

## RX63N グループ

R01AN2098JJ0100

Rev.1.00

2014.07.01

Renesas Starter Kit+ for RX63N を使用した  
RX63N グループ用フラッシュプログラマ(ブートモード)

---

### 要旨

本アプリケーションノートでは、Renesas Starter Kit+ for RX63N(以下 RSK+RX63N と略します)を使用したRX63N グループ用のフラッシュプログラマを実現する方法について説明します。

書き換え対象はRX63N グループです。RX63N グループのユーザ領域書き換えには、ブートモードを使用します。

### 対象デバイス

- ・フラッシュプログラマ
  - RX63N グループ ROM 容量 : 1MB~2MB
- ・ターゲット MCU
  - RX63N グループ ROM 容量 : 256KB~1MB

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

## 目次

|                               |    |
|-------------------------------|----|
| 1. 仕様                         | 3  |
| 1.1 RSK+RX63N のユーザ領域メモリ配置     | 4  |
| 2. 動作確認条件                     | 5  |
| 3. 関連アプリケーションノート              | 5  |
| 4. ハードウェア説明                   | 6  |
| 4.1 ハードウェア構成例                 | 6  |
| 4.2 使用端子一覧                    | 6  |
| 5. ソフトウェア説明                   | 7  |
| 5.1 RSK+RX63N への書き込み方法        | 7  |
| 5.1.1 FDT ワークスペースの準備          | 8  |
| 5.1.2 データの結合と保存               | 9  |
| 5.1.3 RSK+RX63N のユーザ領域書き込み    | 16 |
| 5.2 動作概要                      | 17 |
| 5.2.1 ブートモード起動                | 18 |
| 5.2.2 ビットレート自動調整              | 19 |
| 5.2.3 ターゲット MCU の確定           | 20 |
| 5.2.4 ID コードプロテクトの判定          | 23 |
| 5.2.5 ターゲット MCU のユーザ領域への書き込み  | 26 |
| 5.2.6 ターゲット MCU のリセット         | 30 |
| 5.3 ファイル構成                    | 31 |
| 5.4 オプション設定メモリ                | 32 |
| 5.5 定数一覧                      | 33 |
| 5.6 構造体/共用体一覧                 | 36 |
| 5.7 変数一覧                      | 37 |
| 5.8 関数一覧                      | 39 |
| 5.9 関数仕様                      | 40 |
| 5.10 フローチャート                  | 44 |
| 5.10.1 メイン処理、通信プロトコル制御        | 44 |
| 5.10.2 周辺機能初期設定               | 56 |
| 5.10.3 CMT による待ち時間用タイマ初期設定    | 57 |
| 5.10.4 CMT による待ち時間設定          | 58 |
| 5.10.5 CMT による時間待ち処理          | 59 |
| 5.10.6 CMT0,CMIO 割り込み処理       | 59 |
| 5.10.7 SCI 初期設定               | 60 |
| 5.10.8 SCI ビットレート変更処理         | 61 |
| 5.10.9 “SUM” データ計算処理          | 62 |
| 5.10.10 ターゲット MCU のブートモード起動処理 | 63 |
| 5.10.11 ターゲット MCU のリセット処理     | 64 |
| 5.10.12 コマンド送信処理              | 65 |
| 5.10.13 レスポンス受信処理             | 66 |
| 5.10.14 符号なし 4 バイトデータの複写      | 70 |
| 6. サンプルコード                    | 71 |
| 7. 参考ドキュメント                   | 71 |

## 1. 仕様

フラッシュプログラマは、RSK+RX63N で動作し、ターゲットのRX63N グループをブートモードで起動します。その後、調歩同期式シリアルで通信することでRX63N グループのユーザ領域を書き換えます。

表 1.1に使用する周辺機能と用途を、図 1.1にフラッシュプログラマの使用例を示します。

調歩同期式のシリアル通信には、シリアルコミュニケーションインタフェースのチャンネル 0(SCI0)を使用します。

通信データフォーマットおよび出力形式は以下のとおりです。

- スタートビット : 1 ビット
- 転送データ : 8 ビット
- パリティビット : なし
- ストップビット : 1 ビット
- ビットレート : 19,200bps(新ビットレート選択コマンドの応答まで)  
: 1.5 Mbps(P/E ステータス遷移コマンド以降)
- 出力形式 : CMOS 出力

表1.1 使用する周辺機能と用途

| 周辺機能    | 用途              |
|---------|-----------------|
| SCI0    | 調歩同期式シリアル送受信    |
| CMT0    | 待ち時間用タイマ        |
| I/O ポート | ブートモード制御、LCD 出力 |

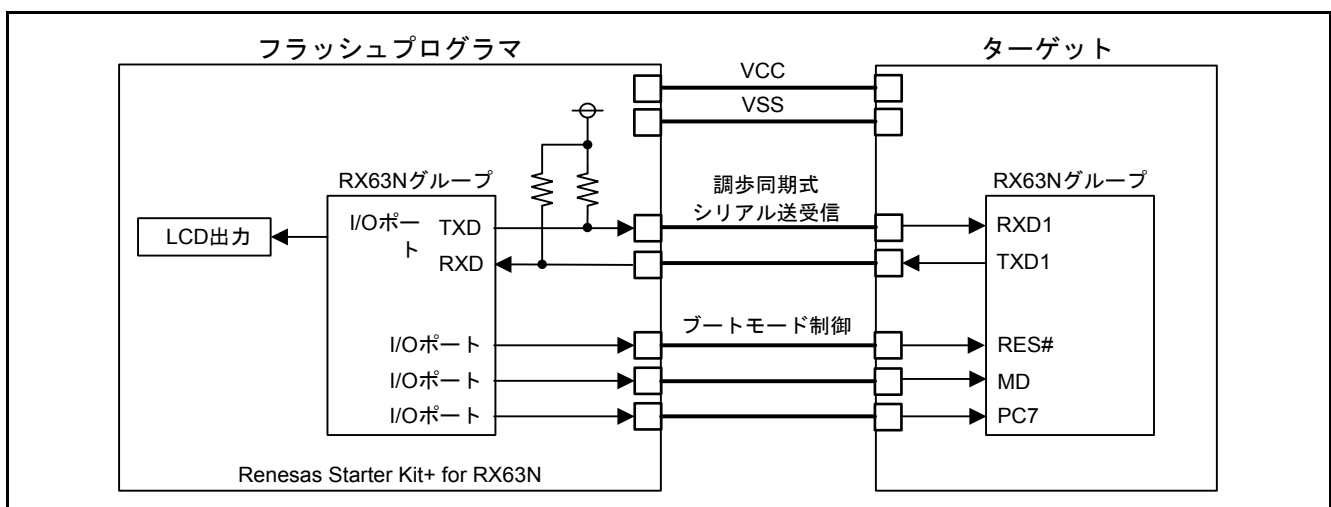


図 1.1 フラッシュプログラマの使用例

## 1.1 RSK+RX63N のユーザ領域メモリ配置

RSK+RX63N のユーザ領域には、フラッシュプログラマのプログラムとターゲットマイコンのユーザ領域に書き込むデータを配置します。図 1.2にRSK+RX63N のユーザ領域メモリ配置を示します。

RSK+RX63N のユーザ領域への書き込みについては「5.1 RSK+RX63N への書き込み方法」を参照してください。

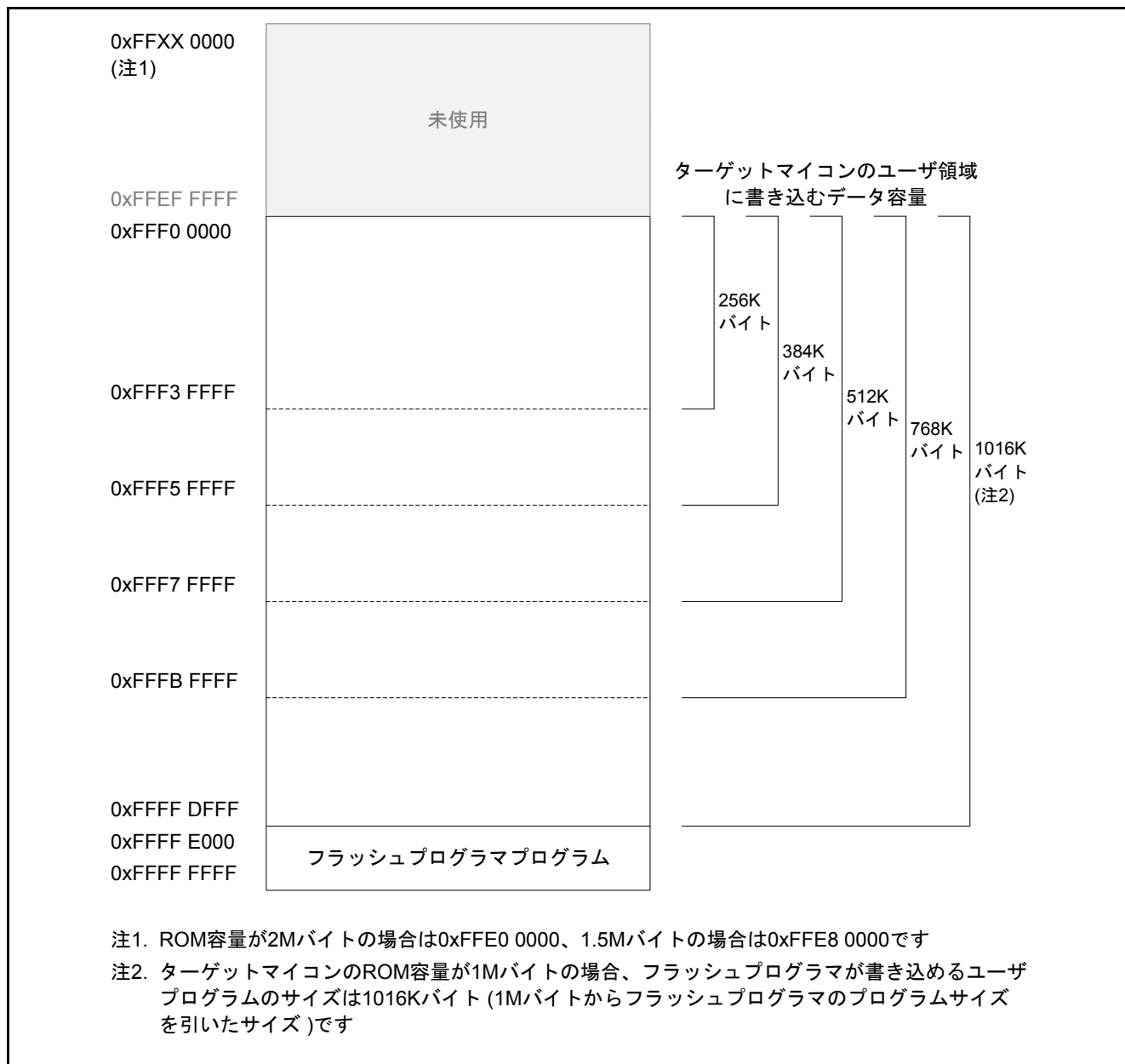


図 1.2 RSK+RX63N のユーザ領域メモリ配置

## 2. 動作確認条件

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

表2.1 動作確認条件

| 項目                | 内容  |
|-------------------|---|
| 使用マイコン            | R5F563NBDDFC (RX63N グループ)   |
| 動作周波数             | メインクロック: 12MHz<br>PLL: 192MHz (メインクロック 1 分周 16 通倍)<br>システムクロック (ICLK): 96MHz (PLL 2 分周)<br>周辺モジュールクロック B (PCLKB): 48MHz (PLL 4 分周)  |
| 動作電圧              | 3.3V  |
| 統合開発環境            | ルネサスエレクトロニクス製<br>High-performance Embedded Workshop Version 4.09.01   |
| C コンパイラ           | ルネサスエレクトロニクス製<br>C/C++ compiler package for RX family V.1.02 Release 01<br>コンパイルオプション<br>-cpu=rx600 -output=obj="\$ (CONFIGDIR)¥\$(FILELEAF).obj" -debug<br>-nologo<br>(統合開発環境のデフォルト設定を使用しています) |
| iodefine.h のバージョン | Version 1.6A  |
| エンディアン            | リトルエンディアン   |
| 動作モード             | シングルチップモード  |
| プロセッサモード          | スーパバイザモード   |
| サンプルコードのバージョン     | Version 1.00  |
| 使用ボード             | Renesas Starter Kit+ for RX63N (製品型名: R0K50563NC010BR)  |

