RZ/T1 グループ

ARM® Development Studio 5（DS-5™）のセミホスティング機能を使用したシリアルフラッシュメモリへのダウンロード例

R01AN2951JJ0120
Rev.1.20
2017.09.15

要旨

本アプリケーションノートは、RZ/T1 グループマイコンの外部アドレス空間（SPI マルチ I/O バス空間）に配置されたシリアルフラッシュメモリにプログラムをダウンロードする方法について説明しています。

なお、本アプリケーションノートで紹介するダウンロード方法は、ARM® Development Studio 5（DS-5™）（以下、DS-5 と略します）のセミホスティング機能を使用します。DS-5 は別途お客様で準備して頂く必要があります。DS-5 のセミホスティング機能の詳細については、ARM® より提供されるドキュメント注 1 を参照してください。

注 1. 「ARM® コンパイラツールチェーン ARM® プロセッサをターゲットとしたソフトウェア開発／セミホスティング」を参照してください。

対象デバイス

RZ/T1 グループ

本アプリケーションノートを他のマイコンに適用する場合、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

シリアルフラッシュメモリは、コードおよびデータを保存するために使用される一般的な不揮発性メモリです。シリアルフラッシュメモリへの書き込みはシリアルフラッシュメモリに応じた適切なアルゴリズムが必要です。本アプリケーションノートでは、このアルゴリズムを RZ/T1 グループマイコンの密結合メモリ（ATCM）上で実行する C ソースプログラムとして提供します。また、DS-5 のセミホスティング機能を使用し、DS-5 を実行するホストコンピュータのハードディスクに格納されている、SPIBSC 初期設定バイナリファイル、及びアプリケーションバイナリファイル注 1 を参照して、シリアルフラッシュメモリに書き込む方法を紹介します。

表 1.1 に　使用する周辺機能と用途を示します。

注 1.　SPIBSC 初期設定バイナリファイル、及びアプリケーションバイナリファイルについては表 5.1 を参照してください。

表 1.1　使用する周辺機能と用途

周辺機能	用途
SPI マルチ I/O バスコントローラ（SPIBSC）	• 外部アドレス空間（SPI マルチ I/O バス空間）に接続されたシリアルフラッシュメモリにアクセスするための信号を生成するために使用します。
ARM® Development Studio 5（DS-5 ™） セミホスティング機能	• ターゲット上で実行中のコード（オンボード実行プログラム）により、デバッガを実行しているホストコンピュータの入出力機能と通信するためにセミホスティング機能を使用します。 • ターゲットからDS-5のアプリケーションコンソールへのターミナル出力、及びホストコンピュータのハードディスクに格納されたアプリケーションバイナリファイルを参照するために使用します。

2. 動作確認条件

本アプリケーションノートのサンプルプログラムは、以下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	RZ/T1グループ
動作周波数	CPUCLK = 450MHz, PCLKA = 150MHz
動作電圧	3.3V
統合開発環境	ARM®製 DS-5 Version 5.25.0
動作モード	SPIブートモード（シリアル・フラッシュ）
使用ボード	RZ/T1 Evaluation Board（RTK7910022C00000BR）
使用デバイス （ボード上で使用する機能）	SPIマルチI/Oバス空間（1または4ビットバス幅）に配置されたシリアルフラッシュメモリ • メーカ名：Macronix International Co.. • 型名：MX25L51245GMI-10G

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RZ/T1 グループ初期設定例 アプリケーションノート（R01AN2554JJ）
- RZ/T1 グループ R-IN Engine 搭載製品　初期設定（R01AN2989JJ）

4. ハードウェア説明

4.1 使用端子一覧

表 4.1 に使用端子と機能を示します。

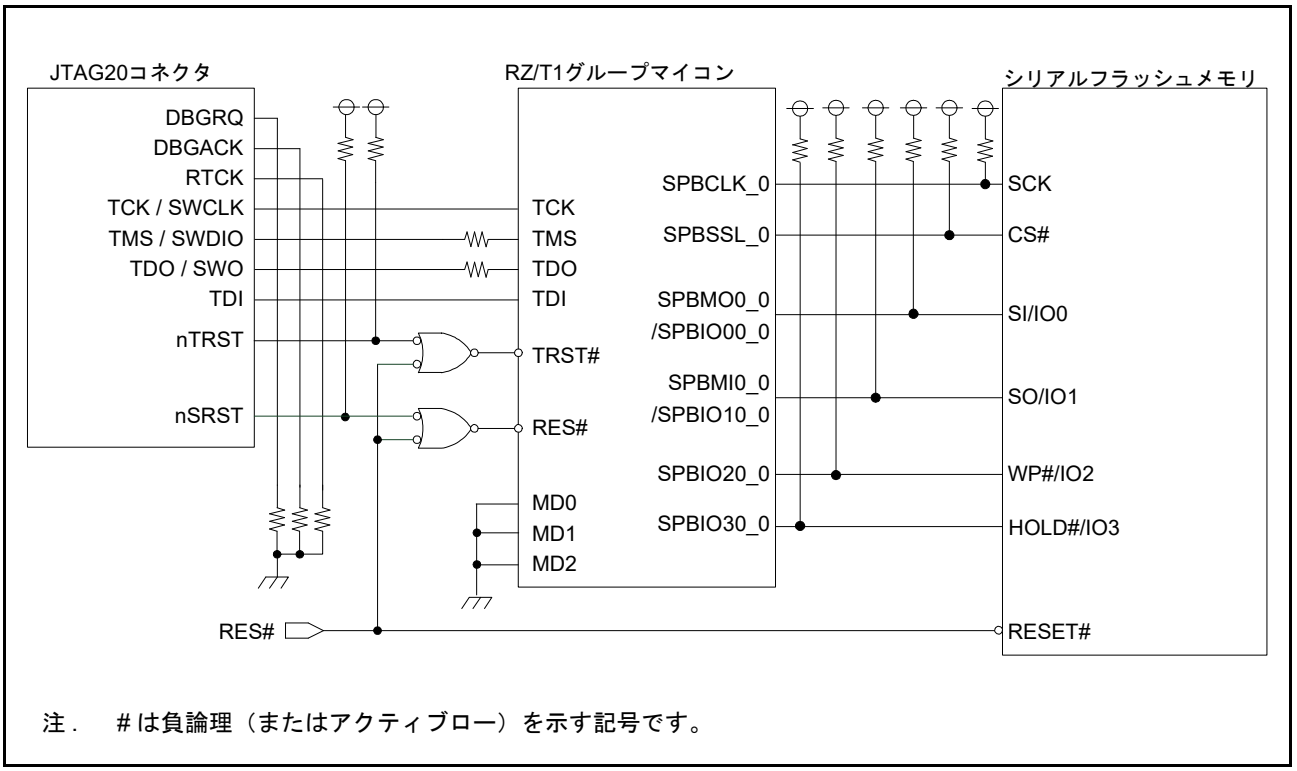
表4.1 使用端子と機能

端子名	入出力	内容
SPBCLK_0	出力	クロック出力
SPBSSL_0	出力	スレーブセレクト
SPBMO0_0/SPBIO00_0	入出力	マスタ送出データ／データ 0
SPBMI0_0/SPBIO10_0	入出力	マスタ入力データ／データ 1
SPBIO20_0	入出力	データ 2
SPBIO30_0	入出力	データ 3
MD2、MD1、MD0	入力	ブートモードの選択（SPI ブートモードに設定） MD2：”L” MD1：”L” MD0：”L”
TCK	入力	ARM®製エミュレータからのクロック入力
TMS	入力	ARM®製エミュレータからのモード選択
TRST#	入力	ARM®製エミュレータからのリセット入力
TDI	入力	ARM®製エミュレータからのデータ入力
TDO	出力	ARM®製エミュレータへのデータ出力
RES#	入力	システムリセット信号

注． #は負論理（またはアクティブロー）を示す記号です。

4.2 参考回路

図 4.1 に接続例を示します。



注． ＃は負論理（またはアクティブロー）を示す記号です。

図 4.1 接続例

5. シリアルフラッシュメモリへのダウンロードの概要

この章では、シリアルフラッシュメモリへのダウンロードの概要について説明します。

5.1 シリアルフラッシュメモリへダウンロードに関連する用語

表 5.1 に本アプリケーションノートで使用するシリアルフラッシュメモリへのダウンロード関連の用語を示します。

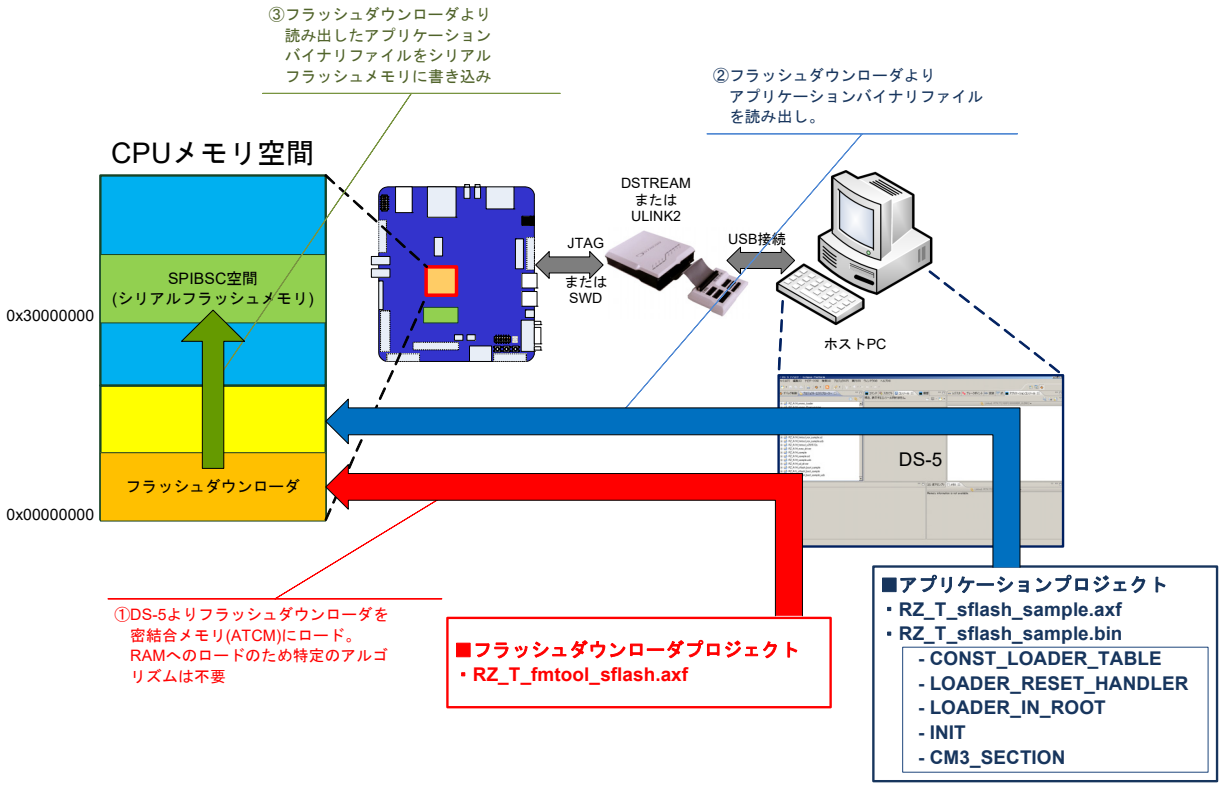
表5.1 シリアルフラッシュメモリへのダウンロード関連の用語

用語	説明
アプリケーションプログラム	アプリケーションプログラムは、お客様がシステムに応じて作成するプログラムです。
フラッシュダウンローダ	フラッシュダウンローダは、SPIマルチI/Oバスコントローラ初期設定プログラムおよびアプリケーションプログラムをシリアルフラッシュメモリに書き込むためのプログラムです。本アプリケーションノートを参考に、お客様がシステムに応じて作成してください。
セミホスティング	セミホスティングは、ARM® CPU上で実行される入出力要求のコードが、デバッガとの通信を通してDS-5の入出力機能を使用するメカニズムです。 セミホスティング機能を使用して、ARM® CPU上でprintf関数やscanf関数等のC言語の標準関数を実行すると、ARM® CPU上が存在するターゲットシステム側の入出力機能に対してではなく、DS-5の入出力機能を通して、ホストPCの画面やキーボード等に対して入出力処理を行うことができます。 詳細はARM®より提供されるドキュメントを参照してください。
アプリケーションプロジェクト	アプリケーションプロジェクトは、DS-5にてアプリケーションプログラム実行形式ファイル（axfファイル）を生成するためのプロジェクトです。 アプリケーションプログラムには、RZ/T1グループマイコンが参照するローダ用パラメータ情報、およびローダプログラムが含まれます。
フラッシュダウンローダプロジェクト	フラッシュダウンローダプロジェクトは、DS-5でフラッシュダウンローダ実行形式ファイル（axfファイル）を生成するためのプロジェクトです。 アプリケーションプログラムには、RZ/T1グループマイコンが参照するローダ用パラメータ情報、およびローダプログラムが含まれます。
アプリケーションバイナリファイル	アプリケーションバイナリファイルは、シリアルフラッシュメモリに書き込むアプリケーションプログラムのデータファイルです。DS-5で、アプリケーションプロジェクトのビルドにより生成したアプリケーションプログラム実行形式ファイル（axfファイル）から、バイナリファイル生成ツール（fromelf.exe）注1を使用して生成します。

注1. バイナリファイル生成ツールはDS-5に付属しています。詳細はARM®より提供される「ARM® DS-5™ DS-5スタートガイド／ARM DS-5の製品概要」を参照してください。

5.2 フラッシュダウンローダの動作イメージ

図 5.1 にフラッシュダウンローダの動作イメージを示します。フラッシュダウンローダは、RZ/T1 グループマイコンの密結合メモリ（ATCM）上で実行され、セミホスティング機能を使用し、DS-5 を実行するホストコンピュータのハードディスクに格納されているアプリケーションバイナリファイルを参照して、シリアルフラッシュメモリに書き込みを行います。



注 . CM3_SECTION は、Cortex-M3 プログラム用のバイナリファイルです。詳細は、アプリケーションノート「RZ/T1 グループ　R-IN Engine 搭載製品　初期設定」を参照ください。

図 5.1 フラッシュダウンローダの動作イメージ

5.3 フラッシュダウンローダの開発方法

図 5.2 にフラッシュダウンローダの開発フローを示します。フラッシュダウンローダは DS-5 プロジェクトとして開発します。このプロジェクトをフラッシュダウンローダプロジェクトと呼びます。フラッシュダウンローダには、セミホスティング機能によるアプリケーションバイナリファイルの読み出し処理、CPU 初期化処理および書き込み対象となるシリアルフラッシュメモリに応じた書き込み処理を実装します。本アプリケーションノートのサンプルプログラムでは、RZ/T1 評価ボードに実装されているシリアルフラッシュメモリの書き込み処理をシリアルフラッシュメモリインタフェース関数として実装しています。シリアルフラッシュメモリインタフェース関数については、「7. フラッシュメモリインタフェース関数」を参照してください。

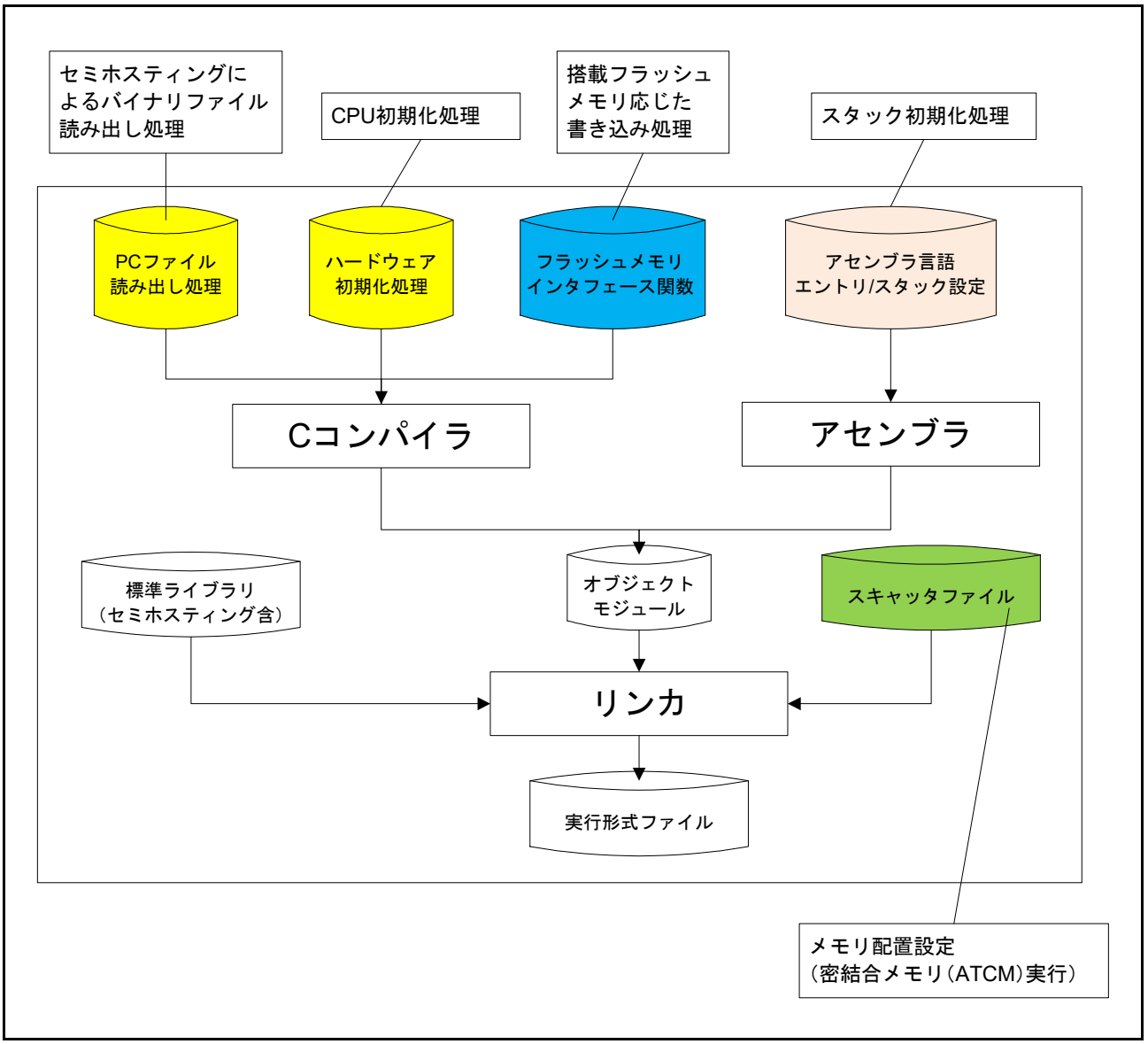


図 5.2 フラッシュダウンローダの開発フロー

5.3.1 メモリマップ

フラッシュダウンローダは RZ/T1 グループマイコンの密結合メモリ（ATCM）上で実行するため、スキャッタファイル注 1 で密結合メモリ（ATCM）に配置します。図 5.3 にフラッシュダウンローダのメモリ配置を示します。

注 1. スキャッタファイルは、メモリレイアウトおよびコードとデータの配置を記述したテキストです。詳細は ARM® より提供される「ARM® コンパイラツールチェーン リンカの使用／イメージの構造と生成」を参照してください。

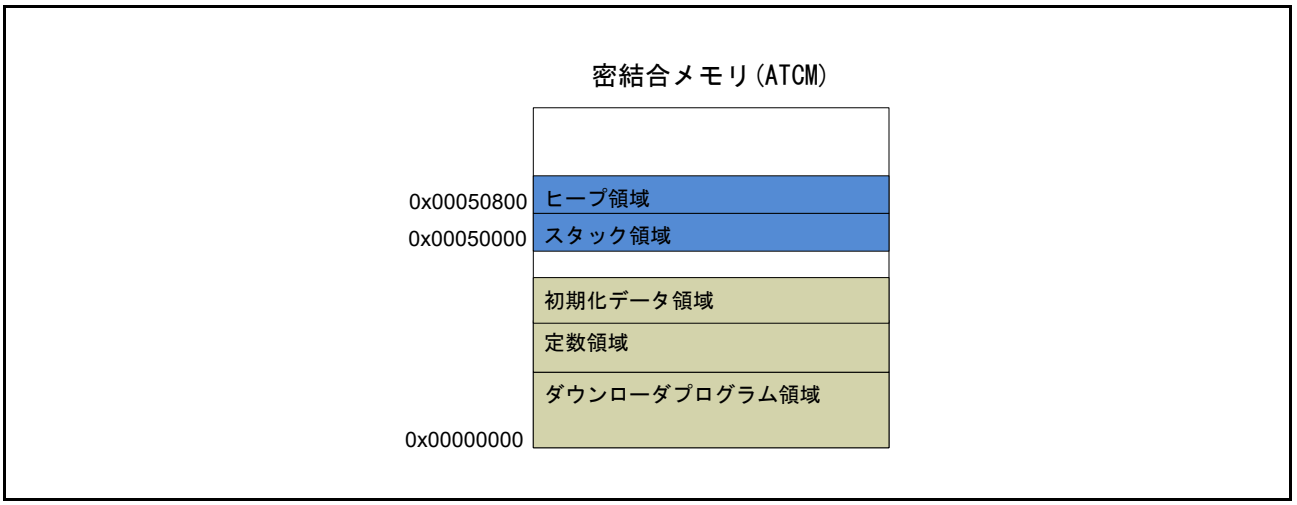


図 5.3 フラッシュダウンローダのメモリ配置

1. フラッシュダウンローダは、RZ/T1 グループマイコンの密結合メモリ（ATCM）領域に配置します。フラッシュダウンローダのエントリポイントは 0x00000000 番地に設定しています。
2. フラッシュダウンローダが使用するスタック領域やヒープ領域等は密結合メモリ（ATCM）領域に配置します。
3. フラッシュダウンローダの例外処理ベクタテーブルはセミホスティング機能により実現されるため、実装は不要です。

5.4 シリアルフラッシュメモリへのダウンロード例のカスタマイズについて

本章では、本アプリケーションノートで紹介するシリアルフラッシュメモリへのダウンロード例のカスタマイズ方法について、説明します。

ダウンロード例は、**表 5.2** に示す項目について、カスタマイズすることができます。お客様のシステムに応じてカスタマイズしてください。

表5.2 カスタマイズできる項目

項目	説明
ダウンロードするアプリケーションプロジェクトに合わせたカスタマイズ	シリアルフラッシュメモリにダウンロードするアプリケーションプロジェクトに合わせて、アプリケーションバイナリファイル名や、書き込み開始アドレスをカスタマイズすることができます。 カスタマイズ方法の詳細は「10.1 バイナリファイル名および書き込みアドレスを変更する方法」を参照してください。
フラッシュメモリインタフェース関数のカスタマイズ	書き込み対象となるシリアルフラッシュメモリに応じて、フラッシュメモリインタフェース関数をカスタマイズすることができます。 カスタマイズ方法の詳細は「10.2 フラッシュメモリに応じたサンプルプログラムの変更方法」を参照してください。

6. RZ/T1 評価ボード（RTK7910022C00000BR）へのダウンロード実行例

この章では、DS-5 および ARM® 製エミュレータを使用し、本アプリケーションノートで紹介するダウンロード方法で、RZ/T1 評価ボード（RTK7910022C00000BR）に搭載されたシリアルフラッシュメモリにアプリケーションプログラム（RZ_T_sflash_sample）をダウンロードする手順を紹介します。

