RL78/G1E グループ
アナログ部との SPI 通信サンプルコード

R01AN1130JJ0120
Rev.1.20
2013.09.30

要旨

本アプリケーションノートは、RL78/G1E（R5F10FMx）のシリアル・アレイ・ユニット 1 チャネル 1 の 3 線シリアル I/O 機能（CSI21）を使用して、SPI 制御レジスタにアクセスするためのサンプルコードについて説明します。

動作確認デバイス

RL78/G1E（R5F10FMx（x = C, D, E））

本サンプルコード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。

目次

1. 仕様

1.1 概要

本アプリケーションノートは、RL78/G1E（R5F10FMx）のシリアル・アレイ・ユニット 1 チャネル 1 の 3 線シリアル I/O 機能を使用して、アナログ部との SPI 通信を行うサンプルコードについて説明します。

本アプリケーションノートのサンプルコードは、コード生成ツール（CubeSuite+）で生成されるシリアル関数の一部を使用しています。また、サンプルコードを呼び出す前には、必ずシリアル初期化関数（自動生成コード関数）を呼び出してください。

サンプルコードを実行するのに必要なシリアル関数（自動生成コード関数）を以下に示します。

- R_SAU1_Create　：本サンプルコード実行前に必ず呼び出してください。
- R_CSI21_Create　：R_SAU1_Create 関数から呼ばれます。
- R_CSI21_Start　：本サンプルコードから呼ばれます。
- R_CSI21_Stop　：本サンプルコードから呼ばれます。
- R_CSI21_SendReceive　：本サンプルコードから呼ばれます。
- r_csi21_interrupt　：CSI21 割り込みハンドラです。
- r_csi21_callback_receiveend　：r_csi21_interrupt から呼ばれます。
- r_csi21_callback_error　：r_csi21_interrupt から呼ばれます。

1.2 使用手順

本アプリケーションノートのサンプルコードを使用する方法について説明します。下記手順に従って、サンプルコードをご使用ください。

(1) CubeSuite+のコード生成ツールを用いて、サンプルコードを実行するのに必要な関数を生成してください。その際、必ず以下の設定を行ってください。

- オーバラン・エラーのコールバック機能を有効にする
  シリアル・アレイ・ユニット 1 チャネル 1 の 3 線シリアル I/O 機能（CSI21）のコールバック機能設定にて、オーバラン・エラーのチェックボックスにチェックを入れてください。
- リセットを“ H”出力にする
  リセット端子が接続されたポートを有効にし、“ H（1）”出力としてください。

(2) r_sa_spi_control_register.c（および r_sa_spi_control_register.h）ファイルをプロジェクトに追加してください。

(3) 自動生成されたファイルに下記ソースコードを追加してください。

r_cg_serial_user.c ファイル　（※自動生成されたファイル）

```
//インクルードファイルを追加
#include "r_sa_spi_control_register.h"

//オーバラン・エラーフラグのグローバル宣言を追加
volatile uint8_t   g_csi21_overrun_flag;

//r_csi21_callback_error 関数内に下記処理を追加

static void r_csi21_callback_error(uint8_t err_type)
{
    /* Start user code. Do not edit comment generated here */
  uint8_t    dummy;

  /* Clear overrun flag of R5F10FMx register */
  dummy = SIO21;
  dummy = (uint8_t)(SSR11 & _0001_SAU_OVERRUN_ERROR);
  SIR11 = (uint16_t)dummy;

  /* Set overrun flag of global variable */
  g_csi21_overrun_flag = 1;

  /* Set CS output level high and stop CSI21 */
  SPI_CS = 1;
  R_CSI21_Stop();
    /* End user code. Do not edit comment generated here */
}

//r_csi21_callback_receiveend 関数内に下記処理を追加

static void r_csi21_callback_receiveend(void)
{
    /* Start user code. Do not edit comment generated here */
  /* Set CS output level high and stop CSI21 */
  SPI_CS = 1;
  R_CSI21_Stop();
    /* End user code. Do not edit comment generated here */
}
```

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2.1　動作確認条件

項目	内容
使用マイコン	RL78/G1E（R5F10FME）
動作周波数	• 高速オンチップ・オシレータ（高速 OCD）クロック：32MHz • CPU/周辺ハードウェア・クロック：32MHz
動作電圧	• V_{DD}, DV_{DD}, AV_{DD1}, AV_{DD2}, AV_{DD3}：5.0V • AV_{DD}：3.3V • LVD 検出（V_{LVIH}）：立ち上がり 4.06V、立ち下がり 3.98V
使用外部デバイス	なし
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V1.01.01 [31 Jan 2012]
C コンパイラ（ビルド・ツール）	ルネサス エレクトロニクス製 CA78K0R V1.30
RL78/G1A コードライブラリ	ルネサス エレクトロニクス製 CodeGenerator for RL78/G1A E1.00.00c [22 Dec 2011]

3. 関連アプリケーションノート

関連するアプリケーションノートを以下に示します。併せてご参照ください。

- RL78/G13　初期設定（R01AN0451J）アプリケーションノート
- RL78/G13　シリアル・アレイ・ユニット 3 線シリアル I/O（マスタ送受信）（R01AN0460J）アプリケーションノート

4. 関数一覧

本アプリケーションノートのサンプルコードの関数一覧を表 4.1に示します。

表 4.1　関数一覧

アクセス	関数名	概要
バイト操作（8 ビット）	R_SPI_SmartAnalogRead	SPI 制御レジスタ読み出し関数
	R_SPI_SmartAnalogWrite	SPI 制御レジスタ書き込み関数
	R_SPI_SmartAnalogWriteVerify	SPI 制御レジスタ書き込みチェック関数
ビット操作（1 ビット）	R_SPI_SmartAnalogReadBit	SPI 制御レジスタビット読み出し関数
	R_SPI_SmartAnalogWriteBit	SPI 制御レジスタビット書き込み関数
	R_SPI_SmartAnalogWriteVerifyBit	SPI 制御レジスタビット書き込みチェック関数