## 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。

RX63N グループ、RX631 グループ 初期設定例 Rev.1.10(R01AN1245JJ)

RX63N High-performance Embedded Workshop 用ルネサススタータキットのサンプルコードRev1.00 (R01AN1395JG)

上記アプリケーションノートの初期設定関数、デバッグ LCD 出力関数を、本アプリケーションノートのサンプルコードで使用しています。Rev は本アプリケーションノート作成時点のものです。

最新版がある場合、最新版に差し替えて使用してください。最新版はルネサスエレクトロニクスホームページで確認および入手してください。

## 4. ハードウェア説明

### 4.1 ハードウェア構成例

図 4.1に接続例を示します。

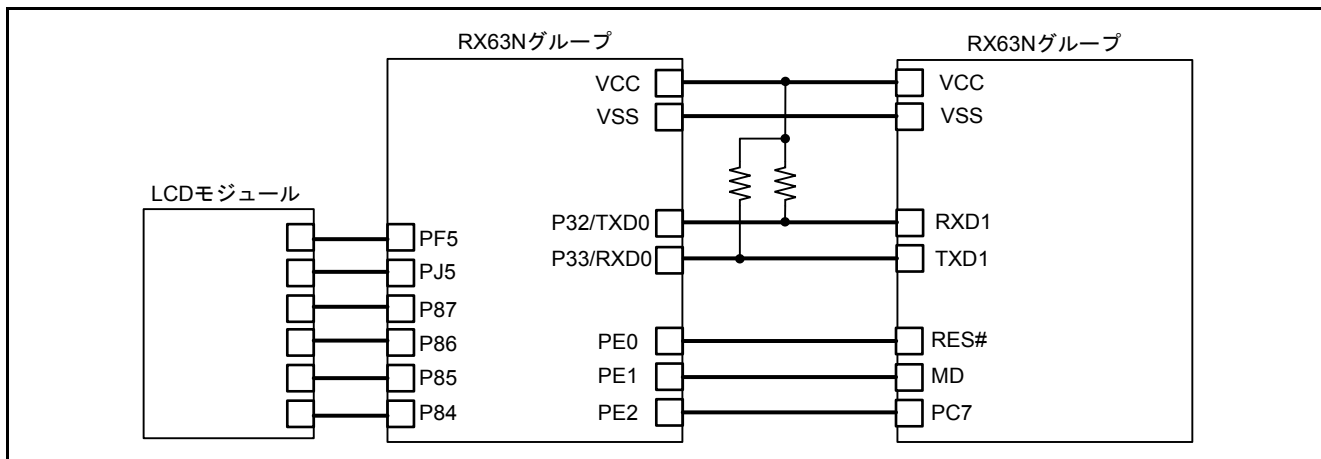


図 4.1 接続例

### 4.2 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                           |
|----------|-----|------------------------------|
| P87      | 出力  | Debug LCD データ 7 出力           |
| P86      | 出力  | Debug LCD データ 6 / バックライト出力   |
| P85      | 出力  | Debug LCD データ 5 / Y ドライブ出力   |
| P84      | 出力  | Debug LCD データ 4 / X ドライブ出力   |
| PF5      | 出力  | Debug LCD Enable 出力          |
| PJ5      | 出力  | Debug LCD Register select 出力 |
| P33/RXD0 | 入力  | SCI0 の受信データ入力端子              |
| P32/TXD0 | 出力  | SCI0 の送信データ出力端子              |
| PE0      | 出力  | RES#端子の制御                    |
| PE1      | 出力  | MD 端子の制御                     |
| PE2      | 出力  | PC7 端子の制御                    |

## 5. ソフトウェア説明

### 5.1 RSK+RX63N への書き込み方法

RSK+RX63N のユーザ領域に書き込むデータは、以下の 2 つです。

- ・ターゲット MCU のユーザ領域に書き込むユーザプログラム
- ・フラッシュプログラマのプログラム

このアプリケーションノートでは、ルネサスフラッシュ開発ツールキット(以下 FDT と略します)を使用した場合を例に説明します。

FDT の S レコード形式、または 16 進数ファイルのエディタ機能を使用して RSK+RX63N のユーザ領域に書き込むデータを結合します。また、FDT を使用して RSK+RX63N のユーザ領域に結合したデータを書き込みます。

FDT の使い方は、ルネサスフラッシュ開発ツールキットのユーザーズマニュアル (R20UT0508JJ)を参照してください。

図 5.1にRSK+RX63N への書き込み方法の流れを示します。

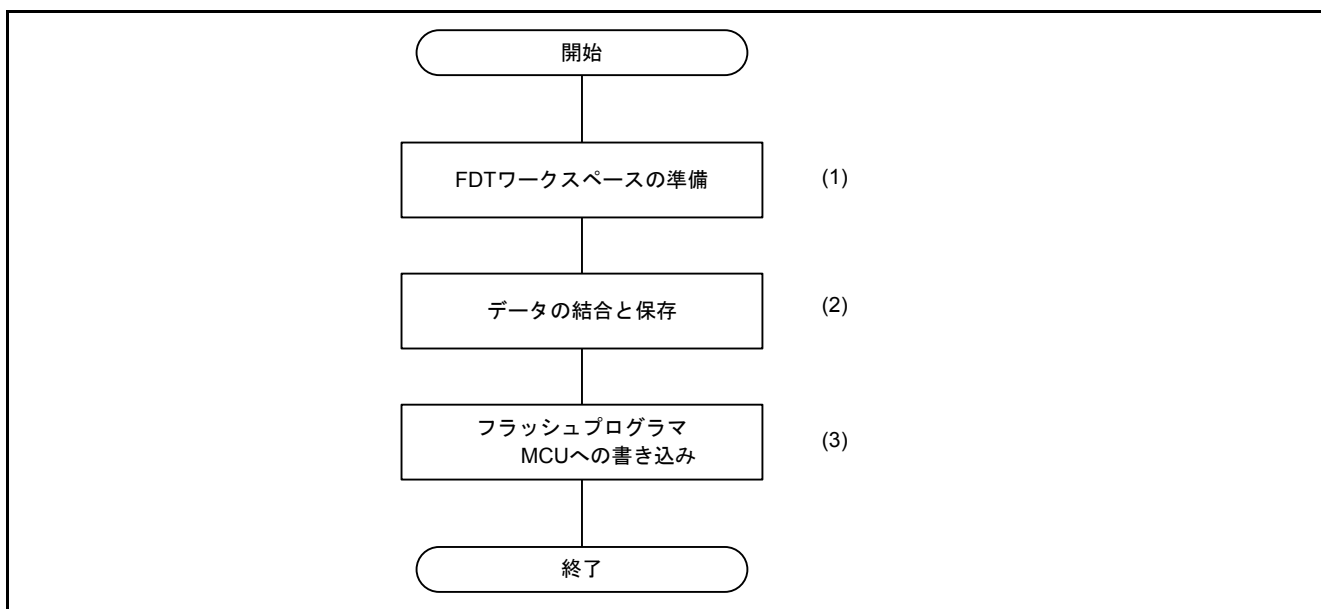


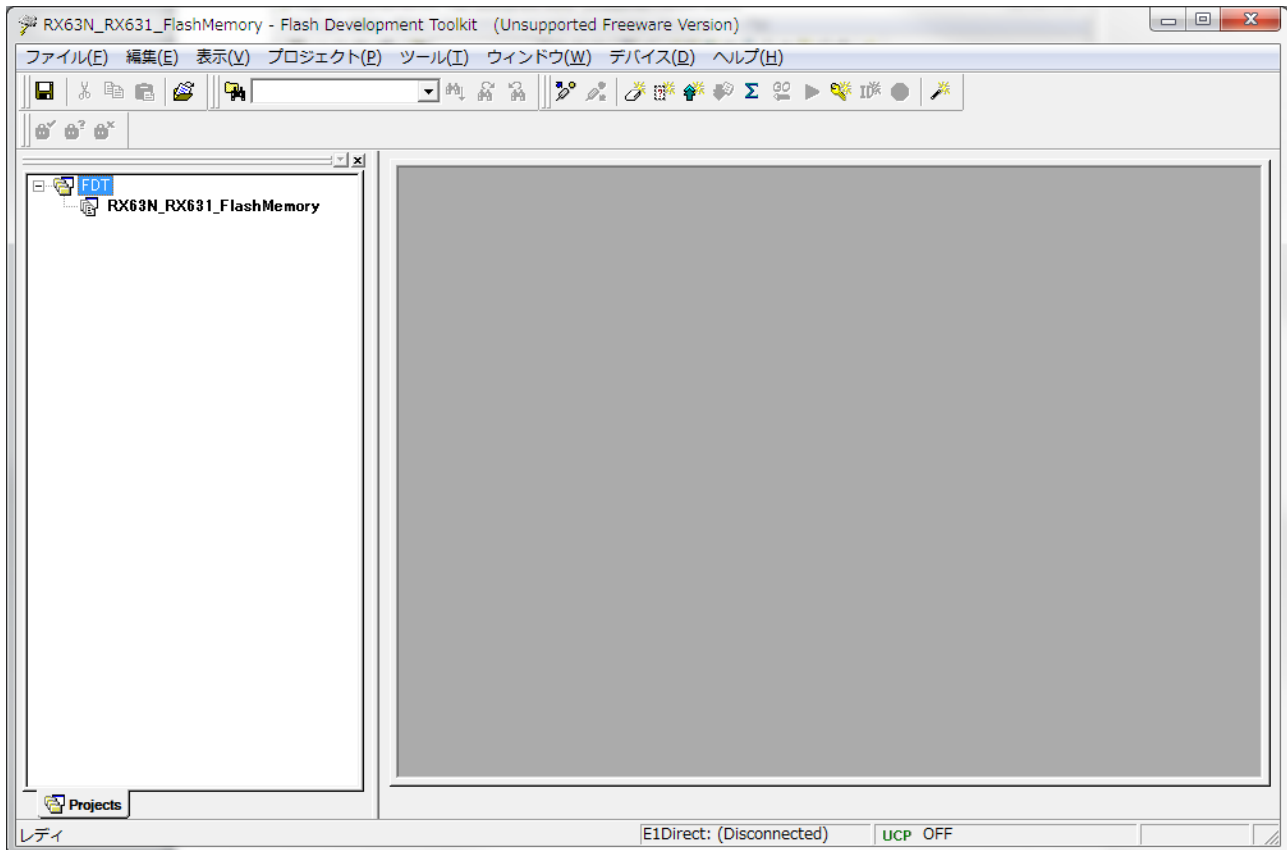
図 5.1 RSK+RX63N への書き込み方法の流れ

- (1) 「5.1.1 FDT ワークスペースの準備」を参照してください。
- (2) 「5.1.2 データの結合と保存」を参照してください。
- (3) 「5.1.3 RSK+RX63N のユーザ領域書き込み」を参照してください。