図 6.1 にダウンロードの概略手順を示します。

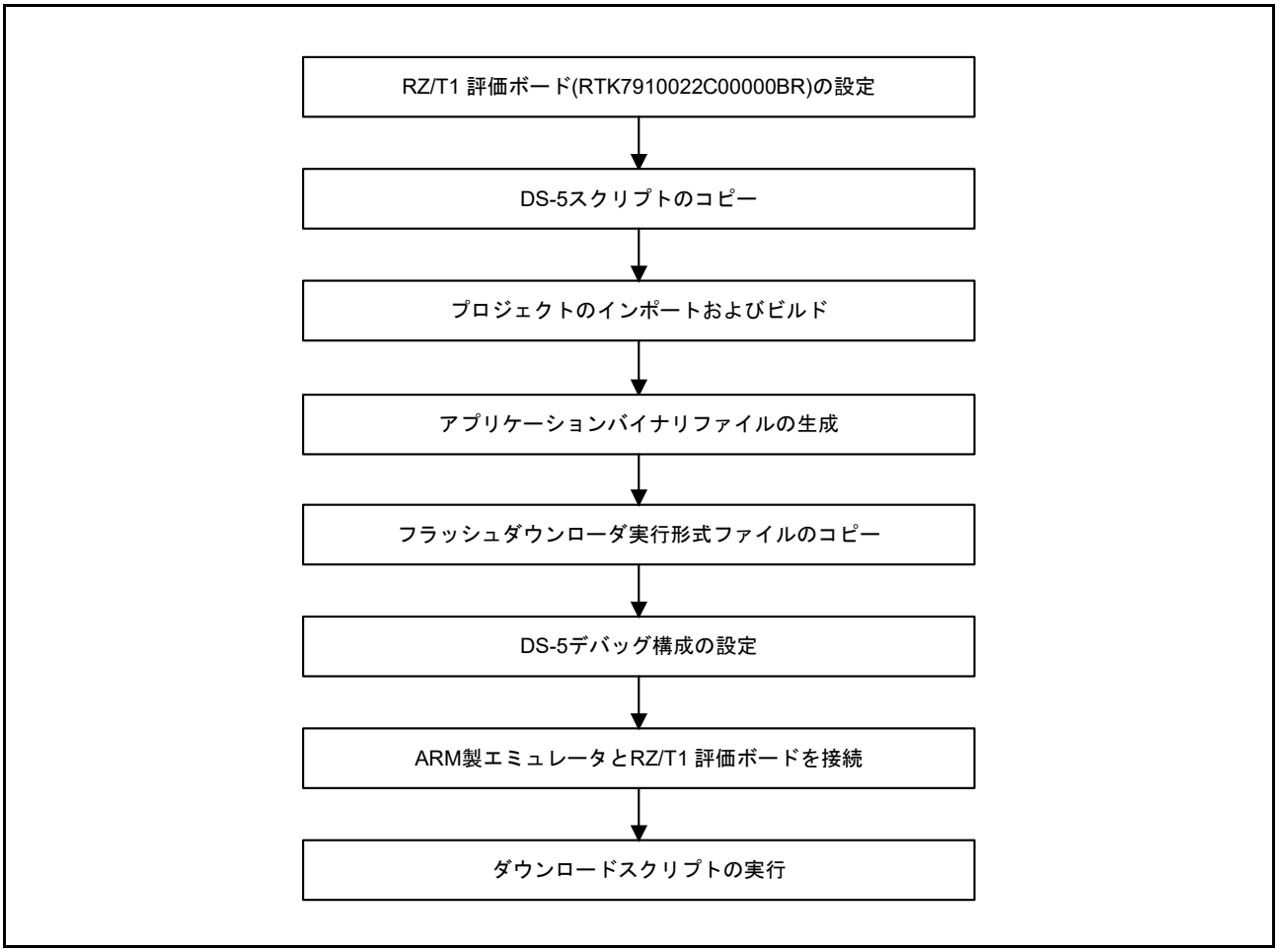


図 6.1　ダウンロードの概略手順

6.1 RZ/T1 評価ボード（RTK7910022C00000BR）の設定

表 6.1 に本アプリケーションノートのサンプルプログラムを動作させるための RZ/T1 評価ボード（RTK7910022C00000BR）の設定を示します。

表 6.1 に従って、RZ/T1 評価ボード（RTK7910022C00000BR）を設定します。

表6.1 RZ/T1評価ボード（RTK7910022C00000BR）の設定

SW	設定	内容
SW4-1	ON	MD0 = L レベル
SW4-2	ON	MD1 = L レベル
SW4-3	ON	MD2 = L レベル
SW4-4	ON	BSCANP = L レベル
SW4-5	ON	OSCTH = L レベル
SW4-6	OFF	PU7 = H レベル

6.2 DS-5 スクリプトのコピー

アプリケーションプロジェクト（RZ_T_sflash_sample）ディレクトリの直下にディレクトリ [script_sflash] を作成し、表 6.2 の DS-5 スクリプトをコピーします。

注 . DS-5 ワークスペースディレクトリについては、ARM® より提供される「ARM® DS-5™ デバッガの使用」を参照してください。

表6.2 DS-5スクリプトファイル一覧

スクリプト名	説明
init_RZ-T.ds	RZ/T1評価ボード初期化スクリプトです。 DS-5とRZ/T1評価ボードを接続した際に、RZ/T1グループマイコンの密結合メモリ（ATCM）の書き込みを許可する等の処理を実行するDS-5スクリプトです。
RZ_T_sflash_sample.ds	アプリケーションダウンロードスクリプトです。 アプリケーションプログラムをRZ/T1グループマイコンの外部アドレス空間（SPIマルチI/Oバス空間）に配置されたシリアルフラッシュメモリに書き込むための一連の作業を記載したDS-5スクリプトです。
init_RZ-T2.ds	アプリケーションダウンロードスクリプトから実行されるRZ/T1評価ボード初期化スクリプトです。DS-5メモリ領域設定を行わない以外は、init_RZ-T.dsと同じです。

6.3 プロジェクトのインポートおよびビルド

表 6.3 に示すプロジェクトを DS-5 ワークスペースディレクトリ以下にインポートします。インポート後、ビルドを実行し、実行形式ファイルを生成します。

【手順】

1. DS-5（スタートメニューから [すべてのプログラム] ー [ARM DS-5 v5.21.1]—[Eclipse for DS-5]）を起動します。
2. [ファイル (F)] ー [インポート (I)] を選択し、[インポートー選択] ウインドウを開きます。
3. [一般] ー [既存プロジェクトをワークスペースへ] を選択し、[次へ] をクリックします。
4. [インポートープロジェクトのインポート] ウインドウで、[参照] をクリックしてプロジェクトを表示させた後、インポートするプロジェクトを選択します。オプションでは [プロジェクトをワークスペースにコピー (C)] にチェックを入れて、[終了] をクリックします。
5. プロジェクト・エクスプローラーでインポートしたプロジェクトを順に選択した後、[プロジェクト (P) ープロジェクトのビルド (B)] を選択してプロジェクトをビルドします。

表6.3　プロジェクト一覧

プロジェクト名	説明	実行形式ファイル
RZ_T_fmtool_sflash	このプロジェクトでフラッシュダウンローダをビルドします。このプロジェクトをフラッシュダウンローダプロジェクトと呼びます。	RZ_T_fmtool_sflash.axf
RZ_T_sflash_sample	このプロジェクトでアプリケーションプログラムをビルドします。このプロジェクトをアプリケーションプロジェクトと呼びます。	RZ_T_sflash_sample.axf

6.4 アプリケーションバイナリファイルの生成

DS-5 の [DS-5 Command Prompt] より、**図 6.3** に示すコマンド注 1 を実行し、アプリケーションバイナリファイル（RZ_T_sflash_sample.bin）を生成します。**表 6.4** にコマンド実行より生成されるバイナリファイル一覧を示します。

なお、本アプリケーションノートに添付されているプロジェクトでは、この処理の実行をバッチファイル（¥RZ_T_sflash_sample¥Debug¥after_build.bat）でビルド時に実行します。

【手順】

1. DS-5 Command Pronpt [スタートメニューからすべてのプログラムー ARM DS-5 v5.21.1—DS-5 Command Pronpt] を起動します。
2. [select_toolchain] と入力して enter を押す。使用するツールチェーンの選択が表示されるので、選択して enter を押す。（**図 6.2** を参照）
3. **「6.3 プロジェクトのインポートおよびビルド」**で作成した [fmtool] フォルダにパスを設定し、**図 6.3** に示すコマンド注 1 を実行する。

注 1. コマンドの詳細は ARM® より提供される「ARM® DS-5™ DS-5 スタートガイド／ ARM DS-5 の製品概要」を参照してください。

```
You can change the compiler toolchain for this environment at any time by
running the 'select_toolchain' command. A default for all future environments
can be set  with the 'select_default_toolchain' command.


C:¥Program Files¥DS-5 v5.21.1¥bin>select_toolchain
Select a toolchain to use in the current environment

 1 - ARM Compiler 5 (DS-5 built-in)
 2 - GCC 4.x [arm-linux-gnueabihf] (DS-5 built-in)
Enter a number or <return> for no toolchain: 1

Environment configured for ARM Compiler 5 (DS-5 built-in)

C:¥Program Files¥DS-5 v5.21.1¥bin>
```

図 6.2 ツールチェーンの設定

```
fromelf --bin --output=RZ_T_sflash_sample.bin RZ_T_sflash_sample.axf
```

図 6.3 アプリケーションバイナリファイル生成コマンド

表 6.4　アプリケーションバイナリファイル一覧

ディレクトリ	バイナリファイル	説明
RZ_T_sflash_sample.bin	CONST_LOADER_TABLE	アプリケーション(1)（ローダ用パラメータ情報） バイナリファイル
	LOADER_RESET_HANDLER	アプリケーション(2)（ローダプログラム） バイナリファイル
	LOADER_IN_ROOT	アプリケーション(3)（ローダプログラム） バイナリファイル
	INIT	アプリケーション(4)（ユーザプログラム） バイナリファイル
	CM3_SECTION 注 1	アプリケーション(5)（ユーザプログラム） バイナリファイル（Cortex-M3 プログラム）

注 1.　CM3_SECTION は、Cortex-M3 プログラム用のバイナリファイルです。詳細は、アプリケーションノート「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を参照ください。

6.5　フラッシュダウンローダ実行形式ファイルのコピー

「6.3　プロジェクトのインポートおよびビルド」でインポートしたアプリケーションプロジェクト（RZ_T_sflash_sample）ディレクトリの直下に、ディレクトリ [fmtool] を作成し、フラッシュダウンローダプロジェクト実行形式ファイル（RZ_T_fmtool_sflash.axf）をコピーします。

なお、本アプリケーションノートに添付されているプロジェクトでは、この処理の実行をバッチファイル（l¥RZ_T_fmtool_sflash¥Debug¥after_build.bat）でビルド時に実行します。

6.6　DS-5 デバッグ構成の設定

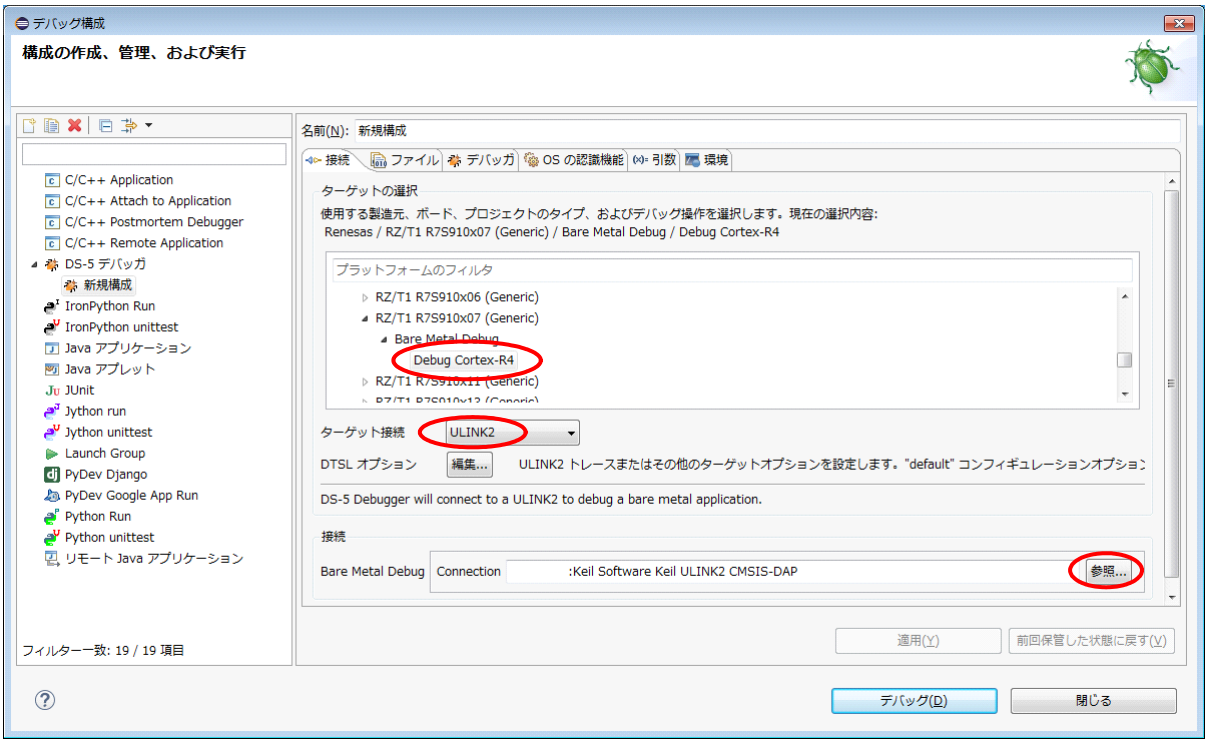
以下の手順で、DS-5 デバッグ構成を設定します。DS-5 デバッグ構成の設定で、DS-5 と RZ/T1 評価ボードを接続 注 1 した際に RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）が実行されるように設定します。なお、RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）の処理内容については、**「9.2　RZ/T1 評価ボード初期化スクリプト」**を参照してください。

【手順】

1. DS-5 の [実行 (R)] ー [デバッグ構成 (B)] により、[デバッグ構成] 画面を表示します。
2. DS-5 の [デバッグ構成] 画面の [接続] タブで、ターゲットを選択します。ターゲットは、[Renesas] /[RZ/T1 R7S910x17(Generic)] / [Bare Metal Debug] / [Debug of Cortex-R4] を選択 注 2 します。
3. DS-5 の [デバッグ構成] 画面の [接続] タブで、ターゲット接続と接続ブラウザを選択します。[ターゲット接続] で接続するデバッガを選択し、[Bare Metal Debug] で [参照] ボタンを押し、[接続ブラウザ] で接続されたデバッガを選択します。
4. DS-5 の [デバッグ構成] 画面の [デバッガ] タブで、実行制御の [接続のみ] にチェックを入れます。
5. DS-5 の [デバッグ構成] 画面の [デバッガ] タブで、実行制御の [ターゲット初期化デバッガスクリプト (.ds / .py) を実行します] にチェックを入れ、RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）への PATH を設定します。

注 1.　本手順は、DS-5 ／プラットフォームに RZ/T1 評価ボードが登録されていることを前提としています。DS-5 ／プラットフォームに RZ/T1 評価ボードが登録されていない場合は、DS-5 Debug Hardware Configuration で、プラットフォームを登録してください。

注 2.　使用する DS-5 のバージョンにより、選択するターゲットの名称が異なる場合があります。



注 .　この画面は、ターゲット接続するデバッガが ULINK2™ の場合です。

図 6.4　DS-5 でのデバッガ選択

6.7　ARM® 製エミュレータにて RZ/T1 評価ボードと接続

以下の手順で、ARM® 製エミュレータにて RZ/T1 評価ボードと接続します。

【手順】

1. DS-5 の [デバッグ制御] タブにより、「6.6　DS-5 デバッグ構成の設定」の手順 (2) で設定した名称の接続を選択し、右クリックにより [Connect to Target] を選択します。
2. 1. で接続を開始し、接続後に「6.6　DS-5 デバッグ構成の設定」の手順 (4) で登録した RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）が実行されます。

6.8 ダウンロードスクリプトの実行

以下の手順により、ダウンロードスクリプト（RZ_T_sflash_sample.ds）を実行します。

【手順】

1. DS-5 の [スクリプトタブ] にダウンロードスクリプト（RZ_T_sflash_sample.ds）を登録します。
2. 1. で登録したダウンロードスクリプト（RZ_T_sflash_sample.ds）をダブルクリックし、ダウンロードスクリプトを実行します。
3. ダウンロードスクリプトを実行すると、フラッシュダウンローダが起動し、フラッシュメモリへの書き込みを開始します。**図 6.5** に [アプリケーションコンソール] に表示されるメッセージを示します。
4. ダウンロードが完了すると、フラッシュダウンローダのシンボル情報を破棄し、ダウンロードスクリプト（RZ_T_sflash_sample.ds）から実行される RZ/T1 評価ボード初期化スクリプト（init_RZ-T2.ds）を実行し、アプリケーションプログラム（RZ_T_sflash_sample）のシンボル情報をロードし、ダウンロードが完了します。

```
RZ/T1 CPU Board S-Flash Programming Sample. Ver.1.00
Copyright (C) 2015 Renesas Electronics Corporation. All rights reserved.

Initializing Flash...
Start to load Binary Data to Flash Memory.
loop=1, file=CONST_LOADER_TABLE, flash address=0x30000000.
Calculating Data Size...
Data Size is 76
Programing Flash...
Calcurating Checksum of Loader Parameter.
Verifying Flash...
loop=1, Flash Programming Success!!
loop=2, file=LOADER_RESET_HANDLER, flash address=0x30000200.
Calculating Data Size...
Data Size is 11812
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x30006200.
Calculating Data Size...
Data Size is 236
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x30010000.
Calculating Data Size...
Data Size is 2592
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, Could not open file. Exiting.
Flash Programming Complete
```

注 . loop = 5 の処理は、Cortex-M3 プログラムの書き込み専用処理です。Cortex-R4F プログラムの書き込み時は loop = 4 の処理が完了した時点で、必要なプログラムが書き込めています。

図 6.5 アプリケーションコンソールに出力されるメッセージ

7. フラッシュメモリインタフェース関数

この章では、フラッシュメモリインタフェース関数について説明します。

7.1 固定幅整数一覧

表 7.1 にサンプルプログラムで使用する固定幅整数を示します。

表7.1 サンプルプログラムで使用する固定幅整数

シンボル	内容
char8_t	8ビット整数、符号あり
int16_t	整数、符号あり
int32_t	32ビット整数、符号あり
uint8_t	8ビット整数、符号なし
uint16_t	16ビット整数、符号なし
uint32_t	32ビット整数、符号なし

7.2 構造体／共用体一覧

表 7.2 ～表 7.9 にサンプルプログラムで使用する構造体を示します。

表7.2 SPIBSC 外部アドレスリード設定構造体（st_spibsc_cfg_t）(1)

メンバ名	内容
uint8_t udef_cmd	リードコマンド • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するリードコマンドを設定します。 • 本メンバに設定した値をデータリードコマンド設定レジスタ（DRCMR）のCMD[7:0]に設定します。
uint8_t udef_cmd_width	リードコマンドビット幅 • リードコマンド発行時のビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のCDB[1:0]に設定します。
uint8_t udef_opd3 uint8_t udef_opd2 uint8_t udef_opd1 uint8_t udef_opd0	オプショナルデータ • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するオプショナルデータを設定します。 • 本メンバに設定した値をデータリードオプション設定レジスタ（DROPR）のOPD3[7:0]、OPD2[7:0]、OPD1[7:0]、OPD0[7:0]に設定します。
uint8_t udef_opd_enable	オプションナルデータイネーブル • オプショナルデータを発行するかどうかを選択します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_OPD_3：OPD3を出力 SPIBSC_OUTPUT_OPD_32：OPD3,OPD2を出力 SPIBSC_OUTPUT_OPD_321：OPD3,OPD2,OPD1を出力 SPIBSC_OUTPUT_OPD_3210：OPD3,OPD2,OPD1,OPD0を出力 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のOPDE[3:0]に設定します。
uint8_t udef_opd_width	オプショナルデータビット幅 • オプショナルデータ発行時のビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のOPDB[1:0]に設定します。

表7.3　SPIBSC 外部アドレスリード設定構造体（st_spibsc_cfg_t）(2)

メンバ名	内容
uint8_t　udef_dmycyc_num	ダミーサイクル数 • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するダミーサイクル数を設定します。 • 設定可能な値： SPIBSC_DUMMY_1CYC：1サイクル SPIBSC_DUMMY_2CYC：2サイクル SPIBSC_DUMMY_3CYC：3サイクル SPIBSC_DUMMY_4CYC：4サイクル SPIBSC_DUMMY_5CYC：5サイクル SPIBSC_DUMMY_6CYC：6サイクル SPIBSC_DUMMY_7CYC：7サイクル SPIBSC_DUMMY_8CYC：8サイクル • 本メンバに設定した値をデータリードダミーサイクル設定レジスタ（DRDMCR）のDMCYC[2:0]に設定します。
uint8_t　udef_dmycyc_enable	ダミーサイクルイネーブル • ダミーサイクルを挿入するかを選択します。 • 設定可能な値： SPIBSC_DUMMY_CYC_DISABLE：挿入しない SPIBSC_DUMMY_CYC_ENABLE：挿入する • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のDMEに設定します。
uint8_t　udef_dmycyc_width	ダミーサイクルビット幅 • ダミーサイクル発行時のビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をデータリードダミーサイクル設定レジスタ（DRDMCR）のDMDB[1:0]に設定します。
uint8_t　udef_data_width	データリードビット幅 • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時のシリアルフラッシュメモリのデータリードビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のDRDB[1:0]に設定します。

表7.4　SPIBSC 外部アドレスリード設定構造体（st_spibsc_cfg_t）(3)

メンバ名	内容
uint8_t　udef_spbr	ビットレート • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するシリアルクロック（SPBCLK）のビットレートを設定します。 • 設定可能な値： ビットレート分周設定（udef_brdv）と合わせて設定を行ってください。 • 本メンバに設定した値をビットレート設定レジスタ（SPBCR）のSPBR[7:0]に設定します。
uint8_t　udef_brdv	ビットレート分周設定 • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するシリアルクロック（SPBCLK）のビットレートを設定します。 • 設定可能な値： ビットレート（udef_spbr）と合わせて設定を行ってください。 • 本メンバに設定した値をビットレート設定レジスタ（SPBCR）のBRDV[1:0]に設定します。
uint8_t　udef_addr_width	アドレスビット幅 • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するアドレスのビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のADB[1:0]に設定します。
uint8_t　udef_addr_mode	アドレスイネーブル • SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するアドレスを設定します。 • 設定可能な値： SPIBSC_OUTPUT_ADDR_24：24ビットのアドレスを出力 SPIBSC_OUTPUT_ADDR_32：32ビットのアドレスを出力 • 本メンバに設定した値をデータリードイネーブル設定レジスタ（DRENR）のADE[3:0]に設定します。

表7.5　SPIBSC SPIモード設定構造体（st_spibsc_spimd_reg_t）(1)

メンバ名	内容
uint32_t　cdb	コマンドビット幅 • SPI 動作モード時のコマンドビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のCDB[1:0]に設定します。
uint32_t　ocdb	オプショナルコマンドビット幅 • SPI 動作モード時のオプショナルコマンドビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のOCDB[1:0]に設定します。
uint32_t　adb	アドレスビット幅 • SPI 動作モード時のアドレスビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のADB[1:0]に設定します。
uint32_t　opdb	オプショナルデータビット幅 • SPI 動作モード時のオプショナルデータビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のOPDB[1:0]に設定します。
uint32_t　spidb	転送データビット幅 • SPI 動作モード時の転送データビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のSPIDB[1:0]に設定します。
uint32_t　cde	SPI 動作モード時にコマンドを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ENABLE：出力する • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のCDEに設定します。

表7.6　SPIBSC SPIモード設定構造体（st_spibsc_spimd_reg_t）(2)

メンバ名	内容
uint32_t　ocde	オプショナルコマンドイネーブル • SPI 動作モード時にオプショナルコマンドを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ENABLE：出力する • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のOCDEに設定します。
uint32_t　ade	アドレスイネーブル • SPI 動作モード時にアドレスを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ADDR_24：ADR[23:0]を出力 SPIBSC_OUTPUT_ADDR_32：ADR[31:0]を出力 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のADE[3:0]に設定します。
uint32_t　opde	オプショナルデータイネーブル • SPI 動作モード時にオプショナルデータを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_OPD_3：OPD3を出力 SPIBSC_OUTPUT_OPD_32：OPD3,OPD2を出力 SPIBSC_OUTPUT_OPD_321：OPD3,OPD2,OPD1を出力 SPIBSC_OUTPUT_OPD_3210：OPD3,OPD2,OPD1,OPD0を出力 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のOPDE[3:0]に設定します。
uint32_t　spide	転送データイネーブル • SPI 動作モード時にデータ転送を行うかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_SPID_8：8（または16）ビット転送 SPIBSC_OUTPUT_SPID_16：16（または32）ビット転送 SPIBSC_OUTPUT_SPID_32：32（または64）ビット転送 • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のSPIDE[3:0]に設定します。
uint32_t　sslkp	SPBSSL信号レベル保持 • SPI 動作モード時に転送終了後のSPBSSL 信号状態を設定します。 • 設定可能な値： SPIBSC_SPISSL_NEGATE：転送終了時にネゲート SPIBSC_SPISSL_KEEP：転送終了後から次アクセス開始までSPBSSL信号レベルを保持 • 本メンバに設定した値をSPI モードコントロールレジスタ（SMCR）のSSLKPに設定します。

