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NOR型フラッシュメモリへのダウンロード例

R01AN2950JJ0120
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（CS0 空間）に配置された NOR 型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

NOR 型フラッシュメモリは、プログラム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。NOR 型フラッシュメモリへの書き込みは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アプリケーションバイナリファイルを参照して、NOR 型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 .　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
バスステートコントローラ（BSC）	• 外部アドレス空間（CS0空間）に接続されたNOR型フラッシュメモリにアクセスするための信号を生成します。 • 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（RTK7910022C00000BR）に実装されているNOR型フラッシュメモリにアクセスするための設定を行っています。注1
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。

注1.　RZ/T1評価ボードに実装されているNOR型フラッシュメモリの型名は表2.1を参照してください。

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, CKIO = 50MHz
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	16ビットバスブートモード（NOR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）	CS0空間（16ビットバス幅）に配置されたNOR型フラッシュメモリ • メーカ名：Macronix International Co.. • 型名：MX29GL512FLT2I-10Q

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。

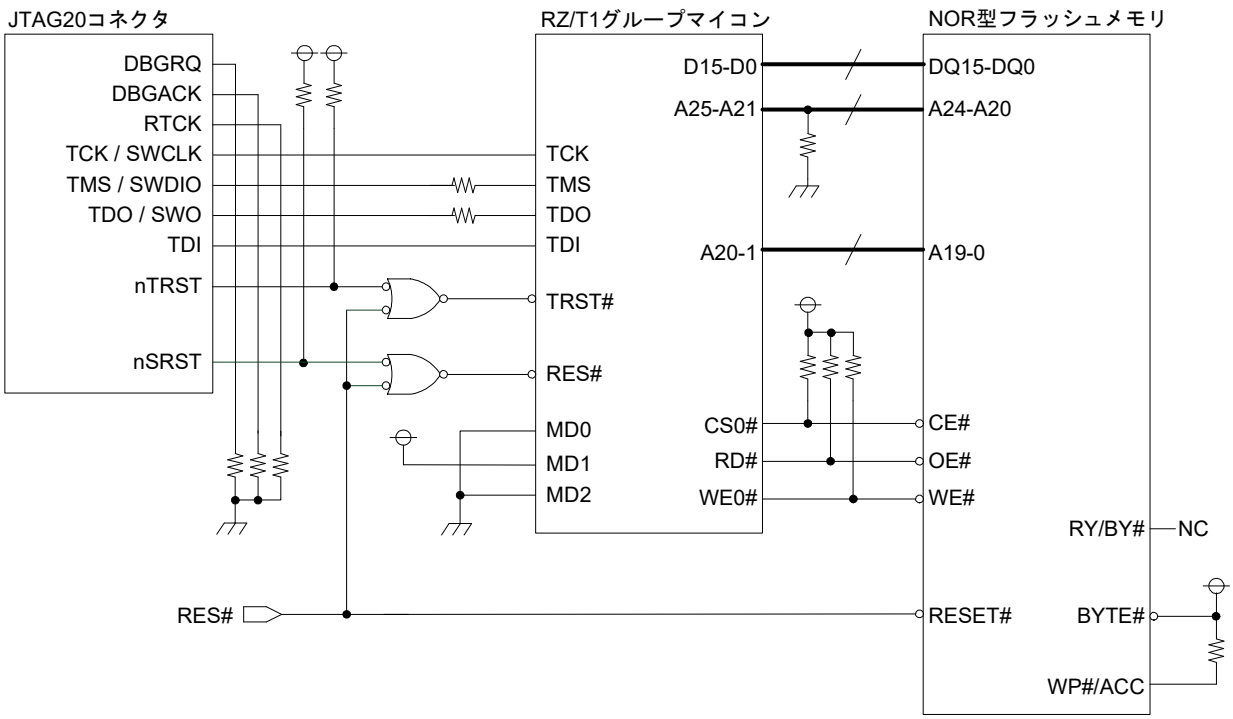
表4.1 使用端子と機能

端子名	入出力	内容
A25～A1	出力	NOR型フラッシュメモリへのアドレス信号出力
D15～D0	入出力	NOR型フラッシュメモリのデータ信号入出力
CS0#	出力	CS0空間に接続されたNOR型フラッシュメモリへのデバイス選択信号出力
RD#	出力	NOR型フラッシュメモリへのリード制御信号出力
WE0#	出力	NOR型フラッシュメモリへのライトイネーブル制御信号出力
MD2、MD1、MD0	入力	ブートモードの選択（16ビットバスブートモードに設定） MD2：”L” MD1：”H” 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ウ）を示す記号です。

図 4.1 接続例

5. NOR 型フラッシュメモリへのダウンロードの概要

この章では、NOR 型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 NOR 型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 NOR 型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。

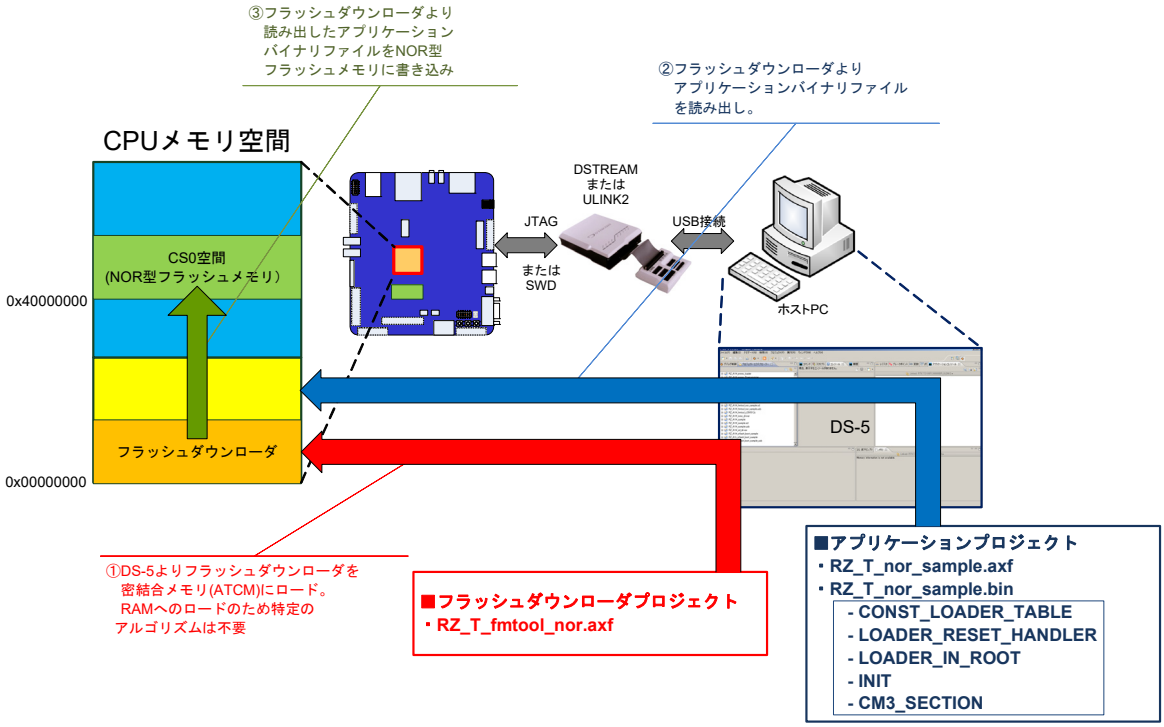
表5.1 NOR 型フラッシュメモリへのダウンロード関連の用語

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、アプリケーションプログラムをNOR型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にて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、NOR型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、NOR型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 NOR 型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 NOR 型フラッシュメモリの書き込み処理を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。NOR 型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。

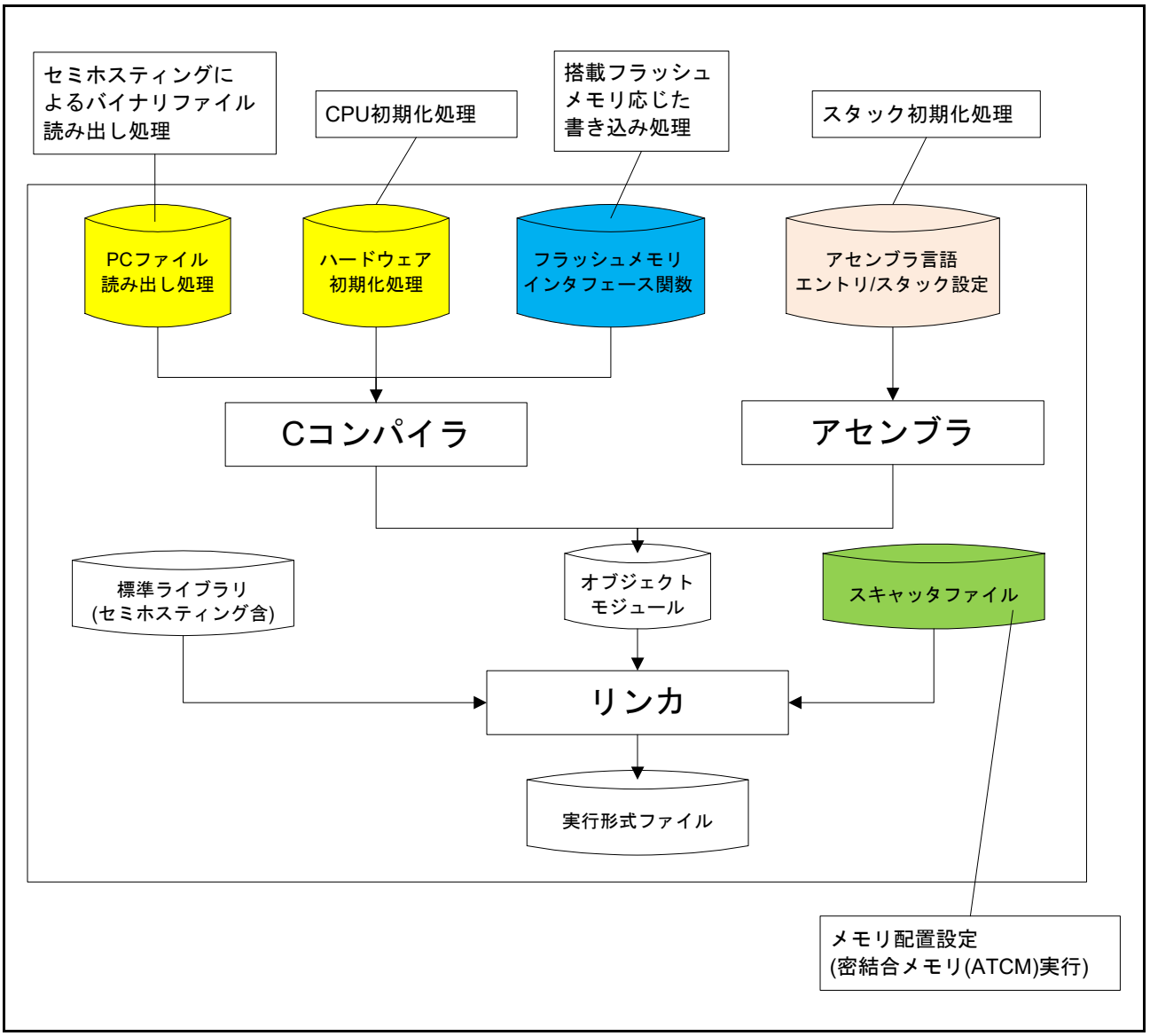


図 5.2 フラッシュダウンローダの開発フロー

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。

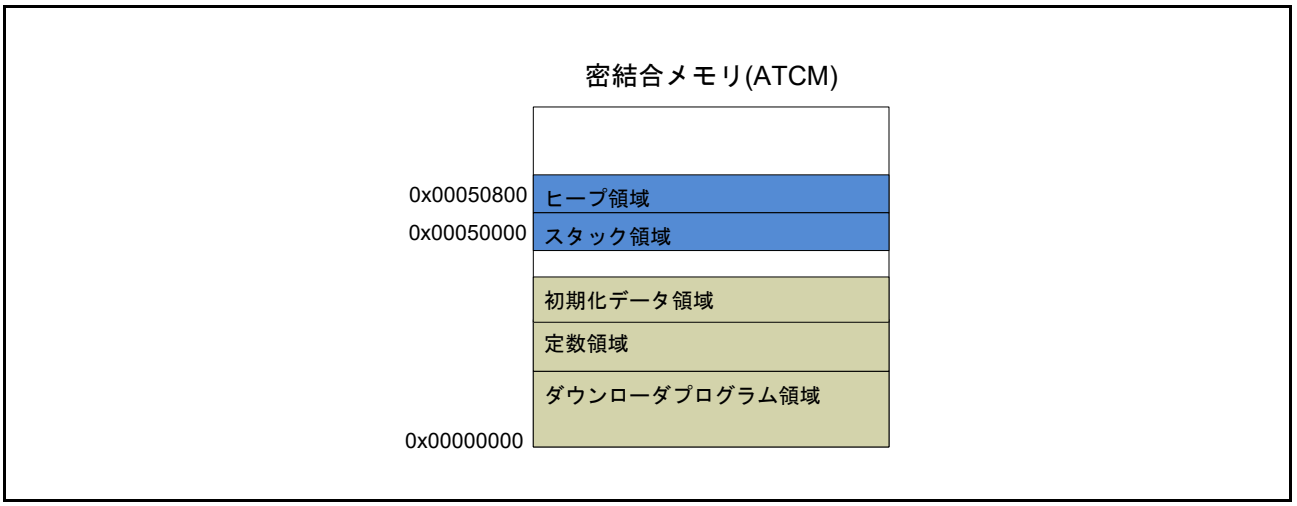


図 5.3 フラッシュダウンローダのメモリ配置

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 NOR 型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 NOR 型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	NOR型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フラッシュメモリインタフェース関数のカスタマイズ」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 NOR 型フラッシュメモリにアプリケーションプログラム（RZ_T_nor_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。