### 5.1.1 FDT ワークスペースの準備

FDT を使用するためにワークスペースとプロジェクトを作成します。ターゲットデバイスはフラッシュプログラマに使用する MCU を指定してください。

例では「ワークスペース名(W)」を“FDT”、「プロジェクト名(P)」を“RX63N\_RX631\_FlashMemory”としています。

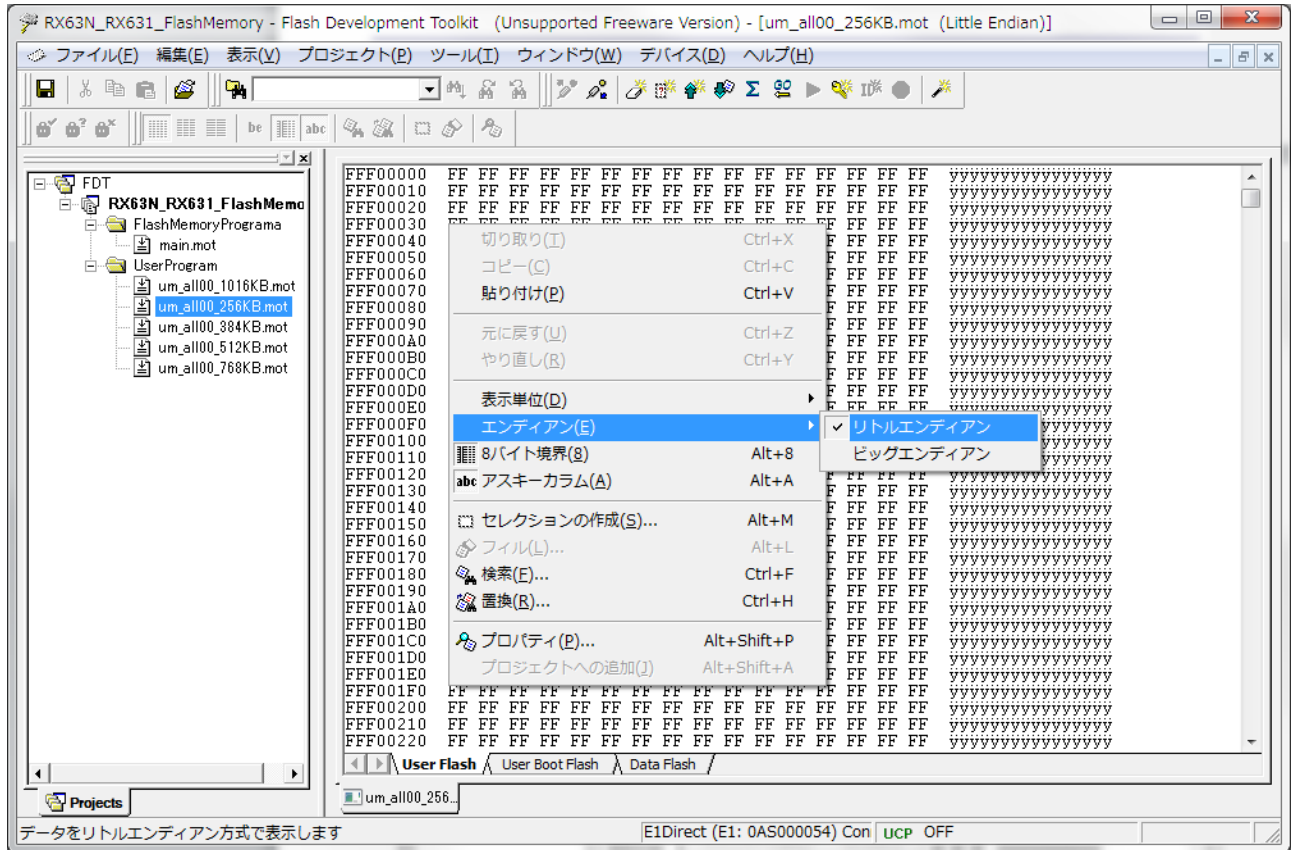






(2) 結合するデータファイルを 16 進数エディタウィンドウに開き、エンディアンを指定します。

例では “main.mot” ファイルと “um\_all00\_256KB.mot” ファイルを 16 進数エディタウィンドウに開いています。また、エンディアンは 2 つのファイル共、リトルエンディアンを選択しています。

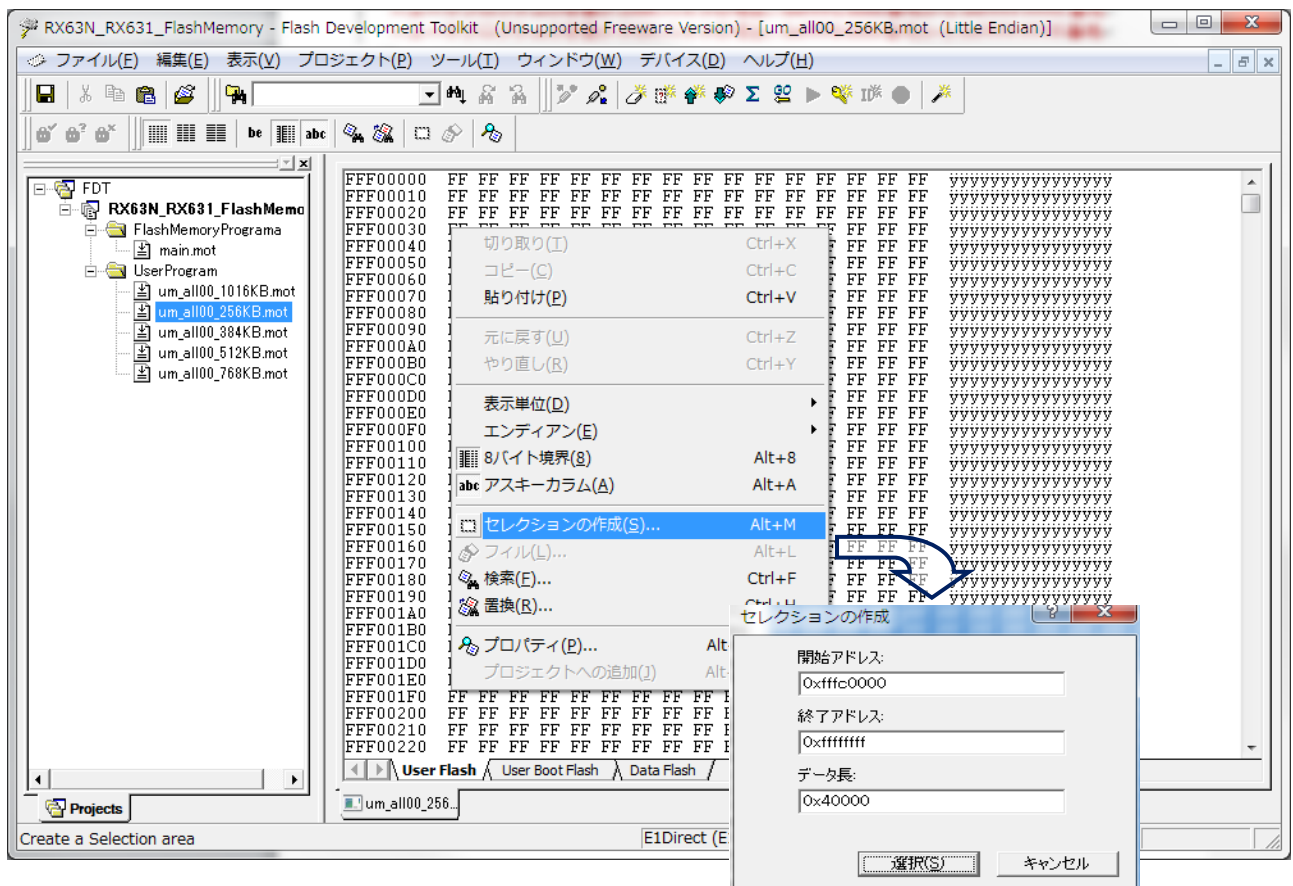


(3) 結合するターゲット MCU のユーザ領域に書き込むユーザプログラムデータを選択します。

選択する範囲は以下のとおりです。

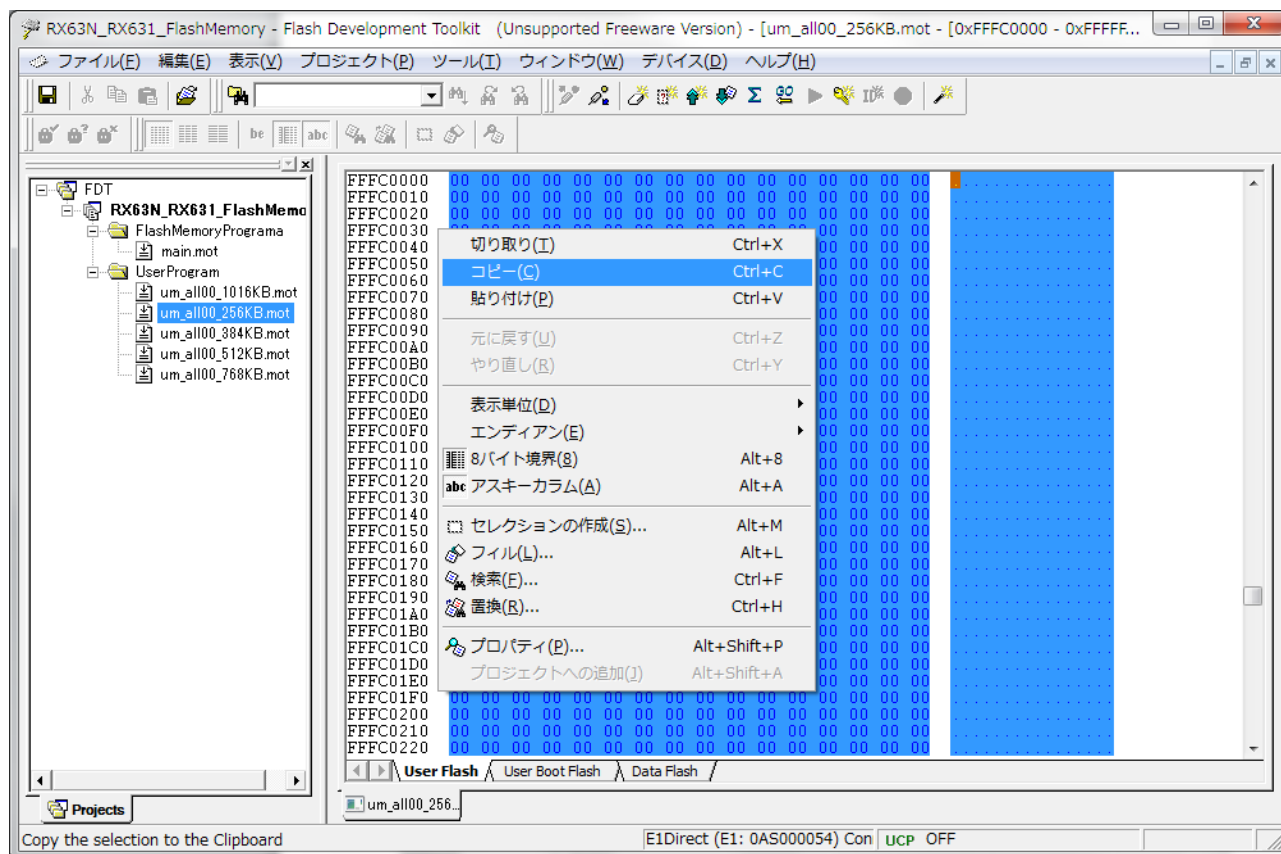
- ・ユーザプログラムの容量が 256K バイトの場合、0xFFFFC 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFFFA 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFFF8 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFFF4 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFFF0 2000 番地から 0xFFFF FFFF 番地