表7.7　SPIBSC SPIモード設定構造体（st_spibsc_spimd_reg_t）(3)

メンバ名	内容
uint32_t　spire	データリードイネーブル • SPI 動作モード時にデータリードするかを設定します。 • 設定可能な値： SPIBSC_SPIDATA_DISABLE：データリードしない SPIBSC_SPIDATA_ENABLE：データリードする • 本メンバに設定した値をSPI モードコントロールレジスタ（SMCR）のSPIREに設定します。
uint32_t　spiwe	データライトイネーブル • SPI 動作モード時にデータライトするかを設定します。 • 設定可能な値： SPIBSC_SPIDATA_DISABLE：データライトしない SPIBSC_SPIDATA_ENABLE：データライトする • 本メンバに設定した値をSPI モードコントロールレジスタ（SMCR）のSPIWEに設定します。
uint32_t　dme	ダミーサイクルイネーブル • SPI 動作モード時にダミーサイクル挿入するかどうかを設定します。 • 設定可能な値： SPIBSC_DUMMY_CYC_DISABLE：挿入しない SPIBSC_DUMMY_CYC_ENABLE：挿入する • 本メンバに設定した値をSPI モードイネーブル設定レジスタ（SMENR）のDMEに設定します。
uint32_t　adder	アドレスDDRイネーブル • SPI 動作モード時に出力するアドレスのSDR／DDR転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR転送 • 本メンバに設定した値をSPI モードDDR イネーブルレジスタ（SMDRENR）のADDREに設定します。
uint32_t　opdre	オプションデータDDRイネーブル • SPI 動作モード時に出力するオプションデータのSDR／DDR転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR転送 • 本メンバに設定した値をSPI モードDDR イネーブルレジスタ（SMDRENR）のOPDREに設定します。
uint32_t　spidre	転送データDDRイネーブル • SPI 動作モード時に転送するデータのSDR／DDR転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR転送 • 本メンバに設定した値をSPI モードDDR イネーブルレジスタ（SMDRENR）のSPIDREに設定します。

表7.8　SPIBSC SPIモード設定構造体（st_spibsc_spimd_reg_t）(4)

メンバ名	内容
uint8_t　dmdb	ダミーサイクルビット幅 • SPI 動作モード時のダミーサイクルのビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1ビット幅 SPIBSC_4BIT：4ビット幅 • 本メンバに設定した値をSPI モードダミーサイクル設定レジスタ（SMDMCR）のDMDB[1:0]に設定します。
uint8_t　dmcyc	ダミーサイクル数 • SPI 動作モード時のダミーサイクル数を設定します。 • 設定可能な値： SPIBSC_DUMMY_1CYC：1サイクル SPIBSC_DUMMY_2CYC：2サイクル SPIBSC_DUMMY_3CYC：3サイクル SPIBSC_DUMMY_4CYC：4サイクル SPIBSC_DUMMY_5CYC：5サイクル SPIBSC_DUMMY_6CYC：6サイクル SPIBSC_DUMMY_7CYC：7サイクル SPIBSC_DUMMY_8CYC：8サイクル • 本メンバに設定した値をSPI モードダミーサイクル設定レジスタ（SMDMCR）のDMCYC[2:0]に設定します。
uint8_t　cmd	コマンド • SPI 動作モード時に出力するコマンドを設定します。 • 本メンバに設定した値をSPI モードコマンド設定レジスタ（SMCMR）のCMD[7:0]に設定します。
uint8_t　ocmd	オプショナルコマンド • SPI 動作モード時に出力するオプショナルコマンドを設定します。 • 本メンバに設定した値をSPI モードコマンド設定レジスタ（SMCMR）のOCMD[7:0]に設定します。
uint32_t　addr	アドレス • SPI 動作モード時に出力するアドレスを設定します。 • 本メンバに設定した値をSPI モードアドレス設定レジスタ（SMADR）のADR[31:0]に設定します。
uint8_t　opd[4]	オプションデータ • SPI 動作モード時に出力するオプションデータを設定します。 • 本メンバに設定した値をSPI モードオプション設定レジスタ（SMOPR）のOPDn[7:0]に以下のように設定します。 OPD3[7:0]←opd[0] OPD2[7:0]←opd[1] OPD1[7:0]←opd[2] OPD0[7:0]←opd[3]

表 7.9　SPIBSC SPI モード設定構造体（st_spibsc_spimd_reg_t）(5)

メンバ名	内容
uint32_t　smrdr[2]	リードデータ格納バッファ • SPI 動作モード時にリードしたデータ（SPI モードリードデータレジスタ n（SMRDRn））を以下のように格納します。 SMRDR0→smrdr[0] SMRDR1→smrdr[1]
uint32_t　smwdr[2]	ライトデータ格納バッファ • SPI 動作モード時にライトするデータ（SPI モードライトデータレジスタ n（SMWDRn））を以下のように格納します。 SMWDR0←swrdr[0] SMWDR1←swrdr[1]

7.3　定数一覧

表 7.10 ～表 7.13 にサンプルプログラムで使用する定数を示します。

表 7.10　サンプルプログラムで使用する定数 (1)

定数名	設定値	内容
SF_REQ_PROTECT	(0)	シリアルフラッシュメモリ・プロテクト設定
SF_REQ_UNPROTECT	(1)	シリアルフラッシュメモリ・プロテクト解除
SFLASHCMD_SECTOR_ERASE	(0xD8)	Erase 256 kB（3-byte Address）コマンド
SFLASHCMD_BYTE_PROGRAM	(0x02)	Page Program（3-byte Address）コマンド
SFLASHCMD_FAST_READ	(0x0B)	Read Fast（3-byte Address）コマンド
SFLASHCMD_QUAD_IO_READ	(0xEB)	Quad I/O Read（3-byte Address）コマンド
SFLASHCMD_WRITE_ENABLE	(0x06)	Write Enable コマンド
SFLASHCMD_READ_STATUS	(0x05)	Read Status Register-1 コマンド
SFLASHCMD_READ_CONFIG	(0x15)	Read Configuration Register-1 コマンド
SFLASHCMD_WRITE_STATUS	(0x01)	Write Register（Status-1, Configuration-1）コマンド
SFLASHCMD_SECTOR_ERASE_4B	(0xDC)	Erase 256 kB（4-byte Address）コマンド
SFLASHCMD_BYTE_PROGRAM_4B	(0x12)	Page Program（4-byte Address）コマンド
SFLASHCMD_FAST_READ_4B	(0x0C)	Read Fast（4-byte Address）コマンド
SFLASHCMD_QUAD_IO_READ_4B	(0xEC)	Quad I/O Read（4-byte Address）コマンド

表7.11　サンプルプログラムで使用する定数(2)

定数名	設定値	内容
STREG_SRWD_BIT	(0x80)	Status Register／SRWDビットマスク値
STREG_QUAD_BIT	(0x40)	Status Register／QUADビットマスク値
STREG_BPROTECT_BIT	(0x3C)	Status Register／Block Protectionビットマスク値
STREG_WEL_BIT	(0x02)	Status Register／Write Enable Latchビットマスク値
STREG_WIP_BIT	(0x01)	Status Register／Write in Progressビットマスク値
CFREG_LC_BIT	(0xC0)	Configuration Register／Latency Codeビットマスク値
CFREG_4BYTE_BIT	(0x20)	Configuration Register／4BYTEビットマスク値

表7.12　サンプルプログラムで使用する定数(3)

定数名	設定値	内容
SF_PAGE_SIZE	(256)	シリアルフラッシュメモリページサイズ（書き込みサイズ）
SF_SECTOR_SIZE	(64 * 1024)	セクタサイズ
SF_NUM_OF_SECTOR	(1024)	セクタ数

表7.13　サンプルプログラムで使用する定数(4)

定数名	設定値	内容
SPIBSC_CMNCR_BSZ_SINGLE	(0)	SPIBSCデータバスに接続するシリアルフラッシュは1つ
SPIBSC_1BIT	(0)	リードコマンド発行時のビット幅を1ビットに設定
SPIBSC_4BIT	(2)	リードコマンド発行時のビット幅を4ビットに設定
SPIBSC_OUTPUT_DISABLE	(0)	リードコマンド発行時のコマンドを出力しない設定
SPIBSC_OUTPUT_ENABLE	(1)	リードコマンド発行時のコマンドを出力する設定
SPIBSC_OUTPUT_ADDR_24	(0x07)	24ビットのアドレスを出力
SPIBSC_OUTPUT_ADDR_32	(0x0f)	32ビットのアドレスを出力
SPIBSC_OUTPUT_OPD_3	(0x08)	リードコマンド発行時のオプションナルデータイネーブルOPD3を出力
SPIBSC_OUTPUT_OPD_32	(0x0c)	リードコマンド発行時のオプションナルデータイネーブルOPD3,OPD2を出力
SPIBSC_OUTPUT_OPD_321	(0x0e)	リードコマンド発行時のオプションナルデータイネーブルOPD3,OPD2,OPD1を出力
SPIBSC_OUTPUT_OPD_3210	(0x0f)	リードコマンド発行時のオプションナルデータイネーブルOPD3,OPD2,OPD1,OPD0を出力
SPIBSC_OUTPUT_SPID_8	(0x08)	SPI 動作モード時に転送データイネーブルを8（または16）ビット転送に設定
SPIBSC_OUTPUT_SPID_16	(0x0c)	SPI 動作モード時に転送データイネーブルを16（または32）ビット転送に設定
SPIBSC_OUTPUT_SPID_32	(0x0f)	SPI 動作モード時に転送データイネーブルを32（または64）ビット転送に設定
SPIBSC_SPISSL_NEGATE	(0)	SPI 動作モード時に転送終了後のSPBSSL 信号状態をネゲートに設定
SPIBSC_SPISSL_KEEP	(1)	SPI 動作モード時に転送終了後から次アクセス開始までSPBSSL信号レベルを保持する設定
SPIBSC_SPIDATA_DISABLE	(0)	SPI 動作モード時にデータリードしない設定
SPIBSC_SPIDATA_ENABLE	(1)	SPI 動作モード時にデータリードする設定
SPIBSC_DUMMY_CYC_DISABLE	(0)	ダミーサイクルを挿入しない設定
SPIBSC_DUMMY_CYC_ENABLE	(1)	ダミーサイクルを挿入する設定
SPIBSC_DUMMY_1CYC	(0)	SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するダミーサイクル数を1に設定
SPIBSC_DUMMY_2CYC	(1)	SPIマルチI/Oバス空間へのリードをSPI通信に変換する時にシリアルフラッシュメモリに出力するダミーサイクル数を2に設定

7.4 変数一覧

表 7.14 に static 型変数を、表 7.15 にグローバル変数を示します。

表7.14 static型変数

型	変数名	内容
static uint8_t	g_erased_flag[];	セクタ消去フラグ シリアルフラッシュメモリの1つのセクタに対して1バイトのフラグを割り当てます。 初期化エントリ関数実行時、セクタ消去フラグを未消去状態(0)に設定します。 セクタ消去時に消去状態(1)に設定します。

表7.15 グローバル変数

型	変数名	内容
st_spibsc_cfg_t	g_spibsc_cfg	SPIBSC 外部アドレスリード設定内容格納変数 • SPIBSC 外部アドレスリード設定内容を格納します。
st_spibsc_spimd_reg_t	g_spibsc_spimd_reg	SPIBSC SPIモード動作設定内容格納変数 • SPIBSC SPIモードで使用する場合に、SPIBSC設定内容を格納します。 サンプルプログラムでは、API関数およびユーザ定義関数内でシリアルフラッシュ制御関数を実行する際の引数として共用で使用しています。

7.5 フラッシュメモリインタフェース関数一覧

表 7.16 にフラッシュメモリインタフェース関数一覧を示します。書き込み対象となるシリアルフラッシュメモリに応じた処理をこれらの関数に実装してください。

表7.16 フラッシュメモリインタフェース関数一覧

関数名	説明
flash_init	初期化インタフェース関数 RZ/T1の外部バス（SPIマルチI/Oバス空間）に接続されているシリアルフラッシメモリへのアクセスに使用する周辺機能を設定します。 フラッシュメモリインタフェース関数の初期化を行います。
flash_write	書き込みインタフェース関数 RZ/T1の外部バス（SPIマルチI/Oバス空間）に接続されているシリアルフラッシュメモリへの書き込み処理を行います。なお、初期化インタフェース関数実行後、指定されたアドレスに対してセクタ消去が実行されていない場合は、セクタ消去処理を行います。
flash_write_entry	シリアルフラッシュメモリ書き込みモードエントリ関数
flash_veify_entry	シリアルフラッシュメモリ読み出しモードエントリ関数

表7.17 フラッシュメモリインタフェース内部関数

関数名	概要
R_SFLASH_Exmode_Setting	SPIBSC初期設定関数 SPIBSCにてシリアルフラッシュメモリを制御するために必要な初期設定およびSPIBSCを外部アドレスリードモードで使用するために必要な初期設定を行います。また初期設定に合わせて、シリアルフラッシュメモリ内のレジスタ設定を行います。初期設定後、外部アドレスリードモードに設定します。
R_SFLASH_WaitTend	SPIBSCデータ転送終了待ち関数 SPIBSCより、データ転送が終了するのを待ちます。
R_SFLASH_Exmode	SPIBSC外部アドレスモード設定関数 SPIBSCを外部アドレスリードモードに設定します。
R_SFLASH_Set_Config	SPIBSC外部アドレスリード設定関数 SPIBSCを外部アドレスリードモードで使用するために必要な初期設定を行います。
R_SFLASH_SpibscStop	SPIBSC停止関数 SPIBSCを停止します。
R_SFLASH_Spimode	SPIBSC SPIモード設定関数 SPIBSCをSPIモードに設定します。
R_SFLASH_Exmode_Init	SPIBSC外部アドレスモード初期設定関数 SPIBSCを外部アドレスリードモードで使用するために必要な初期設定を行います。初期設定後、外部アドレスリードモードに設定します。
R_SFLASH_Spimode_Init	SPIBSC SPIモード初期設定関数 SPIBSCをSPIモードで使用するために必要な初期設定を行います。初期設定後、SPIモードに設定します。
R_SFLASH_EraseSector	シリアルフラッシュ消去関数 SPIBSCのSPIモードを使用して、シリアルフラッシュの消去を行います。
R_SFLASH_ByteProgram	シリアルフラッシュ書き込み関数 SPIBSCのSPIモードを使用して、シリアルフラッシュにデータを書き込みます。
R_SFLASH_Spibsc_Transfer	シリアルフラッシュ制御関数 引数に従い、シリアルフラッシュメモリにコマンドを発行します。
R_SFLASH_Ctrl_Protect	シリアルフラッシュメモリプロテクト解除関数 引数に従い、シリアルフラッシュメモリ内のレジスタを設定し、プロテクトを解除します。

表7.18　フラッシュメモリインタフェースユーザ定義関数

関数名	概要
Userdef_SPIBSC_Set_Config	SPIBSC外部アドレスリード設定関数 使用するシリアルフラッシュメモリに応じて、SPIBSC外部アドレスリードモードの設定する内容を決定します。サンプルプログラムは、本関数で設定された内容を基にSPIBSCを外部アドレスリードモードで使用するために必要な初期設定を行います。 サンプルプログラムでは、Macronix社製シリアルフラッシュメモ（型名：MX25L51245G）を使用する場合のSPIBSC初期設定となっています。
Userdef_SFLASH_Set_Mode	シリアルフラッシュメモリ内レジスタ設定関数 使用するシリアルフラッシュメモリ応じて、SPIBSCを外部アドレスリードモードで使用する場合に必要なシリアルフラッシュメモリ内レジスタの設定を行います。 サンプルプログラムでは、Macronix社製シリアルフラッシュメモリ（型名：MX25L51245G）へのレジスタ設定を行っています。
Userdef_SFLASH_Write_Enable	シリアルフラッシュメモリライト許可関数 使用するシリアルフラッシュメモリ応じて、シリアルフラッシュメモリ内レジスタを設定し、ライト許可に設定します。 サンプルプログラムでは、Macronix社製シリアルフラッシュメモリ（型名：MX25L51245G）へのレジスタ設定を行っています。
Userdef_SFLASH_Busy_Wait	シリアルフラッシュメモリレディー待ち関数 使用するシリアルフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを読み出し、シリアルフラッシュメモリがレディー状態に遷移するのを待ちます。 サンプルプログラムでは、Macronix社製シリアルフラッシュメモリ（型名：MX25L51245G）のレディー待ちを行っています。
Userdef_SFLASH_Ctrl_Protect	シリアルフラッシュメモリプロテクト解除関数 使用するフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを設定し、プロテクトを解除します。 サンプルプログラムでは、Macronix社製シリアルフラッシュメモリ（型名：MX25L51245G）のプロテクトを解除しています。

7.6 フラッシュメモリインタフェース関数詳細

以下にフラッシュメモリインタフェース関数詳細一覧を示します。

flash_init	
概　要	初期化インタフェース関数
宣　言	int32_t flash_init(void);
説　明	RZ/T1 の外部バス（SPI マルチ I/O バス空間）に接続されているシリアルフラッシメモリへのアクセスに使用する周辺機能を設定します。 フラッシュインタフェース関数の初期化を行います。 セクタ消去フラグ（fmtool_pre_erase_sctno）を未消去状態 (0) に設定します。
引　数	なし
リターン値	0：初期化成功（サンプルプログラムでは常に 0 を設定します。） -1：初期化失敗

flash_write		
概　要	書き込みインタフェース関数	
宣　言	int32_t flash_write(uint32_t *fm_adrs, uint32_t *data, int32_t size);	
説　明	RZ/T1 の外部バス（SPI マルチ I/O バス空間）に接続されているシリアルフラッシュメモリへの書き込み処理を行います。 引数 fm_adrs にて指定されたアドレスに、引数 data に指定されたデータを引数 size で指定されたサイズ分書き込みを行います。 初期化エントリ関数実行後、指定された引数 fm_adrs にて指定されたアドレスを含むセクタに対して消去が実行されていない場合は、セクタ消去処理を行います。なおセクタの消去／未消去の判断は、セクタ消去フラグ（fmtool_pre_erase_sctno）の値で判断します。セクタ消去を行った場合は、セクタ消去フラグ（fmtool_pre_erase_sctno）の値を消去状態 (1) に設定します。	
引　数	uint32_t *fm_adrs	書き込み開始アドレス
	uint32_t *data	書き込みデータ格納アドレス
	int32_t size	書き込みサイズ（常に 512 バイト）
リターン値	0：書き込み成功 -1：書き込み失敗 -2：書き込み後のベリファイ失敗	

flash_write_entry	
概　要	シリアルフラッシュメモリ書き込みモードエントリ関数
宣　言	int32_t flash_write_entry(void);
説　明	SPIBSC を SPI モードに設定します。 本関数内より SPIBSC SPI モード設定関数（R_SFLASH_Spimode）を実行します。
引　数	なし
リターン値	0：成功 -1：失敗

flash_veify_entry	
概　要	シリアルフラッシュメモリ読み出しモードエントリ関数
宣　言	int32_t flash_veify_entry(void);
説　明	SPIBSC を外部アドレスリードモードに設定します。 本関数内より SPIBSC 外部アドレスモード設定関数（R_SFLASH_Exmode）を実行します。
引　数	なし
リターン値	0：成功 -1：失敗

R_SFLASH_Exmode_Setting		
概　要	SPIBSC 初期設定関数	
宣　言	int32_t R_SFLASH_Exmode_Setting (st_spibsc_cfg_t *spibsccfg);	
説　明	SPIBSC にてシリアルフラッシュメモリを制御するために必要な初期設定および SPIBSC を外部アドレスリードモードで使用するために必要な初期設定を行います。 また初期設定に合わせて、シリアルフラッシュメモリ内のレジスタ設定を行います。 初期設定後、外部アドレスリードモードに設定します。 本関数内で SPIBSC 外部アドレスモード初期設定関数（R_SFLASH_Exmode_Init）を実行します。	
引　数	st_spibsc_cfg_t *spibsccfg	SPIBSC 外部アドレスリード設定 設定内容は表 7.2 ～表 7.4 を参照してください。
リターン値	0：正常終了 -1: エラー	

R_SFLASH_Set_Config		
概　要	SPIBSC 外部アドレスリード設定関数	
宣　言	void R_SFLASH_Set_Config(st_spibsc_cfg_t *spibsccfg);	
説　明	使用するシリアルフラッシュメモリに応じて、SPIBSC を外部アドレスリードモードで使用するための設定内容を決定します。 本関数内で、ユーザ定義関数（SPIBSC 外部アドレスリード設定関数：Userdef_SPIBSC_Set_Config）を実行します。	
引　数	st_spibsc_cfg_t *spibsccfg	SPIBSC 外部アドレスリード設定 設定内容は表 7.2 ～表 7.4 を参照してください。
リターン値	0：正常終了 -1: エラー	

R_SFLASH_SpibscStop	
概　要	SPIBSC 停止関数
宣　言	int32_t R_SFLASH_SpibscStop(void);
説　明	SPIBSC を停止します。
引　数	なし
リターン値	なし

R_SFLASH_WaitTend	
概　要	SPIBSC データ転送終了待ち関数
宣　言	void　R_SFLASH_WaitTend(void);
説　明	SPIBSC より、データ転送が終了するのを待ちます。
引　数	なし
リターン値	なし

R_SFLASH_Exmode	
概　要	SPIBSC 外部アドレスモード設定関数
宣　言	int32_t R_SFLASH_Exmode(void);
説　明	SPIBSC を外部アドレスリードモードに設定します。
引　数	なし
リターン値	0：設定成功

R_SFLASH_Spimode	
概　要	SPIBSC SPI モード設定関数
宣　言	int32_t R_SFLASH_Spimode(void);
説　明	SPIBSC を SPI モードに設定します。
引　数	なし
リターン値	0：設定成功

R_SFLASH_Exmode_Init		
概　要	SPIBSC 外部アドレスモード初期設定関数	
宣　言	int32_t R_SFLASH_Exmode_Init(st_spibsc_cfg_t *spibsccfg)	
説　明	SPIBSC を外部アドレスリードモードで使用するために必要な初期設定を行います。 初期設定後、外部アドレスリードモードに設定します。	
引　数	st_spibsc_cfg_t *spibsccfg	SPIBSC 外部アドレスリード設定 設定内容は表 7.2 ～表 7.4 を参照してください。
リターン値	0：正常終了 -1: エラー	

R_SFLASH_Spimode_Init		
概　要	SPIBSC SPI モード初期設定関数	
宣　言	int32_t　R_SFLASH_Spimode_Init(uint32_t data_width, uint32_t addr_mode, uint32_t spbr, uint32_t brdv);	
説　明	SPIBSC を SPI モードで使用するために必要な初期設定を行います。初期設定後、SPI モードに設定します。	
引　数	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
	uint32_t addr_mode	アドレスモード設定 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時にシリアルフラッシュメモリに出力するアドレスを設定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
	uint32_t spbr	ビットレート ビットレート分周設定（brdv）と合わせてシリアルクロック（SPBCLK）のビットレートを設定します。
	uint32_t brdv	ビットレート分周設定 ビットレート分周設定（spbr）と合わせてシリアルクロック（SPBCLK）のビットレートを設定します。
リターン値	0：設定成功 -1：設定失敗	

R_SFLASH_EraseSector		
概　要	シリアルフラッシュメモリ消去関数	
宣　言	int32_t R_SFLASH_EraseSector(uint32_t addr, uint32_t data_width, uint32_t addr_mode);	
説　明	SPIBSC の SPI モードを使用して、シリアルフラッシュメモリの消去を行います。	
引　数	uint32_t addr	消去するシリアルフラッシュメモリのアドレス
	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
	uint32_t addr_mode	アドレスモード設定 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時にシリアルフラッシュメモリに出力するアドレスを設定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0：設定成功 -1：設定失敗	

