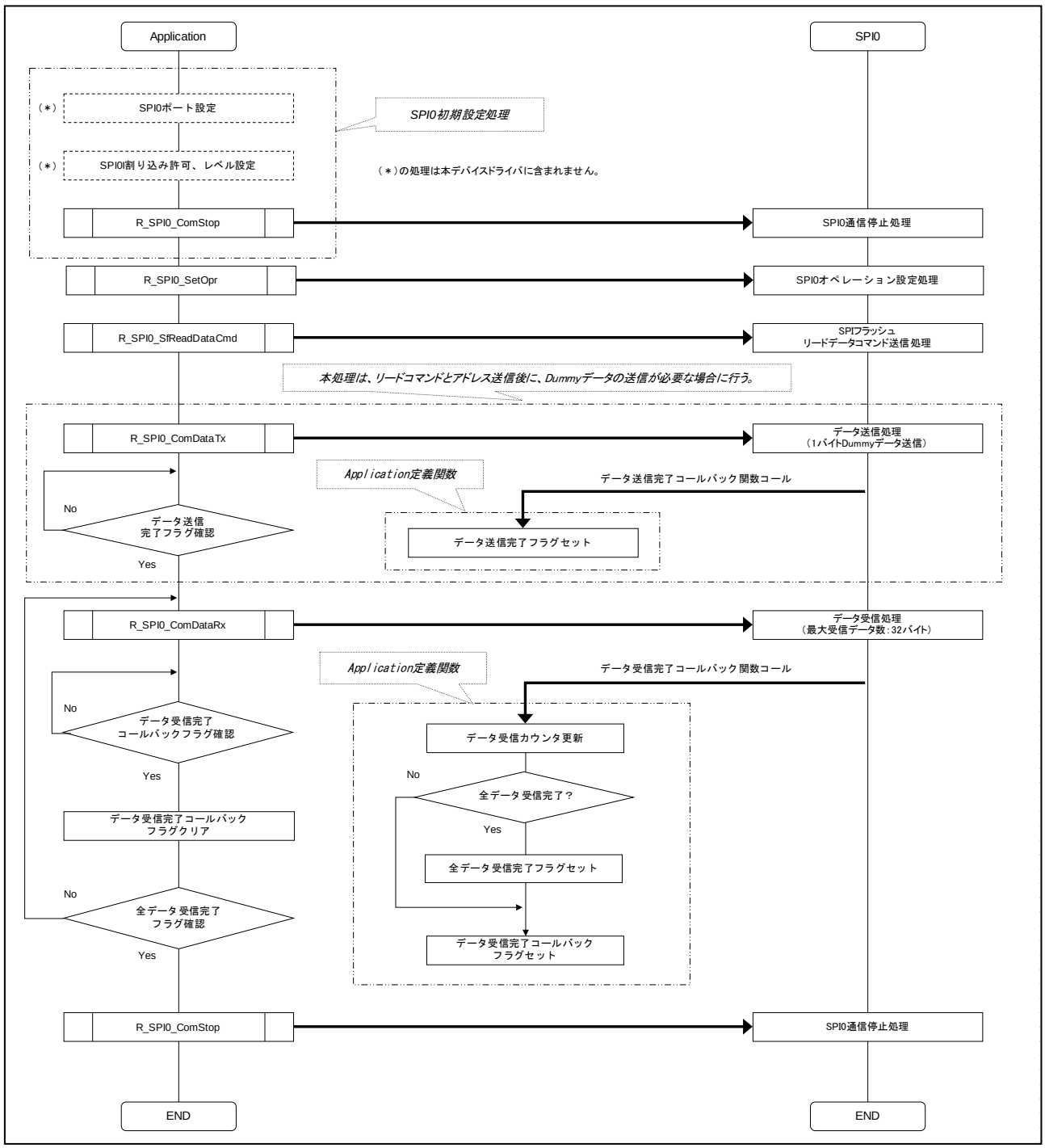

図 1. SPI フラッシュリードデータアクセス時の使用例

6.1.2 SPI フラッシュリードデータアクセス時の使用例（DMAC 連動）

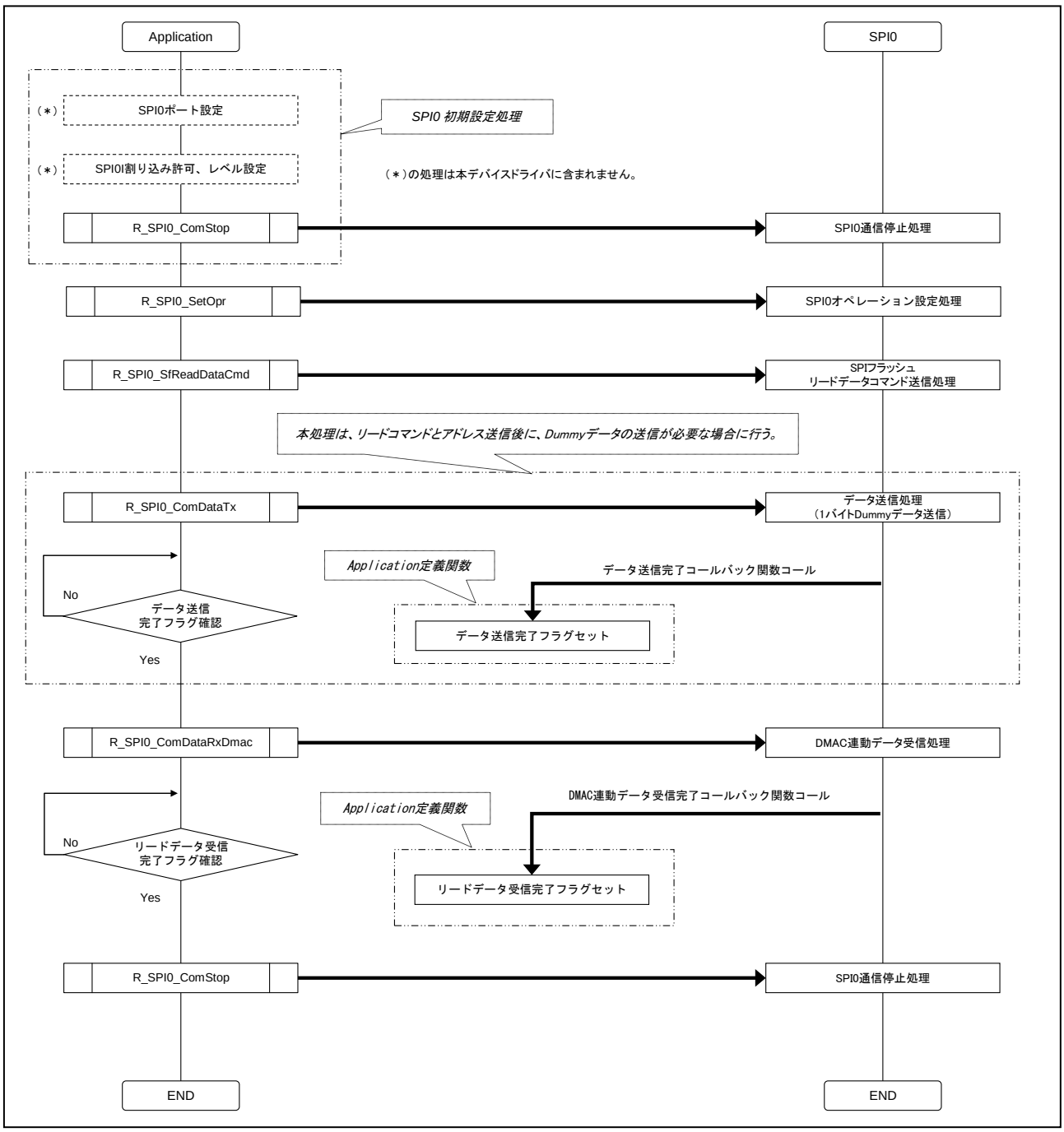

図 2. SPI フラッシュリードデータアクセス時の使用例（DMAC 連動）

6.1.3 SPI フラッシュページライトアクセスの処理例

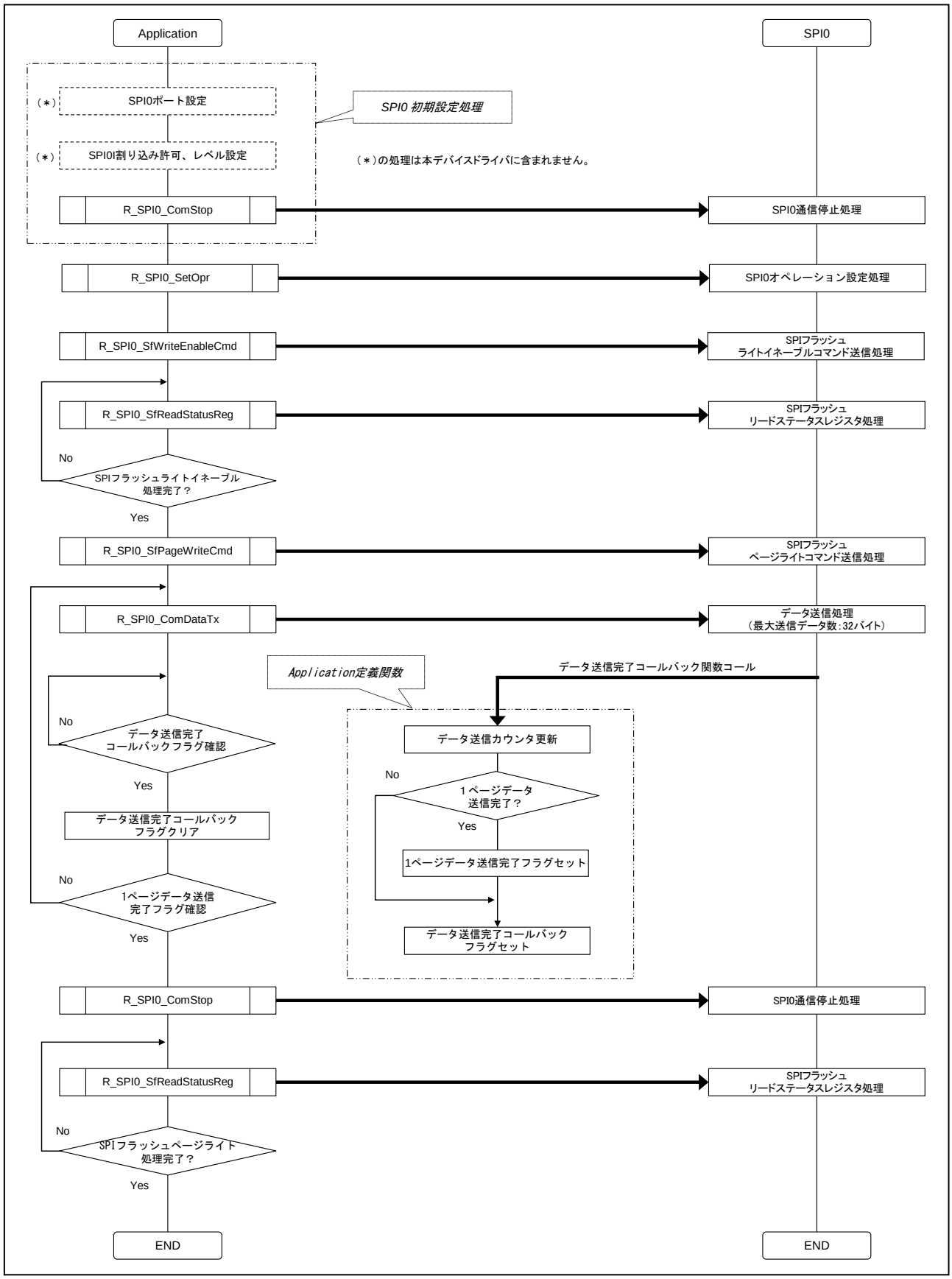