例では “um\_all00\_256KB.mot” ファイルの 0xFFFFC 0000 番地から 0xFFFF FFFF 番地を選択します。



- (4) 反転表示しているターゲット MCU のユーザ領域に書き込むユーザプログラムデータを Windows のクリップボードにコピーします。

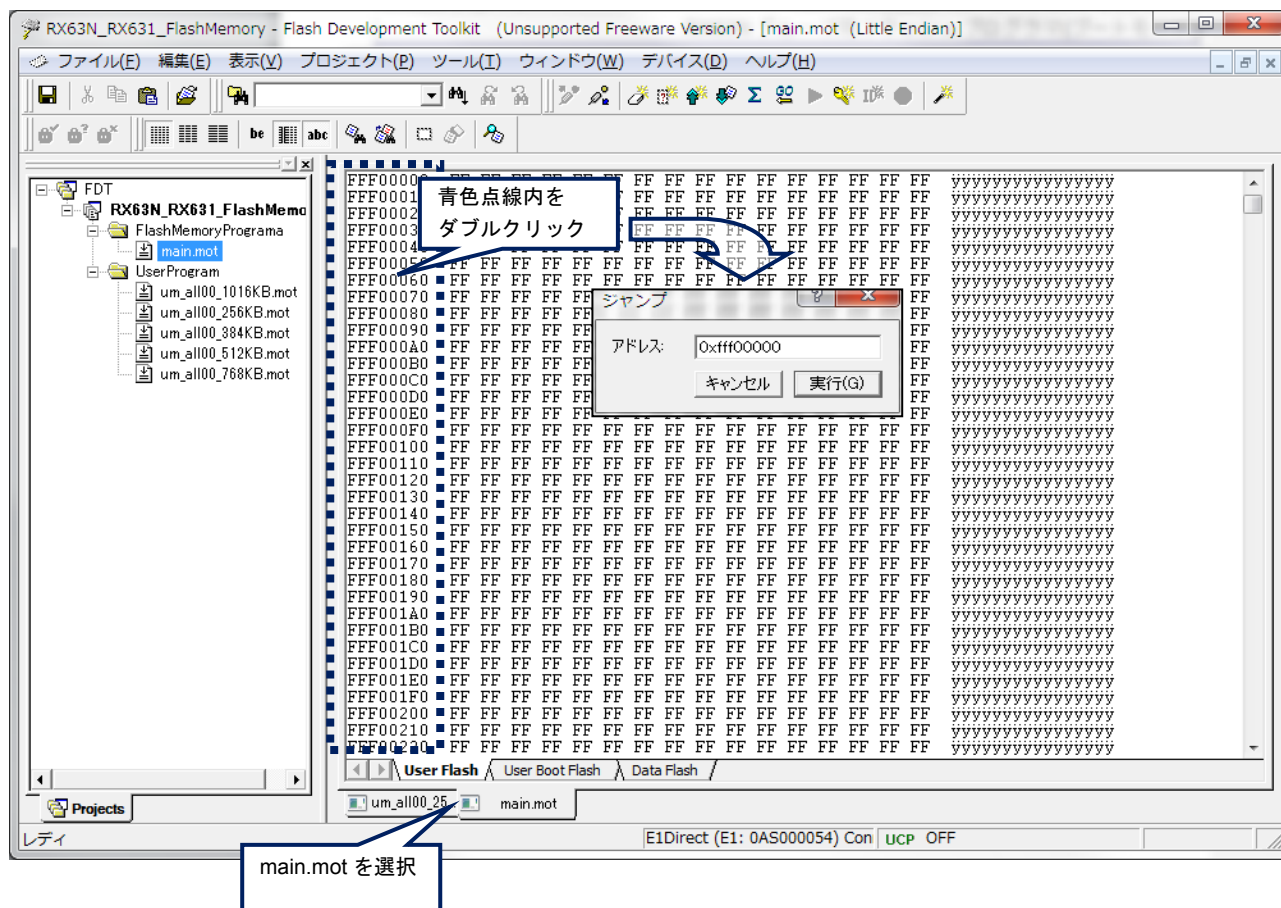
例では“um\_all00\_256KB.mot” ファイルの 0xFFFFC 0000 番地から 0xFFFF FFFF 番地を Windows のクリップボードにコピーします。



(5) RSK+RX63N のユーザ領域に書き込むデータを結合して作成します。

16 進数エディタウィンドウに“main.mot”ファイルを選択し、前項(4)で Windows のクリップボードにコピーしたデータを 0xFFFF0 0000 番地以降に貼り付けます。

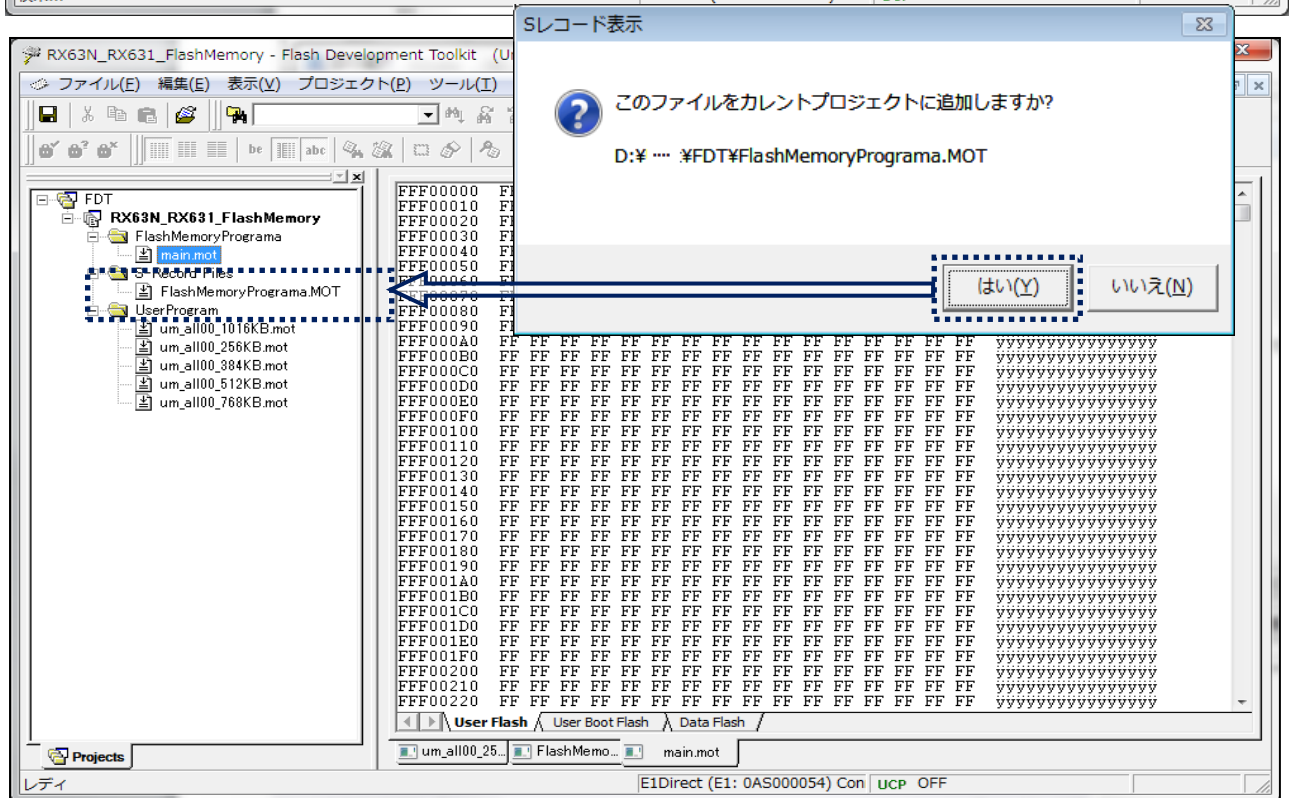
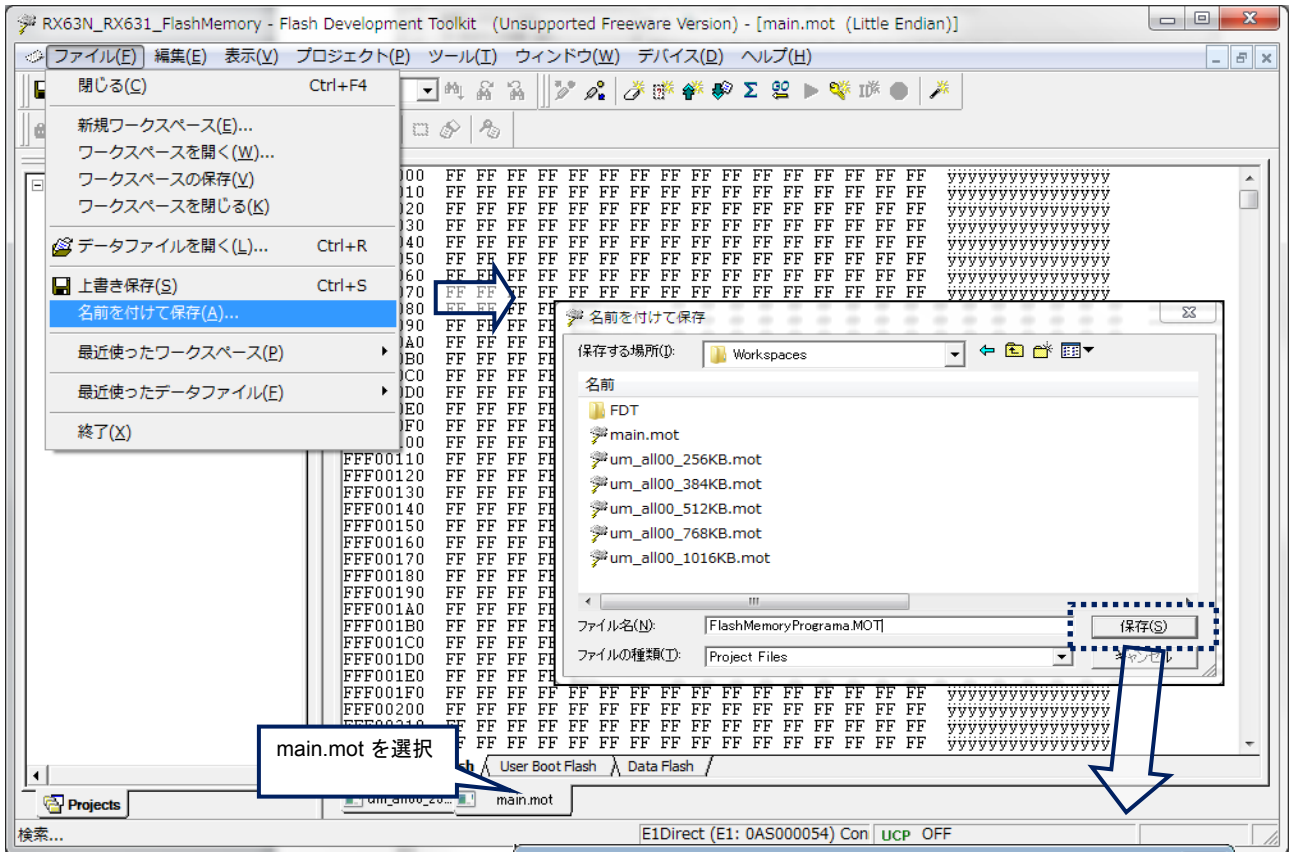
例は“main.mot”ファイルの貼り付け先の開始アドレスを 0xFFFF0 0000 番地に指定しています。その後、クリップボードのデータを貼り付けてください。