R_SFLASH_ByteProgram		
概　要	シリアルフラッシュメモリ書き込み関数	
宣　言	int32_t R_SFLASH_ByteProgram(uint32_t addr, uint8_t *buf, int32_t size, uint32_t data_width, uint32_t addr_mode);	
説　明	SPIBSC の SPI モードを使用して、シリアルフラッシュメモリにデータを書き込みます。	
引　数	uint32_t addr	書き込むシリアルフラッシュメモリのアドレス
	uint8_t *buf	書き込みデータ格納バッファ
	int32_t size	書き込みサイズ（バイト単位）
	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
	uint32_t addr_mode	アドレスモード設定 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時にシリアルフラッシュメモリに出力するアドレスを設定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0：設定成功 -1：設定失敗	

R_SFLASH_Spibsc_Transfer		
概　要	シリアルフラッシュメモリ制御関数	
宣　言	int32_t R_SFLASH_Spibsc_Transfer(st_spibsc_spimd_reg_t *regset);	
説　明	SPIBSC の SPI モードを使用して、シリアルフラッシュメモリにアクセスします。	
引　数	st_spibsc_spimd_reg_t * regset	SPIBSC SPI モード設定 設定内容は表 7.2 ～表 7.4 を参照してください。
リターン値	0：設定成功 -1：設定失敗	

R_SFLASH_Ctrl_Protect		
概　要	シリアルフラッシュメモリプロテクト解除関数	
宣　言	int32_t R_SFLASH_Ctrl_Protect(uint32_t req, uint32_t data_width);	
説　明	シリアルフラッシュメモリ内のレジスタを設定し、プロテクトを解除します。	
引　数	uint32_t_t req	レジスタ設定情報 SF_REQ_PROTECT：プロテクトに設定 SF_REQ_UNPROTECT：プロテクト解除に設定
	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
リターン値	なし	

Userdef_SPIBSC_Set_Config		
概　要	SPIBSC 外部アドレスリード設定関数	
宣　言	void Userdef_SPIBSC_Set_Config(st_spibsc_cfg_t *spibsccfg);	
説　明	使用するシリアルフラッシュメモリに応じて、SPIBSC 外部アドレスリードモードの設定する内容を決定します。本関数にて、引数 spibsccfg にて指定された領域に SPIBSC を外部アドレスリードモードで使用するために必要な初期設定を行ってください。 引数 spibsccfg に設定する内容については、表 7.2 ～表 7.4 を参照してください。	
引　数	st_spibsc_cfg_t *spibsccfg	SPIBSC 外部アドレスリード設定 設定内容は表 7.2 ～表 7.4 を参照してください。
リターン値	なし	
注意事項	サンプルプログラムでは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を使用する場合の SPIBSC 初期設定となっています。	

Userdef_SFLASH_Set_Mode		
概　要	シリアルフラッシュメモリ内レジスタ設定関数	
宣　言	int32_t Userdef_SFLASH_Set_Mode(uint32_t data_width, uint32_t addr_mode);	
説　明	本関数内で使用するシリアルフラッシュメモリ応じて、SPIBSC を外部アドレスリードモードで使用する場合に必要なシリアルフラッシュメモリ内レジスタを設定する処理を実装してください。	
引　数	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
	uint32_t addr_mode	アドレスモード設定 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時にシリアルフラッシュメモリに出力するアドレスを設定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0：設定成功 -1：設定失敗	
注意事項	サンプルプログラムでは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を使用する場合の SPIBSC 初期設定となっています。	

Userdef_SFLASH_Write_Enable	
概　要	シリアルフラッシュメモリライト許可関数
宣　言	int32_t Userdef_SFLASH_Write_Enable(void);
説　明	本関数内で使用するシリアルフラッシュメモリ応じて、シリアルフラッシュメモリ内レジスタを設定し、ライト許可に設定する処理を実装してください。
引　数	なし
リターン値	0：設定成功 -1：設定失敗
注意事項	サンプルプログラムでは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を使用する場合の SPIBSC 初期設定となっています。

Userdef_SFLASH_Busy_Wait		
概　要	シリアルフラッシュメモリレディー待ち関数	
宣　言	int32_t Userdef_SFLASH_Busy_Wait(uint32_t data_width);	
説　明	本関数内で使用するシリアルフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを読み出し、シリアルフラッシュメモリがレディー状態に遷移する処理を実装してください。	
引　数	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
リターン値	なし	
注意事項	サンプルプログラムでは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を使用する場合の SPIBSC 初期設定となっています。	

Userdef_SFLASH_Ctrl_Protect		
概　要	シリアルフラッシュメモリプロテクト解除関数	
宣　言	int32_t Userdef_SFLASH_Ctrl_Protect(uint32_t req, uint32_t data_width);	
説　明	本関数内で使用するシリアルフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを設定し、プロテクトを解除する処理を実装してください。	
引　数	uint32_t req	レジスタ設定情報 SF_REQ_PROTECT：プロテクトに設定 SF_REQ_UNPROTECT：プロテクト解除に設定
	uint32_t data_width	データリードビット幅 SPI マルチ I/O バス空間へのリードを SPI 通信に変換する時のシリアルフラッシュメモリのデータリードビット幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
リターン値	なし	
注意事項	サンプルプログラムでは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を使用する場合の SPIBSC 初期設定となっています。	

7.7 フラッシュメモリインタフェース関数のフロー

7.7.1 初期化インタフェース関数のフロー

図 7.1 に初期化インタフェース関数のフローを示します。

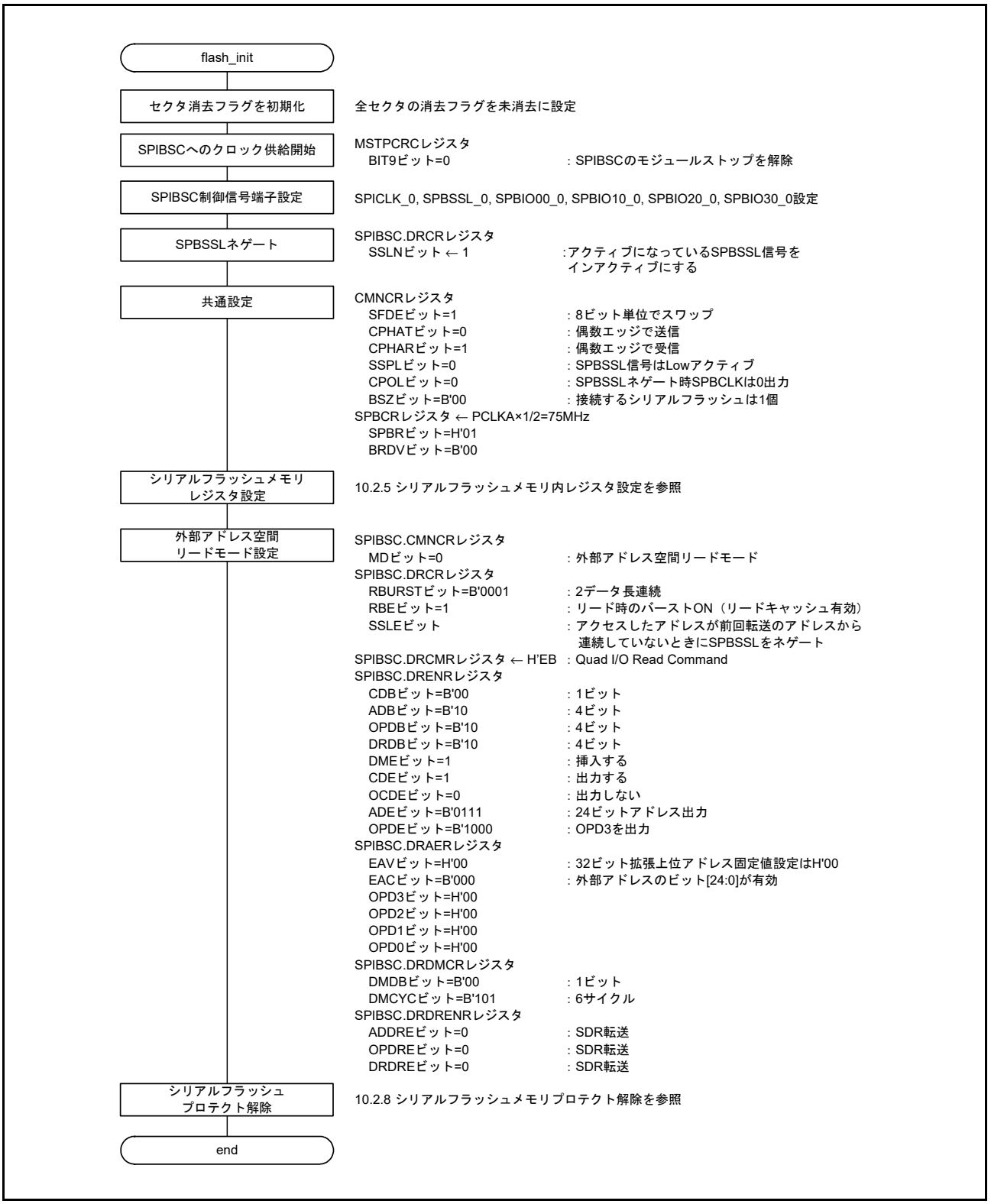


図 7.1 初期化インタフェース関数のフロー

7.7.2 シリアルフラッシュメモリ書き込みモードエントリ関数

図 7.2 にシリアルフラッシュメモリ書き込みモードエントリ関数のフローを示します。

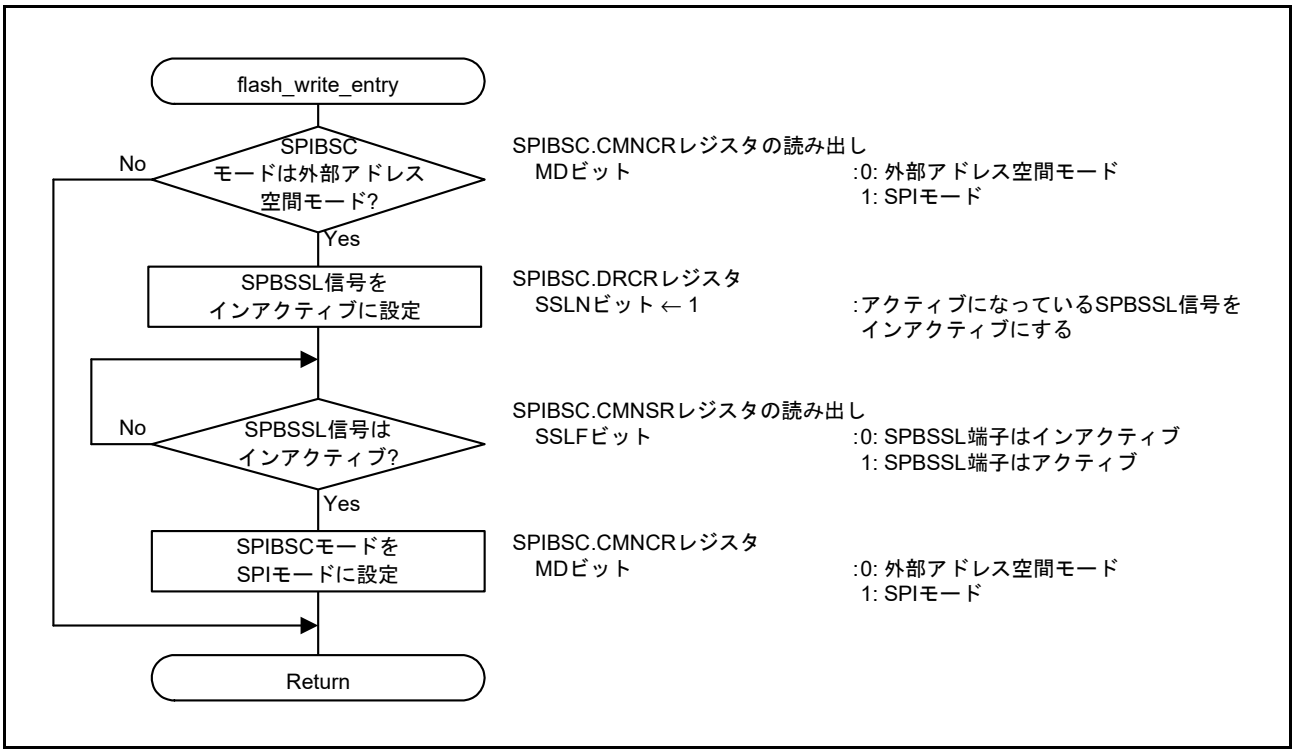


図 7.2 シリアルフラッシュメモリ書き込みモードエントリ関数のフロー

7.7.3 シリアルフラッシュメモリ読み出しモードエントリ関数

図 7.3 にシリアルフラッシュメモリ読み出しモードエントリ関数のフローを示します。

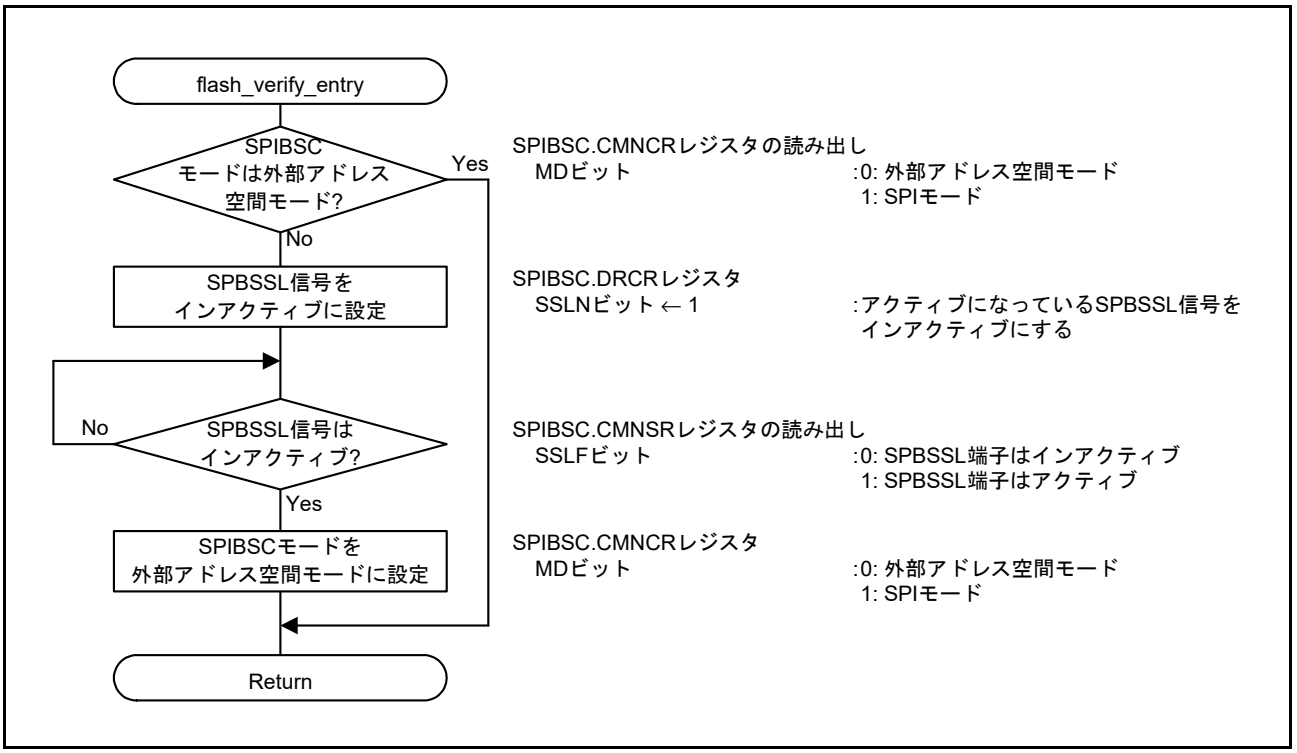


図 7.3 シリアルフラッシュメモリ読み出しモードエントリ関数のフロー

7.7.4 書き込みインタフェース関数のフロー

図 7.4 にシリアルフラッシュメモリ書き込み関数のフローを示します。

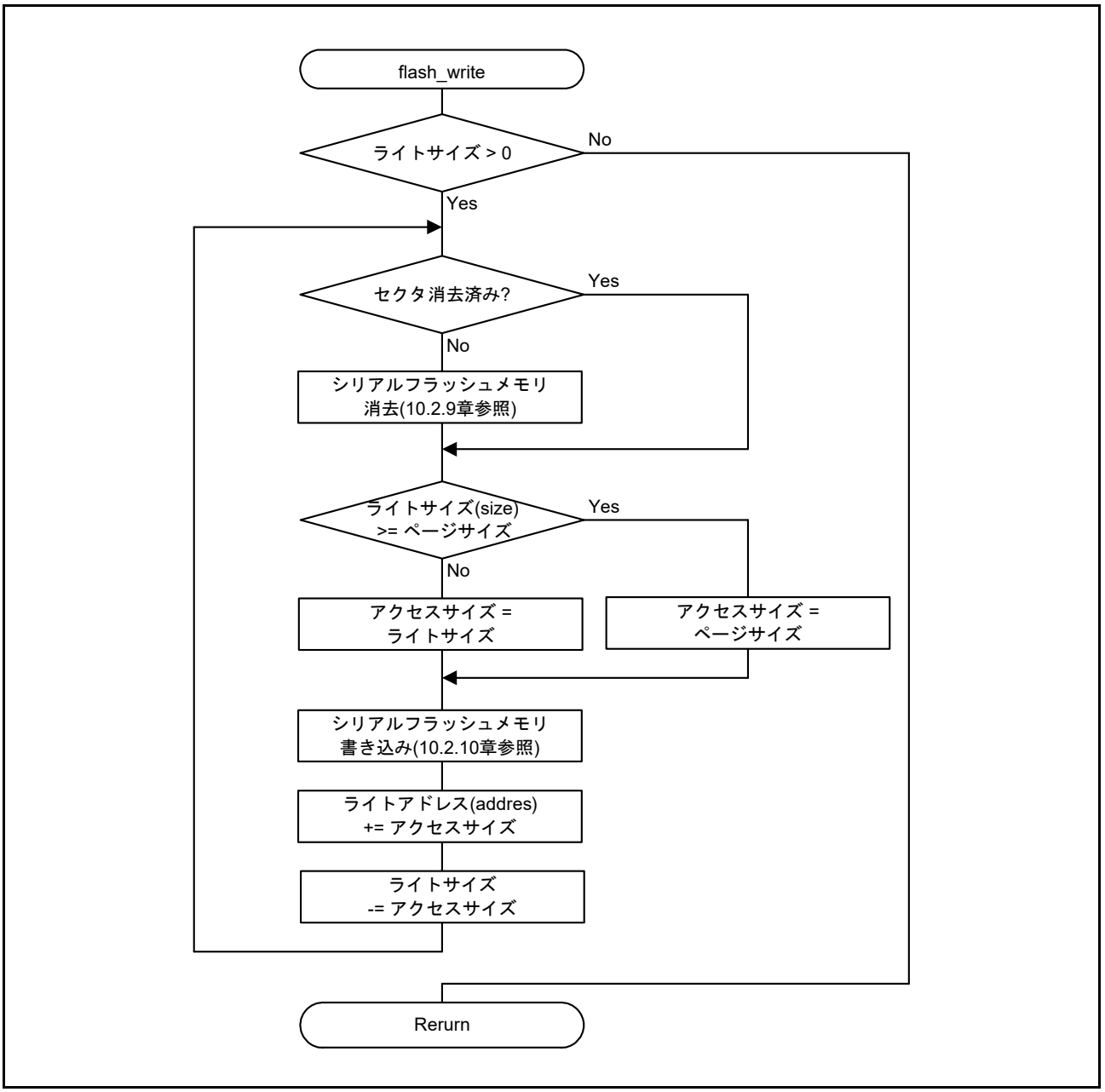


図 7.4 シリアルフラッシュメモリ書き込み関数のフロー

8. フラッシュダウンローダの動作

この章では、フラッシュダウンローダの動作について説明します。

8.1 アプリケーションプログラムのメモリ配置

図 8.1 に本アプリケーションノートにて紹介するフラッシュダウンローダで書き込むアプリケーションプログラムのメモリ配置の例を示します。

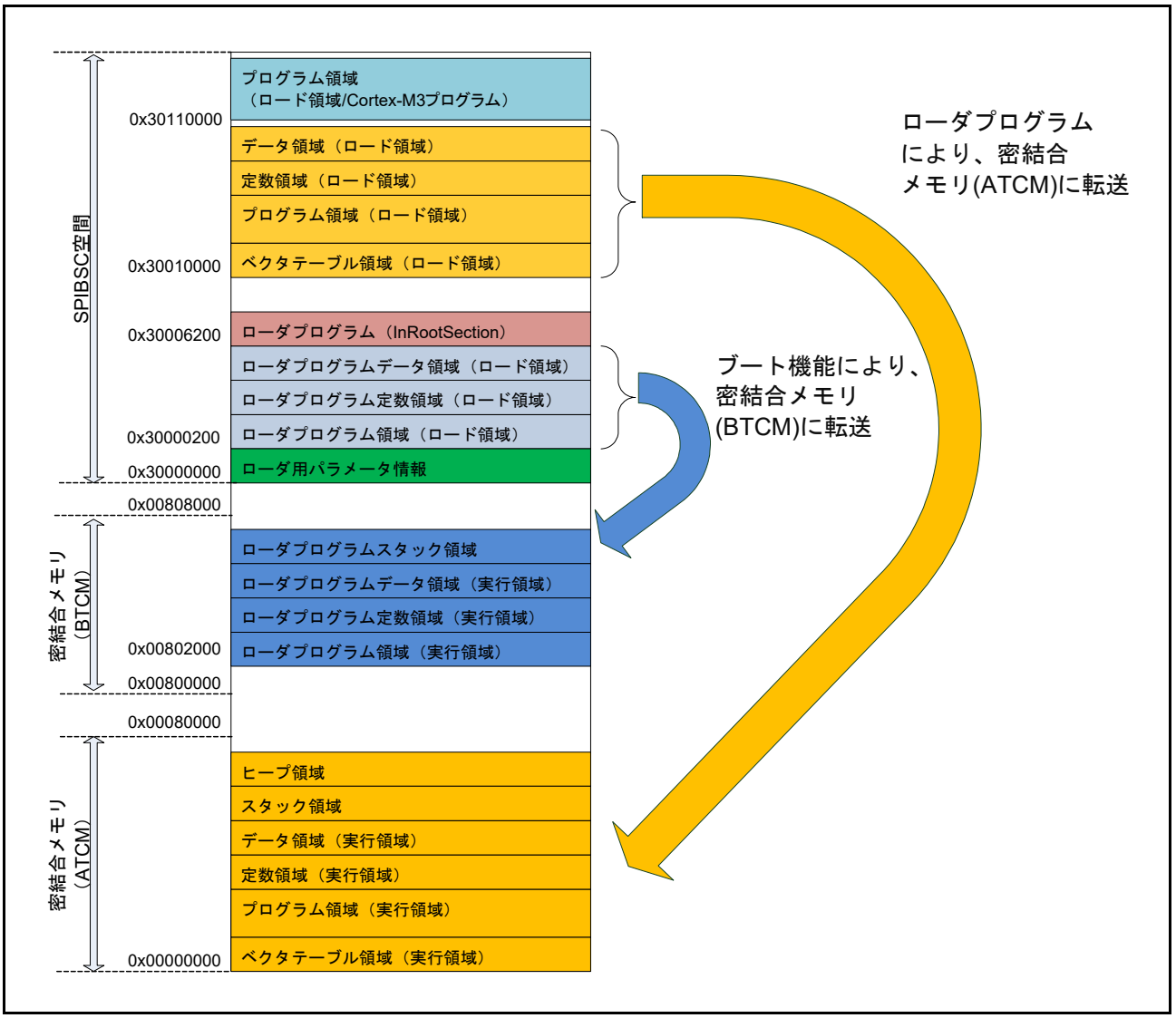


図 8.1 アプリケーションプログラムのメモリ配置の例

1. 本応用でのアプリケーションプログラムは、RZ/T1 グループマイコンが参照するローダ用パラメータ情報およびローダプログラムの領域と、ローダプログラム（InRootSection）領域、アプリケーションプログラムの 4 つの領域で構成されています。
2. 4 つの領域のバイナリデータは、それぞれアプリケーションプロジェクトより生成した実行形式ファイル（axf ファイル）より、バイナリファイル生成ツールにて 3 つのアプリケーションバイナリファイル 注 1 として生成されます。