5. 戻り値一覧

本アプリケーションノートのサンプルコードの戻り値一覧を表 5.1に示します

表 5.1　戻り値一覧

型名	戻り値一覧		
	マクロ名	値	内容
spi_status_t (uint8_t)	SPI_OK	00H	正常終了
	SPI_ERR_PARAM	01H	パラメータ・エラー
	SPI_ERR_COM	02H	SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）
	SPI_ERR_VERIFY	03H	ベリファイ・エラー

6. 構造体一覧

本アプリケーションノートのサンプルコードで使用する構造体について以下に説明します。

6.1 バイト操作関数で使用する構造体

バイト操作（8 ビット）関数で使用する構造体を表 6.1に示します。

表 6.1 バイト操作関数で使用する構造体

構造体型名	spi_data_t		
概要	SPI 制御レジスタの読み出し/書き込みデータ・フォーマット		
型サイズ	2 Byte		
メンバ変数	型	名称	内容
	uint8_t	address	SPI 制御レジスタのアドレス
	uint8_t	data	SPI 制御レジスタのデータ

6.2 ビット操作関数で使用する構造体

ビット操作（1 ビット）関数で使用する構造体を表 6.2に示します。

表 6.2 ビット操作関数で使用する構造体

構造体型名	spi_data_bit_t		
概要	SPI 制御レジスタのビット読み出し/ビット書き込みデータ・フォーマット		
型サイズ	3 Byte		
メンバ変数	型	名称	内容
	uint8_t	address	SPI 制御レジスタのアドレス
	uint8_t	bitNum	SPI 制御レジスタのビット番号（0～7）
	uint8_t	bitData	SPI 制御レジスタのビット・データ（0 / 1）

7. 関数仕様

本アプリケーションノートのサンプルコードの関数仕様について以下に説明します。

7.1 SPI 制御レジスタ読み出し関数

SPI 制御レジスタ読み出し関数（R_SPI_SmartAnalogRead）の関数仕様を表 7.1に示します。

表 7.1 SPI 制御レジスタ読み出し関数の仕様

関数名	R_SPI_SmartAnalogRead		
概要	**SPI 制御レジスタ読み出し関数**		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogRead（spi_data_t　*data,　unit8_t　num）		
関数説明	**CSI21 によるアナログ部の SPI 制御レジスタからデータの読み出し**		
引数	**型**	**名称**	**内容**
	spi_data_t　*	data	**SPI 制御レジスタのアドレスと読み出しデータのセットを格納したバッファへのポインタ**
	uint8_t	num	**spi_data_t の要素数**
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	**正常終了**
	SPI_ERR_COM	02H	**SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）**

7.2 SPI 制御レジスタ書き込み関数

SPI 制御レジスタ書き込み関数（R_SPI_SmartAnalogWrite）の関数仕様を表 7.2に示します。

表 7.2 SPI 制御レジスタ書き込み関数の仕様

関数名	R_SPI_SmartAnalogWrite		
概要	**SPI 制御レジスタ書き込み関数**		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogWrite（spi_data_t　*data,　unit8_t　num）		
関数説明	**CSI21 によるアナログ部の SPI 制御レジスタへのデータの書き込み**		
引数	**型**	**名称**	**内容**
	spi_data_t　*	data	**SPI 制御レジスタのアドレスと書き込みデータのセットを格納したバッファへのポインタ**
	uint8_t	num	**spi_data_t の要素数**
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	**正常終了**
	SPI_ERR_COM	02H	**SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）**

7.3　SPI 制御レジスタ書き込みチェック関数

SPI 制御レジスタ書き込みチェック関数（R_SPI_SmartAnalogWriteVerify）の関数仕様を表 7.3に示します。

表 7.3　SPI 制御レジスタ書き込みチェック関数の仕様

関数名	R_SPI_SmartAnalogWriteVerify		
概要	SPI 制御レジスタ書き込みチェック関数		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogWriteVerify（spi_data_t　*data,　unit8_t　num,　uint8_t　*errIndex）		
関数説明	CSI21 によるアナログ部の SPI 制御レジスタへのデータの書き込みとチェック処理		
引数	**型**	**名称**	**内容**
	spi_data_t　*	data	SPI 制御レジスタのアドレスと書き込みデータのセットを格納したバッファへのポインタ
	uint8_t	num	spi_data_t の要素数
	uint8_t　*	errIndex	ベリファイ・エラーが発生した spi_data_t の要素番号（0～num-1）を格納するバッファへのポインタ
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	正常終了
	SPI_ERR_COM	02H	SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）
	SPI_ERR_VERIFY	03H	ベリファイ・エラー

7.4　SPI 制御レジスタビット読み出し関数

SPI 制御レジスタビット読み出し関数（R_SPI_SmartAnalogReadBit）の関数仕様を表 7.4に示します。

表 7.4　SPI 制御レジスタビット読み出し関数の仕様

関数名	R_SPI_SmartAnalogReadBit		
概要	SPI 制御レジスタビット読み出し関数		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogReadBit（spi_data_bit_t　*data,　unit8_t　num,　uint8_t　*errIndex）		
関数説明	CSI21 によるアナログ部の SPI 制御レジスタからビット・データを読み出し		
引数	**型**	**名称**	**内容**
	spi_data_bit_t　*	data	SPI 制御レジスタのアドレスとビット番号と読み出しビット・データのセットを格納したバッファへのポインタ
	uint8_t	num	spi_data_bit_t の要素数
	uint8_t　*	errIndex	パラメータ・エラーが発生した spi_data_bit_t の要素番号（0～num-1）を格納するバッファへのポインタ
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	正常終了
	SPI_ERR_PARAM	01H	パラメータ・エラー（ビットが不正）
	SPI_ERR_COM	02H	SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）