(6) RSK+RX63N のユーザ領域に書き込むデータを保存します。

16 進数エディタウィンドウに“main.mot”ファイルを選択し、前項(5)で作成したデータに名前を付けてファイルに保存し、プロジェクトに追加します。

例では“S-Record Files”フォルダに“FlashMemoryPrograma.MOT” ファイルを保存します。



(7) RSK+RX63N のユーザ領域に書き込むデータを確認します。

前項(6)で作成したデータファイルの結合したデータの配置が正しいか確認します。

ワークスペースウィンドウで RSK+RX63N のユーザ領域に書き込むデータファイルを選択し、使用ブロックのアドレス範囲を確認します。

確認する使用ブロックのアドレス範囲は以下のとおりです。

- ・ユーザプログラムの容量が 256K バイトの場合、0xFFF0 0000 番地から 0xFFF3 FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFF0 0000 番地から 0xFFF5 FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFF0 0000 番地から 0xFFF7 FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFF0 0000 番地から 0xFFFB FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFF0 0000 番地から 0xFFFF DFFF 番地
- ・フラッシュプログラマのプログラムは、0xFFFF E000 番地から 0xFFFF FFFF 番地

例ではユーザプログラムの容量が 256K バイトの場合の使用ブロックのアドレス範囲であることを確認しています。

開く FlashMemoryPrograma.MOT

- ファイルの追加... INS
- ファイルの削除...
- ドッキングビュー
- 非表示
- プロパティ
- 使用ブロックの表示...**
- Exclude FlashMemoryPrograma.MOT
- User Boot Flash
- ダウンロード [User/Data Area]
- ファイルのチェックサム
- ファイルの比較->デバイスチェックサム
- ファイルの比較->リ
- ファイルの比較->リ

データファイルプロパティ 'FlashMemoryPrograma.MOT'

| 使用ブロック                  | カーソル値      | チェックサム |
|-------------------------|------------|--------|
| Address                 | Length     |        |
| 0xFFF00000 - 0xFFF3FFFF | 0x00040000 |        |
| 0xFFFFE000 - 0xFFFFE030 | 0x00000031 |        |
| 0xFFFFE034 - 0xFFFFE5F1 | 0x000005BE |        |
| 0xFFFFE5F4 - 0xFFFFF526 | 0x00000F33 |        |
| 0xFFFFF528 - 0xFFFFF54B | 0x00000024 |        |
| 0xFFFFF5D0 - 0xFFFFF5FF | 0x00000030 |        |

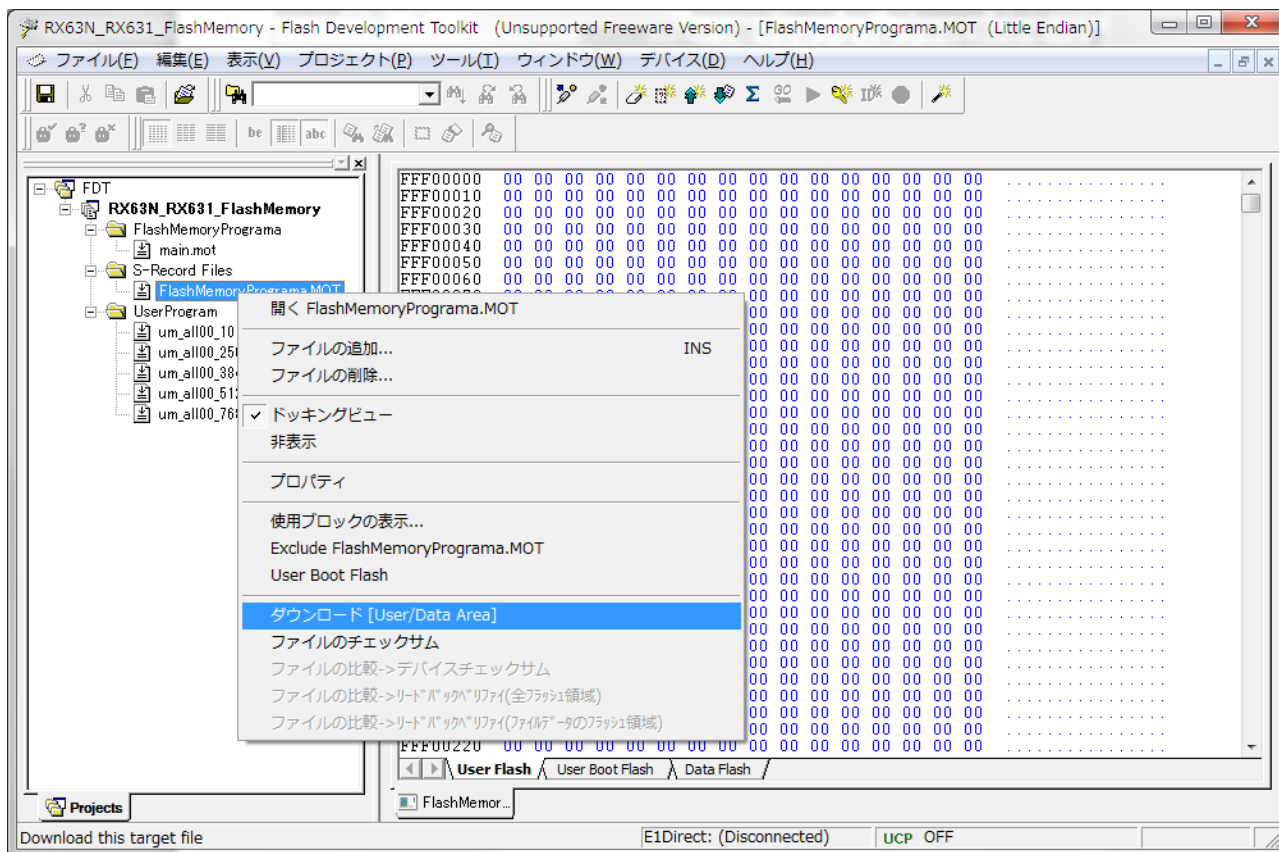
[0xFFF0 0000 から 0xFFF3 FFFF]  
ユーザプログラムのアドレス範囲

[0xFFFF E000 から 0xFFFF FFFF]  
フラッシュプログラマ プログラム  
の使用するアドレス範囲

## 5.1.3 RSK+RX63N のユーザ領域書き込み

ワークスペースウィンドウで RSK+RX63N のユーザ領域に書き込むデータファイルを選択してダウンロードします。

例では“S-Record Files”フォルダの“FlashMemoryPrograma.MOT” ファイルをダウンロードします。





## 5.2 動作概要

ターゲット MCU をブートモードで起動してビットレート自動調整を行い通信ビットレート 19,200bps で接続します。

接続後、サポートデバイス問い合わせコマンドを送信してターゲット MCU の情報を取得します。その後、デバイス選択コマンド、クロックモード選択コマンドを送信し、新ビットレート選択コマンドを送信して通信ビットレートを 1.5Mbps に変更します。

P/E ステータス遷移コマンドを送信してターゲット MCU の ID コードプロテクト状態を判断し ID コードプロテクト(ブートモード)の処置を実行します。

取得したターゲット MCU の情報よりターゲット MCU にユーザプログラムを書き込みます。書き込み完了後、ターゲット MCU に書き込んだ領域を読み出して書き込んだデータとベリファイチェックします。

図 5.2に図 5.2 フラッシュプログラマの状態遷移を示します。

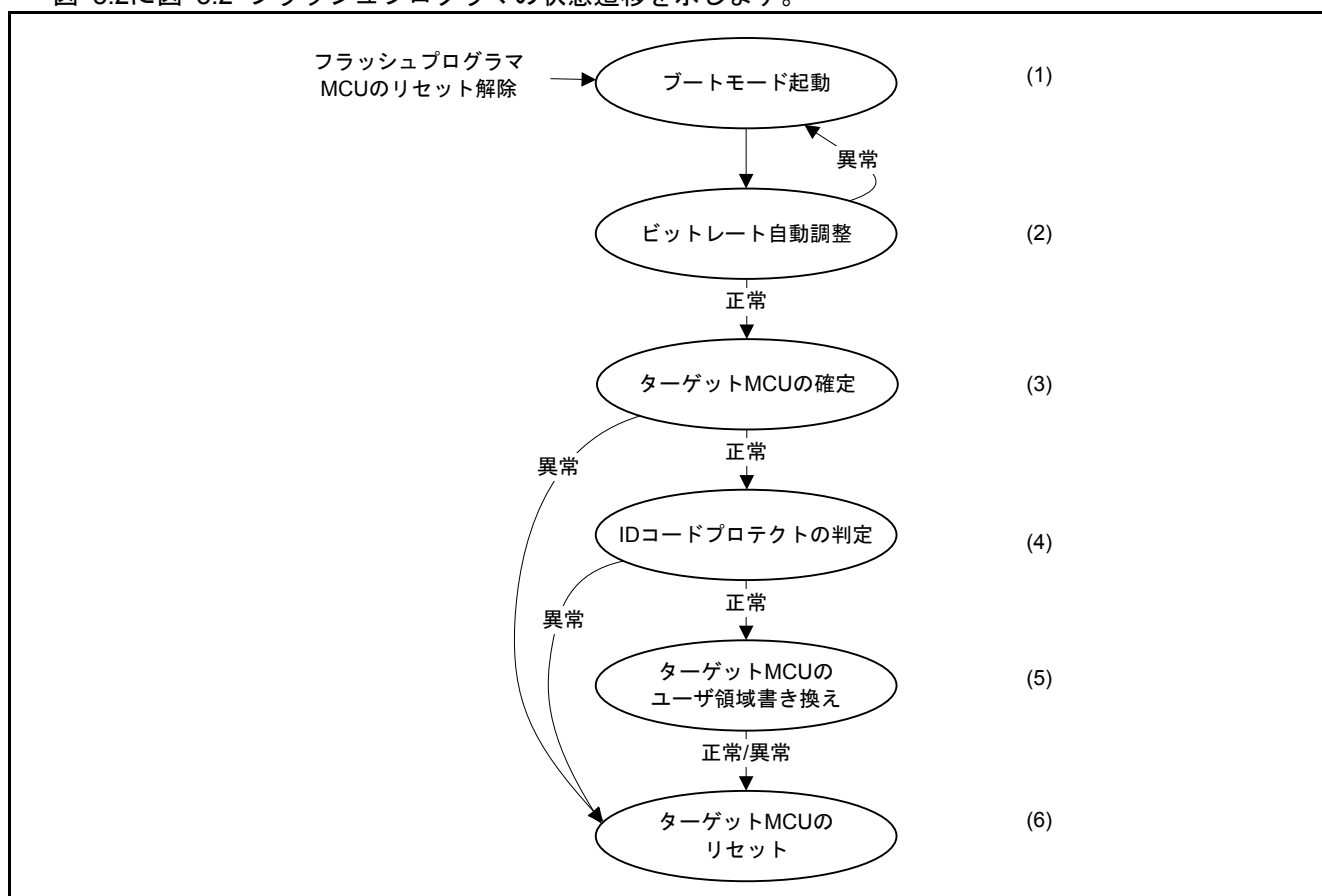


図 5.2 フラッシュプログラマの状態遷移

- (1) 詳細は「5.2.1 ブートモード起動」を参照してください。
- (2) 詳細は「5.2.2 ビットレート自動調整」を参照してください。
- (3) 詳細は「5.2.3 ターゲット MCU の確定」を参照してください。
- (4) 詳細は「5.2.4 ID コードプロテクトの判定」を参照してください。
- (5) 詳細は「5.2.5 ターゲット MCU のユーザ領域への書き込み」を参照してください。
- (6) 詳細は「5.2.6 ターゲット MCU のリセット」を参照してください。