注 1. 生成されるアプリケーションバイナリファイルについては表 9.3 を参照してください。

8.2 フラッシュダウンローダの処理フロー

図 8.2 ～図 8.4 にフラッシュダウンローダ処理フローを示します。

フラッシュダウンローダは、DS-5 より RZ/T1 グループマイコンの密結合メモリ（ATCM）領域にロードされ、エントリポイント 0x00000000 番地より実行されます。フラッシュダウンローダは、スタックポインタを初期化後、main 関数へのエントリ関数である __main() を実行します。__main() は ARM コンパイラの標準ライブラリ関数として提供されており、本関数を実行することで、セミホスティング機能が有効になります注 1。__main 関数から実行される $Sub$$main() からフラッシュダウンロードメイン関数（flash_main）を実行します。flash_main 実行後、後述するアプリケーションダウンロードスクリプトでダウンロード終了を判定するための prg_complete 関数を実行し、無限ループします。

注 1. 詳細は、「ARM® コンパイラツールチェーン ARM® プロセッサをターゲットとしたソフトウェア開発／組み込みソフトウェアの開発」を参照してください。

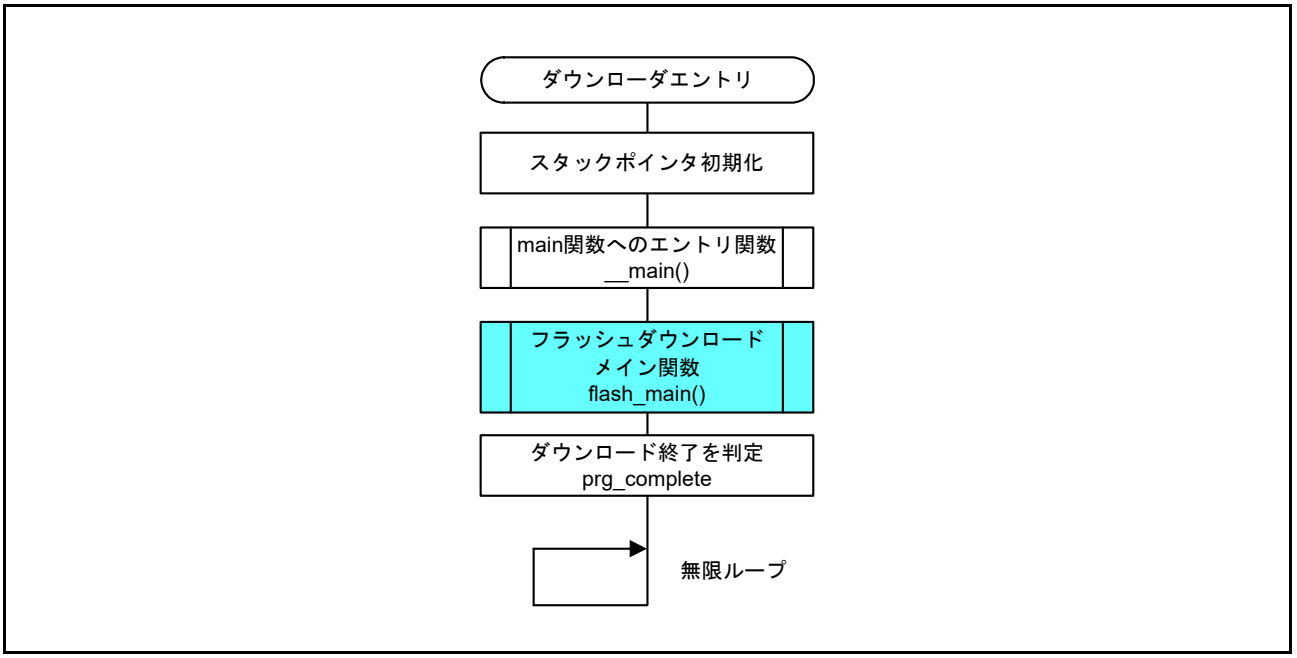


図 8.2 フラッシュダウンローダ処理フロー (1/3)

flash_main 関数では、セミホスティングターミナル入力より、ダウンロードを実行するかを判断します。セミホスティングターミナル入力より ”Y” が入力されると、フラッシュメモリへの書き込みを開始します。セミホスティングターミナル入力より ”N” が入力されると、ダウンロードが終了したと判断し、flash_main 関数を終了します。図 8.3 に flash_main 関数の処理フローを示します。

フラッシュメモリへの書き込みは、RZ_T1_FlashProgram_Sub 関数で行います。セミホスティング機能を使用して、アプリケーションバイナリファイルよりデータを読み出して、シリアルフラッシュメモリにデータを書き込みます。図 8.4 に RZ_T1_FlashProgram_Sub 関数の処理フローを示します。

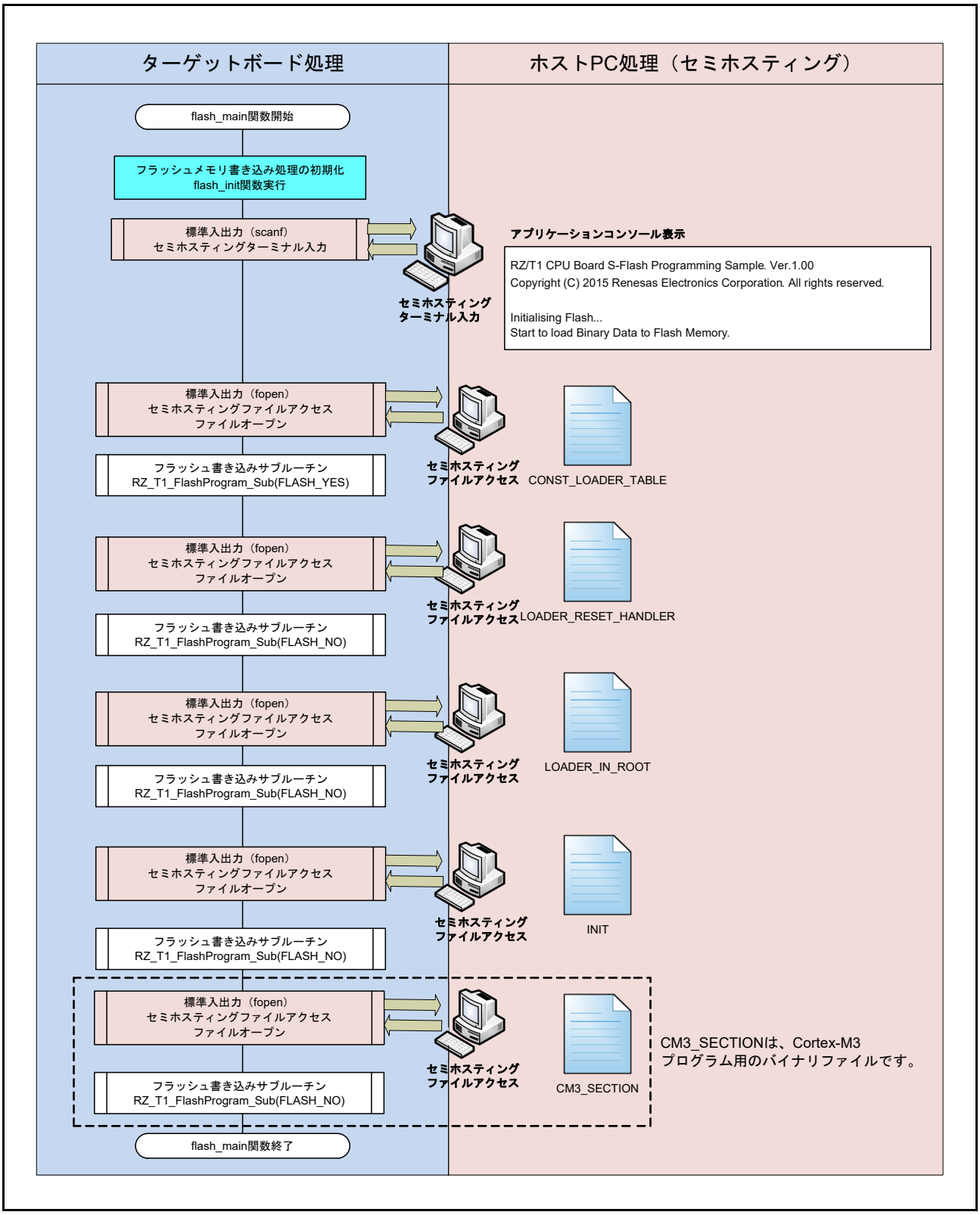


図 8.3　フラッシュダウンローダ処理フロー (2/3)

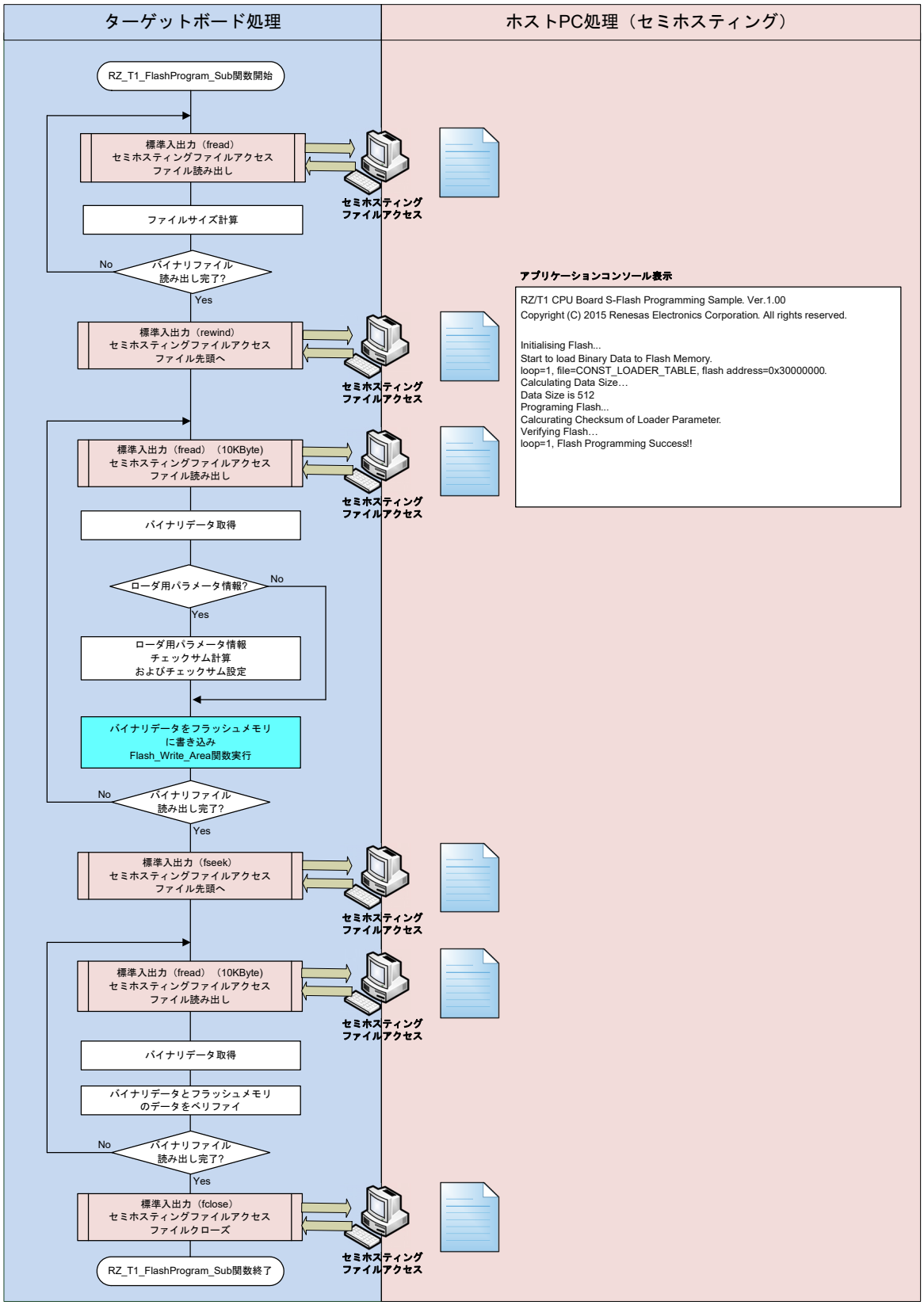


図 8.4　フラッシュダウンローダ処理フロー (3/3)

8.2.1 ローダ用パラメータ情報チェックサムの計算

フラッシュダウンローダは、RZ/T1 グループマイコンがブート機能で参照するローダ用パラメータ情報のチェックサムを計算し、フラッシュメモリに書き込むことができます。

RZ_T1_FlashProgram_Sub 関数の引数 check_sum_flag に FLASH_YES を指定して実行した場合、引数 srcfile にて指定されたバイナリファイルは、ローダ用パラメータ情報バイナリファイルと判断し、バイナリファイル先頭データから 72 バイト（H'48 バイト）分のチェックサムを計算します。バイナリファイル先頭データから 72 バイト（H'48 バイト）目のデータが H'17320508 である場合、計算したチェックサムをローダ用パラメータのチェックサム値としてフラッシュメモリに書き込みます。バイナリファイル先頭データから 72 バイト（H'48 バイト）目のデータが H'17320508 ではない場合、計算したチェックサムと当該データを比較し、値が異なる場合は、以降の書き込み処理を行わず、エラー終了します。

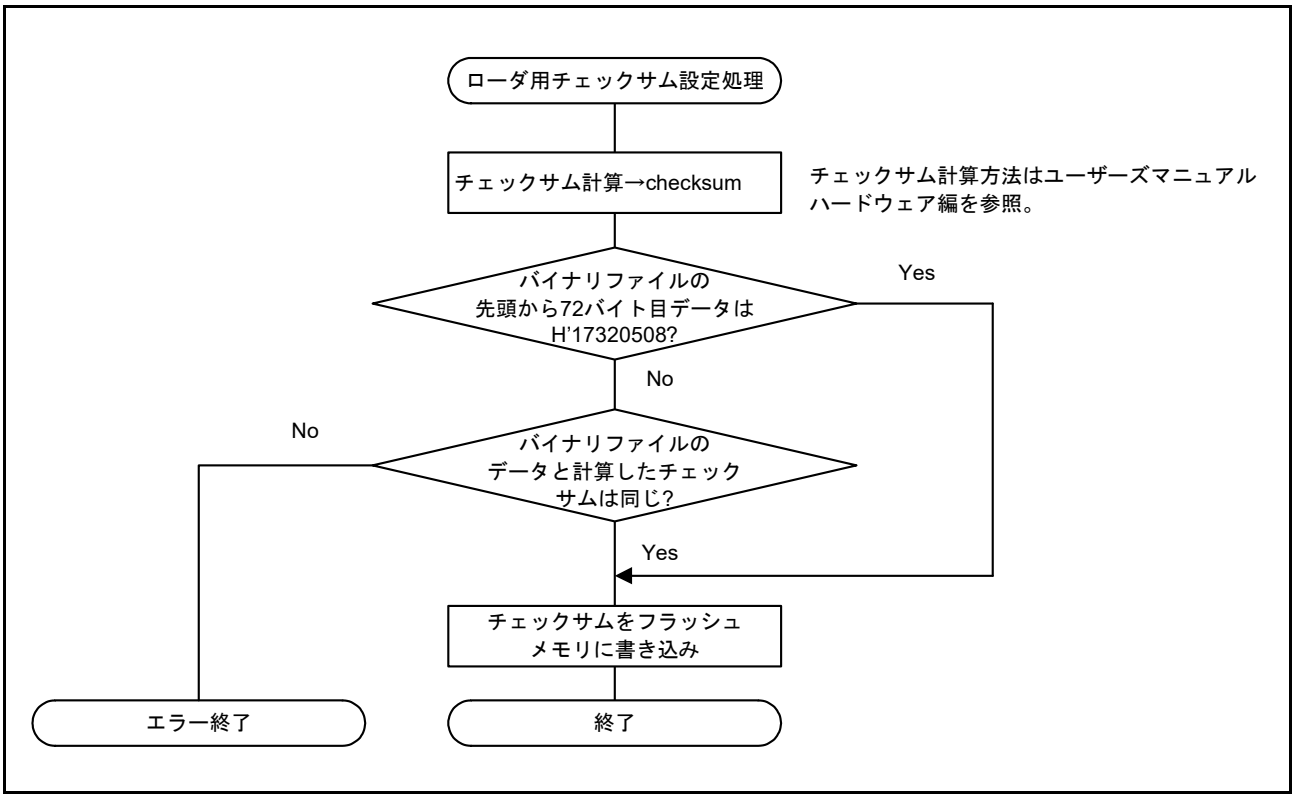


図 8.5 ローダ用パラメータ情報チェックサム設定処理フロー

9. フラッシュダウンローダの構成

9.1 プロジェクトの構成

フラッシュダウンローダは、表 9.1、表 9.2 に示す DS-5 プロジェクトおよび DS-5 スクリプトで構成されています。また表 9.3 にアプリケーションプロジェクトより生成されるアプリケーションバイナリファイルを示します。これらのアプリケーションバイナリファイルを、「8.2　フラッシュダウンローダの処理フロー」で説明したフローで、フラッシュメモリに書き込みます。

表9.1　プロジェクト一覧

プロジェクト名	説明
RZ_T_fmtool_sflash	このプロジェクトでフラッシュダウンローダをビルドします。このプロジェクトをフラッシュダウンローダプロジェクトと呼びます。
RZ_T_sflash_sample	このプロジェクトでアプリケーションプログラムをビルドします。このプロジェクトをアプリケーションプロジェクトと呼びます。また、ビルドで生成した実行形式ファイル（axfファイル）により、バイナリ生成ツール（fromelf.exe）にてアプリケーションバイナリファイルを生成します。

表9.2　スクリプトファイル一覧

スクリプト名	説明
init_RZ-T.ds	RZ/T1評価ボード初期化スクリプトです。 DS-5とRZ/T1評価ボードを接続した際に、RZ/T1グループマイコンの密結合メモリ（ATCM）の書き込みを許可する等の処理を実行するDS-5スクリプトです。
RZ_T_sflash_sample.ds	アプリケーションダウンロードスクリプトです。 アプリケーションプログラムをRZ/T1グループマイコンの外部アドレス空間（SPIマルチI/Oバス空間）に配置されたシリアルフラッシュメモリに書き込むための一連の作業を記載したDS-5スクリプトです。
init_RZ-T2.ds	アプリケーションダウンロードスクリプトから実行されるRZ/T1評価ボード初期化スクリプトです。DS-5メモリ領域設定を行わない以外は、init_RZ-T.ds と同じです。

表9.3　アプリケーションバイナリファイル一覧

バイナリファイル名	書き込み開始アドレス注1	説明
CONST_LOADER_TABLE	H'30000000	アプリケーション(1)（ローダ用パラメータ情報） バイナリファイル
LOADER_RESET_HANDLER	H'30000200	アプリケーション(2)（ローダプログラム） バイナリファイル
LOADER_IN_ROOT	H'30006200	アプリケーション(3)（ローダプログラム） バイナリファイル
INIT	H'30010000	アプリケーション(4)（ユーザプログラム） バイナリファイル
CM3_SECTION注2	H'30110000	アプリケーション(5)（ユーザプログラム） バイナリファイル（Cortex-M3プログラム）

注1.　図8.1に示すアプリケーションプログラムのメモリ配置の場合
注2.　CM3_SECTIONは、Cortex-M3プログラム用のバイナリファイルです。詳細は、アプリケーションノート「RZ/T1グループ R-IN Engine搭載製品　初期設定」を参照ください。

9.2 RZ/T1 評価ボード初期化スクリプト

図 9.1 に RZ/T1 評価ボード初期化スクリプト処理内容を示します。

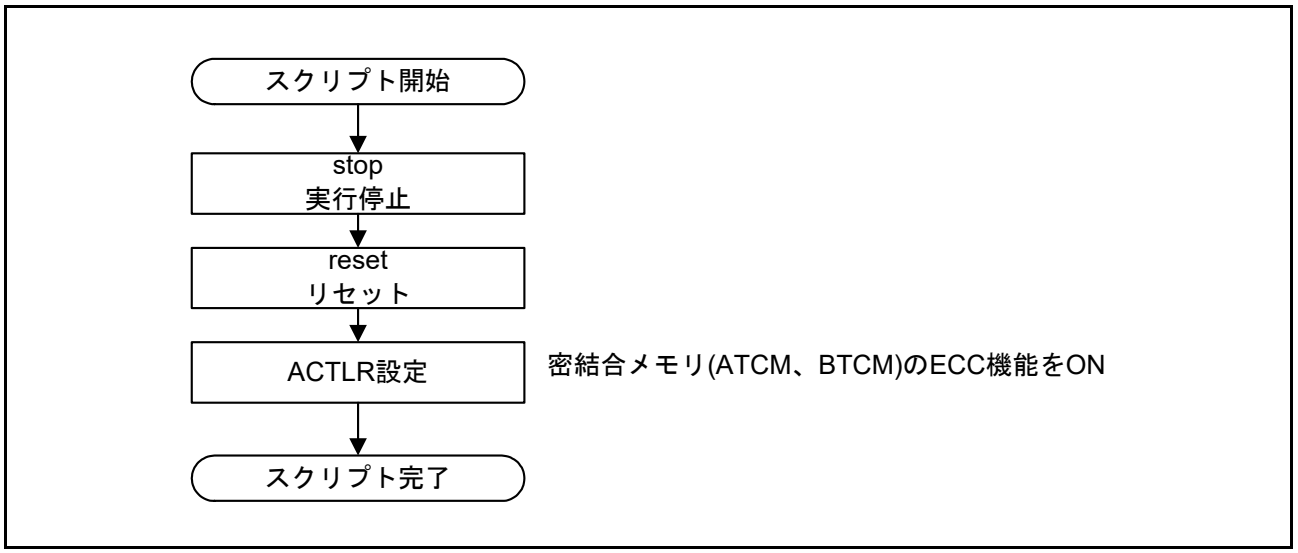


図 9.1 RZ/T1 評価ボード初期化スクリプト処理内容

9.3 アプリケーションダウンロードスクリプト

図 9.2 にアプリケーションプログラム RZ_T_sflash_sample を RZ/T1 グループマイコンの外部アドレス空間（SPI マルチ I/O バス空間）に配置されたシリアルフラッシュメモリに書き込むためのアプリケーションダウンロードスクリプト処理内容を示します。DS-5 より本スクリプトを実行することで、RZ/T1 グループマイコンの外部アドレス空間（SPI マルチ I/O バス空間）に配置されたシリアルフラッシュメモリにアプリケーションプログラム RZ_T_sflash_sample が書き込まれ、DS-5 にアプリケーションプログラム RZ_T1_sflash_sample のシンボル情報がロードされます。