7.5　SPI 制御レジスタビット書き込み関数

SPI 制御レジスタビット書き込み関数（R_SPI_SmartAnalogWriteBit）の関数仕様を表 7.5に示します。

表 7.5　SPI 制御レジスタビット書き込み関数の仕様

関数名	R_SPI_SmartAnalogWriteBit		
概要	**SPI 制御レジスタビット書き込み関数**		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogWriteBit（spi_data_bit_t　*data,　unit8_t　num,　uint8_t　*errIndex）		
関数説明	**CSI21 によるアナログ部の SPI 制御レジスタにビット・データを書き込み**		
引数	**型**	**名称**	**内容**
	spi_data_bit_t　*	data	**SPI 制御レジスタのアドレスとビット番号と書き込みビット・データのセットを格納したバッファへのポインタ**
	uint8_t	num	**spi_data_bit_t の要素数**
	uint8_t　*	errIndex	**パラメータ・エラーが発生した spi_data_bit_t の要素番号（0～num-1）を格納するバッファへのポインタ**
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	**正常終了**
	SPI_ERR_PARAM	01H	**パラメータ・エラー（ビット番号またはビット・データが不正）**
	SPI_ERR_COM	02H	**SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）**

7.6　SPI 制御レジスタビット書き込みチェック関数

SPI 制御レジスタビット書き込みチェック関数（R_SPI_SmartAnalogWriteVerifyBit）の関数仕様を表 7.6に示します。

表 7.6　SPI 制御レジスタビット書き込みチェック関数の仕様

関数名	R_SPI_SmartAnalogWriteVerifyBit		
概要	SPI 制御レジスタビット書き込みチェック関数		
ヘッダ・ファイル	r_sa_spi_control_register.h		
宣言	spi_status_t　R_SPI_SmartAnalogWriteVerify Bit（spi_data_bit_t　*data,　unit8_t num,　uint8_t　*errIndex）		
関数説明	CSI21 によるアナログ部の SPI 制御レジスタにビット・データを書き込みとチェック処理		
引数	**型**	**名称**	**内容**
	spi_data_bit_t　*	data	SPI 制御レジスタのアドレスとビット番号と書き込みビット・データのセットを格納したバッファへのポインタ
	uint8_t	num	spi_data_bit_t の要素数
	uint8_t　*	errIndex	パラメータ・エラー、またはベリファイ・エラーが発生した spi_data_bit_t の要素番号（0～num-1）を格納するバッファへのポインタ
戻り値	**マクロ名**	**値**	**内容**
	SPI_OK	00H	正常終了
	SPI_ERR_PARAM	01H	パラメータ・エラー（ビット番号またはビット・データが不正）
	SPI_ERR_COM	02H	SPI 通信エラー（オーバラン・エラー、タイムアウト・エラー）
	SPI_ERR_VERIFY	03H	ベリファイ・エラー