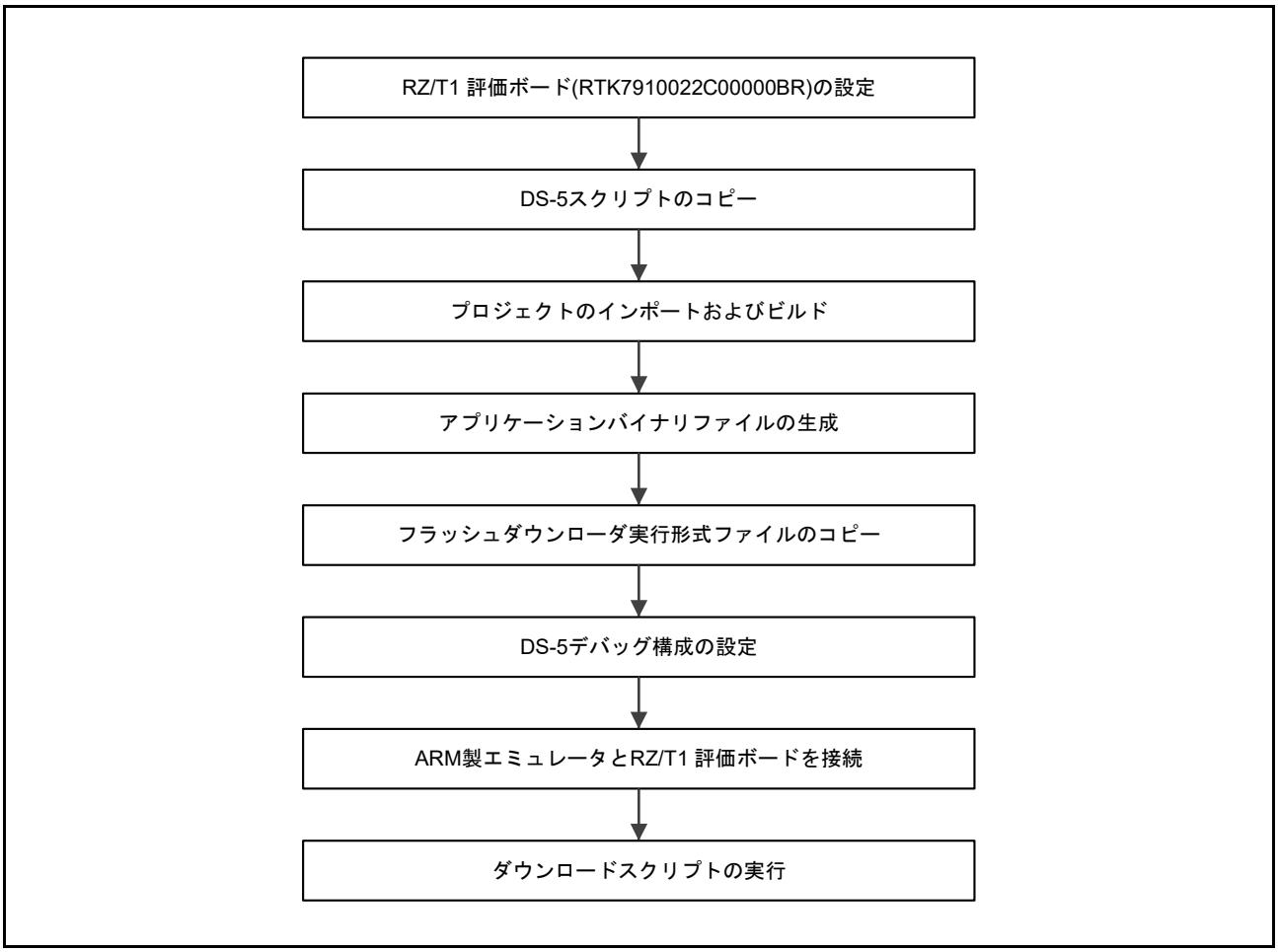


図 6.1　ダウンロードの概略手順

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	OFF	MD1 = H 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_nor_sample）ディレクトリの直下にディレクトリ [script_nor] 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_nor_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（CS0空間）に配置されたNOR型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_nor	このプロジェクトで簡易版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簡易版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_nor.axf
RZ_T_nor_sample	このプロジェクトでユーザアプリケーションをビルドします。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_nor_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 Pronpt] 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_nor_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アプリケーション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_nor_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― 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_nor_sample.bin RZ_T_nor_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

アプリケーションバイナリファイル		説明
RZ_T_nor_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_nor_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_nor.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（¥RZ_T_fmtool_nor¥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

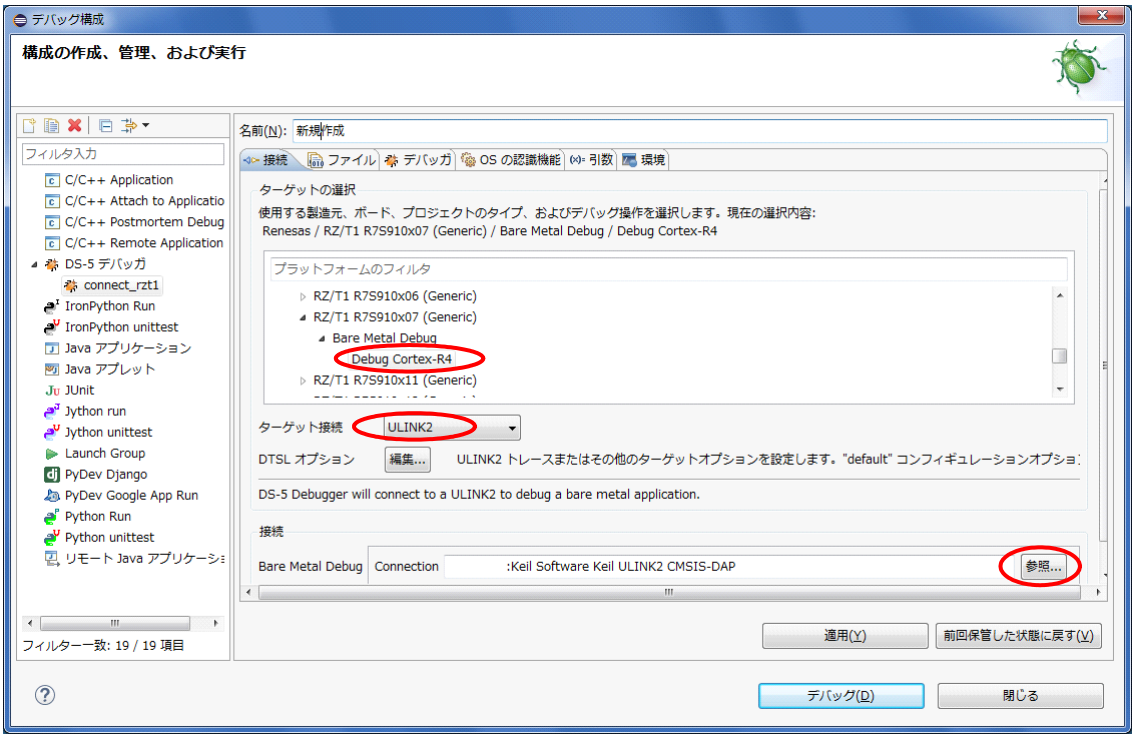
以下の手順で、DS-5 デバッグ構成を設定します。DS-5 デバッグ構成の設定で、DS-5 と RZ/T1 評価ボードを接続注1 した際に RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）が実行されるように設定します。なお、RZ/T1 評価ボード初期化スクリプト（init_RZ-T.ds）の処理内容については、**「9.2　RZ/T1 評価ボード初期化スクリプト」**を参照してください。

【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] タブにて、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。（図 6.4 を参照）
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_nor_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_nor_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_nor_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_nor_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_nor_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 NOR-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=0x40000000.
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=0x40000200.
Calculating Data Size...
Data Size is 3288
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x40006200.
Calculating Data Size...
Data Size is 192
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x40020000.
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 定数一覧

表 7.2 ～表 7.5 にサンプルプログラムで使用する定数を示します。

表7.2 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
FM_TOOL_OK	(0)	成功
FM_TOOL_E_ERASE	(-1)	セクタ消去エラー
FM_TOOL_E_WRITE	(-2)	書き込みエラー
FM_TOOL_E_VERIFY	(-3)	ベリファイエラー
FM_TYPE_BYTE	(0x4220)	CS0空間バス幅 = 8ビット
FM_TYPE_WORD	(0x5720)	CS0空間バス幅 = 16ビット
FM_TYPE_LONG	(0x4C20)	CS0空間バス幅 = 32ビット
FM_ACCESS_SIZE	(FM_TYPE_WORD)	サンプルプログラムのCS0空間バス幅 FM_TYPE_BYTE、FM_TYPE_WORD、FM_TYPE_LONGのいずれか注1を設定します。
FM_UNIFORM	(0)	NOR型フラッシュメモリのセクタタイプ = ユニフォームタイプ
FM_TOP_BOOT	(1)	NOR型フラッシュメモリのセクタタイプ = トップタイプ
FM_BOTTOM_BOOT	(2)	NOR型フラッシュメモリのセクタタイプ = ボトムタイプ
FM_DUAL_BOOT	(3)	NOR型フラッシュメモリのセクタタイプ = デュアルタイプ
FM_BOOT_TYPE	(FM_UNIFORM)	サンプルプログラムのセクタタイプ FM_UNIFORM、FM_TOP_BOOT、FM_BOTTOM_BOOT、FM_DUAL_BOOTのいずれかを設定します。

注1. サンプルプログラムはFM_TYPE_BYTEに対応していません。

表7.3　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
FM_CS0_NON_CACHE_START	(0x40000000uL)	CS0空間の先頭アドレス
FM_B_BOOT_SECTOR_START	(0x00000000uL)	ボトムブートタイプセクタの先頭アドレス 注.　サンプルプログラムはユニフォームタイプなので、(0x00000000uL)を設定しています。
FM_B_BOOT_SECTOR_SIZE	(0x00000000uL)	ボトムブートタイプセクタのセクタサイズ 注.　サンプルプログラムはユニフォームタイプなので、(0x00000000uL)を設定しています。
FM_B_BOOT_SECTOR_NUM	(0)	ボトムブートタイプセクタのセクタ数 注.　サンプルプログラムはユニフォームタイプなので、(0)を設定しています。
FM_NORMAL_SECTOR_START	(0x00000000uL)	ノーマルタイプセクタの先頭アドレス
FM_NORMAL_SECTOR_SIZE	(0x00020000uL)	ノーマルタイプセクタのセクタサイズ
FM_NORMAL_BOOT_SECTOR_NUM	(512)	ノーマルタイプセクタのセクタ数
FM_T_BOOT_SECTOR_START	(0x00000000uL)	トップブートタイプセクタの先頭アドレス 注.　サンプルプログラムはユニフォームタイプなので、(0x00000000uL)を設定しています。
FM_T_BOOT_SECTOR_SIZE	(0x00000000uL)	トップブートタイプセクタのセクタサイズ 注.　サンプルプログラムはユニフォームタイプなので、(0x00000000uL)を設定しています。
FM_T_BOOT_SECTOR_NUM	(0)	トップブートタイプセクタのセクタ数 注.　サンプルプログラムはユニフォームタイプなので、(0)を設定しています。
FM_END_ADDRESS	(0x03FFFFFEuL)	NOR型フラッシュメモリの最終アドレス注1

注1.　CS0空間のバス幅が16ビットなので、16ビットアライメントを取ったアドレスを設定します。

表7.4　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
FM_CMD_S_ERASE_ADDR_1	(FM_CS0_NON_CACHE_START | 0x0AAA)	セクタ消去コマンド発行シーケンスの第一サイクルで発行するアドレス注1
FM_CMD_S_ERASE_ADDR_2	(FM_CS0_NON_CACHE_START | 0x0554)	セクタ消去コマンド発行シーケンスの第二サイクルで発行するアドレス注1
FM_CMD_S_ERASE_ADDR_3	(FM_CS0_NON_CACHE_START | 0x0AAA)	セクタ消去コマンド発行シーケンスの第三サイクルで発行するアドレス注1
FM_CMD_S_ERASE_ADDR_4	(FM_CS0_NON_CACHE_START | 0x0AAA)	セクタ消去コマンド発行シーケンスの第四サイクルで発行するアドレス注1
FM_CMD_S_ERASE_ADDR_5	(FM_CS0_NON_CACHE_START | 0x0554)	セクタ消去コマンド発行シーケンスの第五サイクルで発行するアドレス注1
FM_CMD_PROGRAM_ADDR_1	(FM_CS0_NON_CACHE_START | 0x0AAA)	書き込みコマンド発行シーケンスの第一サイクルで発行するアドレス注1
FM_CMD_PROGRAM_ADDR_2	(FM_CS0_NON_CACHE_START | 0x0554)	書き込みコマンド発行シーケンスの第二サイクルで発行するアドレス注1
FM_CMD_PROGRAM_ADDR_3	(FM_CS0_NON_CACHE_START | 0x0AAA)	書き込みコマンド発行シーケンスの第三サイクルで発行するアドレス注1

注1.　CS0空間のバス幅が16ビットなので、16ビットアライメントを取ったアドレスを設定します。

表7.5　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
FM_CMD_RESET	(0x00F0)	リセットコマンドを設定します。
FM_CMD_S_ERASE_DATA_1	(0x00AA)	セクタ消去コマンド発行シーケンスの第一サイクルで発行するコマンド
FM_CMD_S_ERASE_DATA_2	(0x0055)	セクタ消去コマンド発行シーケンスの第二サイクルで発行するコマンド
FM_CMD_S_ERASE_DATA_3	(0x0080)	セクタ消去コマンド発行シーケンスの第三サイクルで発行するコマンド
FM_CMD_S_ERASE_DATA_4	(0x00AA)	セクタ消去コマンド発行シーケンスの第四サイクルで発行するコマンド
FM_CMD_S_ERASE_DATA_5	(0x0055)	セクタ消去コマンド発行シーケンスの第五サイクルで発行するコマンド
FM_CMD_SECTOR_ERASE	(0x0030)	セクタ消去コマンド発行シーケンスの第六サイクルで発行するコマンド
FM_CMD_PROGRAM_DATA_1	(0x00AA)	書き込みコマンド発行シーケンスの第一サイクルで発行するコマンド
FM_CMD_PROGRAM_DATA_2	(0x0055)	書き込みコマンド発行シーケンスの第二サイクルで発行するコマンド
FM_CMD_PROGRAM_DATA_3	(0x00A0)	書き込みコマンド発行シーケンスの第三サイクルで発行するコマンド
FM_CHK_DQ7	(0x0080)	DQ7：DATA#pollingビットのマスク値
FM_CHK_DQ6	(0x0040)	DQ6：toggleビットのマスク値
FM_CHK_DQ5	(0x0020)	DQ5：timing limit excessビットのマスク値