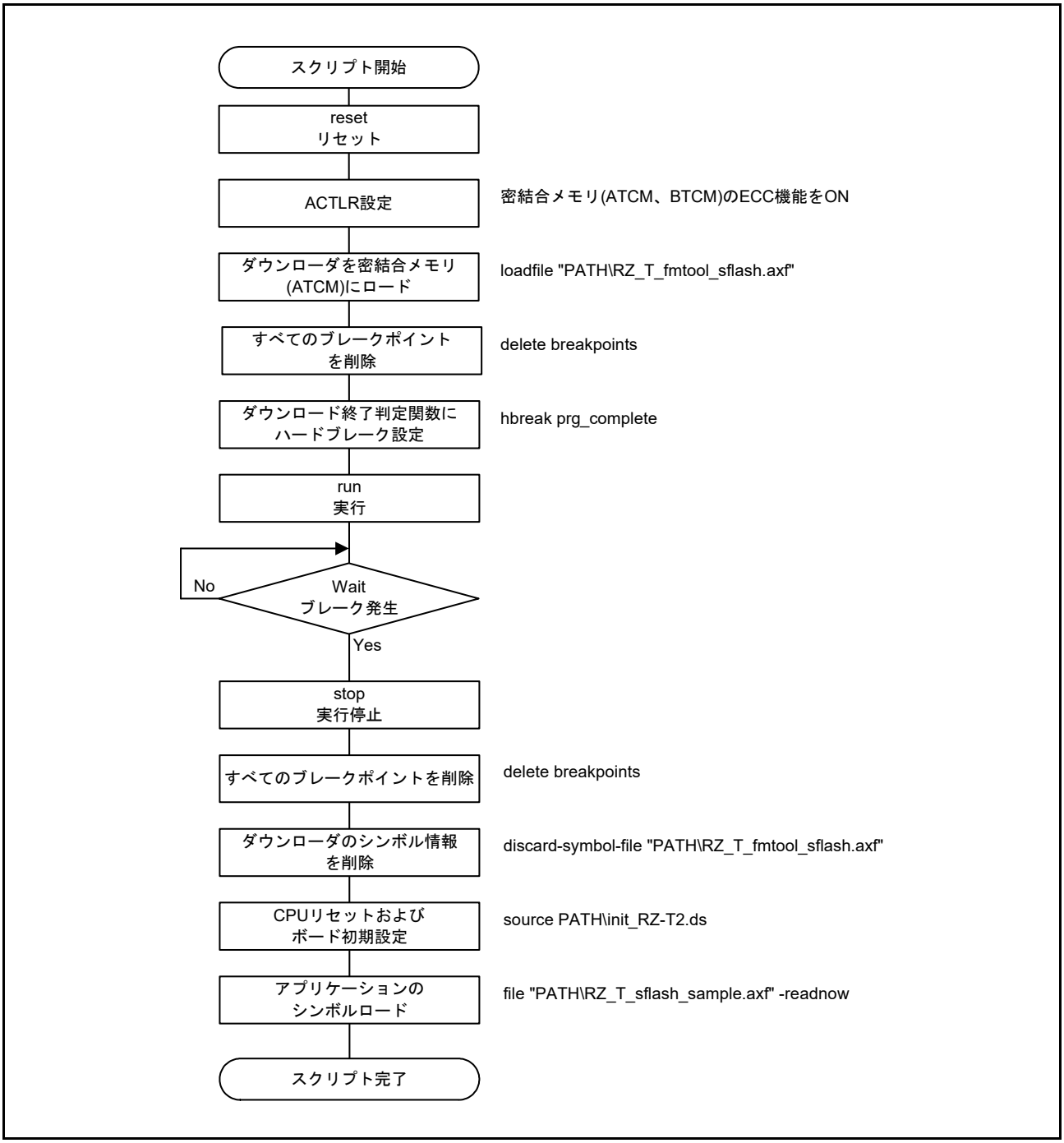


図 9.2 アプリケーションダウンロードスクリプト処理内容

10. 応用例

本章では、サンプルプログラムの応用例として、お客様がフラッシュメモリに書き込むバイナリファイルを変更する方法、およびお客様が使用するフラッシュメモリに応じたサンプルプログラムの変更方法について説明します。

10.1 バイナリファイル名および書き込みアドレスを変更する方法

この章では、「8.2 フラッシュダウンローダの処理フロー」にて説明した、フラッシュメモリに書き込むバイナリファイル名、およびフラッシュメモリ書き込みアドレスを変更する方法について説明します。

10.1.1 フラッシュメモリに書き込むバイナリファイル名を変更する方法

フラッシュメモリに書き込むバイナリファイル名は、Flash_Programming.c ファイル内の RZ_T1_FlashProgrammer 関数に実装されています。RZ_T1_FlashProgrammer 関数に実装されているバイナリファイル名を変更するすることで、フラッシュメモリに書き込むバイナリファイル名を変更することができます。

なお、DS-5 ／セミホスティング機能が実行される場合のカレントディレクトリは、デフォルトで DS-5 ワークスペースディレクトリに設定されています注 1 ので、相対パスを使用することでより別ホスト PC で実行されることを考慮したフラッシュダウンローダを開発することが可能です。

図 10.1、図 10.2 に実装例を示します。

注 1. カレントディレクトリについては、ARM® より提供される「ARM® コンパイラツールチェーン ARM® プロセッサをターゲットとしたソフトウェア開発／セミホスティング」を参照してください。

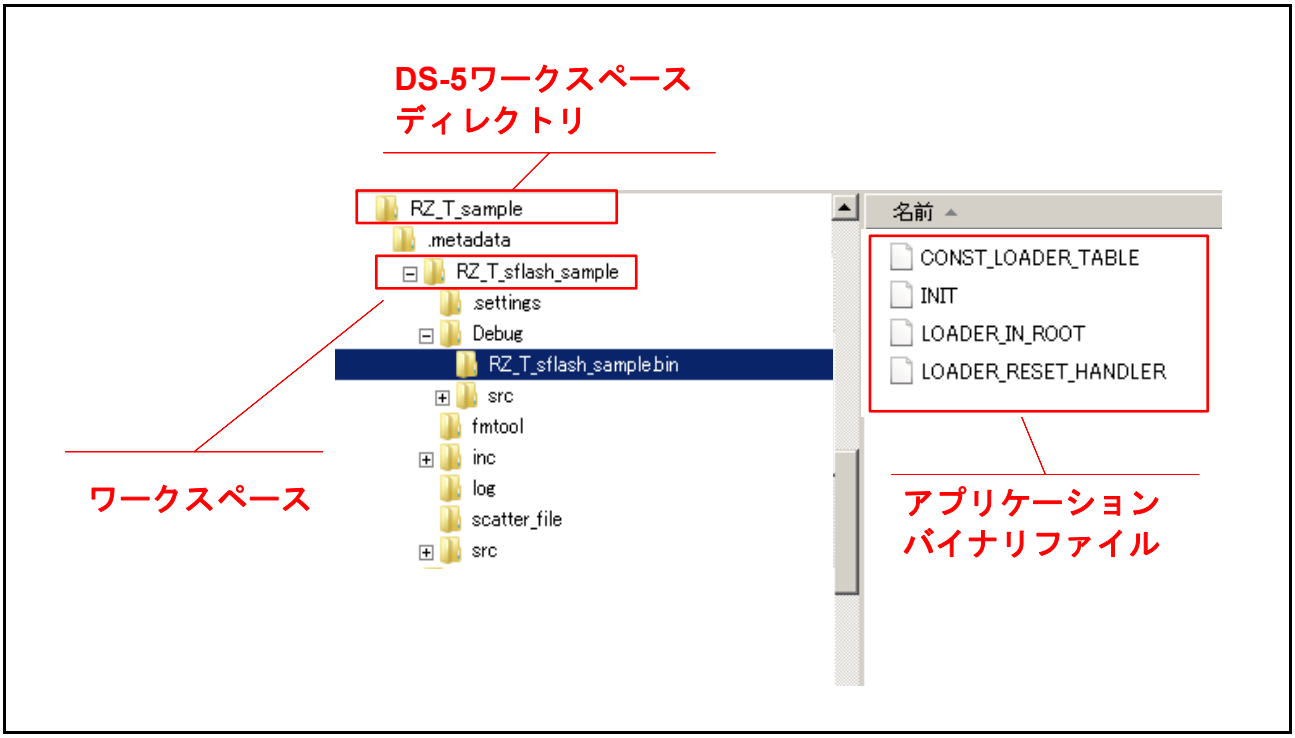


図 10.1 実装例におけるディレクトリ構成

変更前	srcfile = fopen(".¥¥RZ_T_sflash_sample¥¥Debug¥¥RZ_T_sflash_sample.bin¥¥INIT", "r");
	if(srcfile == 0)
	{
	    printf("loop=%d, Could not open file. Exiting.¥n", loop);
	    return;
	}
	address = (uint32_t *)0x30010000;
	printf("loop=%d, file=INIT, flash address=0x%08x.¥n", loop, (uint32_t)address);
	
	ret = RZ_T1_FlashProgram_Sub(srcfile, address, FLASH_NO);
	fclose(srcfile);
	if(ret != 0)
	{
	    printf("loop=%d, Flash Programming Error!!¥n", loop);
	    return;
	}

変更後	srcfile = fopen(".¥¥RZ_T_sflash_sample¥¥Debug¥¥RZ_T_sflash_sample.bin¥¥INIT2", "r");
	if(srcfile == 0)
	{
	    printf("loop=%d, Could not open file. Exiting.¥n", loop);
	    return;
	}
	address = (uint32_t *)0x30040000;
	printf("loop=%d, file=INIT2, flash address=0x%08x.¥n", loop, (uint32_t)address);
	
	ret = RZ_T1_FlashProgram_Sub(srcfile, address, FLASH_NO);
	fclose(srcfile);
	if(ret != 0)
	{
	    printf("loop=%d, Flash Programming Error!!¥n", loop);
	    return;
	}

図10.2　実装例

10.1.2 フラッシュメモリ書き込みアドレスを変更する方法

ファイルパス入力と同様に、フラッシュメモリに書き込むアドレスは、Flash_Programming.c ファイル内の RZ_T1_FlashProgrammer 関数に実装されています。RZ_T1_FlashProgrammer 関数に実装されている書き込みアドレスを変更するすることで、フラッシュメモリ書き込みアドレスを変更することができます。

スキャッタファイルでイメージレイアウトを設定した場合、アプリケーションバイナリファイルはロードモジュール（LOAD_MODULE）毎に生成されます。生成されたアプリケーションバイナリファイル毎のフラッシュメモリ書き込みアドレスは、アプリケーションプロジェクトで設定したイメージレイアウトに依存します。

図 8.1 に示すメモリ配置を持つアプリケーションプログラム RZ_T_sflash_sample のイメージレイアウト（スキャッタファイル）例を**図 10.3** に、生成されるアプリケーションバイナリファイル毎のフラッシュメモリ書き込み開始アドレスを**表 10.1** に示します。

実装例については、**図 10.1** ～**図 10.3** を参照してください。

```
LOAD_MODULE1 0x30000000    0x00000200
{
     CONST_LOADER_TABLE    0x30000000    FIXED
    {
        * (CONST_LOADER_TABLE)
    }{
}
LOAD_MODULE2 0x30000200    0x00006000
{
    LOADER_RESET_HANDLER    0x00802000    FIXED
    {
        * (LOADER_RESET_HANDLER, +FIRST)
        * (USER_PROG_JUMP)
    }
    以下省略
}
LOAD_MODULE3 0x30006200    (0x30020000 - 0x30006200)
{
    LOADER_IN_ROOT    0x40006200    FIXED
    {
        * (InRoot$$Sections)
    }
}
LOAD_MODULE4 0x30010000    (0x30110000 - 0x30010000)
{
    INIT              0x00000000    FIXED
    {
        * (VECTOR_TABLE, +FIRST)
        * (RESET_HANDLER)
        * (IRQ_FIQ_HANDLER)
    }
    以下省略
}
LOAD_MODULE5 0x40120000    (0x34000000 - 0x30110000)
{
    CM3_SECTION       0x30120000    FIXED
    {
        cm3.o(sdram)
    }
}
```

図 10.3　イメージレイアウト（スキャッタファイル）例

表10.1　生成されるアプリケーションバイナリファイル毎のフラッシュメモリ書き込み開始アドレス

バイナリファイル名	書き込み開始アドレス	説明
CONST_LOADER_TABLE	H'30000000	アプリケーション(1)（ローダ用パラメータ情報） バイナリファイル
LOADER_RESET_HANDLER	H'30000200	アプリケーション(2)（ローダプログラム） バイナリファイル
LOADER_IN_ROOT	H'30006200	アプリケーション(3)（ローダプログラム） バイナリファイル
INIT	H'30010000	アプリケーション(4)（ユーザプログラム） バイナリファイル
CM3_SECTION注1	H'30110000	アプリケーション(5)（ユーザプログラム） バイナリファイル

注1.　CM3_SECTIONは、Cortex-M3プログラム用のバイナリファイルです。詳細は、アプリケーションノート「RZ/T1グループ R-IN Engine搭載製品　初期設定」を参照ください。

10.2　フラッシュメモリに応じたサンプルプログラムの変更方法

本章では、サンプルプログラムの応用例として、お客様が使用するシリアルフラッシュメモリに応じたサンプルプログラムの変更方法について説明します。

10.2.1　サンプルプログラムの条件

サンプルプログラムは、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）を表 10.2 に示す条件で使用する場合に最適な設定を行っています。

これらの条件を変更する場合のサンプルプログラムの変更方法について説明します。

表10.2　サンプルプログラムの条件

条件	設定内容	備考
シリアルフラッシュメモリ	Macronix社製シリアルフラッシュメモリ （型名：MX25L51245G）	—
データバス幅	4ビット	データリード時のビット幅
	1ビット	データライト時のビット幅
アドレスバイト数	4バイト	アドレス指定時に発行するバイト数

10.2.2　シリアルフラッシュメモリを変更しない場合のサンプルプログラム変更方法

表 10.3 に使用するシリアルフラッシュメモリを変更しない場合のサンプルプログラムの変更方法を示します。

表10.3　シリアルフラッシュを変更しない場合のサンプルプログラム変更方法

条件	変更内容	変更方法
データリード時のデータバス幅	1ビット	マクロ定義（SPIBSC_BUS_WITDH）注1に(1)を定義
	4ビット	マクロ定義（SPIBSC_BUS_WITDH）に(4)を定義
アドレスバイト数	3バイト	マクロ定義（SPIBSC_OUTPUT_ADDR）注2に （SPIBSC_OUTPUT_ADDR_24）を定義
	4バイト	マクロ定義（SPIBSC_OUTPUT_ADDR）に （SPIBSC_OUTPUT_ADDR_32）を定義

注1.　マクロ定義（SPIBSC_BUS_WITDH）はspibsc_ioset_userdef.cファイル内に定義されています。
注2.　マクロ定義（SPIBSC_OUTPUT_ADDR）はspibsc_ioset_userdef.cファイル内に定義されています。

10.2.3 シリアルフラッシュメモリを変更する場合のサンプルプログラム変更方法

シリアルフラッシュメモリを変更する場合、使用するシリアルフラッシュメモリの仕様に合わせてサンプルプログラムを変更する必要があります。

表 10.4 にサンプルプログラムの変更のポイント示します。

表10.4 サンプルプログラムの変更のポイント

変更のポイント	内容
リードコマンド波形	外部アドレス空間リードモードにより、SPIマルチI/O バス空間へのリードをSPI通信に変換する際にシリアルフラッシュメモリに出力する信号を使用するシリアルフラッシュメモリのリードコマンドに合わせて変更します。
シリアルフラッシュメモリ内レジスタ設定	使用するシリアルフラッシュメモリ応じて、SPIBSCを外部アドレスリードモードで使用する場合に必要なシリアルフラッシュメモリ内レジスタの設定を行います。
シリアルフラッシュメモリライト許可	使用するシリアルフラッシュメモリ応じて、シリアルフラッシュメモリ内レジスタを設定し、ライト許可に設定します。注1
シリアルフラッシュメモリレディー待ち	使用するフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを読み出し、シリアルフラッシュメモリがレディー状態に遷移するのを待ちます。
シリアルフラッシュメモリプロテクト解除	使用するフラッシュメモリに応じて、シリアルフラッシュメモリ内のレジスタを設定し、プロテクトを解除します。注2
シリアルフラッシュメモリ消去	使用するフラッシュメモリに応じて、シリアルフラッシュメモリのセクタ消去を行います。
シリアルフラッシュメモリ書き込み	使用するフラッシュメモリに応じて、シリアルフラッシュメモリの書き込みを行います。

注1. シリアルフラッシュメモリにより、シリアルフラッシュメモリ内のレジスタを設定するために、ライト許可が必要な場合があります。

注2. シリアルフラッシュメモリにより、シリアルフラッシュメモリ内のレジスタを設定するために、プロテクト解除が必要な場合があります。

10.2.4 リードコマンド波形の変更

外部アドレス空間リードモードにより、SPI マルチ I/O バス空間へのリードを SPI 通信に変換する際にシリアルフラッシュメモリに出力する信号を使用するシリアルフラッシュメモリのリードコマンドに合わせて変更します。

SPIBSC は、SPIBSC 制御レジスタ設定より、外部アドレス空間リードモードにてシリアルフラッシュメモリに出力する信号を変更することが可能です。

サンプルプログラムでは、SPIBSC 制御レジスタに設定する値をグローバル変数（SPIBSC 外部アドレスリード設定内容格納変数：spibsc_cfg）で変更することが可能です。spibsc_cfg の設定はユーザ定義関数（SPIBSC 外部アドレスリード設定関数：Userdef_SPIBSC_Set_Config）で行っています。

図 10.4 に SPIBSC 制御レジスタと SPIBSC 外部アドレスリード時にシリアルフラッシュメモリに出力される波形の関係を、**表 10.5** にサンプルプログラムの SPIBSC 制御レジスタ設定値を示します。

本設定例を参考に、使用するシリアルフラッシュメモリのリードコマンドに合わせて spibsc_cfg の設定を行ってください。

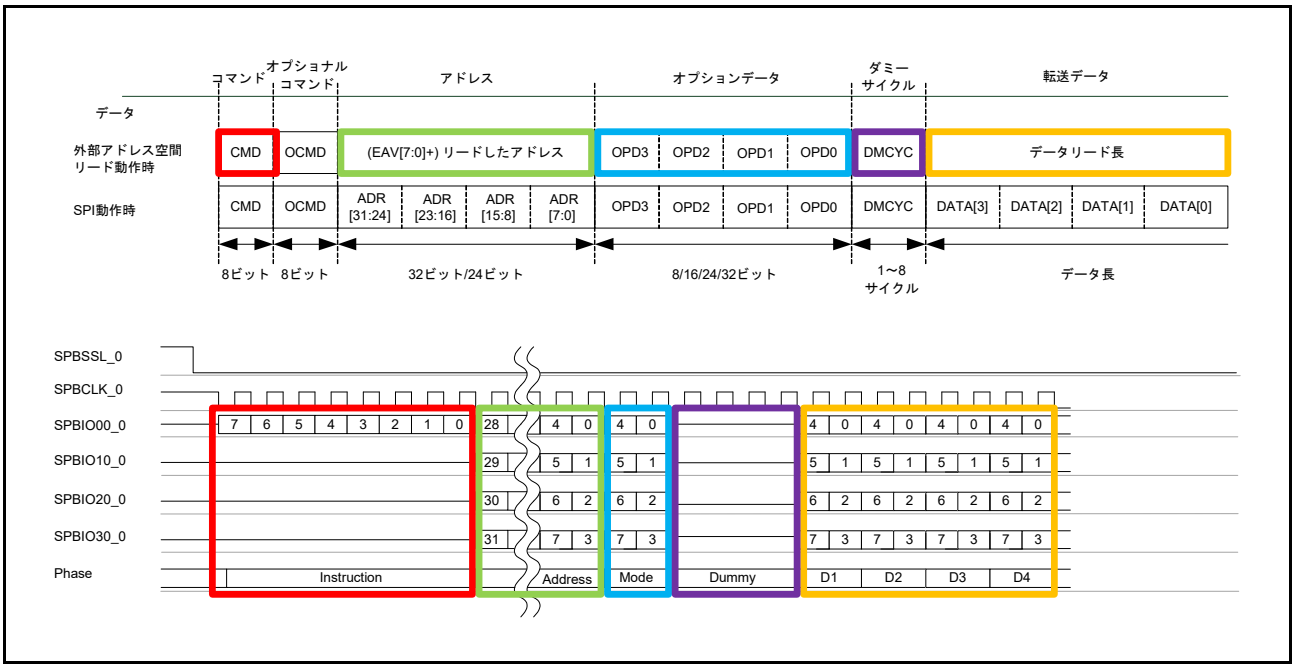


図 10.4 SPIBSC 制御レジスタと SPIBSC 外部アドレスリード時にシリアルフラッシュメモリに出力される波形の関係

表10.5　サンプルプログラムのSPIBSC制御レジスタ設定内容

SPIBSCレジスタ名		設定値	備考
DRCMR	CMD[7:0]	H’EB	Quad I/O Read Command
	OCMD[7:0]	H’00	—
DROPR	OPD3[7:0]	H’00	—
	OPD2[7:0]	H’00	—
	OPD1[7:0]	H’00	—
	OPD0[7:0]	H’00	—
DRENR	CDB[1:0]	B’00	コマンドビット幅：1ビット幅
	OCDB[3:0]	B’0000	—
	ADB[1:0]	B’10	アドレスビット幅：4ビット幅
	OPDB[1:0]	B’10	オプショナルデータビット幅：4ビット幅
	DRDB[1:0]	B’10	データリードビット幅：4ビット幅
	DME	B’1	ダミーサイクル：挿入する
	CDE	B’1	コマンド：発行する
	OCDE	B’0	オプションコマンド：発行しない
	ADE[3:0]	B’0111	アドレスイネーブル：24ビットアドレス出力
	OPDE[3:0]	B’1000	オプションナルデータ：OPD3を出力
DRDMCR	DMDB[1:0]	B’00	ダミーサイクルビット幅：1ビット幅
	DMCYC[2:0]	B’101	ダミーサイクル数：6サイクル
SPBCR	SPBR[7:0]	H’01	ビットレート：PCLKA/2
	BRDV[1:0]	B’00	

10.2.5　シリアルフラッシュメモリ内レジスタ設定

「10.2.4　リードコマンド波形の変更」で、シリアルフラッシュメモリからリードする場合、シリアルフラッシュメモリ内のレジスタを設定する必要があります。

サンプルプログラムでは、ユーザ定義関数（シリアルフラッシュメモリ内レジスタ設定関数：Userdef_SFLASH_Set_Mode）に処理を実装しており、Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）内の Status Register 内の QUAD ビットを 1（= Quad）に、Configuration Register 内の DC1（Dummy Cycle1）ビットを 1 に、DC0（Dummy Cycle0）ビットを 0 に設定しています。

なおシリアルフラッシュメモリ内のレジスタの設定には SPIBSC の SPI モードを使用します。また Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）内のレジスタを設定する場合、Write Enable Command（WREN）より、Write Enable Latch（WEL）bit を 1 に設定する必要があるため、Status Register および Configuration Register 設定前に Write Enable Command（WREN）を発行しています。サンプルプログラムではユーザ定義関数（シリアルフラッシュメモリライト許可関数：Userdef_SFLASH_Write_Enable）にて、Write Enable Command（WREN）発行処理を実装しています。