図 3. SPI フラッシュページライトアクセスの処理例

6.1.4 SPI フラッシュセクタイレースアクセスの処理例

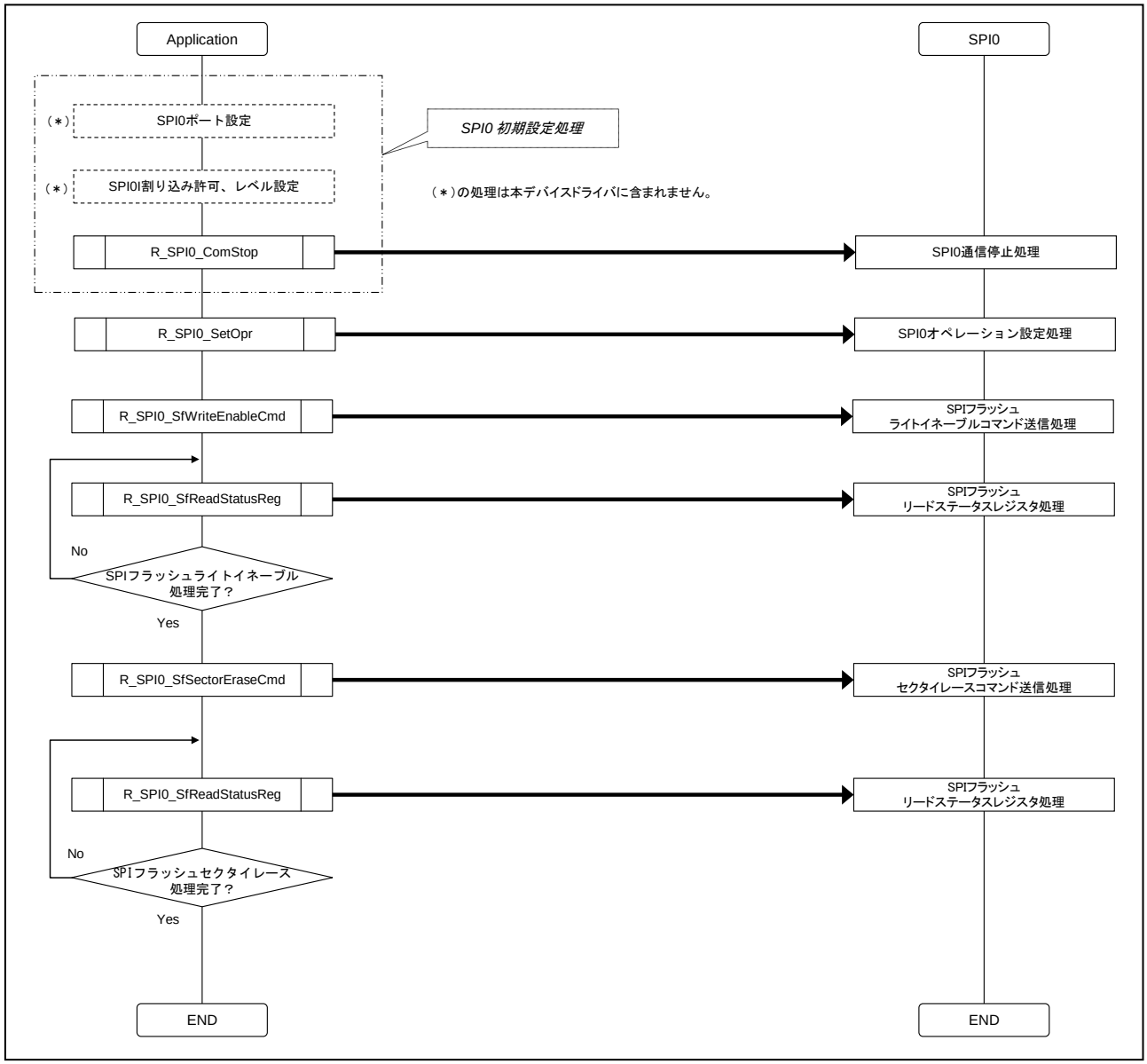

図 4. SPI フラッシュセクタイレースアクセスの処理例

6.1.5 SPI フラッシュバルクイレースアクセスの処理例

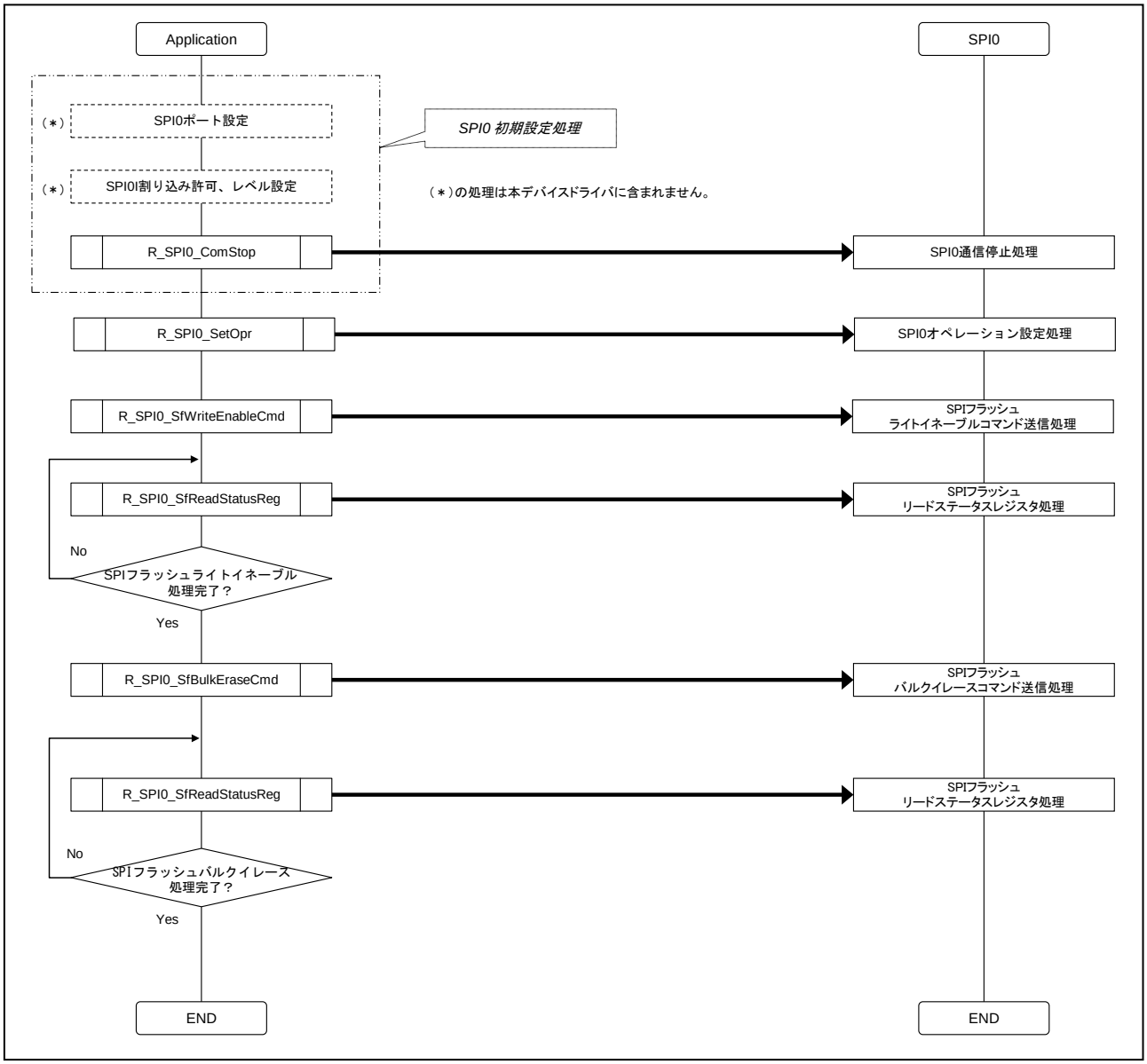

図 5. SPI フラッシュバルクイレースアクセスの処理例

6.1.6 SPI デバイスデータ送受信アクセス時の使用例