7.3　変数一覧

表 7.6 に static 型変数を示します。

表7.6　static型変数

型	変数名	内容	使用関数
static uint8_t	fmtool_pre_erase_sctno[];	セクタ消去フラグ NOR型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。 初期化インタフェース関数実行時、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。	flash_init flash_write

7.4 フラッシュメモリインタフェース関数一覧

表 7.7 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7.7 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（CS0空間）に接続されているNOR型フラッシュ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（CS0空間）に接続されているNOR型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。

7.5 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
ヘッダ	”flash.h”
宣　言	int32_t flash_init(void);
説　明	RZ/T1 の外部バス（CS0 空間）に接続されている NOR 型フラッシュ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。 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	
ヘッダ	”flash.h”	
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（CS0 空間）に接続されている NOR 型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。 初期化インタフェース関数実行後、引数 fm_adrs に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	書き込み先アドレス
	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（バイト単位）
リターン値	フラッシュメモリへの書き込み結果をリターン値として設定してください。 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エラー	

7.6 フラッシュインタフェース関数のフロー

7.6.1 初期化インタフェース関数

図 7.1 に初期化インタフェース関数のフローチャートを示します。

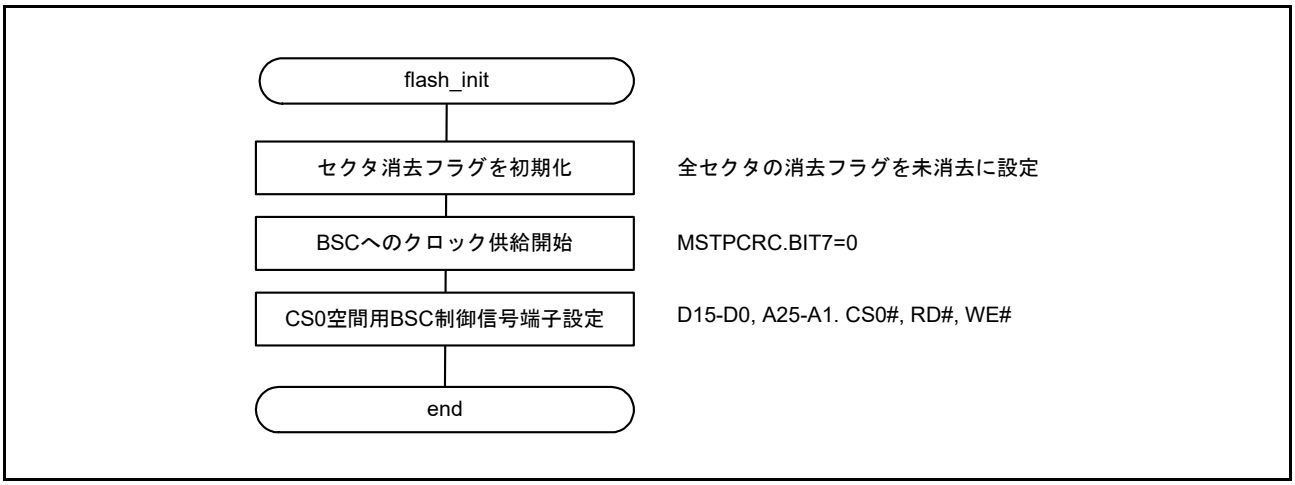


図 7.1 初期化インタフェース関数

7.6.2 書き込みインタフェース関数

図 7.2 に書き込みインタフェース関数のフローチャートを示します。

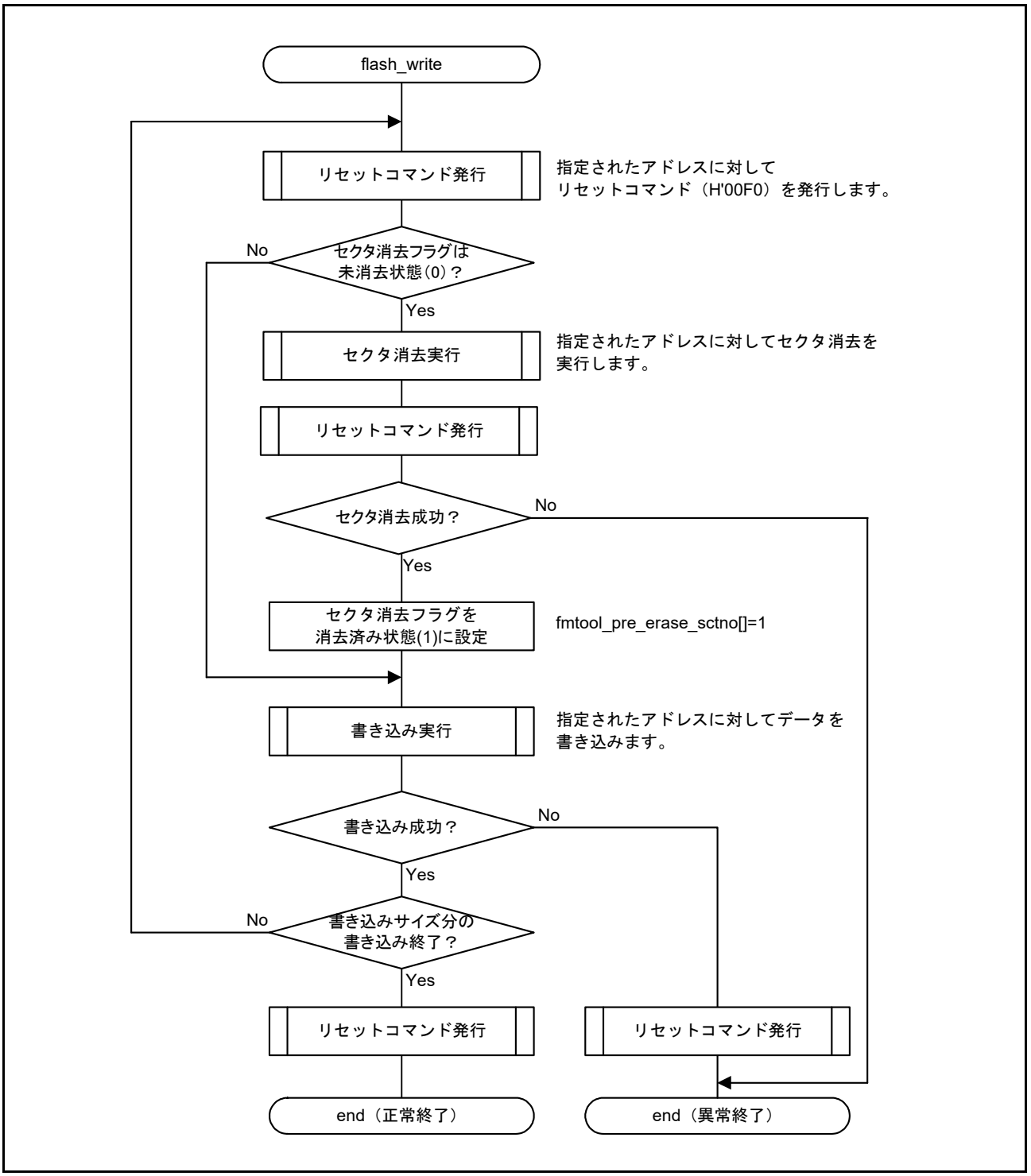


図 7.2 書き込みインタフェース関数

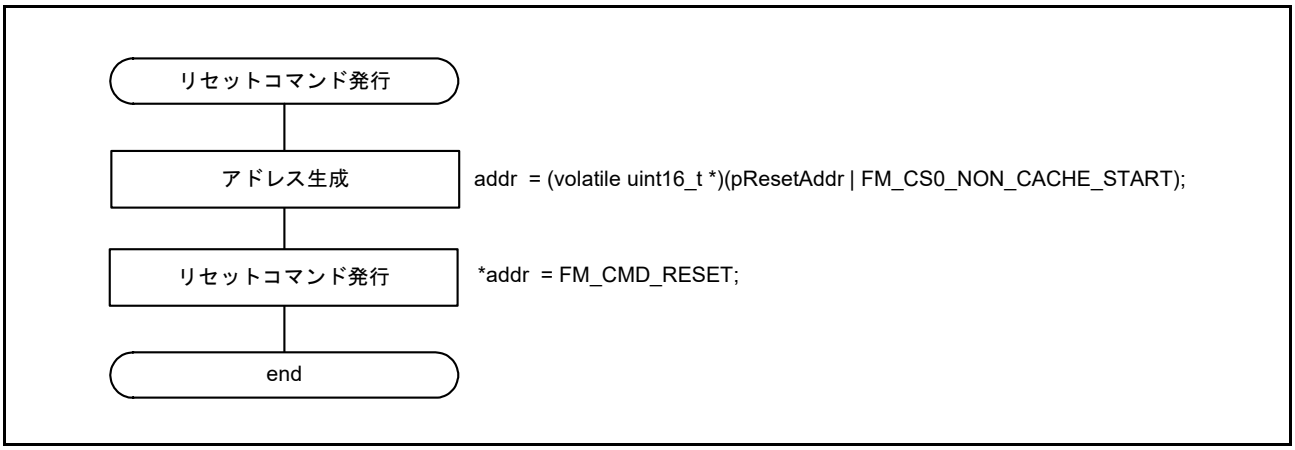


図 7.3　リセットコマンド発行

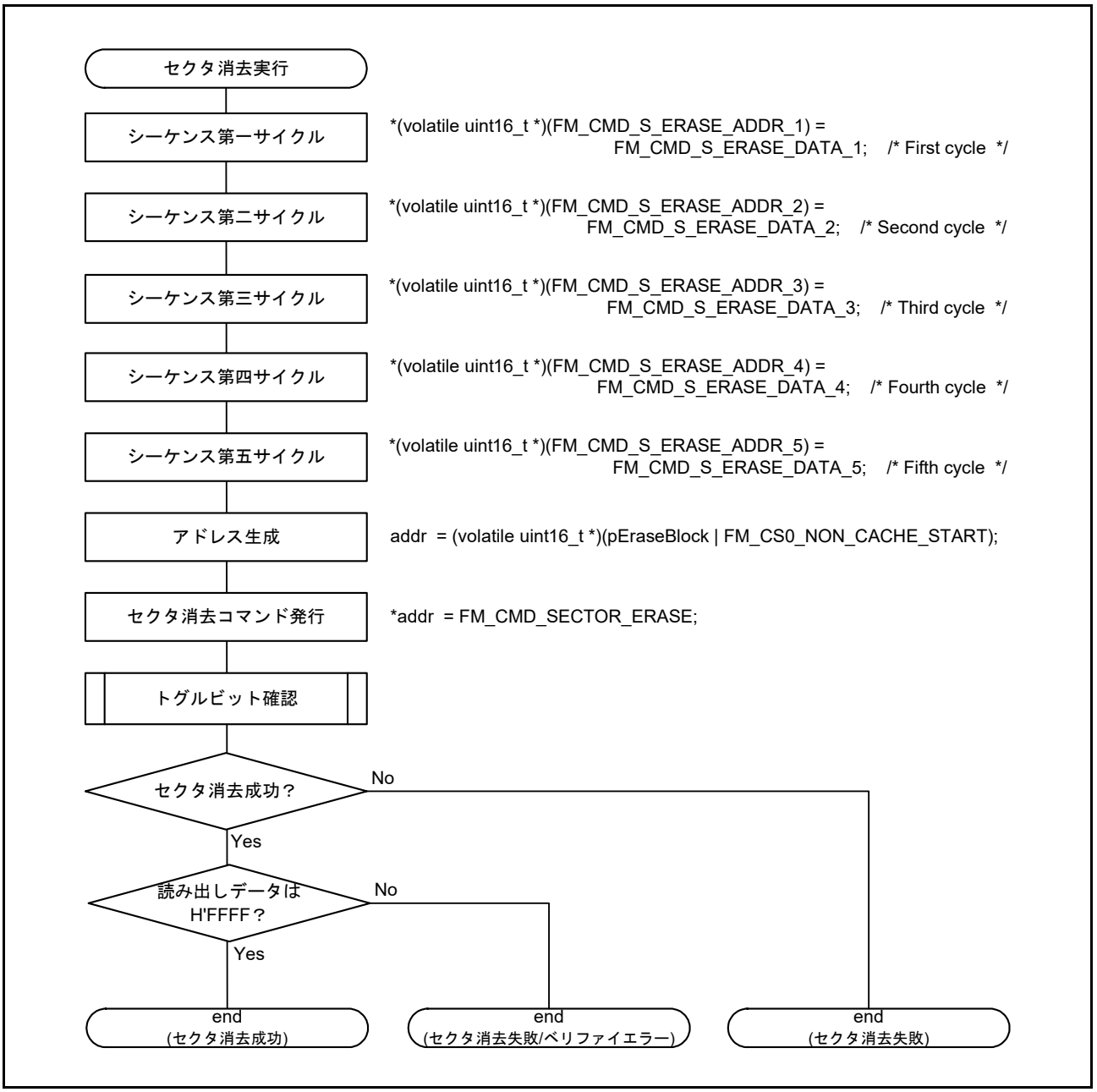

セクタ消去実行
シーケンス第一サイクル
*(volatile uint16_t *)(FM_CMD_S_ERASE_ADDR_1) =
FM_CMD_S_ERASE_DATA_1; /* First cycle */
シーケンス第二サイクル
*(volatile uint16_t *)(FM_CMD_S_ERASE_ADDR_2) =
FM_CMD_S_ERASE_DATA_2; /* Second cycle */
シーケンス第三サイクル
*(volatile uint16_t *)(FM_CMD_S_ERASE_ADDR_3) =
FM_CMD_S_ERASE_DATA_3; /* Third cycle */
シーケンス第四サイクル
*(volatile uint16_t *)(FM_CMD_S_ERASE_ADDR_4) =
FM_CMD_S_ERASE_DATA_4; /* Fourth cycle */
シーケンス第五サイクル
*(volatile uint16_t *)(FM_CMD_S_ERASE_ADDR_5) =
FM_CMD_S_ERASE_DATA_5; /* Fifth cycle */
アドレス生成
addr = (volatile uint16_t *)(pEraseBlock | FM_CS0_NON_CACHE_START);
セクタ消去コマンド発行
*addr = FM_CMD_SECTOR_ERASE;
トグルビット確認
セクタ消去成功？
No
Yes
読み出しデータは
H'FFFF？
No
Yes
end
(セクタ消去成功)
end
(セクタ消去失敗/ベリファイエラー)
end
(セクタ消去失敗)