8. フローチャート

本アプリケーションノートのサンプルコードのフローチャートを以下に示します。

8.1 SPI 制御レジスタ読み出し関数

図 8.1に SPI 制御レジスタ読み出し関数のフローチャートを示します。

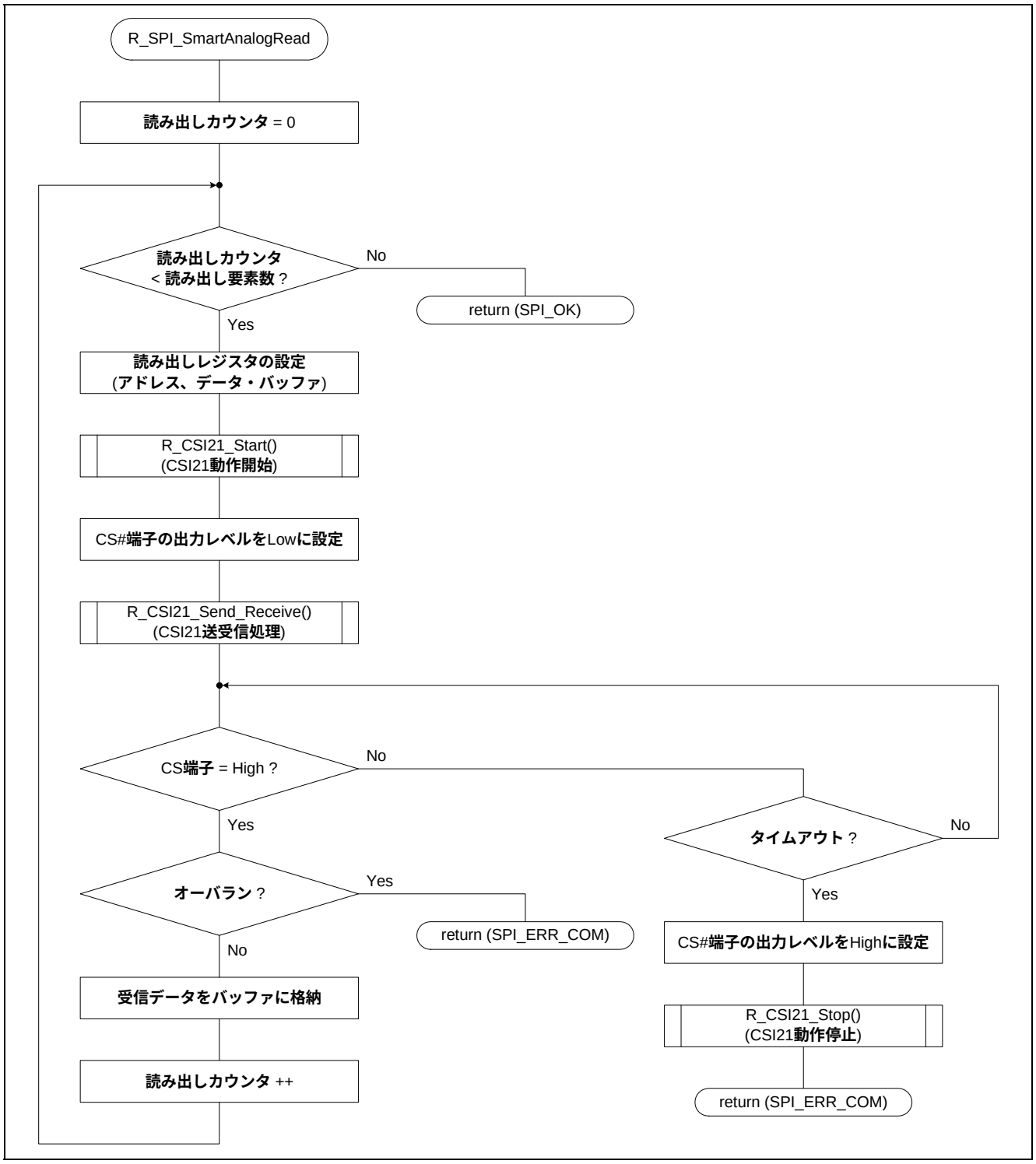

図 8.1　SPI 制御レジスタ読み出し関数のフローチャート

8.2　SPI 制御レジスタ書き込み関数

図 8.2に SPI 制御レジスタ書き込み関数のフローチャートを示します。

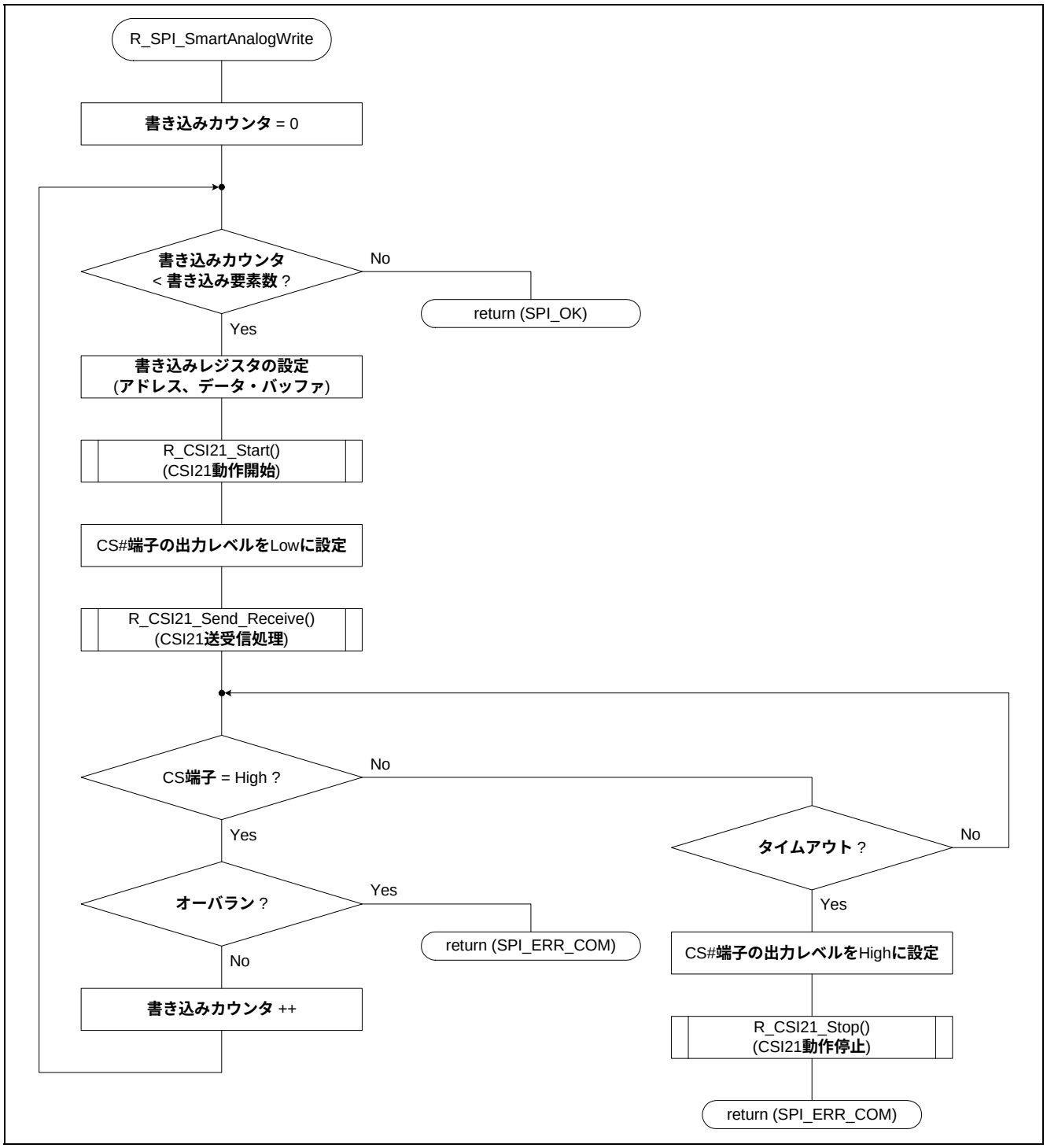

図 8.2　SPI 制御レジスタ書き込み関数のフローチャート

8.3　SPI 制御レジスタ書き込みチェック関数

図 8.3に SPI 制御レジスタ書き込みチェック関数のフローチャートを示します。

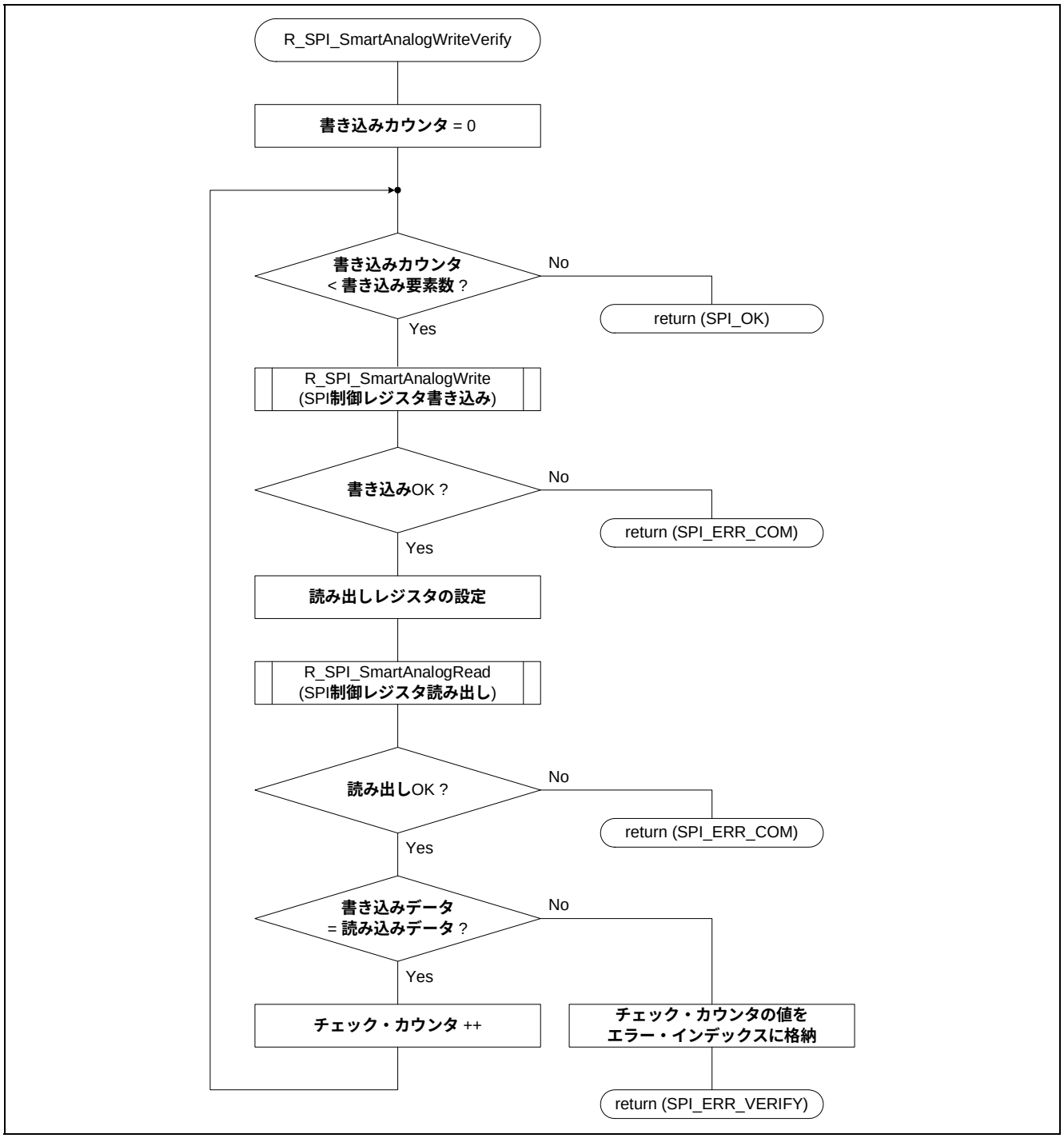

図 8.3　SPI 制御レジスタ書き込みチェック関数のフローチャート

8.4　SPI 制御レジスタビット読み出し関数

図 8.4に SPI 制御レジスタビット読み出し関数のフローチャートを示します。

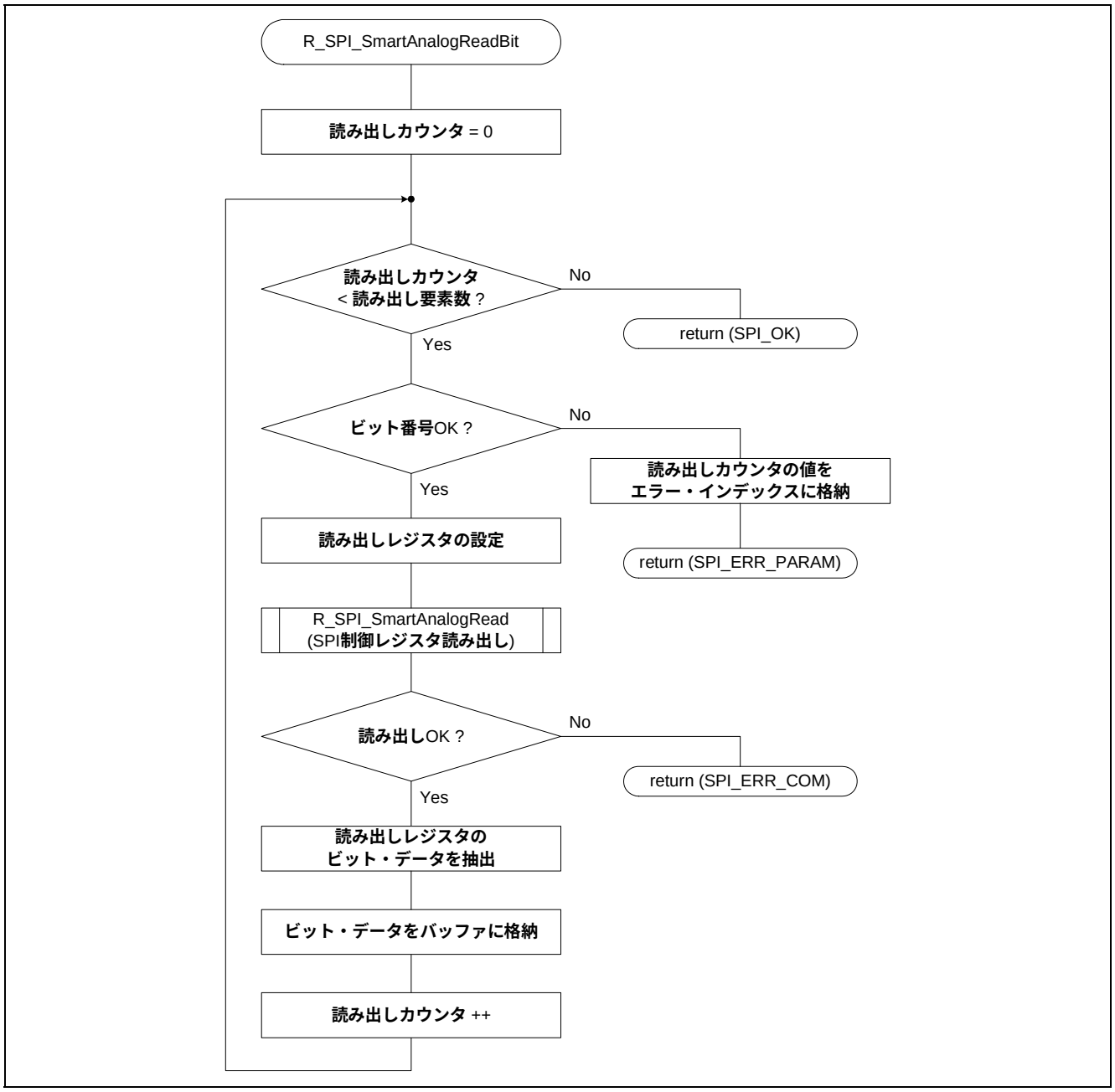

図 8.4　SPI 制御レジスタビット読み出し関数のフローチャート

8.5　SPI 制御レジスタビット書き込み関数

図 8.5に SPI 制御レジスタビット書き込み関数のフローチャートを示します。

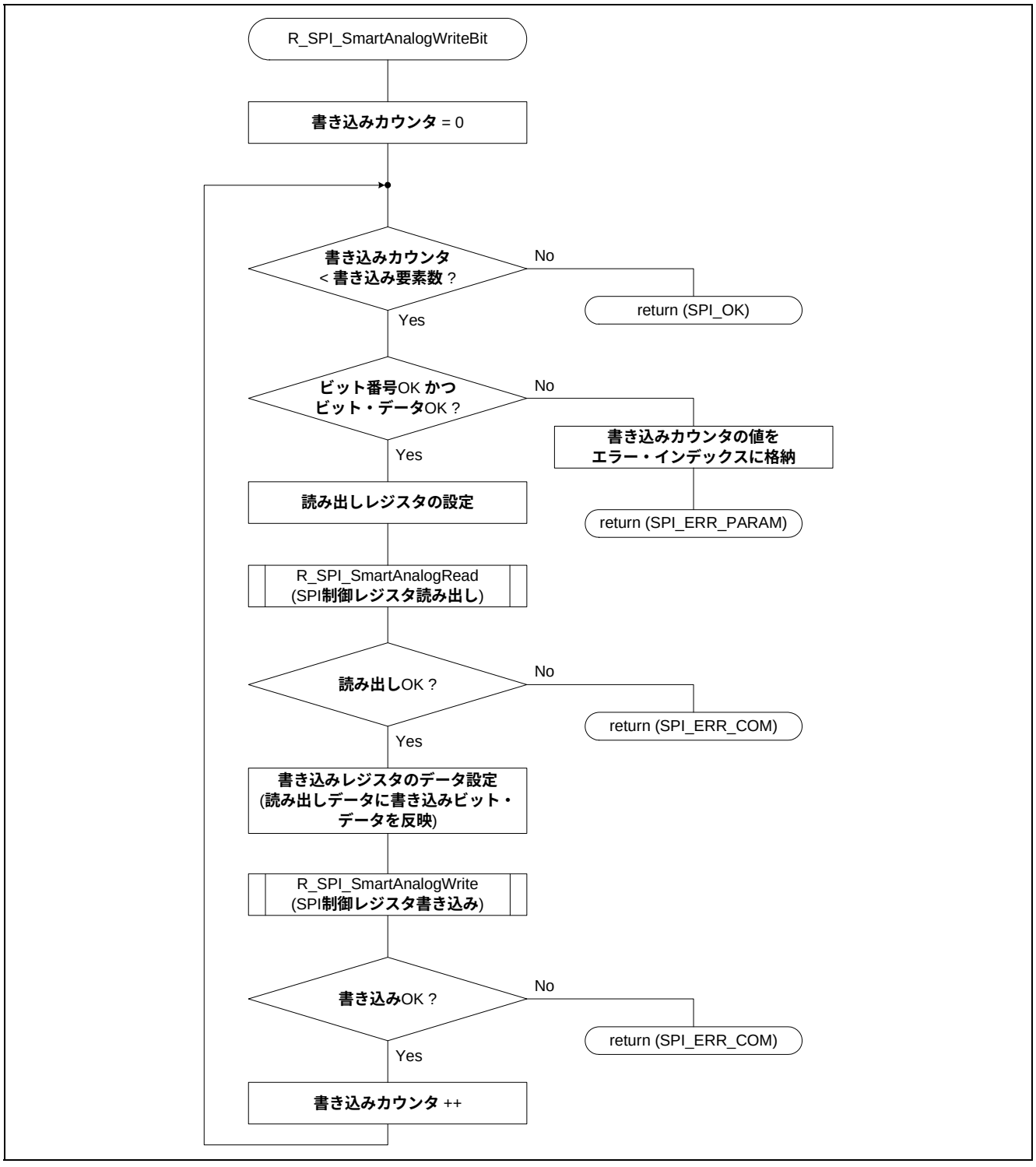