## 5.2.1 ブートモード起動

- (1) ターゲット MCU の RES#端子を“L” にします。
- (2) ターゲット MCU の MD 端子を“L” にします。
- (3) ターゲット MCU の PC7 端子を“L” にします。
- (4) 2ms のウェイト後、ターゲット MCU の RES#端子を“H” にします。

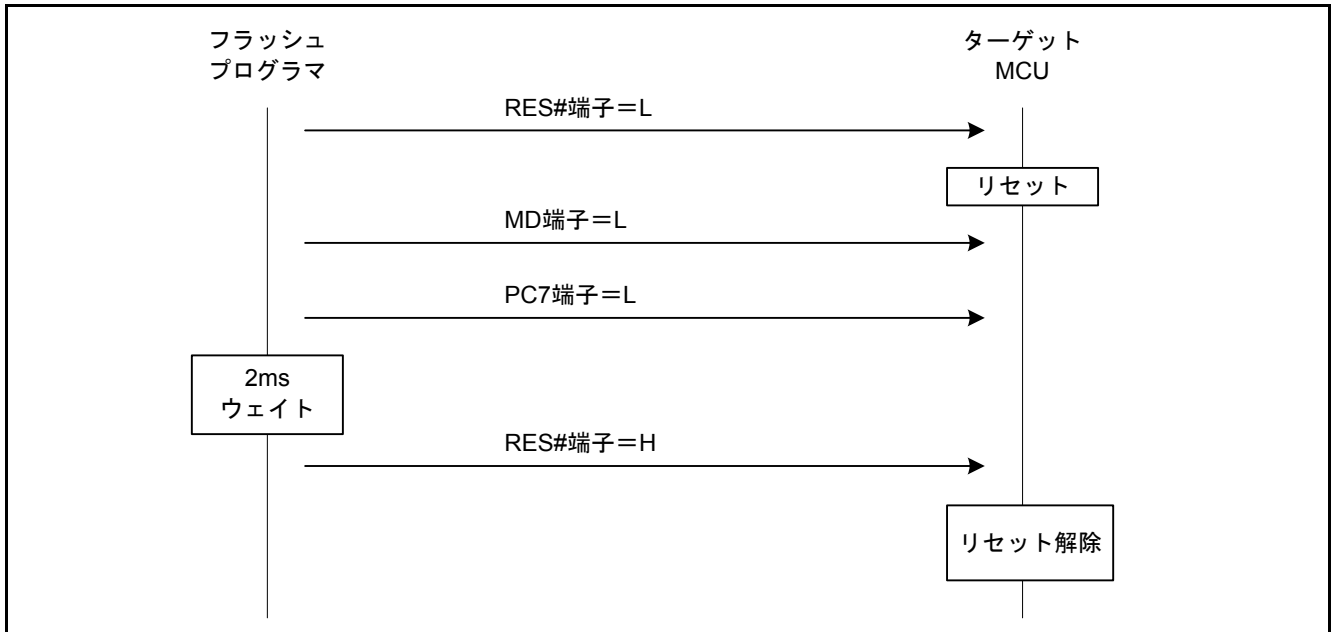


図 5.3 ブートモード起動の手順

## 5.2.2 ビットレート自動調整

フラッシュプログラマは、ターゲット MCU をブートモードで起動して、400ms 経過した後にビットレートを 19,200bps に調整するために “00h” を 30 回送信します。

フラッシュプログラマが “00h” を受信した場合、ターゲット MCU に “55h” を送信してください。“00h” を受信できなかった場合、ターゲット MCU をブートモードで再起動し、再度ビットレート自動調整を実施します。

フラッシュプログラマは “55h” を送信後に “E6h” を受信することでビットレートの自動調整を完了します。“55h” を送信後に “FFh” を受信した場合、ターゲット MCU をブートモードで再起動し、再度ビットレート自動調整を実施します。

図 5.4にビットレート自動調整の手順を示します。

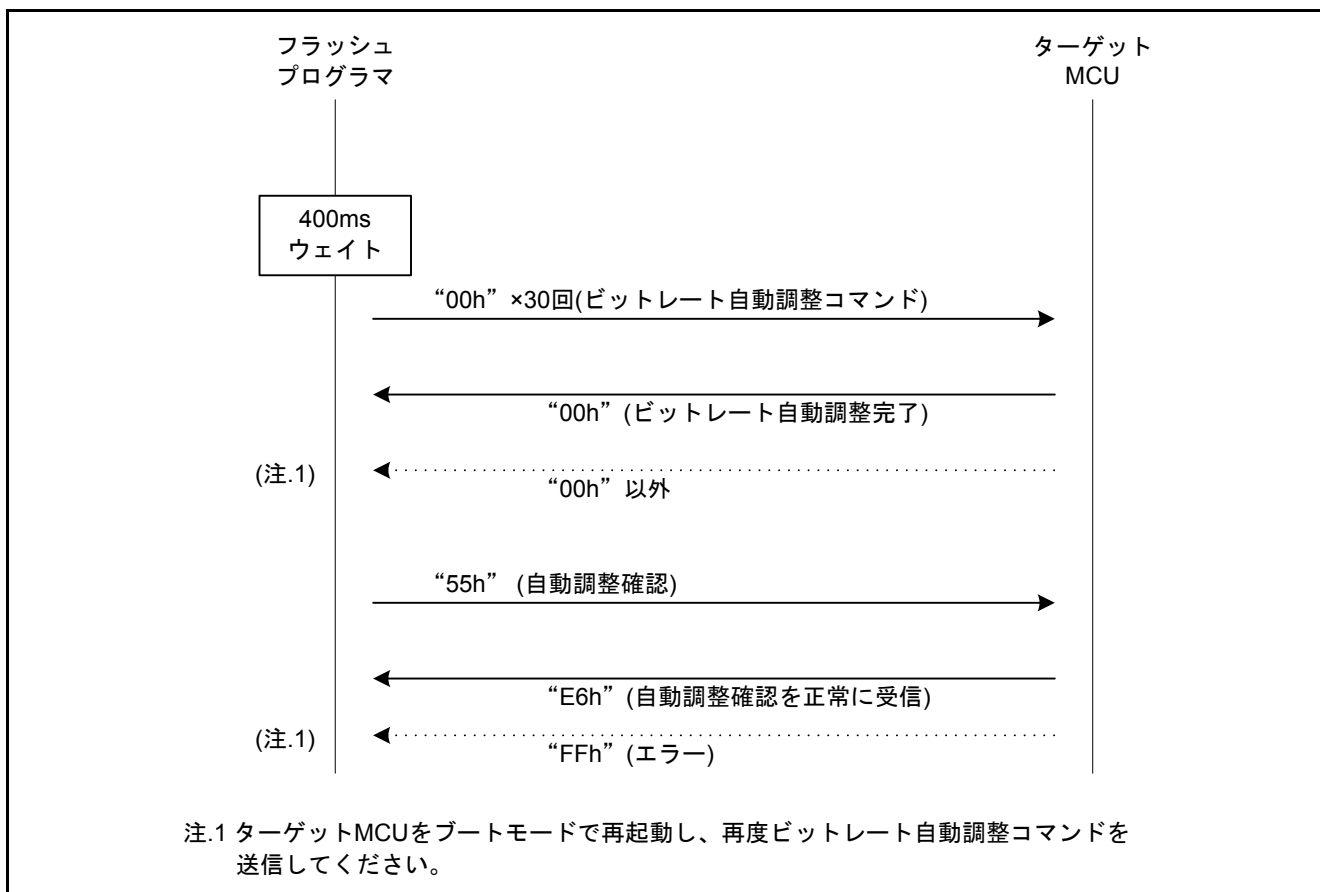


図 5.4 ビットレート自動調整の手順

## 5.2.3 ターゲット MCU の確定

ターゲット MCU の確定は以下の(1)から(4)の処理を行います。

- (1) サポートデバイス問い合わせコマンドを送信してユーザ領域に書き込むデータのエンディアンを選択するデバイスコードを記憶します。

フラッシュプログラマはサポート問い合わせコマンドのレスポンス(30h から始まるデータ)を受信することでユーザ領域に書き込むデータのエンディアンを選択するデバイスコードを記憶します。レスポンス(30h から始まるデータ)以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.5にデバイスコードの記憶手順を示します。

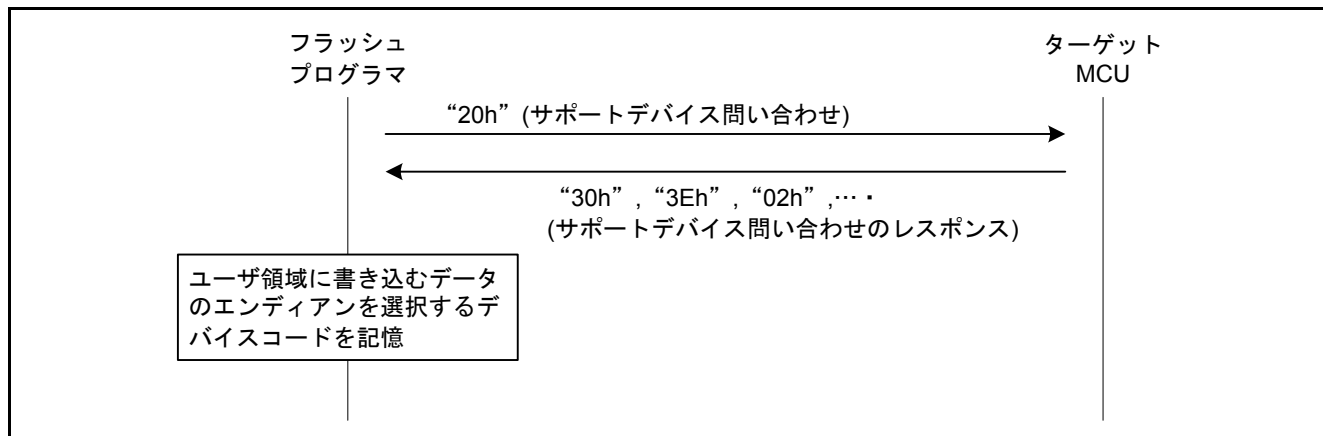


図 5.5 デバイスコードの記憶手順

- (2) デバイス選択コマンドを送信して、ユーザ領域に書き込むデータのエンディアンを選択します。

フラッシュプログラマはユーザ領域に書き込むデータのエンディアンを指定するため、デバイス選択コマンド(10h)を送信します。このとき、指定するデバイスコードは、サポートデバイス問い合わせコマンドで記憶したデバイスコードの内、フラッシュプログラマのエンディアンに合わせたデバイスコードを用います。

フラッシュプログラマはデバイス選択コマンドを送信後にレスポンス(06h)を受信することでエンディアンの選択を完了します。デバイス選択コマンドを送信後にレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.6にエンディアンの選択手順を示します。