図 7.4　セクタ消去実行

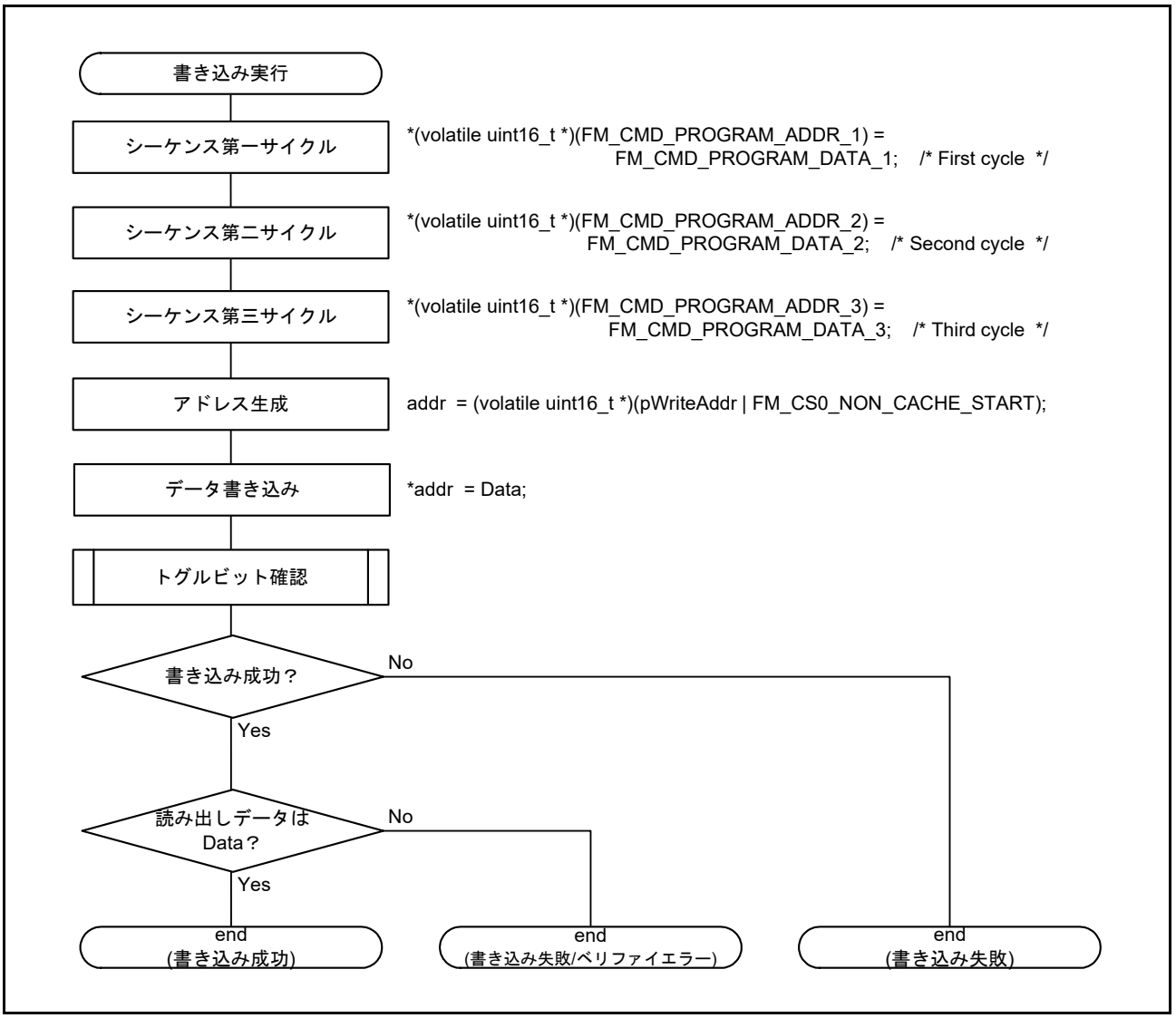

書き込み実行
シーケンス第一サイクル
*(volatile uint16_t *)(FM_CMD_PROGRAM_ADDR_1) =
FM_CMD_PROGRAM_DATA_1; /* First cycle */
シーケンス第二サイクル
*(volatile uint16_t *)(FM_CMD_PROGRAM_ADDR_2) =
FM_CMD_PROGRAM_DATA_2; /* Second cycle */
シーケンス第三サイクル
*(volatile uint16_t *)(FM_CMD_PROGRAM_ADDR_3) =
FM_CMD_PROGRAM_DATA_3; /* Third cycle */
アドレス生成
addr = (volatile uint16_t *)(pWriteAddr | FM_CS0_NON_CACHE_START);
データ書き込み
*addr = Data;
トグルビット確認
書き込み成功？
No
Yes
読み出しデータは
Data？
No
Yes
end
(書き込み成功)
end
(書き込み失敗/ベリファイエラー)
end
(書き込み失敗)

図 7.5　書き込み実行

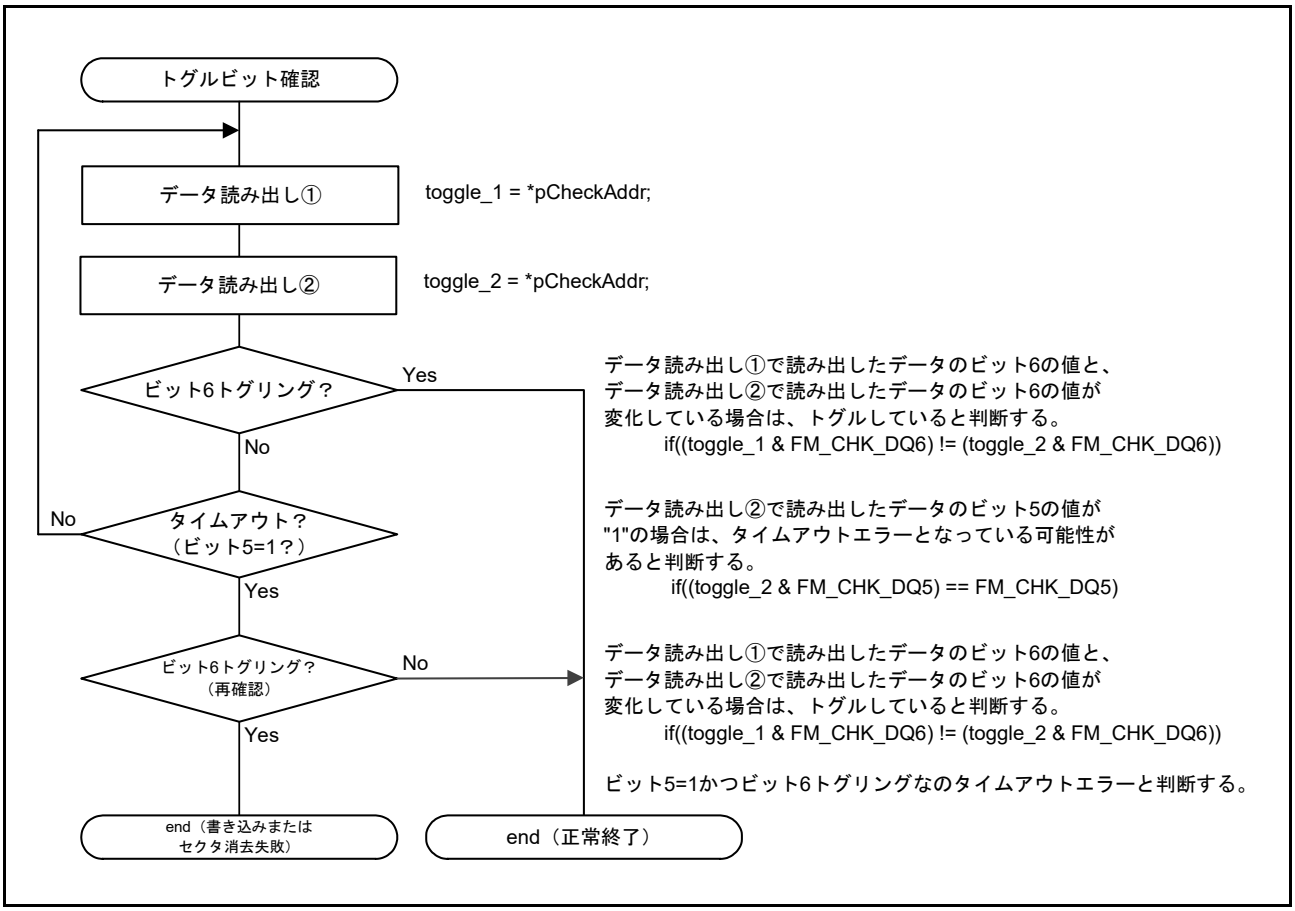


図 7.6　トグルビット確認

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。

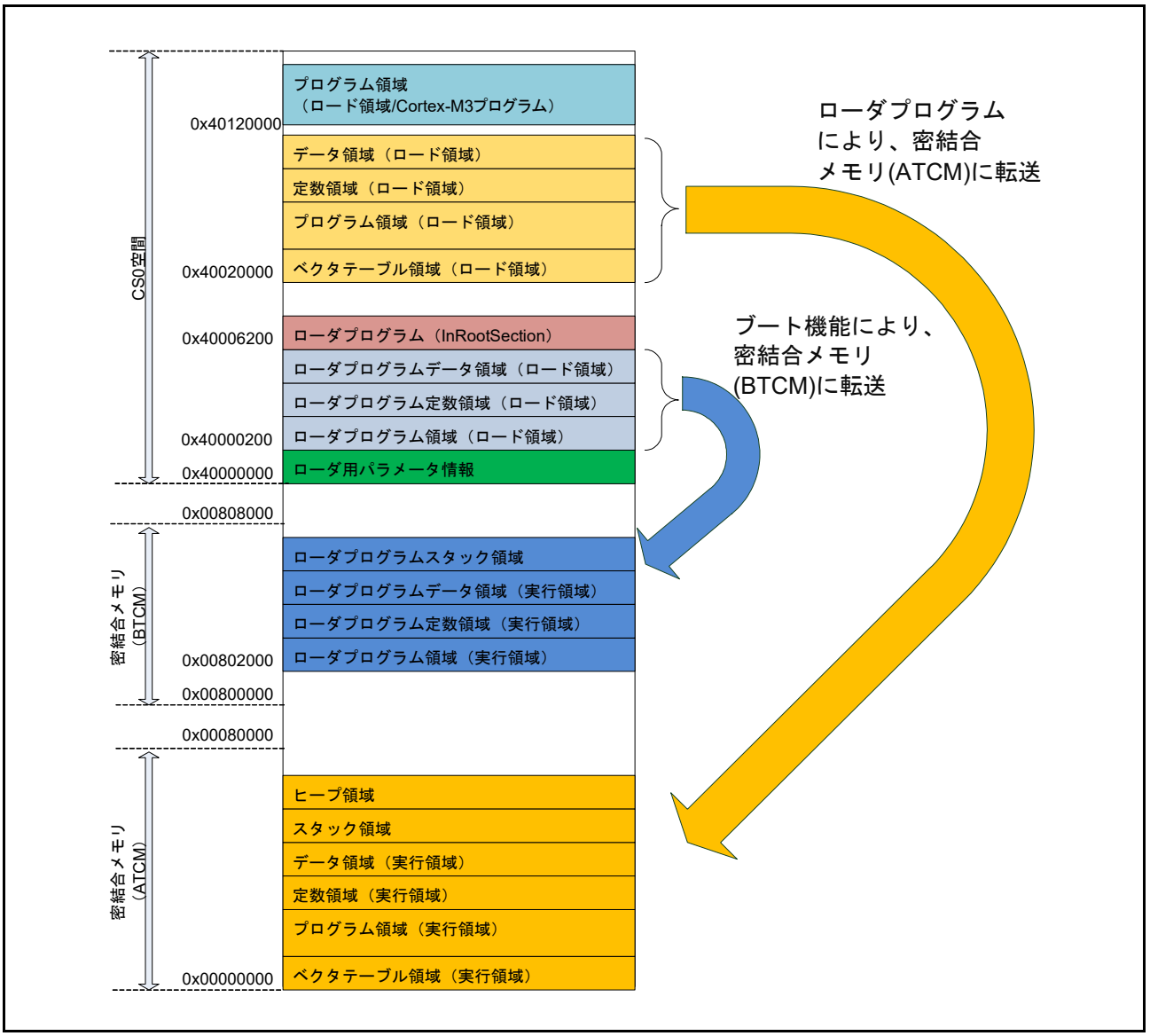


図 8.1 アプリケーションプログラムのメモリ配置の例

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ブート機能で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。

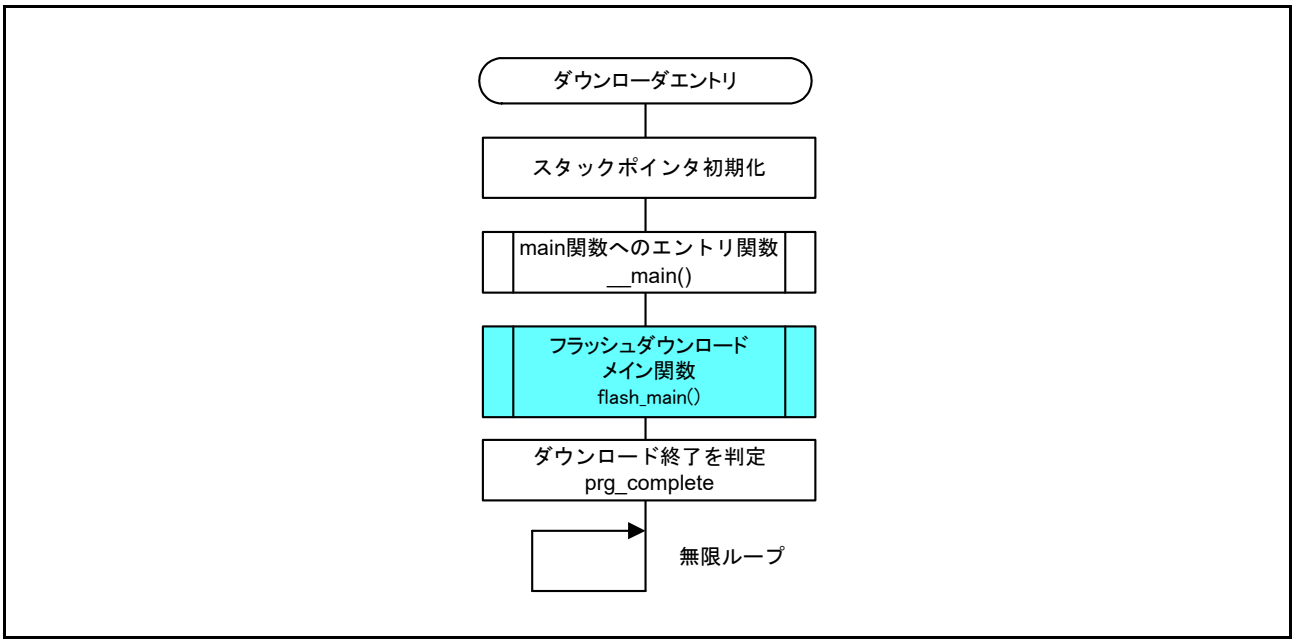


図 8.2 フラッシュダウンローダ処理フロー (1/3)

flash_main 関数では、フラッシュメモリへの書き込みを開始します。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、NOR 型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。