図 8.5　SPI 制御レジスタビット書き込み関数のフローチャート

8.6　SPI 制御レジスタビット書き込みチェック関数

図 8.6に SPI 制御レジスタビット書き込みチェック関数のフローチャートを示します。

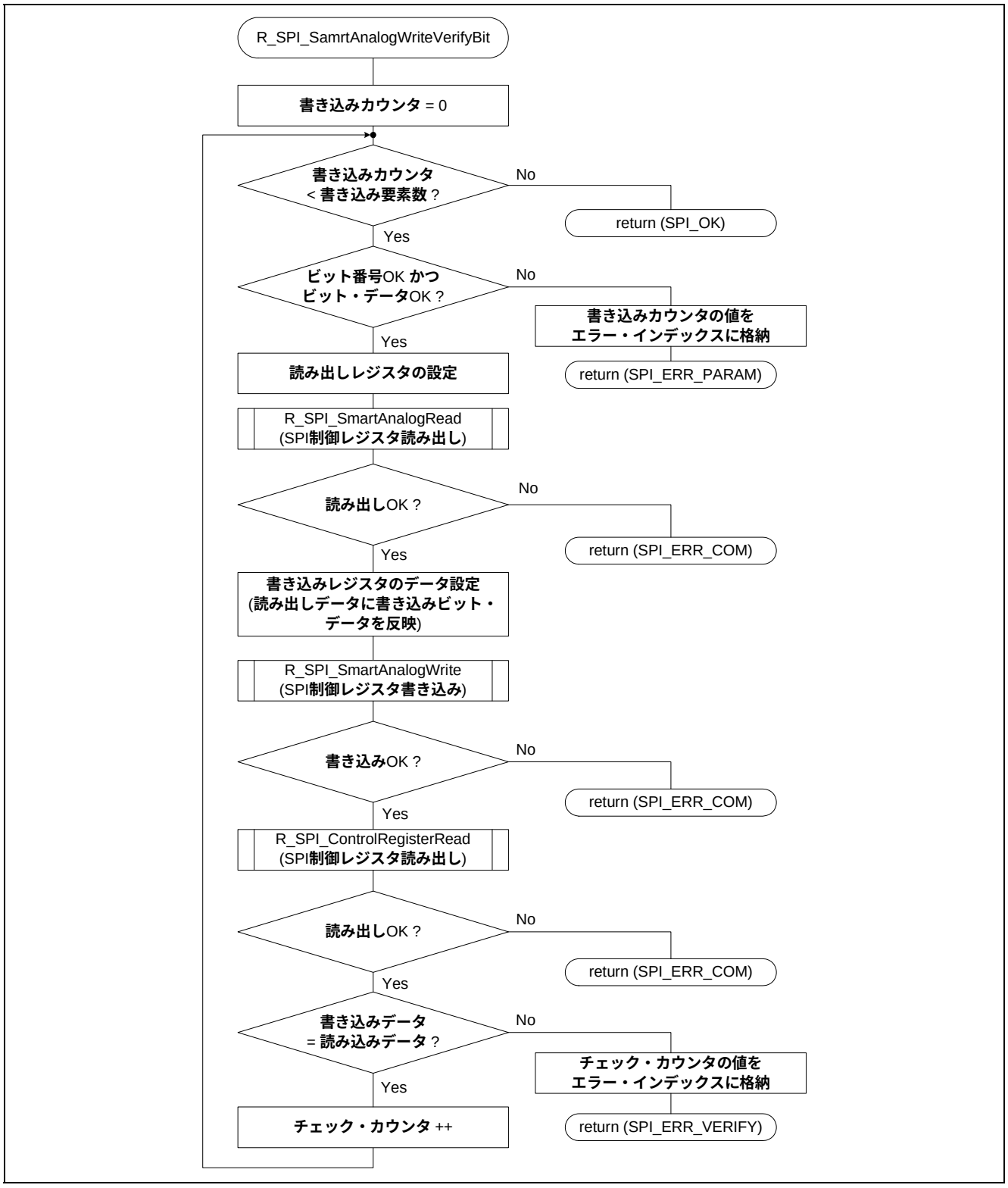

図 8.6　SPI 制御レジスタビット書き込みチェック関数のフローチャート

9. サンプルコードの使用例

本アプリケーションノートのサンプルコードの使用例について以下に示します。

9.1 SPI 制御レジスタ読み出し関数の使用例

SPI 制御レジスタ読み出し関数（R_SPI_SmartAnalogRead）の使用例を図 9.1に示します。

■ 使用例

(1) アドレス 00H を読み出し
(2) アドレス 01H を読み出し
(3) アドレス 03H を読み出し
(4) アドレス 04H を読み出し
(5) アドレス 05H を読み出し

```
#include "r_cg_macrodriver.h"
#include "r_sa_control_register.h"

void main(void)
{
     uint8_t        errCode;   // 関数戻り値格納用
     uint8_t        temp[5];

     // 読み出し用のバッファを用意し、アドレスを設定
     // (読み出し時のデータ初期値は任意)
     spi_data_t     readData[5] = {
          {0x00, 0x00},        // address: 00H, data: 00H(dummy)