図 10.5 にサンプルプログラムのシリアルフラッシュメモリ内レジスタ設定フローを示します。

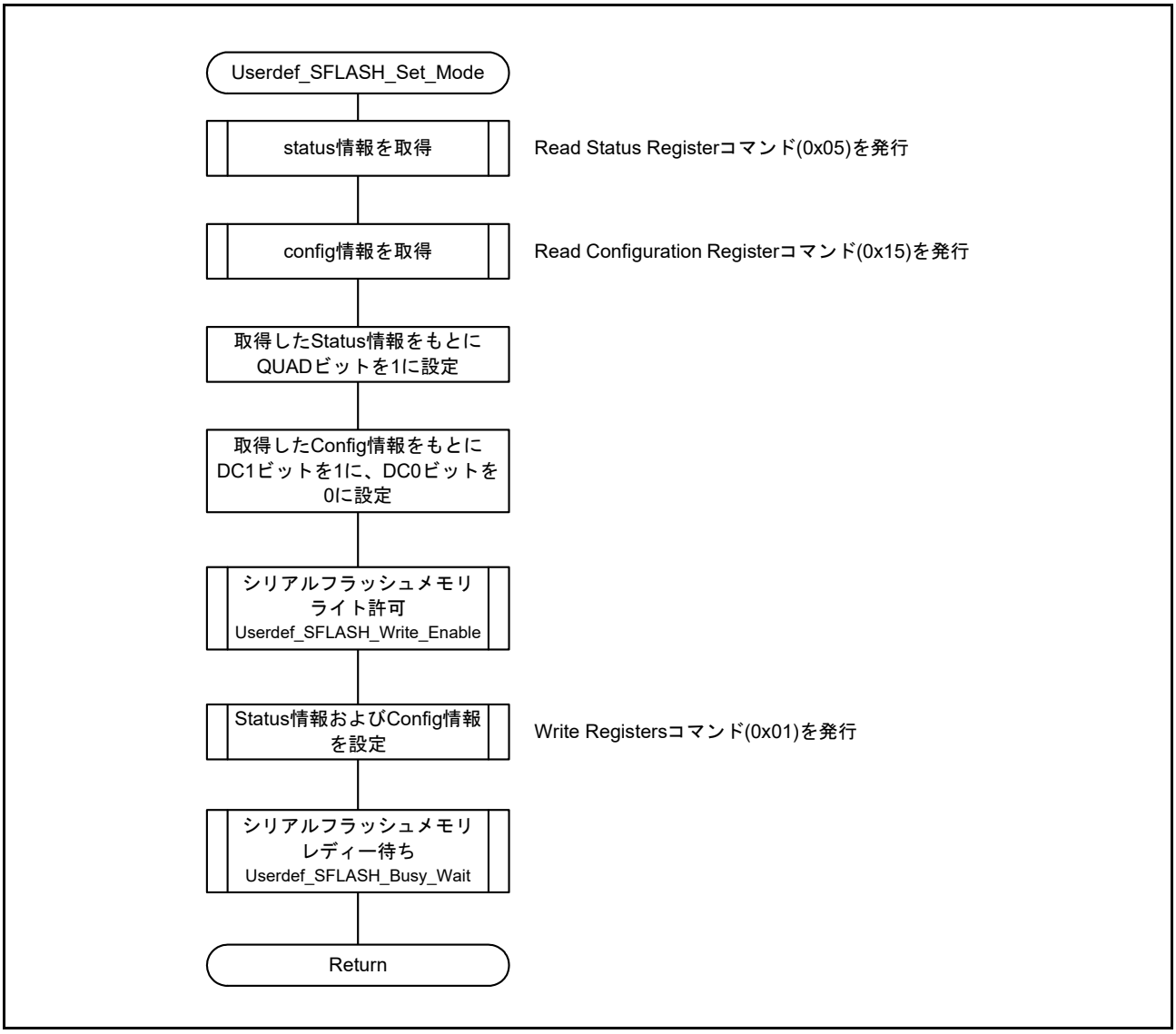


図 10.5　シリアルフラッシュメモリ内レジスタ設定フロー

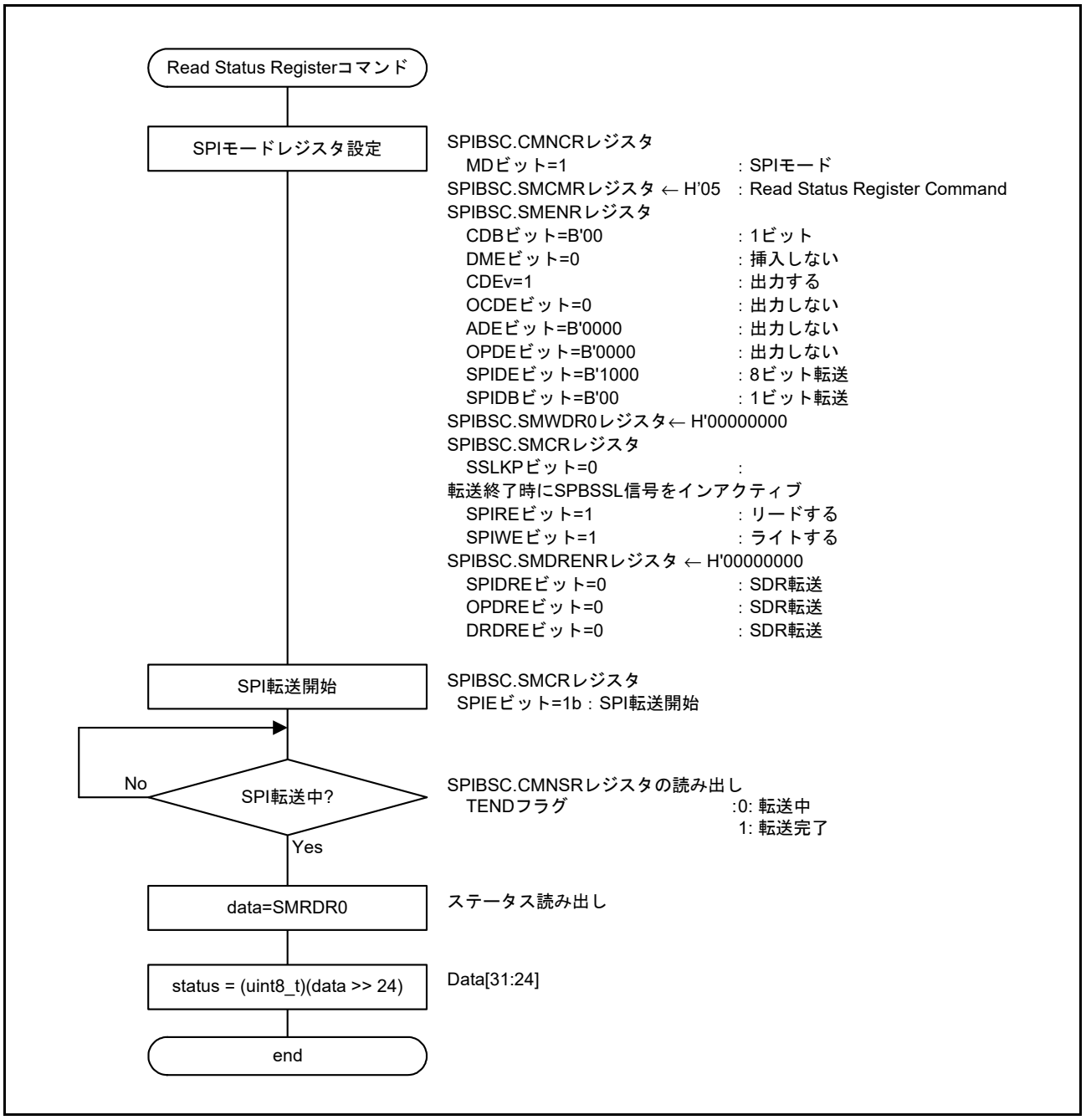


図 10.6　Read Status Register コマンドフロー

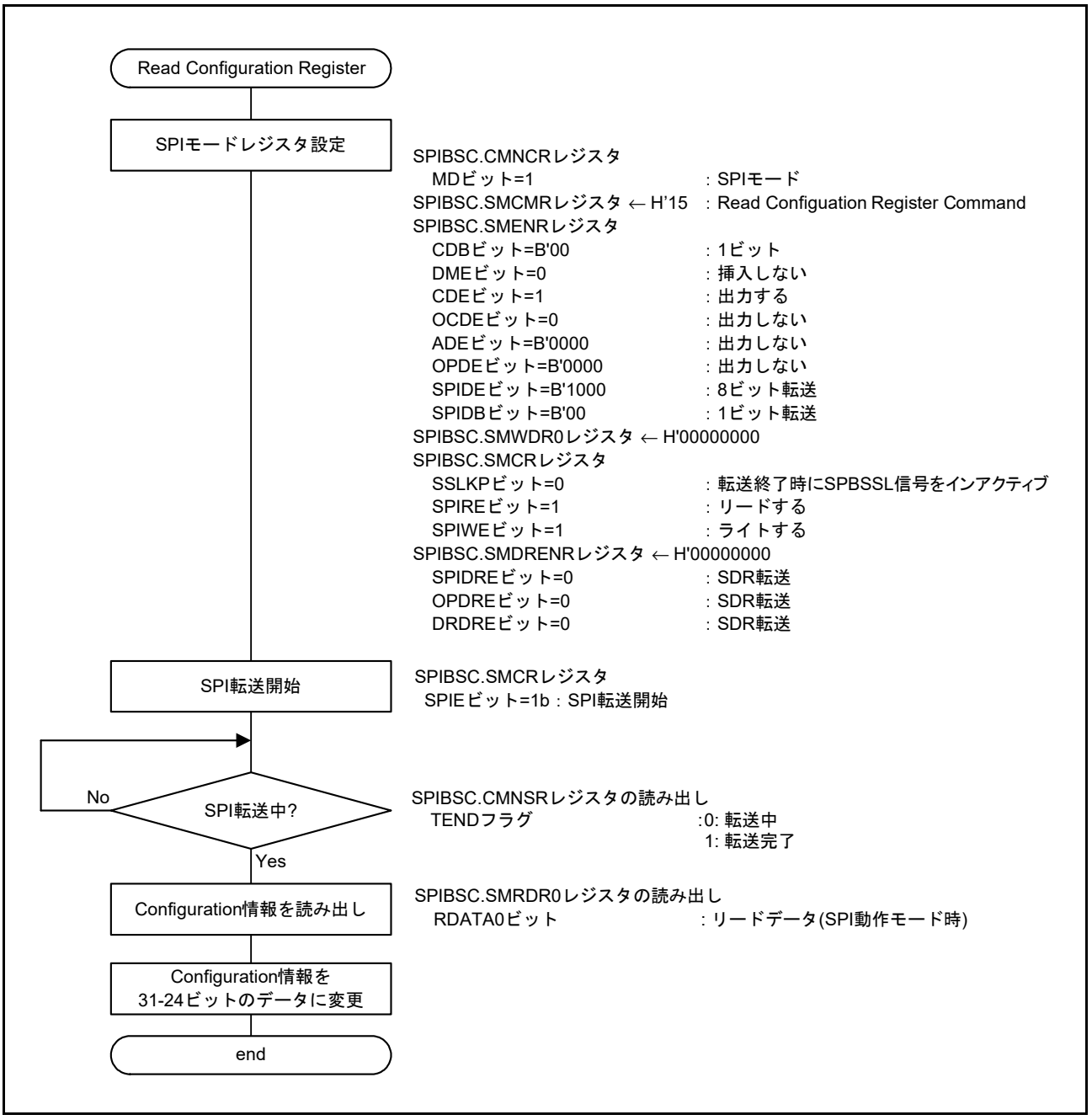


図 10.7　Read Configuration Register コマンドフロー

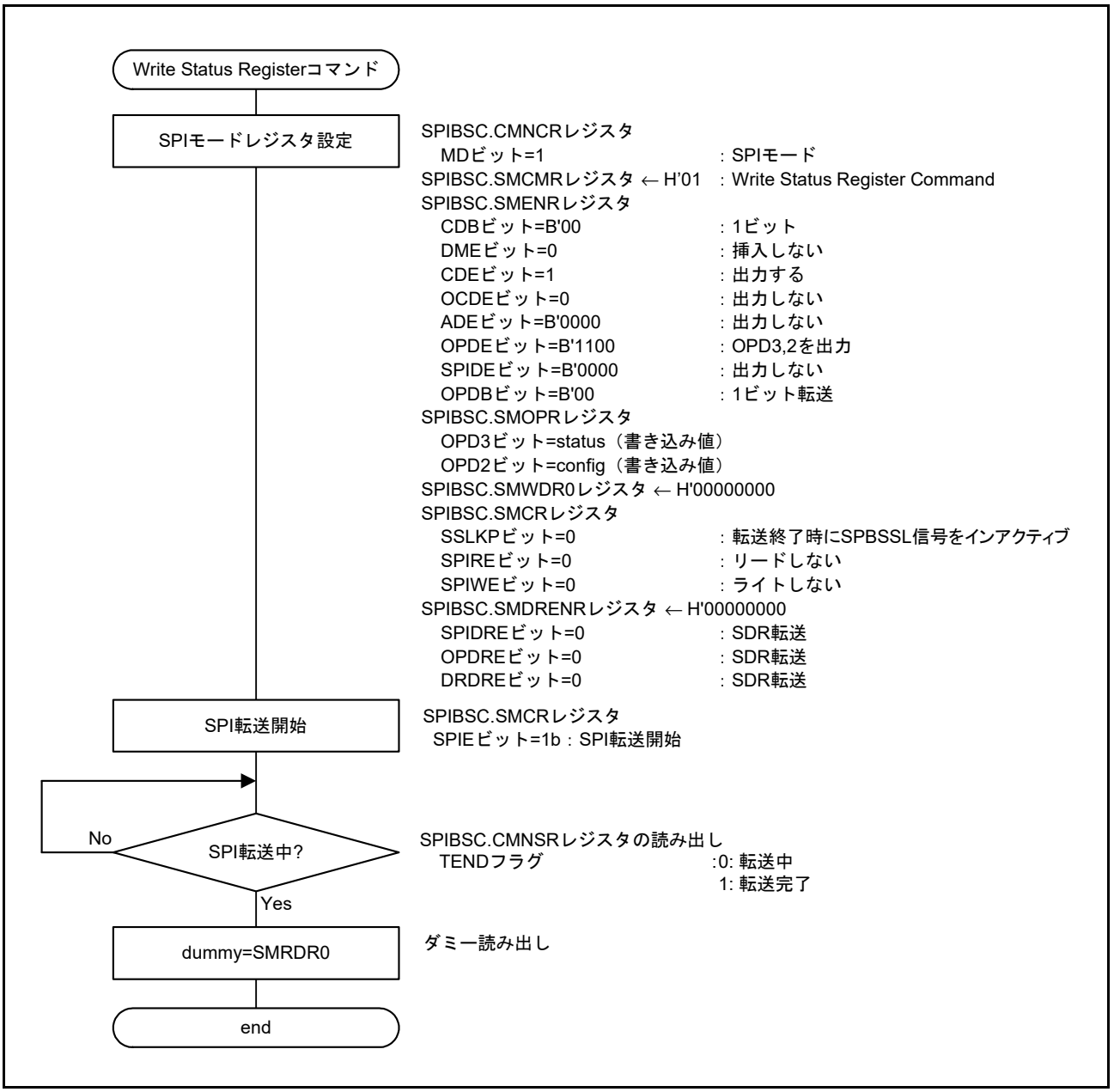


図 10.8　Write Status Register コマンドフロー

10.2.6　シリアルフラッシュメモリライト許可

「10.2.5　シリアルフラッシュメモリ内レジスタ設定」より、シリアルフラッシュメモリ内レジスタを設定する際に必要なライト許可処理をユーザ定義関数（シリアルフラッシュメモリライト許可関数：Userdef_SFLASH_Write_Enable）に実装してください。

図 10.9 にサンプルプログラムのシリアルフラッシュメモリライト許可フローを示します。

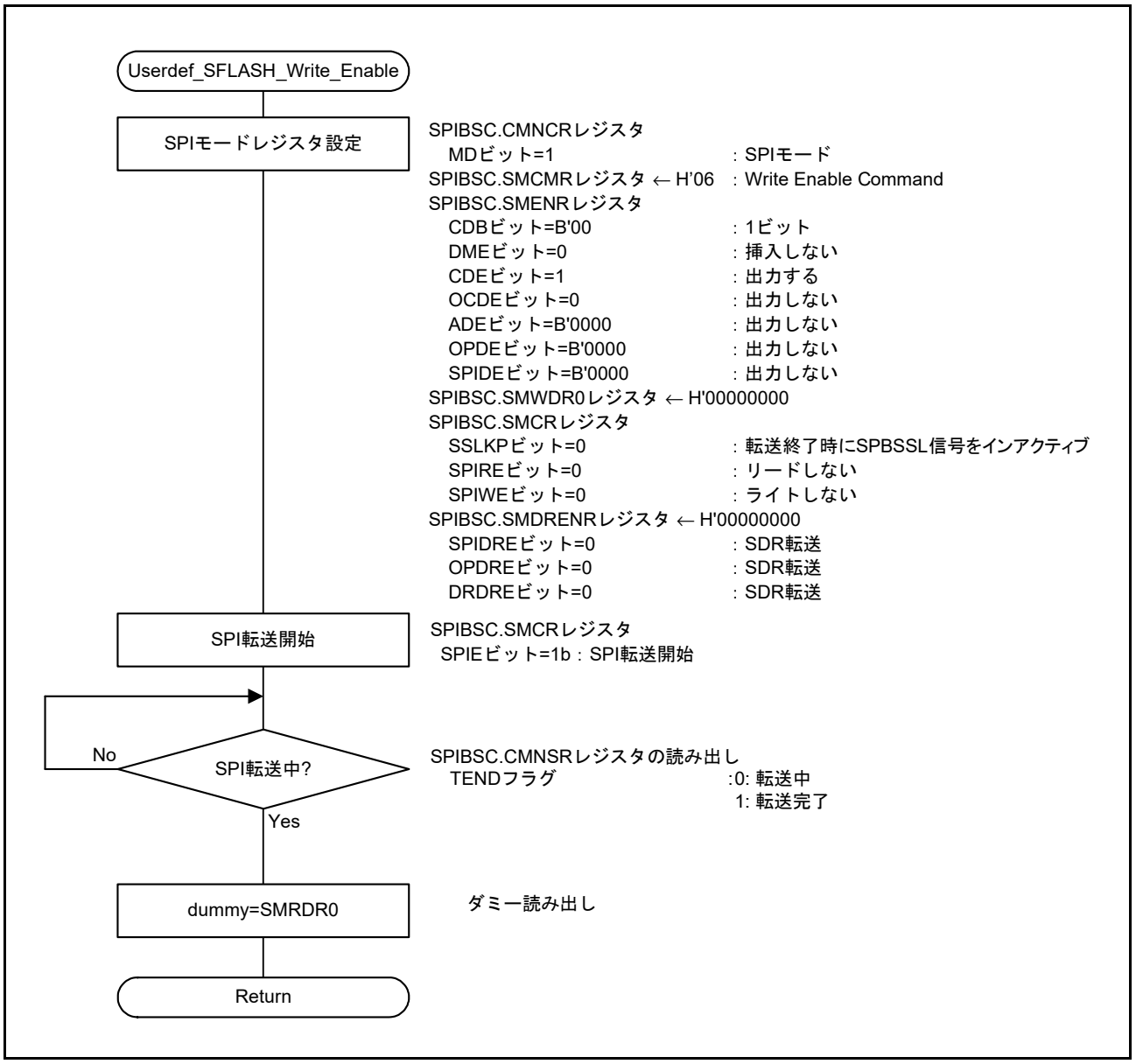


図 10.9　シリアルフラッシュメモリライト許可フロー

10.2.7　シリアルフラッシュメモリレディー待ち

シリアルフラッシュメモリに対して、書き込みコマンド（Page Program）または消去コマンド（Sector Erase）を発行した場合、シリアルフラッシュメモリはビジー状態に遷移します。ビジー状態からレディー状態への遷移を待つ処理をユーザ定義関数（シリアルフラッシュメモリレディー待ち関数：Userdef_SFLASH_Busy_Wait）に実装してください。

Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）では、レディー状態への遷移をシリアルフラッシュメモリ内のレジスタ（Status Register）にて確認することができます。

図 10.10 にサンプルプログラムのシリアルフラッシュメモリレディー待ちフローを示します。

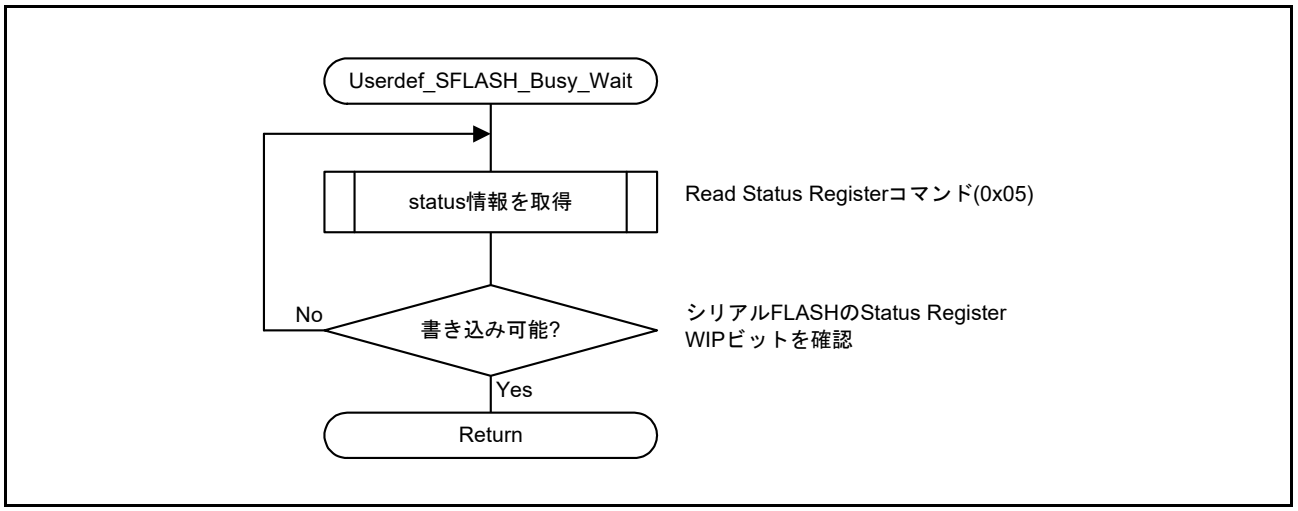


図 10.10　シリアルフラッシュメモリレディー待ちフロー

10.2.8　シリアルフラッシュメモリプロテクト解除

シリアルフラッシュメモリの仕様により、シリアルフラッシュメモリ内のデータを変更する場合、シリアルフラッシュメモリ内のレジスタを操作し、プロテクトを解除する必要があります。

Macronix 社製シリアルフラッシュメモリ（型名：MX25L51245G）では、シリアルフラッシュメモリがプロテクト状態にある場合、シリアルフラッシュメモリへの書き込みおよび消去を実行することができません。プロテクトを解除するためには、Status Register 内の Block Protection（BP3, BP2, BP1, BP0）ビットに 0 を設定する必要があります。

図 10.11 にサンプルプログラムのシリアルフラッシュメモリプロテクト解除フローを示します。

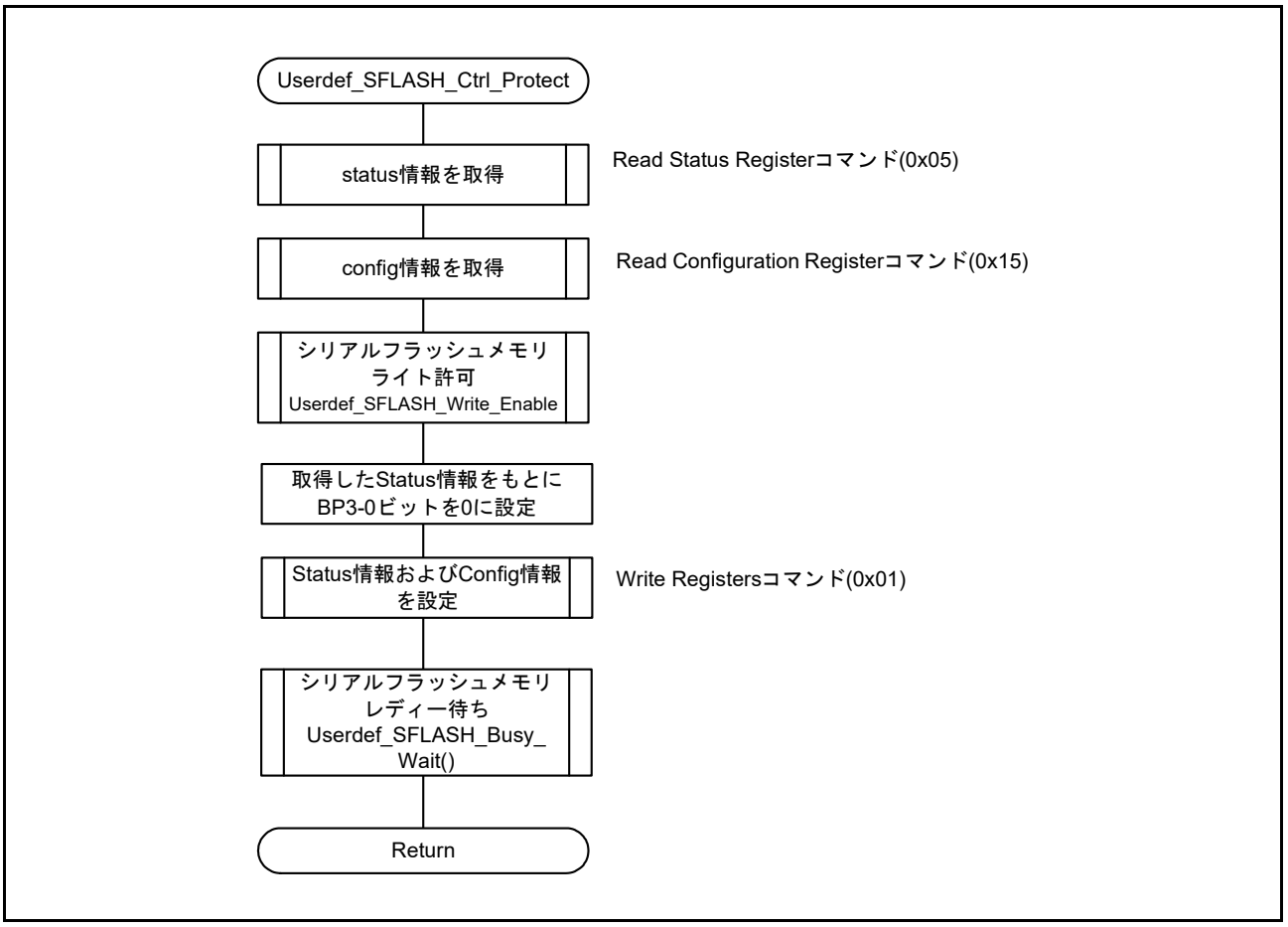


図 10.11　シリアルフラッシュメモリプロテクト解除フロー

10.2.9　シリアルフラッシュメモリ消去

サンプルプログラムでは、シリアルフラッシュメモリのセクタ消去に、消去コマンド（Sector Erase）を使用しています。

シリアルフラッシュメモリの仕様により、消去コマンドを変更する必要がある場合は、シリアルフラッシュ消去関数（R_SFLASH_EraseSector）を変更してください。