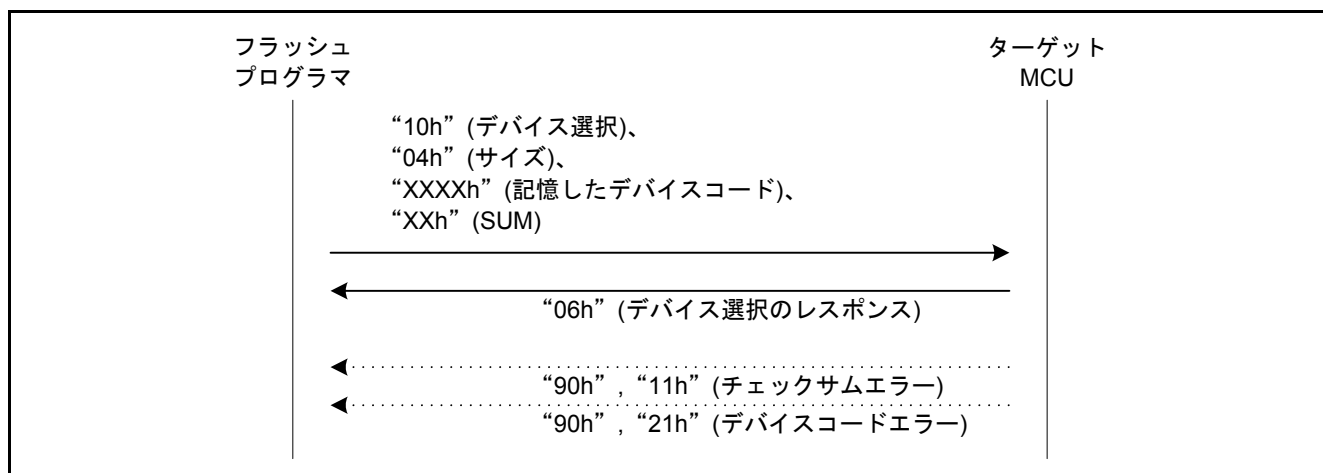


図 5.6 エンディアンの選択手順

(3) クロックモード選択コマンドを送信してターゲット MCU のクロックモードを選択します。

フラッシュプログラマはクロックモードを指定するため、クロックモード選択コマンド(11h)を送信します。クロックモード選択のレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.7にクロックモード選択の手順を示します。

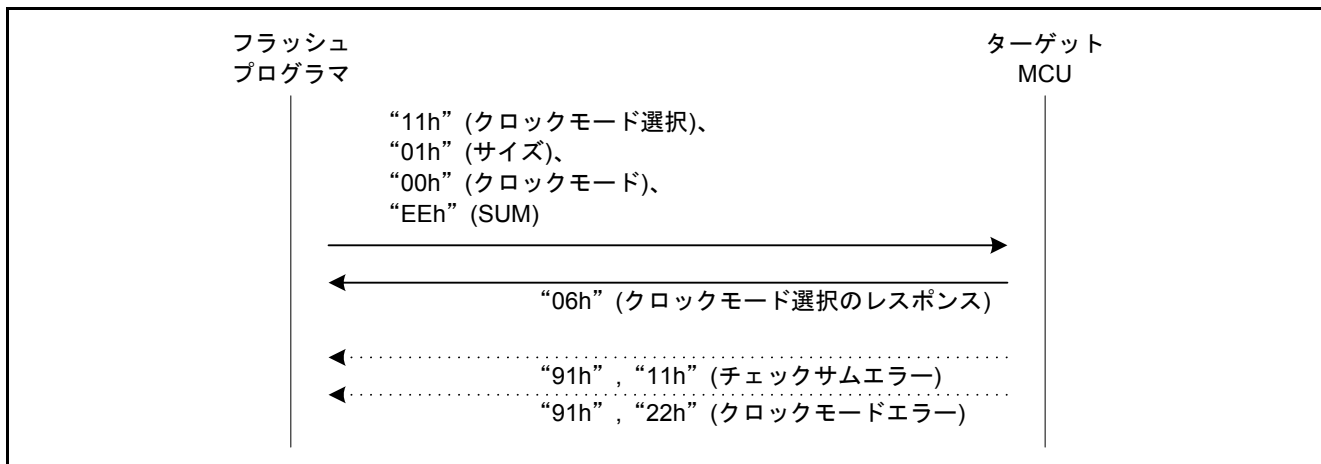


図 5.7 クロックモード選択の手順

- (4) 新ビットレート選択コマンドを送信して、ターゲット MCU との通信ビットレートを 1.5Mbps に変更します。

フラッシュプログラマはビットレートを 1.5Mbps に変更するため、新ビットレート選択コマンド(3Fh)を送信します。また、このときに送信する入力周波数は、ターゲット MCU の入力周波数です。フラッシュプログラマは新ビットレート選択コマンドを送信後に ACK(06h)を受信すると新ビットレート選択コマンド送信時のビットレートで 25 ミリ秒ウェイトし、ビットレートを 1.5Mbps に変更します。その後、変更後のビットレートで通信確認データ(06h)を送信し、通信確認データのレスポンス(06h)を受信することでビットレートの変更を完了します。

フラッシュプログラマは新ビットレート選択コマンドを送信後にレスポンス(06h)以外を受信した場合や、通信確認データ(06h)を送信後に、通信確認データのレスポンス(06h) 以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.8にビットレートの変更手順を示します。

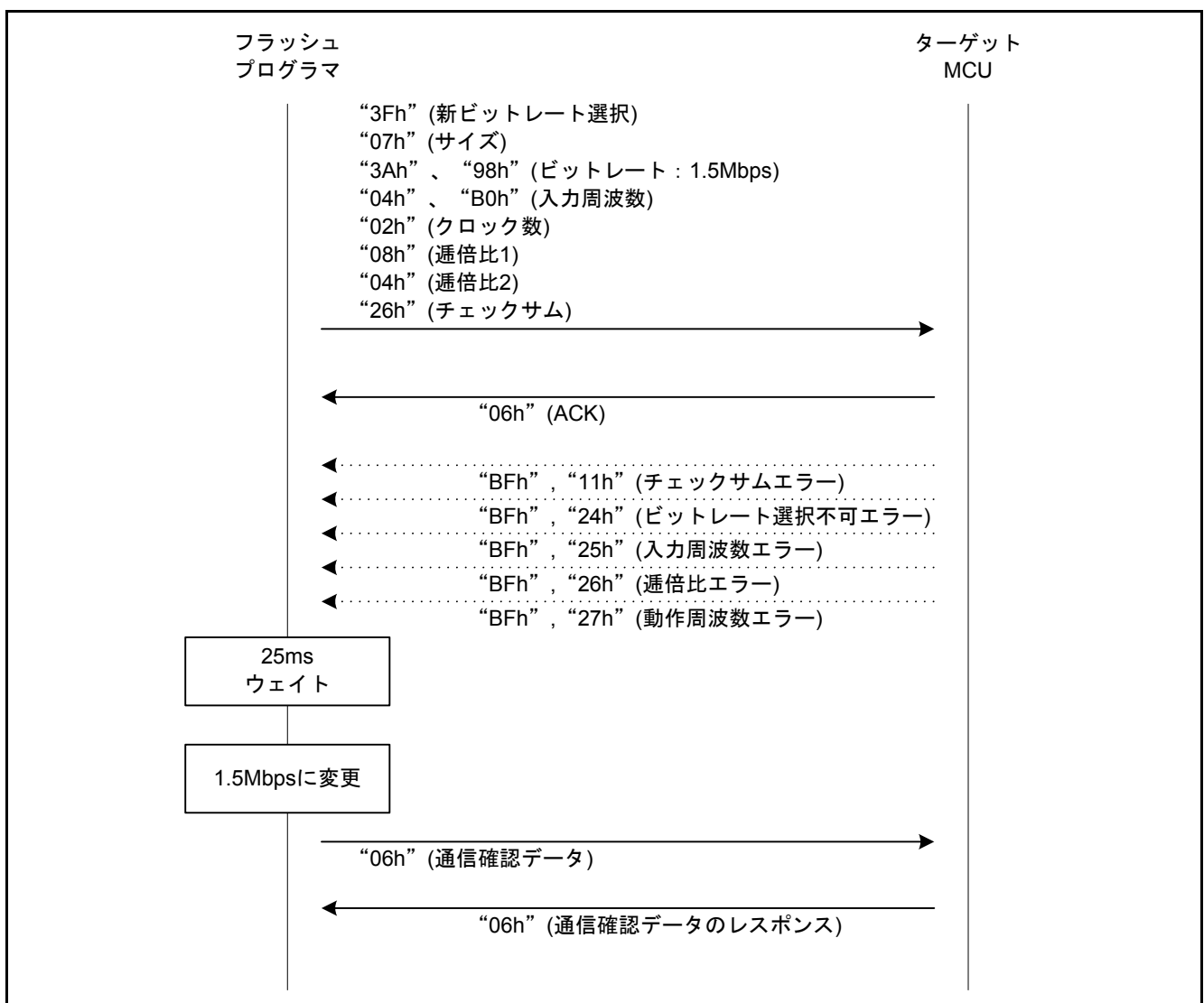


図 5.8 ビットレートの変更手順

## 5.2.4 ID コードプロテクトの判定

ID コードプロテクトの判定は以下の(1)~(3)の処理を行います。

- (1) P/E ステータス遷移コマンドを送信してターゲット MCU の ID コードプロテクト状態を判定し記録します。

フラッシュプログラマはターゲット MCU の ID コードプロテクト状態を判定するため、P/E ステータス遷移コマンド(40h)を送信します。

フラッシュプログラマは P/E ステータス遷移コマンドを送信後に受信したレスポンスを判定して ID コードプロテクト状態判定バッファに記録します。

判定するレスポンスと ID コードプロテクト状態判定バッファへの記録値を表 5.1 に示します。

表5.1 ID コードプロテクト状態判定バッファへの記録値

| レスポンス | ID コードプロテクト状態判定バッファへの記録値 |
|-------|--------------------------|
| "26h" | "00h"                    |
| "16h" | "01h"                    |