          {0x01, 0x00},        // address: 01H, data: 00H(dummy)
          {0x03, 0x00},        // address: 03H, data: 00H(dummy)
          {0x04, 0x00},        // address: 04H, data: 00H(dummy)
          {0x05, 0x00}         // address: 05H, data: 00H(dummy)
     };

     // SPI制御レジスタ読み出し
     errCode = R_SPI_SmartAnalogRead(readData, 5);

     // エラーチェック
     if (errCode != SPI_OK) {
          // エラー処理
     }
     else {
          // 読み出しデータの取得
          temp[0] = readData[0].data;     //アドレス00Hのデータを取得
          temp[1] = readData[1].data;     //アドレス01Hのデータを取得
          temp[2] = readData[2].data;     //アドレス03Hのデータを取得
          temp[3] = readData[3].data;     //アドレス04Hのデータを取得
          temp[4] = readData[4].data;     //アドレス05Hのデータを取得
     }
     while(1);
}
```

図 9.1　SPI 制御レジスタ読み出し関数の使用例

9.2　SPI 制御レジスタ書き込み関数の使用例

SPI 制御レジスタ書き込み関数（R_SPI_SmartAnalogWrite）の使用例を図 9.2に示します。

■　使用例

(1)　アドレス 11H にデータ 57H を書き込み

```
#include "r_cg_macrodriver.h"
#include "r_sa_spi_control_register.h"

void main(void)
{
      uint8_t         errCode;   // 関数戻り値格納用

      // 書き込み用のバッファを用意し、アドレスとデータを設定
      spi_data_t      writeData[1] = {
           {0x11, 0x57}          // address: 11H, data: 57H
      };

      // SPI制御レジスタ書き込み
      errCode = R_SPI_SmartAnalogWrite(writeData, 1);

      // エラーチェック
      if (errCode != SPI_OK) {
           // エラー処理
      }
      else {
           // 正常処理
      }

      while(1);
}
```