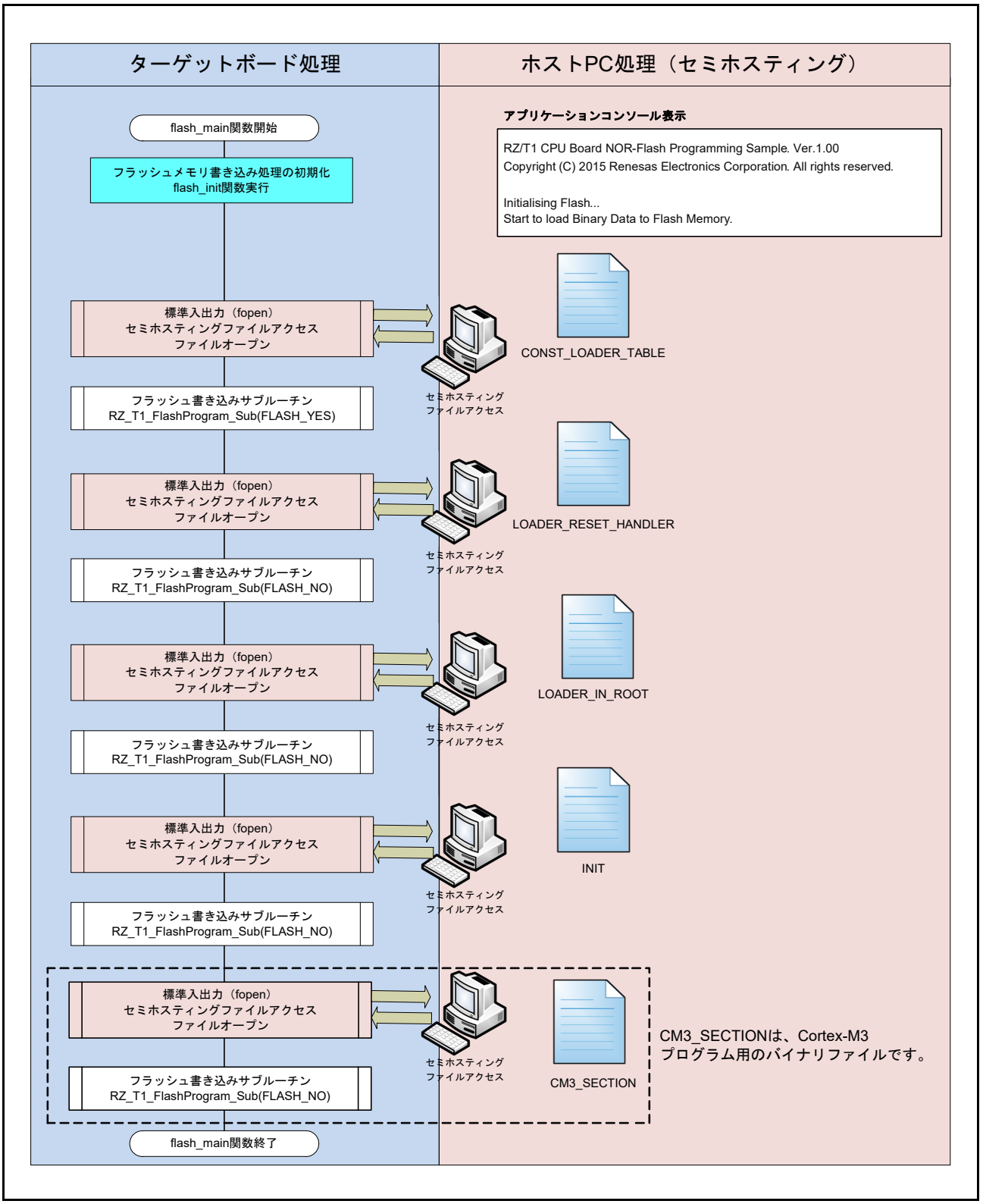


図 8.3　フラッシュダウンローダ処理フロー (2/3)

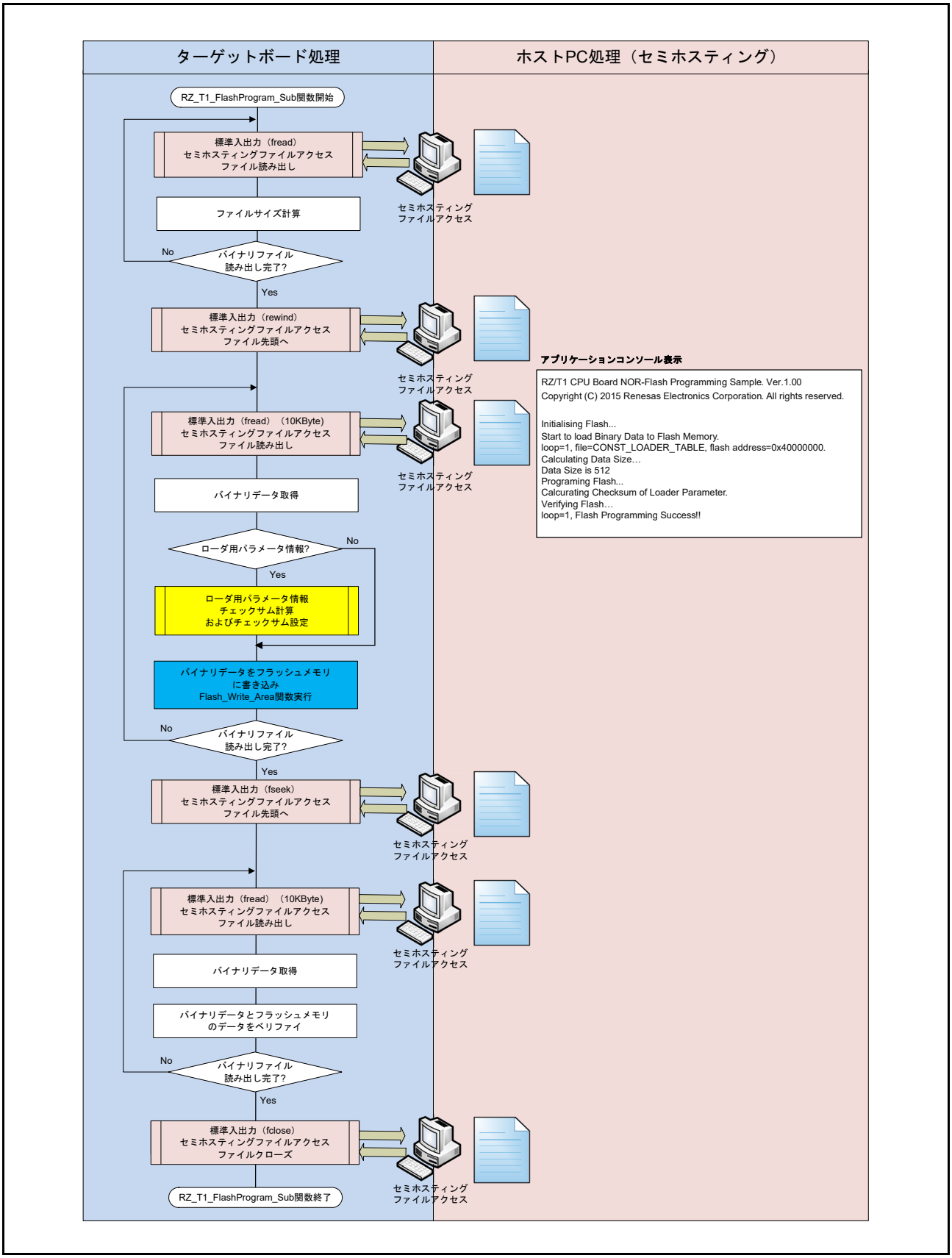


図 8.4　フラッシュダウンローダ処理フロー (3/3)

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。

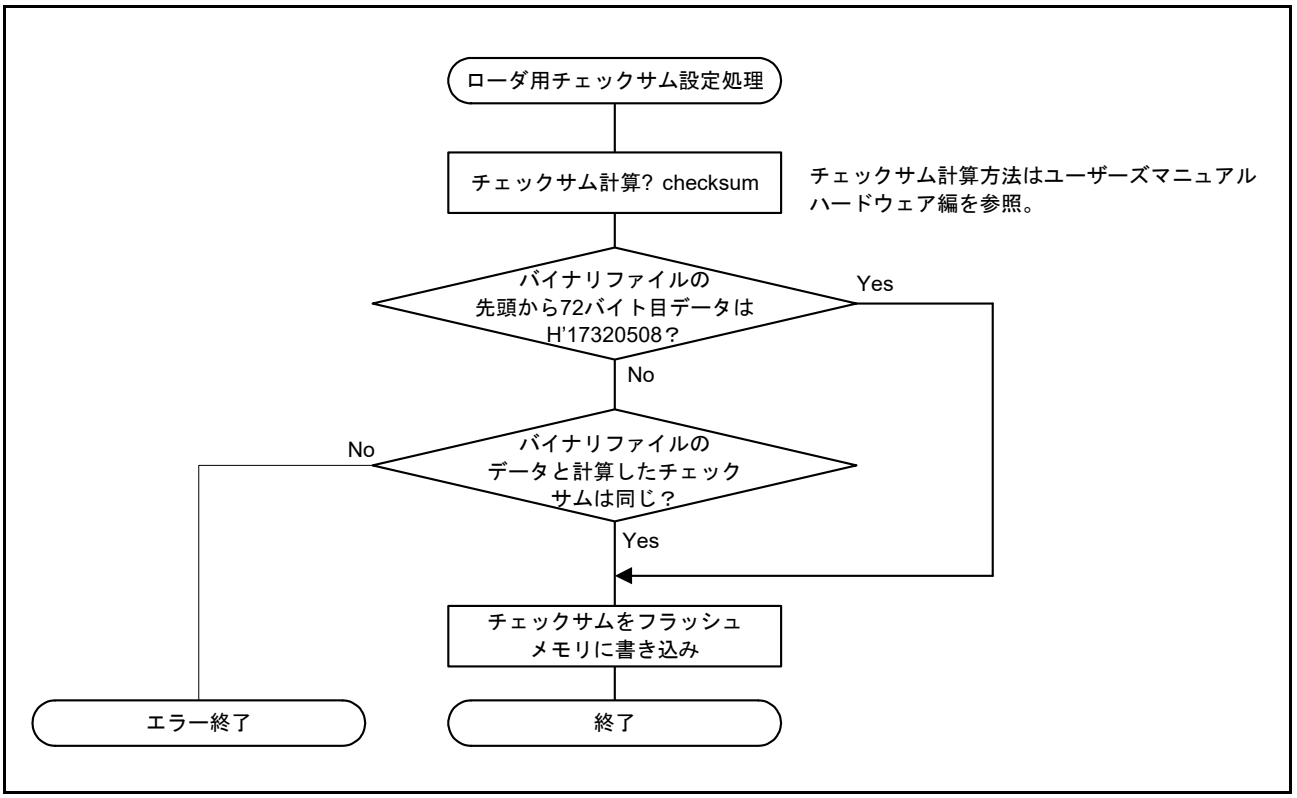


図 8.5 ローダ用パラメータ情報チェックサム設定処理フロー

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_nor	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_nor_sample	このプロジェクトでユーザアプリケーションをビルドします。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_nor_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（CS0空間）に配置されたNOR型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'40000000	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'40000200	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'40006200	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'40020000	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	H'40120000	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

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。

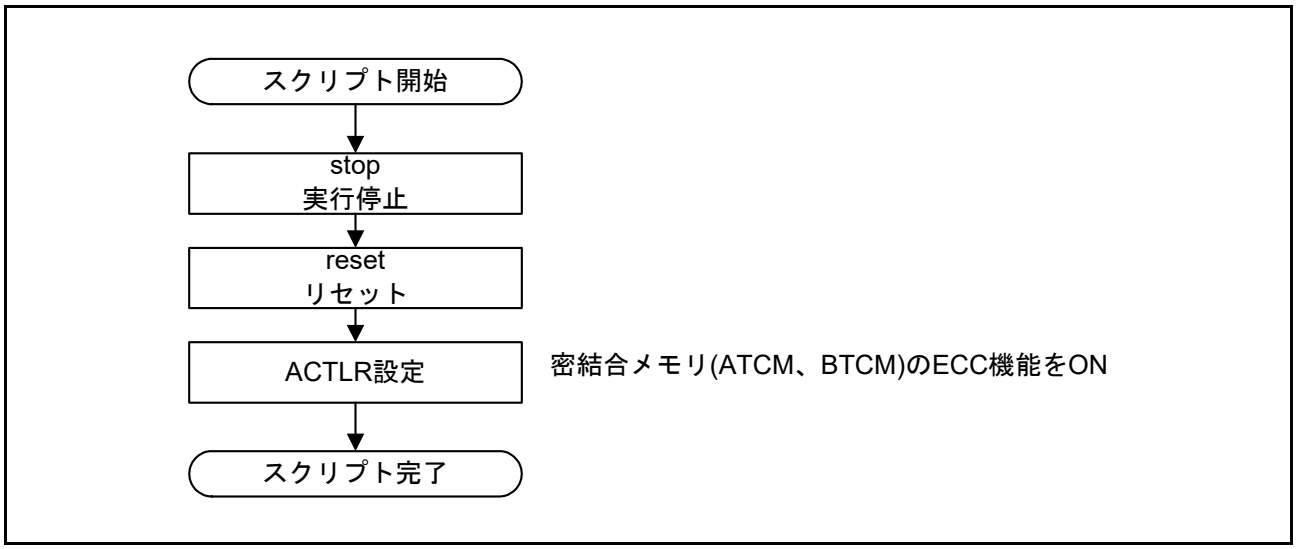


図 9.1 RZ/T1 評価ボード初期化スクリプト処理内容

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_nor_sample を RZ/T1 グループマイコンの外部アドレス空間（CS0 空間）に配置された NOR 型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（CS0 空間）に配置された NOR 型フラッシュメモリにアプリケーションプログラム RZ_T_nor_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_nor_sample のシンボル情報がロードされます。