図 10.12 にサンプルプログラムのシリアルフラッシュメモリ消去フローを示します。

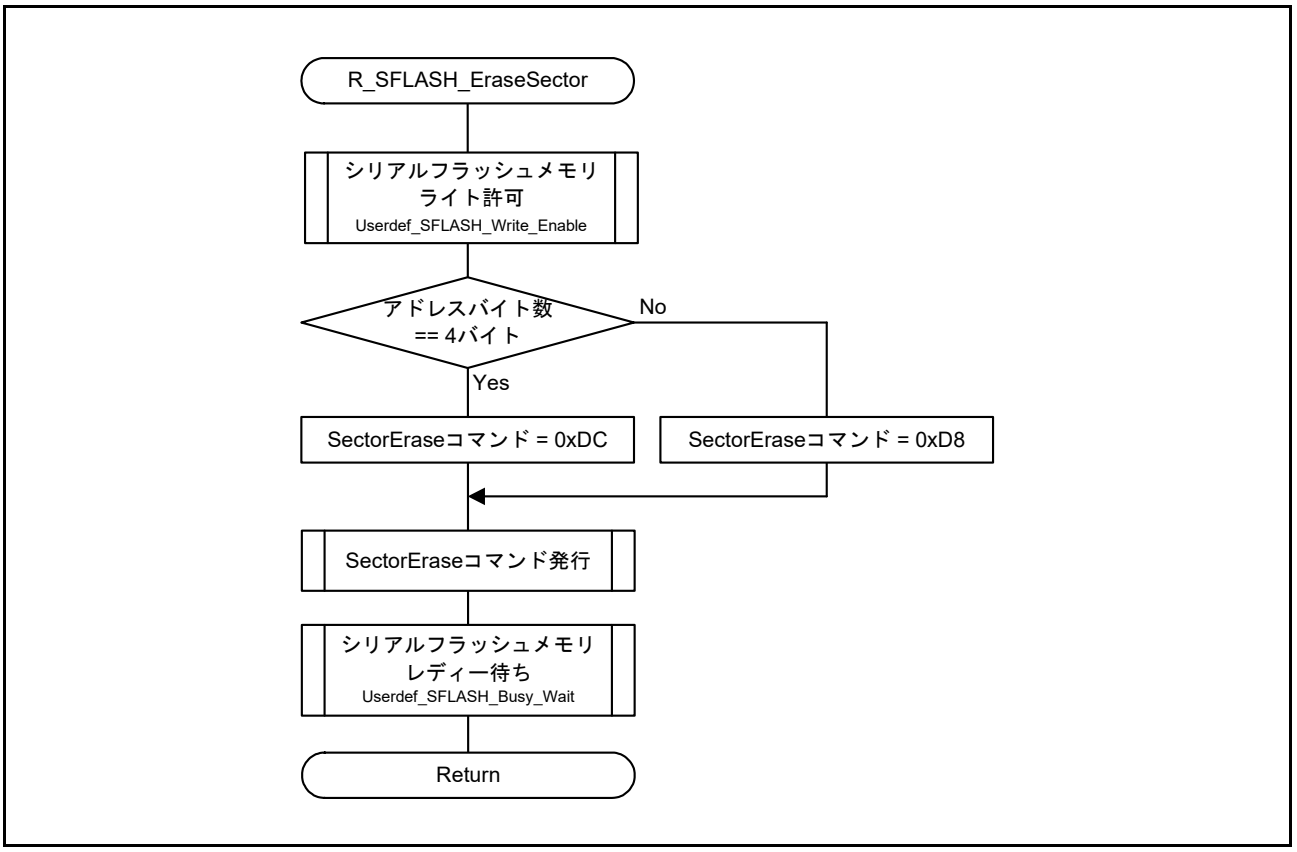


図 10.12　シリアルフラッシュメモリ消去フロー

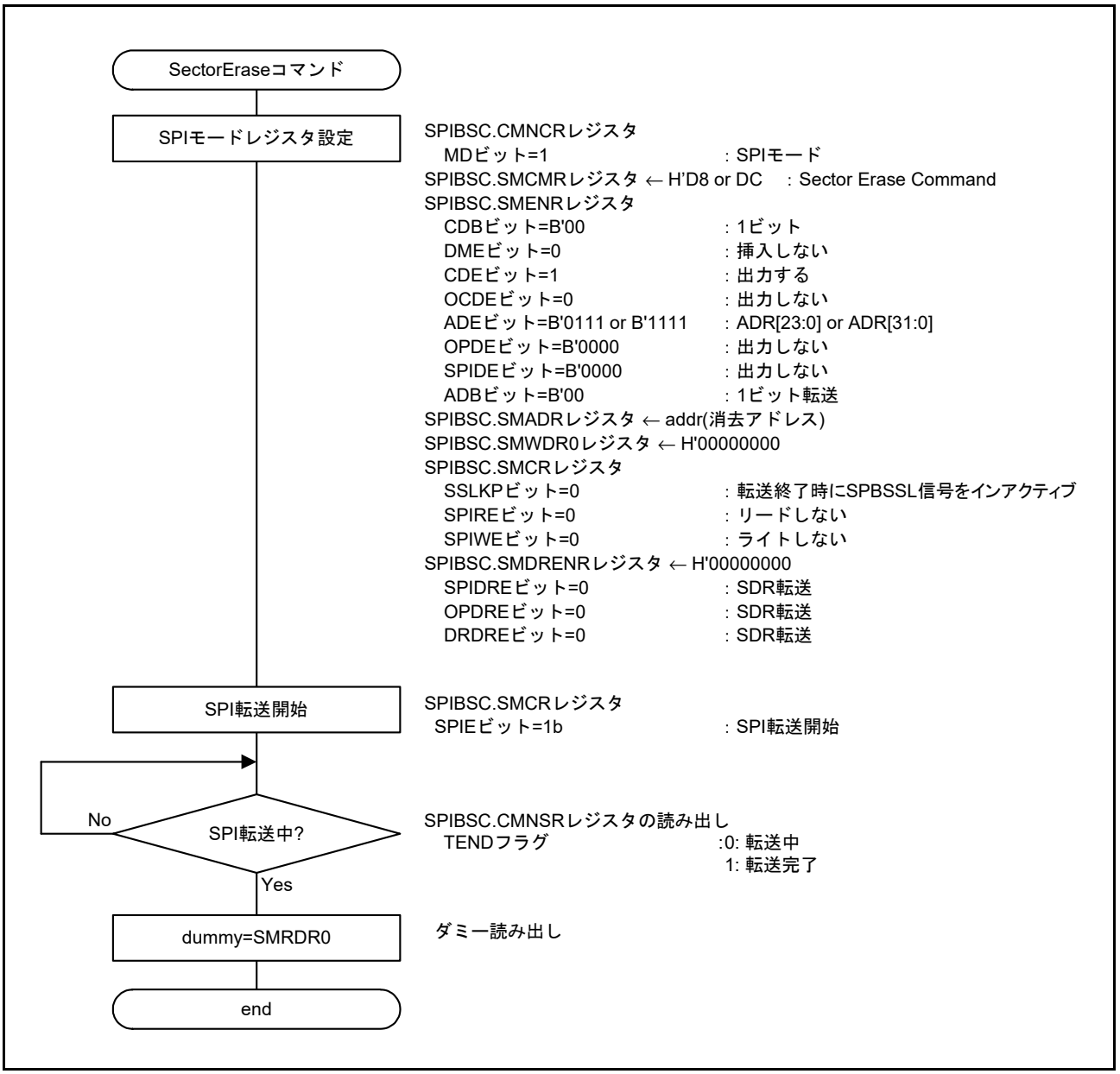


図 10.13　SectorErase コマンドフロー

10.2.10　シリアルフラッシュメモリ書き込み

サンプルプログラムでは、シリアルフラッシュメモリの書き込み去に、書き込みコマンド（Page Program）を使用しています。

シリアルフラッシュメモリの仕様により、書き込みコマンドを変更する必要がある場合は、シリアルフラッシュ書き込み関数（R_SFLASH_ByteProgram）を変更してください。

図 10.14 にサンプルプログラムのシリアルフラッシュメモリ書き込みフローを示します。

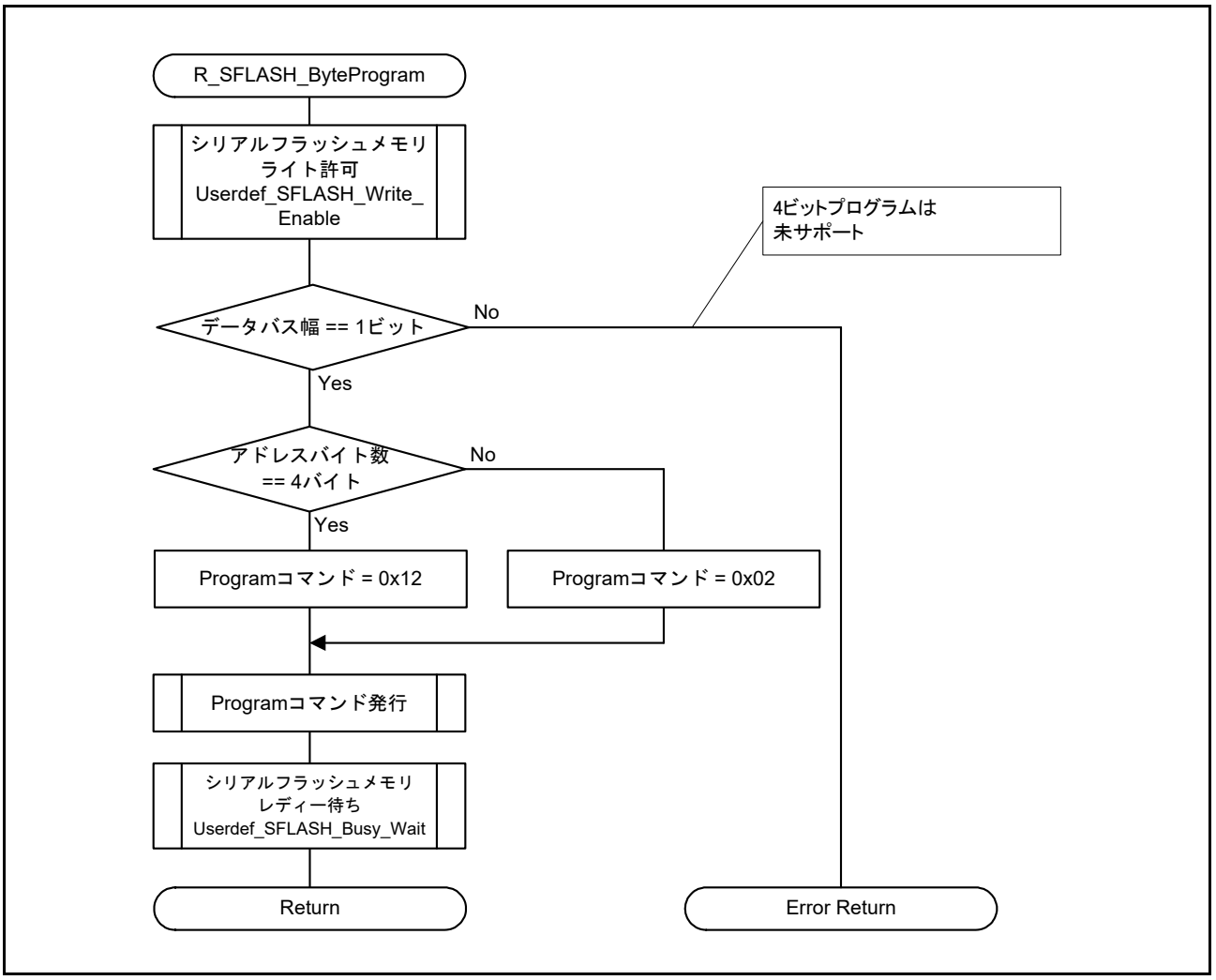


図 10.14　シリアルフラッシュメモリ書き込みフロー

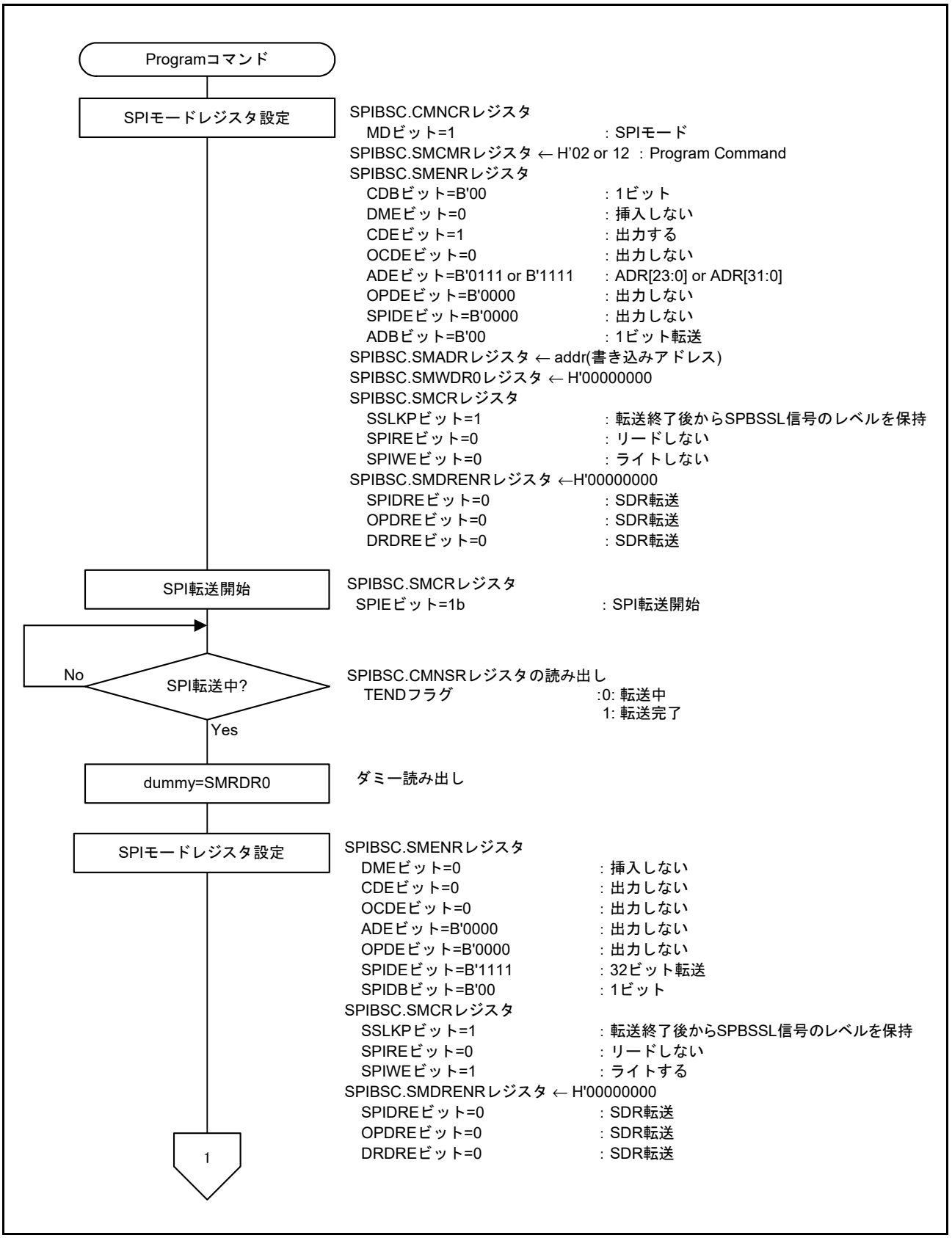


図 10.15　Program コマンドフロー (1)

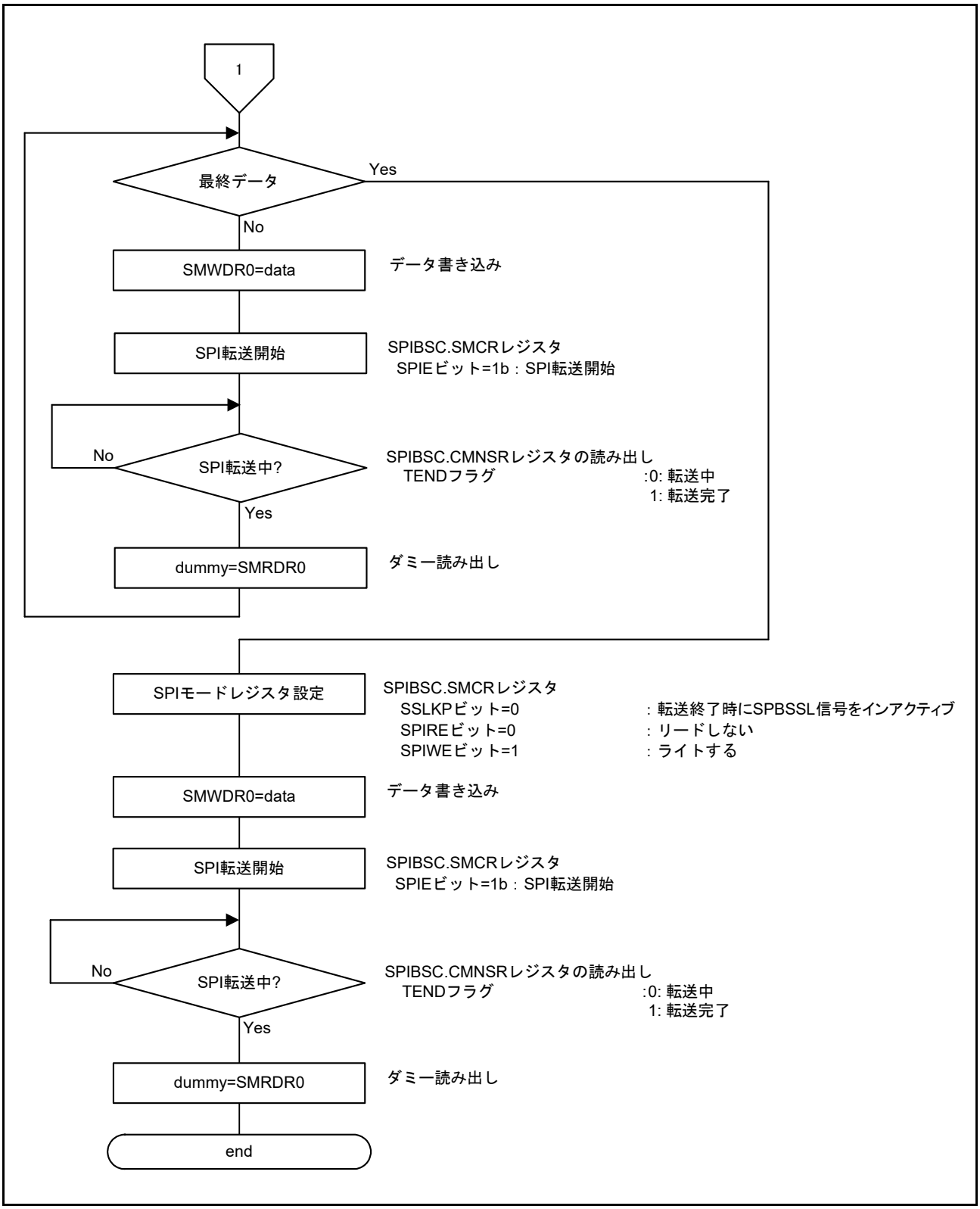


図 10.16　Program コマンドフロー (2)

10.3 R-IN Engine 搭載製品（Cortex-M3）初期設定サンプルプログラムのカスタマイズ

R-IN Engine 搭載製品　初期設定サンプルプログラムでは、アプリケーションバイナリファイルを生成する際に CM3_SECTION が生成されます。ダウンロード時には R-IN Engine 搭載製品用の処理（loop = 5）として Cortex-M3 用バイナリファイルのダウンロードも実行されます。

R-IN Engine 搭載製品　初期設定の詳細は、アプリケーションノート「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を参照してください。

表 10.6　アプリケーションバイナリファイル一覧

ディレクトリ	バイナリファイル	説明
RZ_T_sflash_sample.bin	CONST_LOADER_TABLE	アプリケーション(1)（ローダ用パラメータ情報） バイナリファイル
	LOADER_RESET_HANDLER	アプリケーション(2)（ローダプログラム） バイナリファイル
	LOADER_IN_ROOT	アプリケーション(3)（ローダプログラム） バイナリファイル
	INIT	アプリケーション(4)（ユーザプログラム） バイナリファイル
	CM3_SECTION	アプリケーション(5)（ユーザプログラム） バイナリファイル（Cortex-M3 プログラム）

```
RZ/T1 CPU Board S-Flash Programming Sample. Ver.1.00
Copyright (C) 2015 Renesas Electronics Corporation. All rights reserved.

Initializing Flash...
Start to load Binary Data to Flash Memory.
loop=1, file=CONST_LOADER_TABLE, flash address=0x30000000.
Calculating Data Size...
Data Size is 76
Programing Flash...
Calcurating Checksum of Loader Parameter.
Verifying Flash...
loop=1, Flash Programming Success!!
loop=2, file=LOADER_RESET_HANDLER, flash address=0x30000200.
Calculating Data Size...
Data Size is 11812
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x30006200.
Calculating Data Size...
Data Size is 236
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x30010000.
Calculating Data Size...
Data Size is 2592
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, file=CM3_SECTION, flash address=0x30110000.
Calculating Data Size...
Data Size is 1296
Programing Flash...
Verifying Flash...
loop=5, Flash Programming Success!!

 finish
Flash Programming Complete
```

図 10.17　アプリケーションコンソールに出力されるメッセージ

11. サンプルプログラム

サンプルプログラムは、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。

12. 参考ドキュメント

- ユーザーズマニュアル：ハードウェア
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （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。）

  RZ/T1 評価ボード RTK7910022C00000BR ユーザーズマニュアル
  （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。）

  ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
  （最新版を ARM® ホームページから入手してください。）

  ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
  （最新版を ARM® 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
  ARM ソフトウェア開発ツール（ARM Compiler toolchain、ARM DS-5 等）に関しては、ARM® ホームページから入手してください。
  （最新版を ARM® ホームページから入手してください。）

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	ARM® Development Studio 5（DS-5™）のセミホスティング機能を使用した シリアルフラッシュメモリへのダウンロード例 アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.10.21	—	初版発行
1.10	2016.08.11	全般	アプリケーションノート「RZ/T1 グループ　デュアルコア制御」をアプリケーションノート「RZ/T1 グループ　R-IN Engine搭載製品　初期設定」に変更
1.20	2017.09.15	2. 動作確認条件	
		5	表2.1 動作確認条件　ARM®製DS-5のVersion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. 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、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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。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、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、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損害が生じても、当社は一切その責任を負いません。
6. 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。
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。
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。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. 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、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1　2016.11)

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 エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：https://www.renesas.com/contact/