図 9.2　SPI 制御レジスタ書き込み関数の使用例

9.3　SPI 制御レジスタ書き込みチェック関数の使用例

SPI 制御レジスタ書き込みチェック関数（R_SPI_SamrtAnalogWriteVerify）の使用例を図 9.3に示します。

■　使用例

(1)　アドレス 0BH にデータ 0DH を書き込み
(2)　アドレス 12H にデータ 02H を書き込み
(3)　（1）、（2）を実行後、ベリファイ処理を実行

```
#include "r_cg_macrodriver.h"
#include "r_sa_spi_control_register.h"

void main(void)
{
      uint8_t         errCode;   // 関数戻り値格納用
      uint8_t         errIndex;  // エラーインデックス格納用
      uint8_t         temp;

      // 書き込み用のバッファを用意し、アドレスとデータを設定
      spi_data_t      writeData[2] = {
           {0x0B, 0x0D},         // address: 0BH, data: 0DH
           {0x12, 0x02}          // address: 12H, data: 02H
      };

      // SPI制御レジスタ書き込みチェック
      errCode = R_SPI_SmartAnalogWriteVerify(writeData, 2, &errIndex);

      // エラーチェック
      if (errCode == SPI_OK) {
           // 正常処理
      }
      else if (errCode == SPI_ERR_VERIFY){
           // ベリファイエラー時
           temp = errIndex;      // ベリファイエラーが発生したインデックスを取得
      }
      else {
           // その他エラー処理
      }

      while(1);
}
```

図 9.3　SPI 制御レジスタ書き込みチェック関数の使用例

9.4　SPI 制御レジスタビット読み出し関数の使用例

SPI 制御レジスタビット読み出し関数（R_SPI_SmartAnalogReadBit）の使用例を図 9.4に示します。

■ 使用例

(1) アドレス 01H のビット 6 を読み出し

```
#include "r_cg_macrodriver.h"
#include "r_sa_spi_control_register.h"

void main(void)
{
      uint8_t         errCode;   // 関数戻り値格納用
      uint8_t         errIndex;  // エラーインデックス格納用
      uint8_t         temp;

      // 読み出し用のバッファを用意し、アドレスとビット番号を設定
      // (読み出し時のビットデータ初期値は任意)
      spi_data_bit_t  readData[1] = {
           {0x01, 6, 0}                              // address: 01H, bitNum: 6, bitData: 0(dummy)
      };

      // SPI制御レジスタビット読み出し
      errCode = R_SPI_SmartAnalogReadBit(readData, 1, &errIndex);

      // エラーチェック
      if (errCode == SPI_OK) {
           // 読み出しビットデータの取得
           temp = readData[0].bitData;     //アドレス01Hのビット6のデータを取得
      }
      else if (errCode == SPI_ERR_PARAM){
           // パラメータエラー時
           temp = errIndex;                // パラメータエラーが発生したインデックスを取得
      }
      else {
           // その他エラー処理
      }

      while(1);
}
```

図 9.4　SPI 制御レジスタビット読み出し関数の使用例

9.5　SPI 制御レジスタビット書き込み関数の使用例

SPI 制御レジスタビット書き込み関数（R_SPI_SmartAnalogWriteBit）の使用例を図 9.5 に示します。

■　使用例

(1)　アドレス 11H のビット 2 に 1 を書き込み

(2)　アドレス 12H のビット 4 に 0 を書き込み

```
#include "r_cg_macrodriver.h"
#include "r_sa_spi_control_register.h"

void main(void)
{
      uint8_t         errCode;   // 関数戻り値格納用
      uint8_t         errIndex;  // エラーインデックス格納用
      uint8_t         temp;

      // 書き込み用のバッファを用意し、アドレスとビット番号とビットデータを設定
      spi_data_bit_t  writeData[2] = {
           {0x11, 2, 1},                  // address: 11H, bitNum: 2, bitData: 1
           {0x12, 4, 0}                   // address: 12H, bitNum: 4, bitData: 0
      };

      // SPI制御レジスタビット書き込み
      errCode = R_SPI_SmartAnalogWriteBit(writeData, 2, &errIndex);

      // エラーチェック
      if (errCode == SPI_OK) {
           // 正常処理
      }
      else if (errCode == SPI_ERR_PARAM){
           // パラメータエラー時
           temp = errIndex;      // パラメータエラーが発生したインデックスを取得
      }
      else {
           // その他エラー処理
      }

      while(1);
}
```

図 9.5　SPI 制御レジスタビット書き込み関数の使用例

9.6　SPI 制御レジスタビット書き込みチェック関数の使用例

SPI 制御レジスタビット書き込みチェック関数（R_SPI_SmartAnalogWriteVerifyBit）の使用例を図 9 6 に示します。

■ 使用例

(1) アドレス 01H のビット 1 に 1 を書き込み
(2) アドレス 11H のビット 2 に 0 を書き込み
(3) アドレス 12H のビット 3 に 1 を書き込み
(4) （1）、（2）、（3）を実行後、ベリファイ処理を実施

```
#include "r_cg_macrodriver.h"
#include "r_sa_spi_control_register.h"

void main(void)
{
     uint8_t        errCode;  // 関数戻り値格納用
     uint8_t        errIndex; // エラーインデックス格納用
     uint8_t        temp;

     // 書き込み用のバッファを用意し、アドレスとビット番号とビットデータを設定
     spi_data_bit_t  writeData[3] = {
          {0x01, 1, 1},                  // address: 01H, bitNum: 1, bitData: 1
          {0x11, 2, 0},                  // address: 11H, bitNum: 2, bitData: 0
          {0x12, 3, 1}                   // address: 12H, bitNum: 3, bitData: 1
     };

     // SPI制御レジスタビット書き込みチェック
     errCode = R_SPI_SmartAnalogWriteVerifyBit(writeData, 3, &errIndex);

     // エラーチェック
     if (errCode == SPI_OK) {
          // 正常処理
     }
     else if (errCode == SPI_ERR_PARAM){
          // パラメータエラー時
          temp = errIndex;      // パラメータエラーが発生したインデックスを取得
     }
     else if (errCode == SPI_ERR_VERIFY){
          // ベリファイエラー時
          temp = errIndex;      // ベリファイエラーが発生したインデックスを取得
     }
     else {
          // その他エラー処理
     }

     while(1);
}
```

図 9.6　SPI 制御レジスタビット書き込みチェック関数の使用例

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/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.09.30	—	初版発行
1.10	2013.03.29	—	説明内容変更
1.20	2013.09.30	—	1.2 使用手順追加，関数名変更

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。

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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

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. **リザーブアドレスのアクセス禁止**
   【注意】リザーブアドレス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、事前に問題ないことをご確認下さい。
   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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・電話番号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） (03)5201-5307

■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/

© 2013 Renesas Electronics Corporation. All rights reserved.
Colophon 2.0