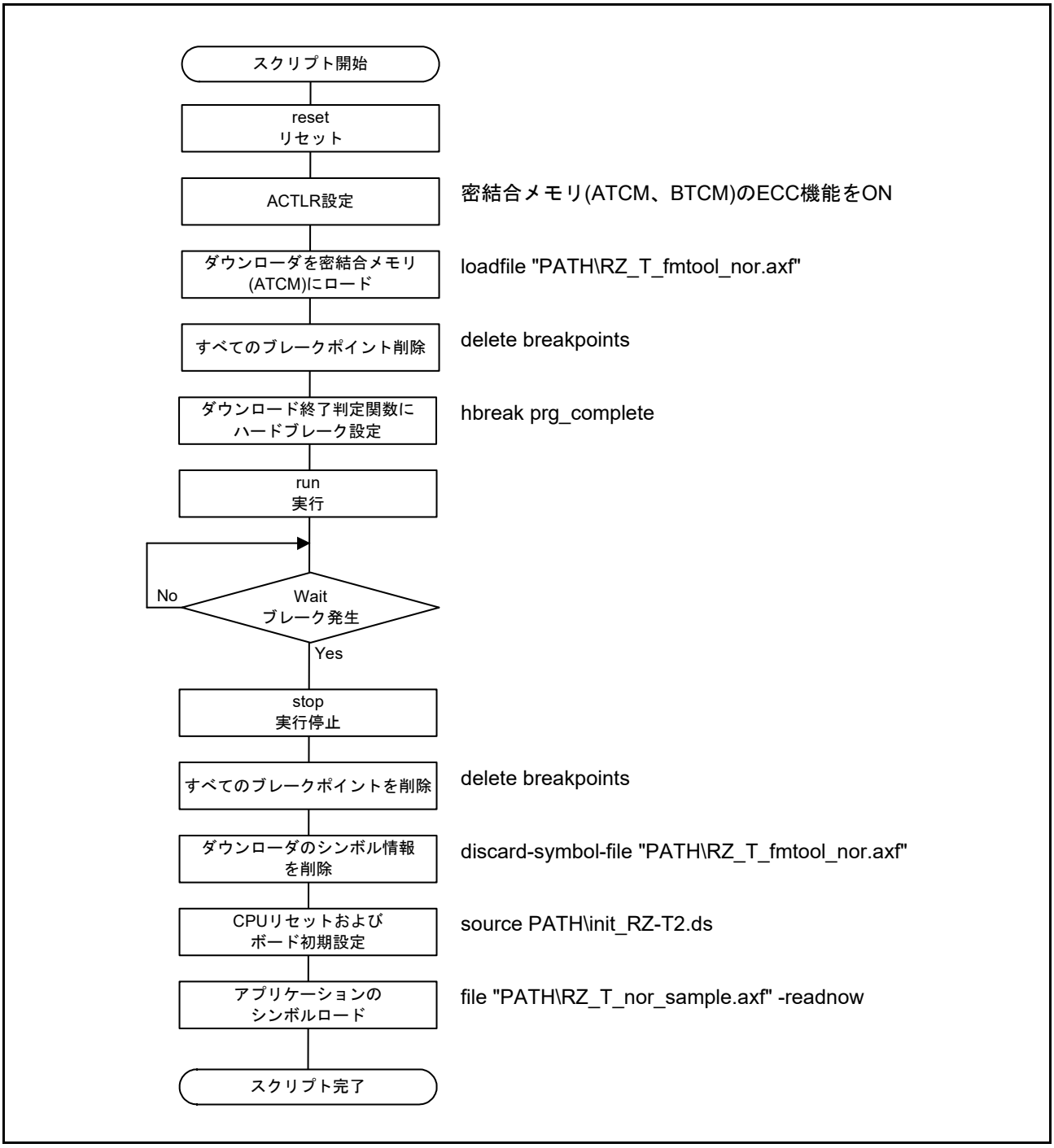


図 9.2 アプリケーションダウンロードスクリプト処理内容

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できます。

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。

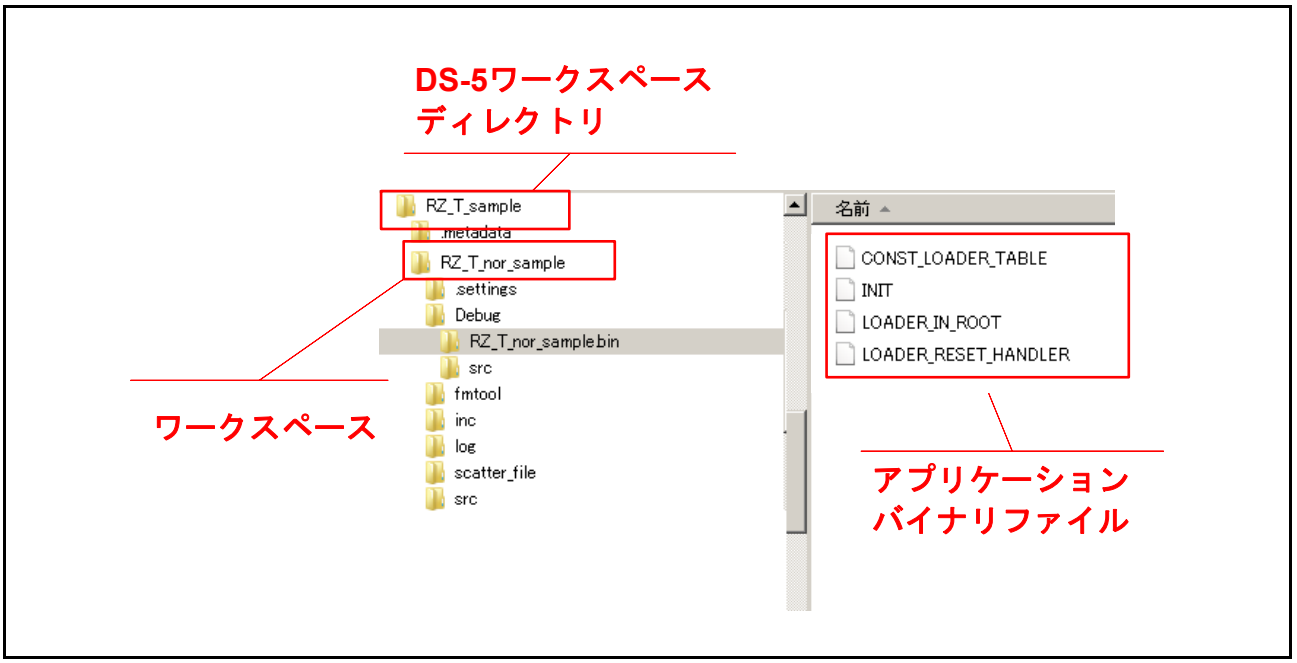


図 10.1 実装例におけるディレクトリ構成

変更前	srcfile = fopen(".¥¥RZ_T_nor_sample¥¥Debug¥¥RZ_T_nor_sample.bin¥¥INIT", "r");
	if(srcfile == 0)
	{
	    printf("loop=%d, Could not open file. Exiting.¥n", loop);
	    return;
	}
	address = (uint32_t *)0x40020000;
	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_nor_sample¥¥Debug¥¥RZ_T_nor_sample.bin¥¥INIT2", "r");
	if(srcfile == 0)
	{
	    printf("loop=%d, Could not open file. Exiting.¥n", loop);
	    return;
	}
	address = (uint32_t *)0x40040000;
	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）ごとに生成されます。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_nor_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** から**図 10.2** を参照してください。

```
LOAD_MODULE1 0x40000000     0x00000200
{
     CONST_LOADER_TABLE     0x40000000     FIXED
    {
        * (CONST_LOADER_TABLE)
    }
}
LOAD_MODULE2 0x40000200     0x00006000
{
    LOADER_RESET_HANDLER     0x00802000     FIXED
    {
        * (LOADER_RESET_HANDLER, +FIRST)
        * (USER_PROG_JUMP)
    }
    以下省略
}
LOAD_MODULE3 0x40006200     (0x40020000 - 0x40006200)
{
    LOADER_IN_ROOT    0x40006200     FIXED
    {
        * (InRoot$$Sections)
    }
}
LOAD_MODULE4 0x40020000     (0x40120000 - 0x40020000)
{
    INIT              0x00000000     FIXED
    {
        * (VECTOR_TABLE, +FIRST)
        * (RESET_HANDLER)
        * (IRQ_FIQ_HANDLER)
    }
    以下省略
}
LOAD_MODULE5 0x40120000     (0x44000000 - 0x40120000)
{
    CM3_SECTION       0x40120000     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'40000000	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'40000200	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'40006200	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'40020000	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'40120000	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フラッシュメモリインタフェース関数のカスタマイズ

本章では、サンプルプログラムの応用例として、お客様が使用するフラッシュメモリに応じたフラッシュメモリインタフェース関数のカスタマイズ方法について説明します。

フラッシュメモリインタフェース関数はフラッシュメモリのデバイス仕様に依存するため、デバイス変更時はプログラムのカスタマイズが必要な場合があります。

本アプリケーションノートのサンプルプログラムは、JEDEC 標準型コマンド互換方式に対応しています。JEDEC 標準型コマンド互換方式のフラッシュメモリをご使用の場合は、サンプルプログラムを流用することが可能です。

CUI（Command User Interface）コマンド方式のフラッシュメモリをご使用の場合は、サンプルプログラムは流用できません。この場合は、お客様が新規にダウンロードプログラムを作成してください。

注 1.　JEDEC 標準型コマンド互換方式は、あらかじめ決められたアドレス（H'AAA、H'554 など）にコマンドを発行して書き込む方式です。
注 2.　CUI コマンド方式は、CUI に書き込みコマンド（H'40）、消去コマンド（H'20）を発行する方式です。

10.2.1　サンプルプログラムに対応するデバイス仕様

表 10.2 から表 10.3 に、使用デバイスの詳細仕様およびサンプルプログラムで使用するコマンドを示します。

表10.2　使用デバイスの詳細仕様

項目	内容
メーカ	Macronix International Co..
型名	MX29GL512F
容量	64Mバイト
データバス幅	16ビット
アクセスタイム	90ns
セクタ構造	ユニフォーム型
セクタサイズ	128Kバイト