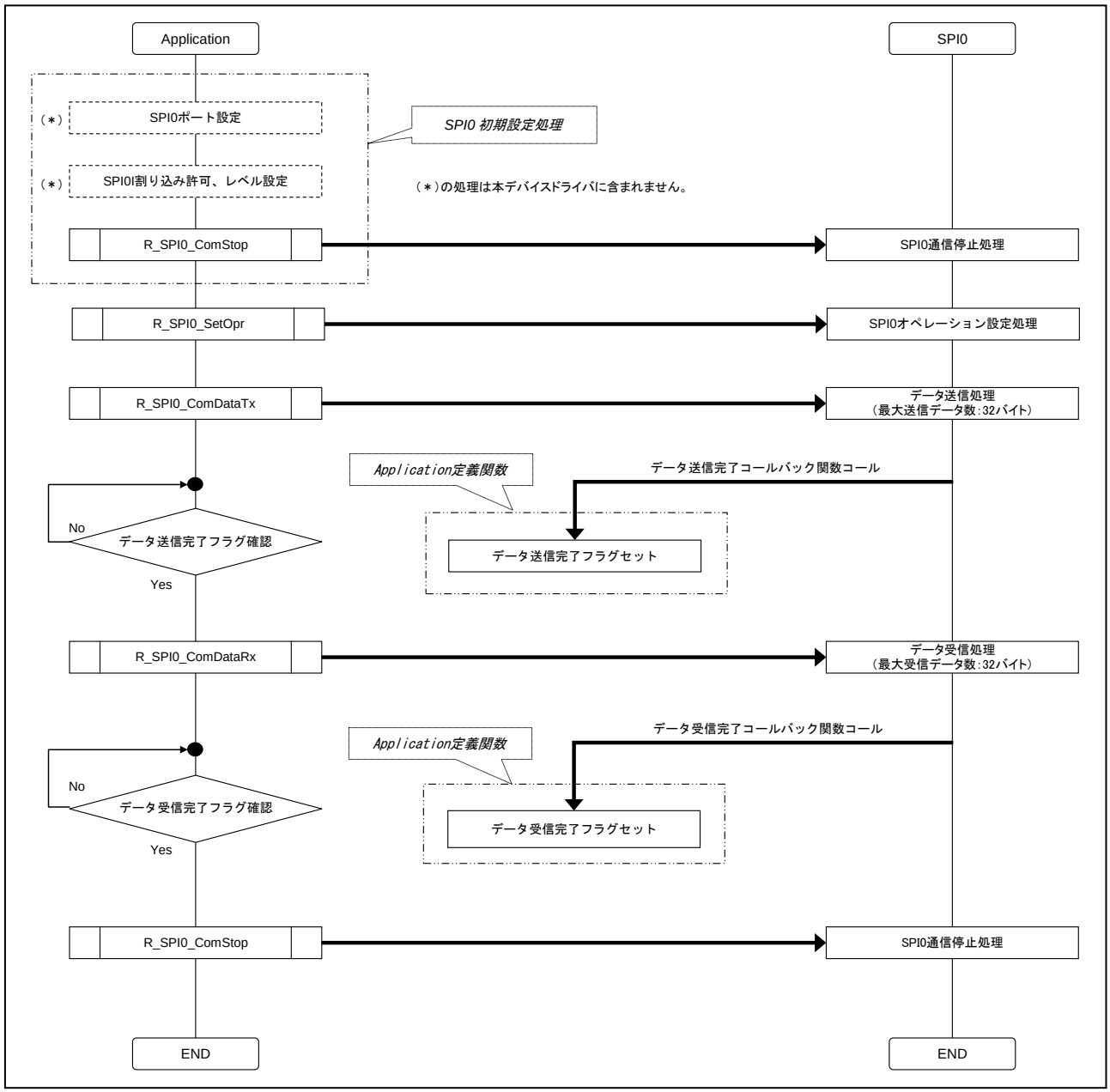

図 6. SPI デバイスデータ送受信アクセス時の使用例

ホームページとサポート窓口

- ルネサス エレクトロニクスホームページ
  http://japan.renesas.com/
- お問合せ先
  http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.7.1	—	初版発行

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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部 ROM、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、ノイズ輻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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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。
5. 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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
7. 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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。
8. 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。お客様が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。
9. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 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。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町 2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 2.1