セクタ数	512
書き込み方式	JEDEC標準型コマンド互換方式

表10.3　サンプルプログラムで使用するコマンド

項目	内容		
	サイクル	アドレス	データ
イレーズコマンド（セクタイレーズ）	第一サイクル	H'AAA	H'AA
	第二サイクル	H'554	H'55
	第三サイクル	H'AAA	H'80
	第四サイクル	H'AAA	H'AA
	第五サイクル	H'554	H'55
	第六サイクル	SA注1	H'30
プログラムコマンド	第一サイクル	H'AAA	H'AA
	第二サイクル	H'554	H'55
	第三サイクル	H'AAA	H'A0
	第四サイクル	PA注2	PD 注3
リセットコマンド	第一サイクル	— 注4	H'F0

注1.　SAはセクタアドレスを示します。イレーズするセクタアドレスを指定します。
注2.　PAはプログラムアドレスを示します。ライトするアドレスを指定します。
注3.　PDはプログラムデータを示します。ライトするデータを指定します。
注4.　”ー”はフラッシュメモリが配置されている空間のアドレスを指定します。フラッシュメモリが配置されている空間であれば、どのアドレスを指定しても問題はありません。

10.2.2 カスタマイズで対応可能なフラッシュメモリのブート型

サンプルプログラムをカスタマイズすることにより、以下の 4 種類のフラッシュメモリのブート型に対応することが可能です。

1. ユニフォーム型
2. ボトムブート型
3. トップブート型
4. デュアルブート型

図 10.4 にサンプルプログラムのカスタマイズで対応可能なフラッシュメモリのブート型のメモリマップ例を示します。

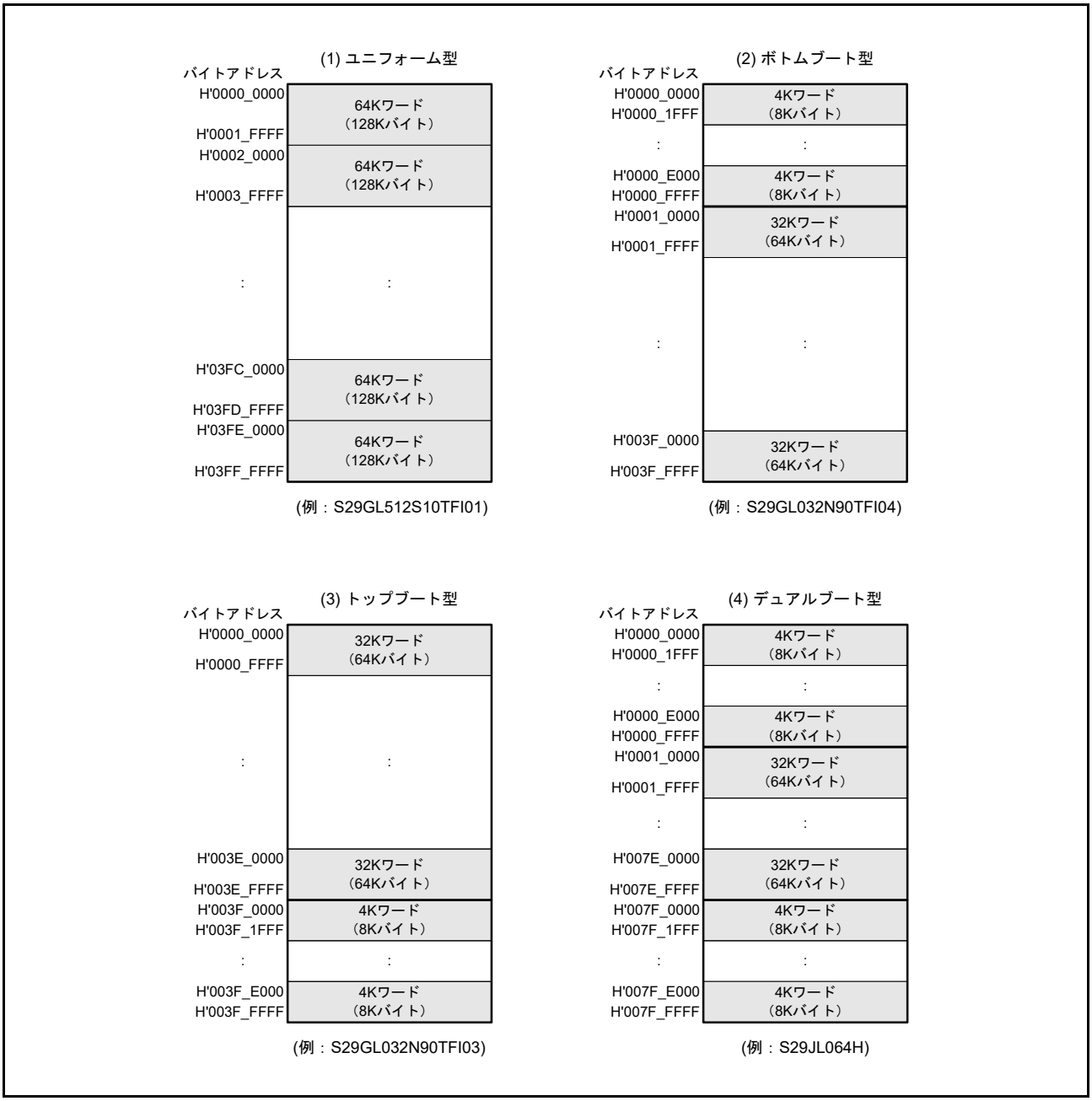


図 10.4 サンプルプログラムのカスタマイズで対応可能なフラッシュメモリのブート型

10.2.3 カスタマイズの内容

以下にカスタマイズが必要なケースと、その変更内容を示します。

表10.4 カスタマイズが必要なケースとカスタマイズ内容(1/2)

項目	カスタマイズ内容
フラッシュメモリのブート型が異なる場合	サンプルプログラムはユニフォーム型のフラッシュメモリの消去および書き込みに対応しています。 トップブート型、ボトムブート型、およびデュアルブート型のフラッシュメモリを使用する場合は、FM_BOOT_TYPEのマクロ定義を以下の値に変更してください。 トップブート型：FM_TOP_BOOT ボトムブート型：FM_BOTTOM_BOOT デュアルブート型：FM_DUAL_BOOT （初期値は、ユニフォーム型の”FM_UNIFORM”を定義しています。）
フラッシュメモリのブート型を変更する場合1 （ボトムブート型に変更）	セクタの定義を変更してください。 • FM_B_BOOT_SECTOR_START • FM_B_BOOT_SECTOR_SIZE • FM_B_BOOT_SECTOR_NUM • FM_NORMAL_SECTOR_START • FM_NORMAL_SECTOR_SIZE • FM_NORMAL_SECTOR_NUM 以下のセクタ定義の初期値は”0”で定義しています。そのまま使用してください。 • FM_T_BOOT_SECTOR_START • FM_T_BOOT_SECTOR_SIZE • FM_T_BOOT_SECTOR_NUM
フラッシュメモリのブート型を変更する場合2 （トップブート型に変更）	セクタの定義を変更してください。 • FM_NORMAL_SECTOR_START • FM_NORMAL_SECTOR_SIZE • FM_NORMAL_SECTOR_NUM • FM_T_BOOT_SECTOR_START • FM_T_BOOT_SECTOR_SIZE • FM_T_BOOT_SECTOR_NUM 以下のセクタ定義の初期値は”0”で定義しています。そのまま使用してください。 • FM_B_BOOT_SECTOR_START • FM_B_BOOT_SECTOR_SIZE • FM_B_BOOT_SECTOR_NUM

注. フラッシュメモリインタフェース関数はフラッシュメモリの仕様に依存するため、使用されるデバイスのデータシートをご確認の上、デバイスの仕様に合わせてフラッシュメモリインタフェース関数を修正してください。

表10.5　カスタマイズが必要なケースとカスタマイズ内容(2/2)

項目	カスタマイズ内容
フラッシュメモリのブート型を変更する場合3（デュアルブート型に変更）	セクタの定義を変更してください。 • FM_B_BOOT_SECTOR_START • FM_B_BOOT_SECTOR_SIZE • FM_B_BOOT_SECTOR_NUM • FM_NORMAL_SECTOR_START • FM_NORMAL_SECTOR_SIZE • FM_NORMAL_SECTOR_NUM • FM_T_BOOT_SECTOR_START • FM_T_BOOT_SECTOR_SIZE • FM_T_BOOT_SECTOR_NUM
セクタのサイズ、セクタ数が異なる場合	セクタの定義を変更してください。 FM_NORMAL_SECTOR_START FM_NORMAL_SECTOR_SIZE FM_NORMAL_SECTOR_NUM 以下のセクタ定義の初期値は”0”で定義しています。そのまま使用してください。 FM_B_BOOT_SECTOR_START FM_B_BOOT_SECTOR_SIZE FM_B_BOOT_SECTOR_NUM FM_T_BOOT_SECTOR_START FM_T_BOOT_SECTOR_SIZE FM_T_BOOT_SECTOR_NUM
メモリ容量が異なる場合	FM_END_ADDRESSマクロの設定値を変更してください。
図10.4に示す4種類のブート型とは異なる場合	フラッシュメモリの操作関数のカスタマイズが必要です。詳細はサンプルプログラムを参照してください。
イレーズ、ライト、およびリセットのコマンド仕様が異なる場合	
書き込みコマンドがCUIコマンド方式の場合	
RZ/T1グループマイコンとフラッシュメモリを16ビット以外で接続している場合	

注.　フラッシュメモリインタフェース関数はフラッシュメモリの仕様に依存するため、使用されるデバイスのデータシートをご確認の上、デバイスの仕様に合わせてフラッシュメモリインタフェース関数を修正してください。

10.2.4　ユニフォーム型のセクタサイズやセクタ数をカスタマイズする方法

図 10.5 にユニフォーム型のフラッシュメモリのセクタサイズやセクタ数を変更するカスタマイズ方法を示します。フラッシュメモリ MX29GL512F とセクタサイズやセクタ数の異なるユニフォーム型のフラッシュメモリを使用する場合は、ユニフォーム型のセクタ情報のマクロの設定値を変更します。フラッシュメモリの容量が異なる場合は、最終アドレスのマクロの設定値も変更します。

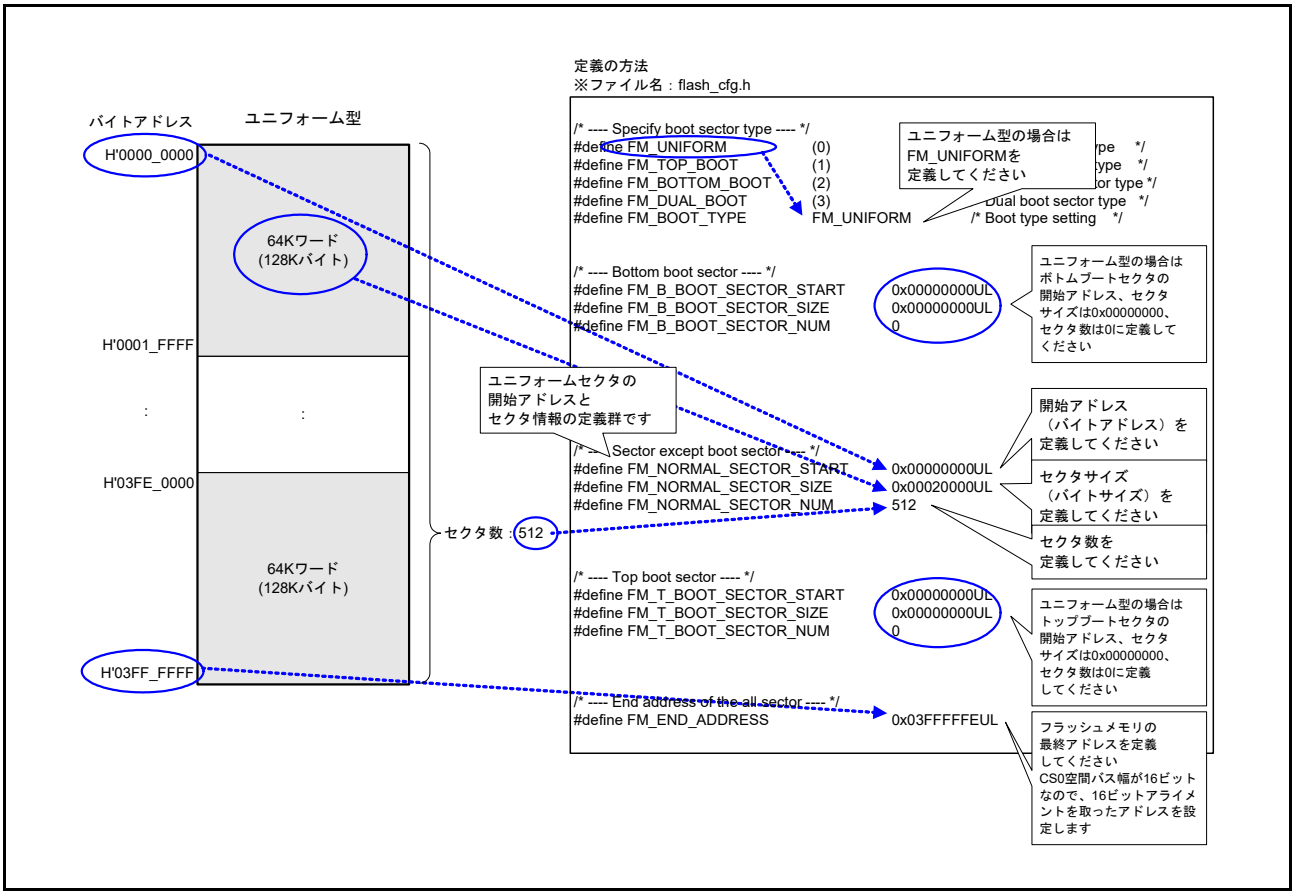


図 10.5　ユニフォーム型のフラッシュメモリのセクタサイズやセクタ数を変更するカスタマイズ方法

10.2.5　フラッシュメモリのブート型をボトムブート型にカスタマイズする方法

図 10.6 にフラッシュメモリのブート型をボトムブート型に変更する場合のカスタマイズ方法を示します。ボトムブート型のフラッシュメモリを使用する場合は、ボトムブートセクタ領域およびブートセクタ以外の領域の 2 種類のセクタ情報のマクロの設定値を変更します。フラッシュメモリの容量が異なる場合は、最終アドレスのマクロの設定値も変更します。

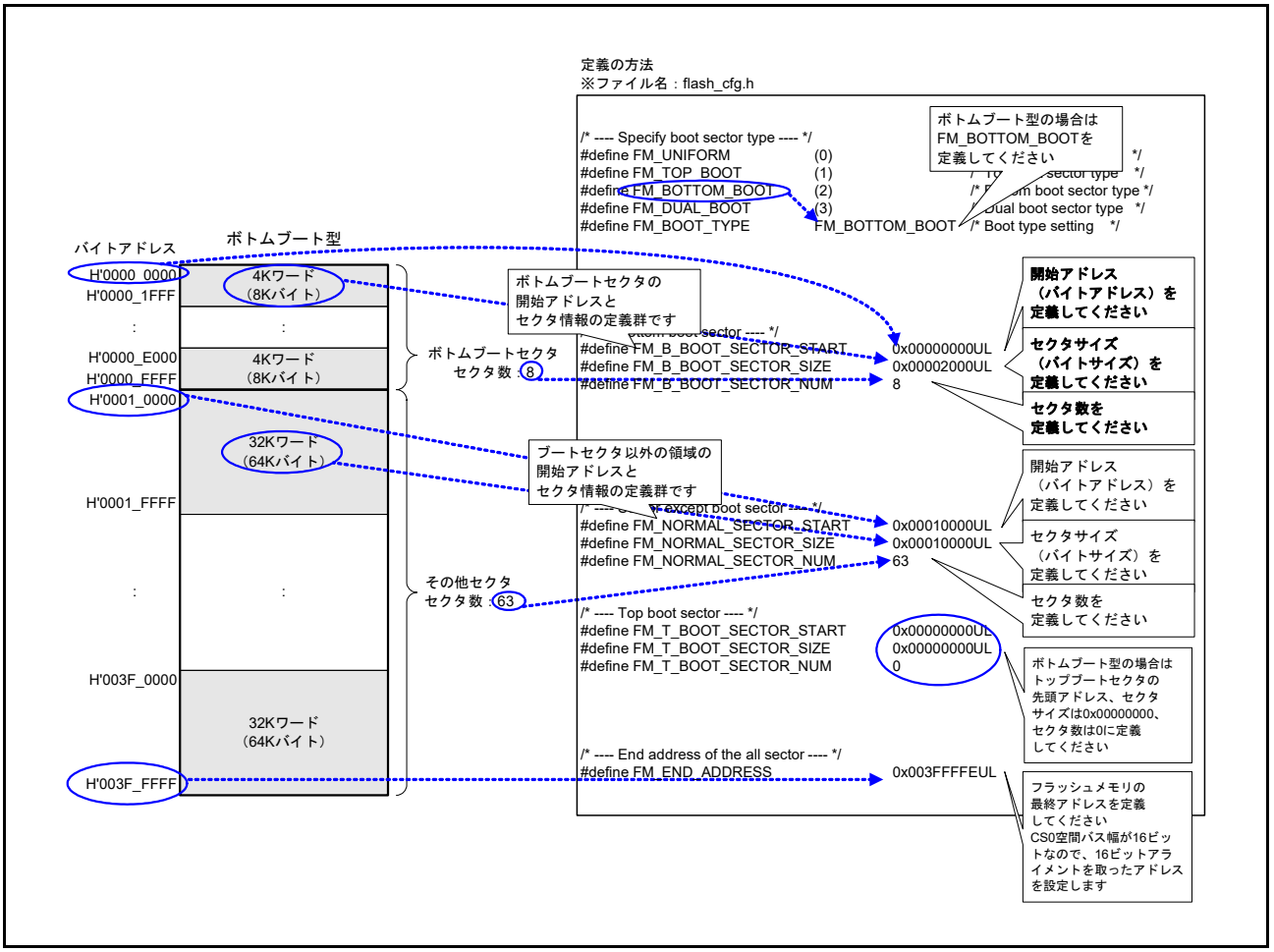


図 10.6　フラッシュメモリのブート型をボトムブート型に変更する場合のカスタマイズ方法

10.2.6　フラッシュメモリのブート型をトップブート型にカスタマイズする方法

図 10.7 にフラッシュメモリのブート型をトップブート型に変更する場合のカスタマイズ方法を示します。トップブート型のフラッシュメモリを使用する場合は、トップブートセクタ領域およびブートセクタ以外の領域の 2 種類のセクタ情報のマクロの設定値を変更します。フラッシュメモリの容量が異なる場合は、最終アドレスのマクロの設定値も変更します。

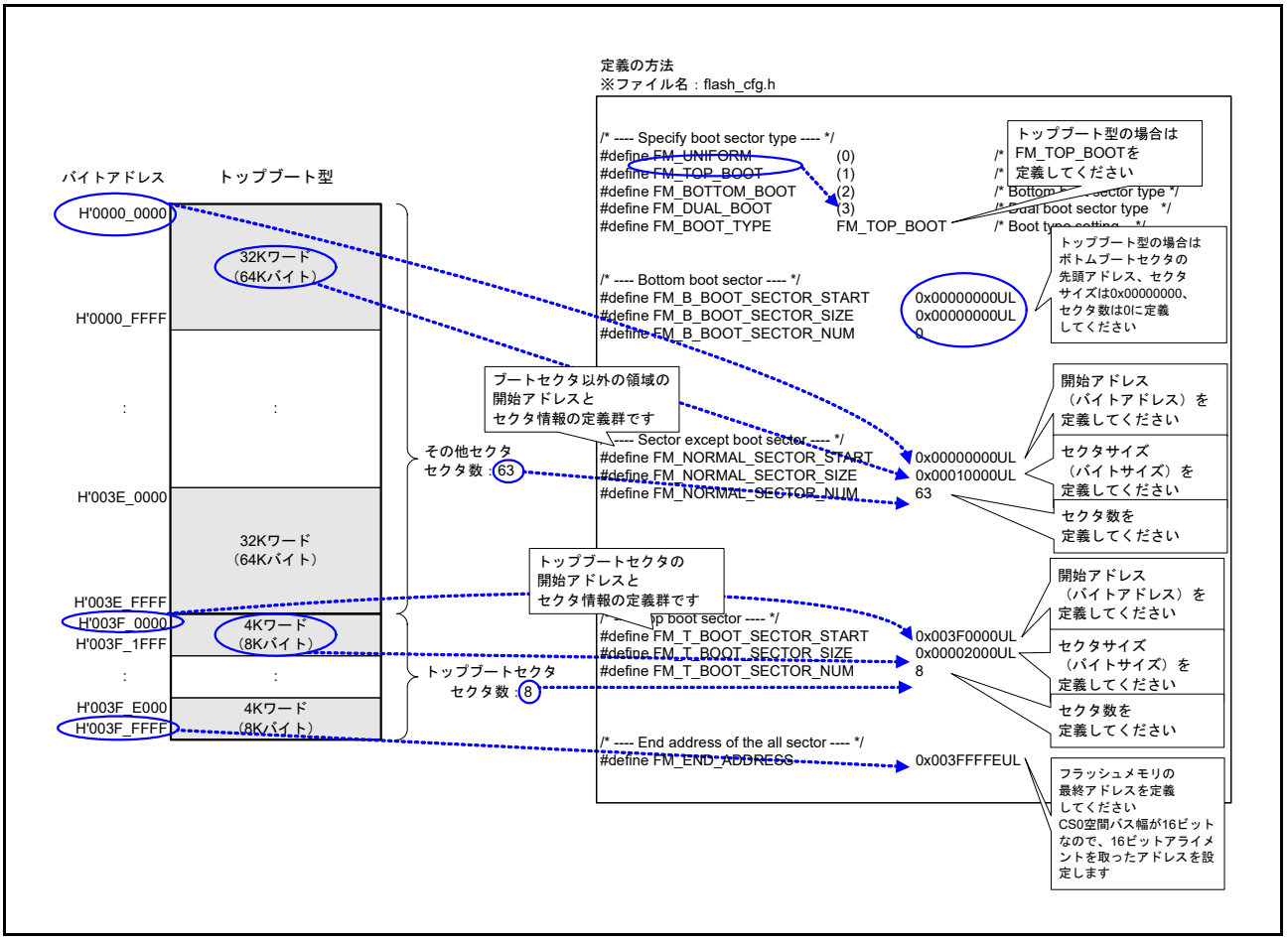


図 10.7　フラッシュメモリのブート型をトップブート型に変更する場合のカスタマイズ方法

10.2.7　フラッシュメモリのブート型をデュアルブート型にカスタマイズする方法

図 10.8 に、フラッシュメモリのブート型をデュアルブート型に変更する場合のカスタマイズ方法を示します。デュアルブート型のフラッシュメモリを使用する場合は、トップブートセクタ領域、ボトムブートセクタ領域、およびブートセクタ以外の領域の 3 種類のセクタ情報のマクロの設定値を変更します。フラッシュメモリの容量が異なる場合は、最終アドレスのマクロの設定値も変更します。

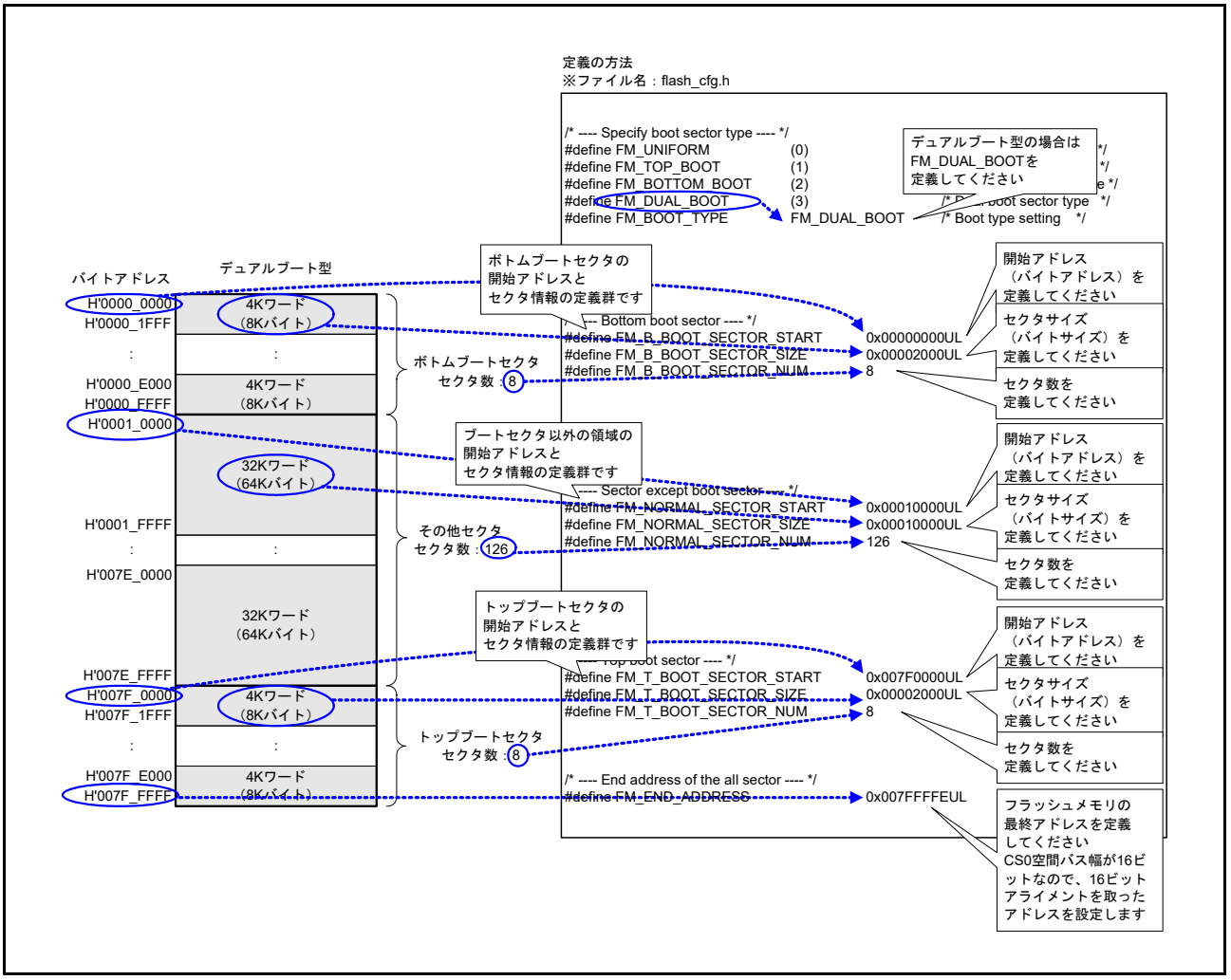


図 10.8　フラッシュメモリのブート型をデュアルブート型に変更する場合のカスタマイズ方法

10.2.8 フラッシュメモリのコマンドの確認

図 10.9 に、フラッシュメモリのコマンド定義を示します。サンプルプログラムは、JEDEC 標準型コマンド互換方式に対応しており、RZ/T1 グループマイコンとフラッシュメモリを 16 ビットで接続する場合のコマンドを定義しています。フラッシュメモリのデータシートでワードモード（x16 ビットモード）のコマンドを参照して、リセットコマンド、セクタイレーズコマンド、およびプログラムコマンド（ライトコマンド）のコマンド内容に相違がないことを確認してください。なお、サンプルプログラムでは、ページプログラム（ページライト）には対応していません。

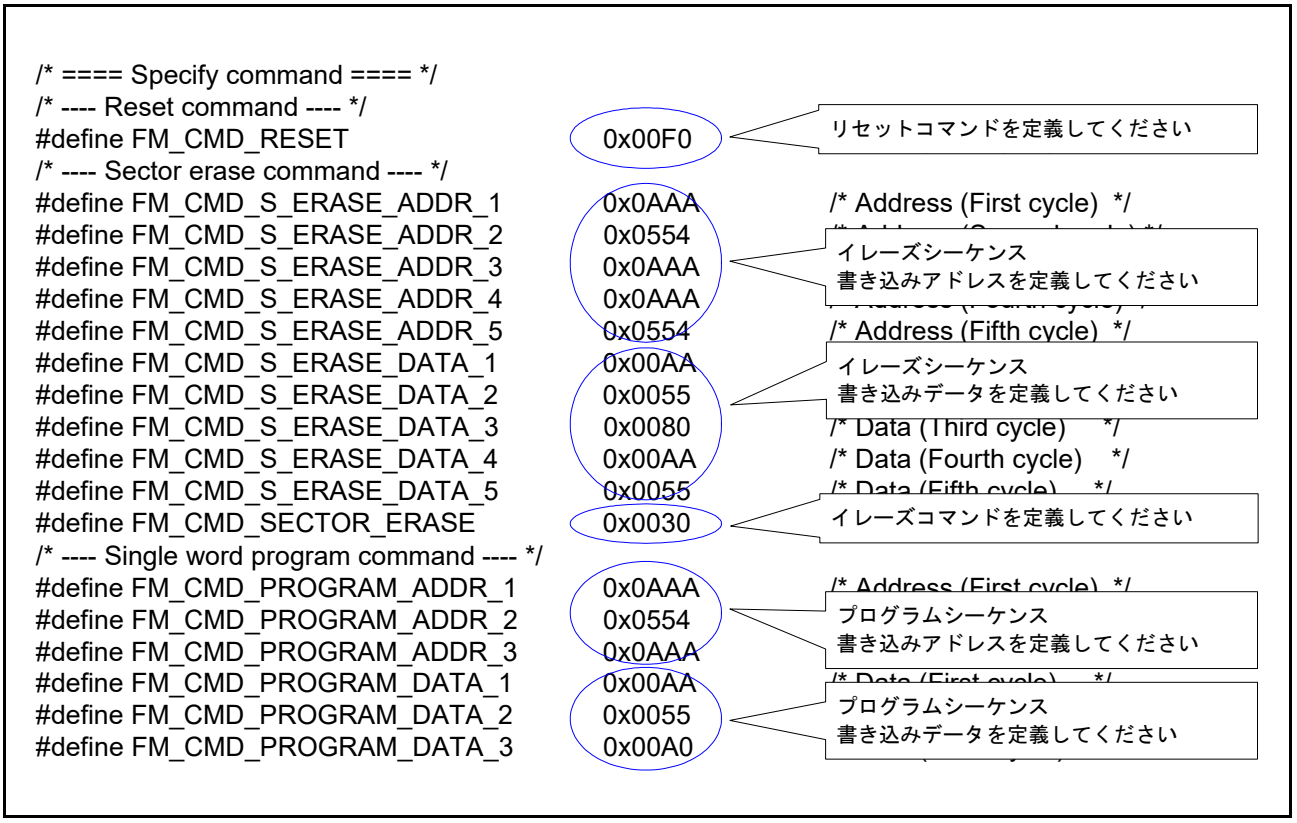


図 10.9 フラッシュメモリのコマンド定義（flash_cfg.h）

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**表 10.6** に示すように CM3_SECTION が生成されます。また、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、アプリケーションコンソールには**図 10.10** のように表示されます。

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

アプリケーションバイナリファイル		説明
RZ_T_nor_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 NOR-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=0x40000000.
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=0x40000200.
Calculating Data Size...
Data Size is 3288
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x40006200.
Calculating Data Size...
Data Size is 192
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x40020000.
Calculating Data Size...
Data Size is 3008
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, file=CM3_SECTION, flash address=0x40120000.
Calculating Data Size...
Data Size is 1296
Programing Flash...
Verifying Flash...
loop=5, Flash Programming Success!!

 finish
Flash Programming Complete
```

図 10.10　アプリケーションコンソールに出力されるメッセージ（R-IN Engine 搭載製品使用時）

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 NOR型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	2015.09.25	—	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/