フラッシュプログラマは P/E ステータス遷移コマンドを送信後に受信したレスポンス値が表 5.1 にない値の場合、ターゲット MCU をリセットして中断します。

図 5.9 に P/E ステータス遷移コマンドの ID コードプロテクト状態判定の手順を示します。

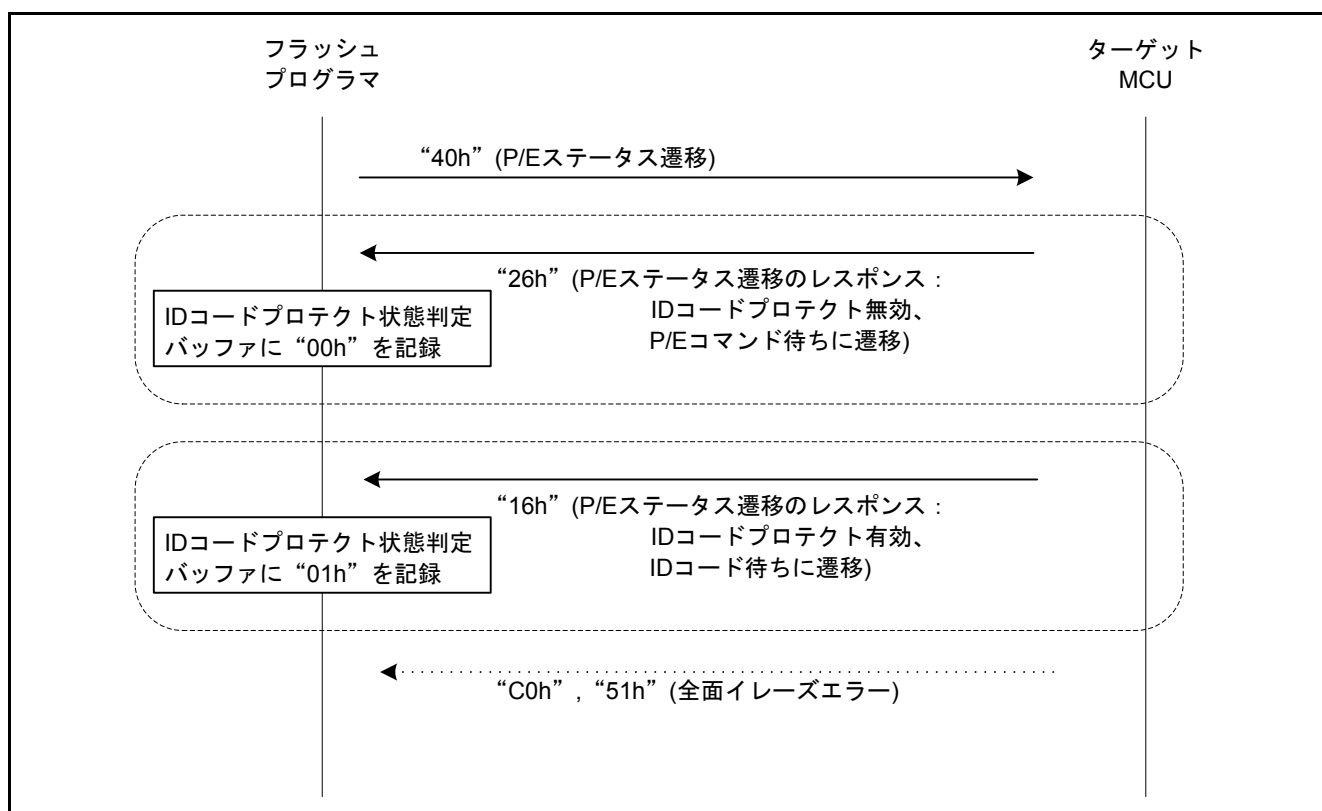


図 5.9 P/E ステータス遷移コマンドの ID コードプロテクト状態判定の手順

- (2) IDコードチェックコマンドを送信してターゲットMCUのIDコードプロテクト状態を判定し記録します。

この処理はIDコードプロテクト状態判定バッファの記録値が“01h”の場合に実施します。

フラッシュプログラマはターゲットMCUのIDコードプロテクト状態を判定するため、IDコードチェックコマンド(60h)を送信します。このとき、指定する制御コード、およびIDコード1からIDコード15はターゲットMCUのユーザ領域に書き込むデータを読み出して用います。フラッシュプログラマは、IDコードチェックコマンドを送信後に受信したレスポンス値が26h以外の場合、ターゲットMCUをリセットして中断します。

図 5.10にIDコードチェックコマンドのIDコードプロテクト状態判定の手順を示します。

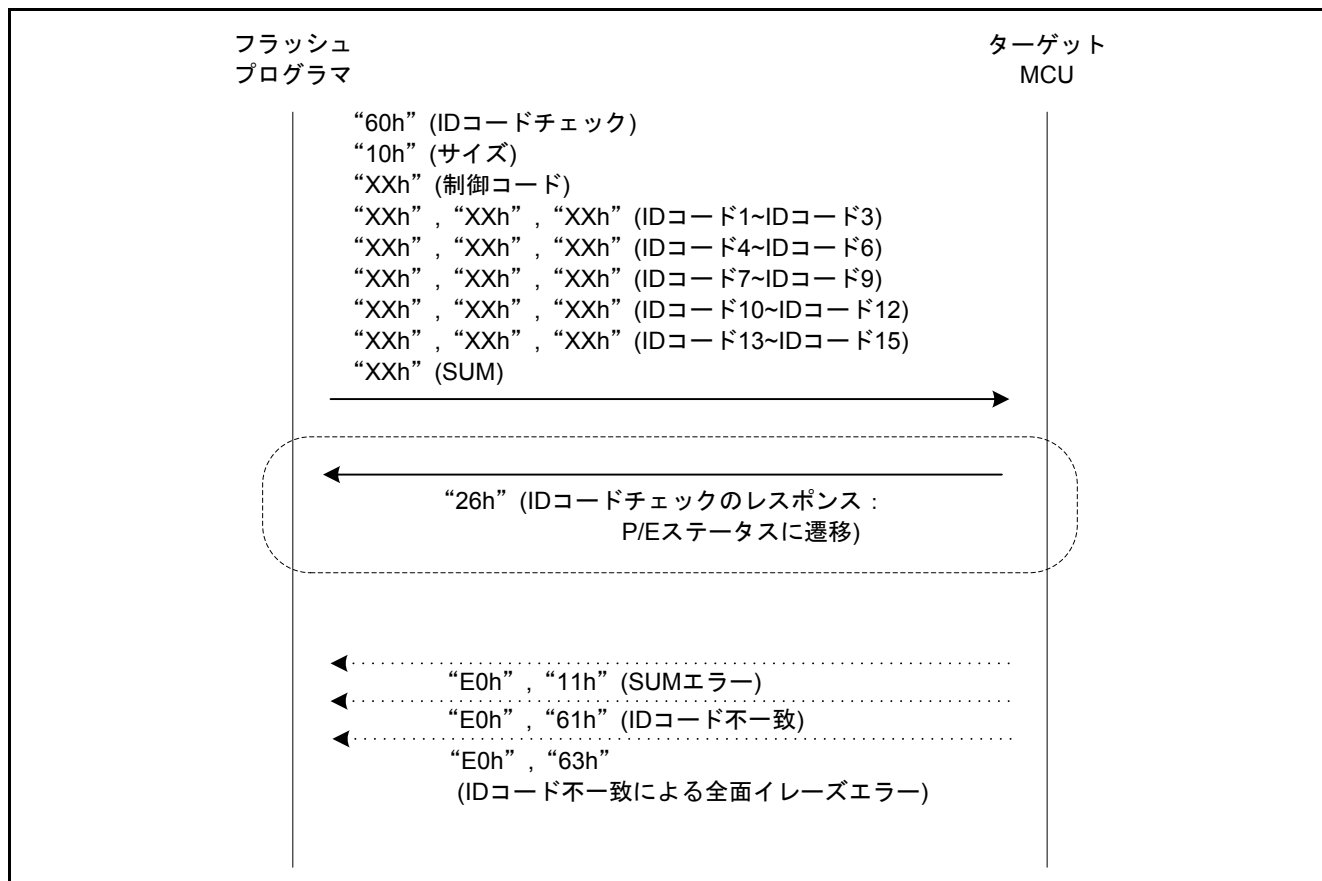


図 5.10 IDコードチェックコマンドのIDコードプロテクト状態判定の手順



- (3) ターゲット MCU のユーザ領域にユーザプログラムを書き込むため、ターゲット MCU のフラッシュメモリを消去します。

この処理は ID コードプロテクト状態判定バッファの記録値が “01h” の場合に実施します。

フラッシュプログラマは先ずイレーズ選択コマンド(48h)を送信します。イレーズ選択コマンドを送信後にイレーズ選択のレスポンス(06h)を受信することでイレーズ選択を完了します。イレーズ選択コマンドを送信後にイレーズ選択のレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

次にブロックイレーズコマンド(58h)の送信を、ターゲット MCU に存在するブロック数分繰り返します。

ターゲット MCU に存在するブロック数分のブロックイレーズコマンドを送信すると、ブロックイレーズを終了するブロックイレーズコマンド(58h 04h FFh A5h)を送信します。ブロックイレーズコマンドを送信後にブロックイレーズのレスポンス(06h)を受信すると、1 回のブロックイレーズを完了します。ブロックイレーズコマンドを送信後にブロックイレーズのレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.11にターゲット MCU のフラッシュメモリ消去の手順を示します。

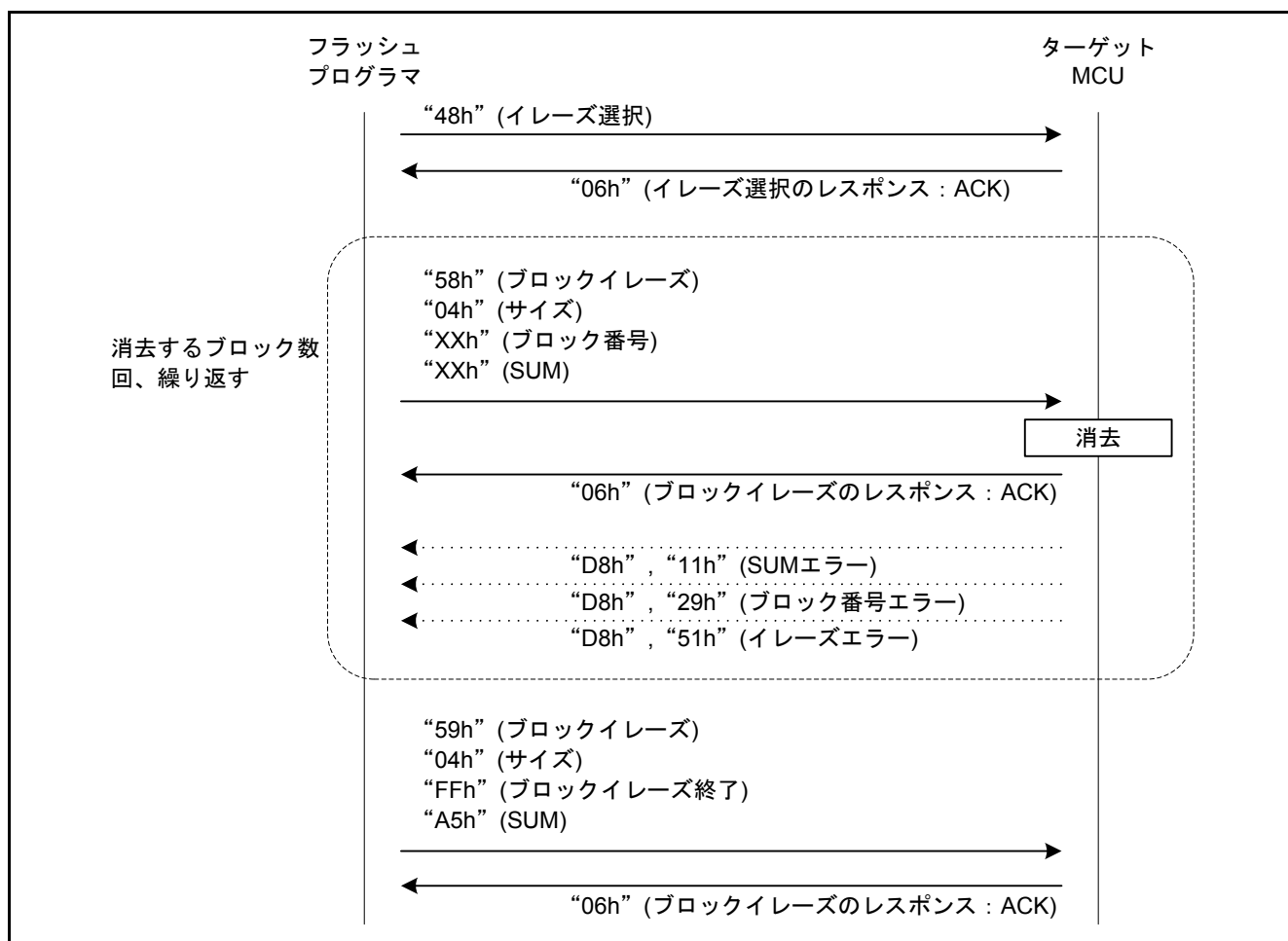


図 5.11 ターゲット MCU のフラッシュメモリ消去の手順

### 5.2.5 ターゲット MCU のユーザ領域への書き込み

ターゲット MCU のユーザ領域への書き込みは以下の(1)から(2)の処理を行います。

- (1) ターゲット MCU のユーザ領域にユーザプログラムを書き込みます。

フラッシュプログラマは先ずユーザ/データ領域プログラム選択コマンド(43h)を送信します。ユーザ/データ領域プログラム選択コマンドを送信後にユーザ/データ領域プログラム選択のレスポンス(06h)を受信することでユーザ/データ領域プログラム選択を完了します。ユーザ/データ領域プログラム選択コマンドを送信後にユーザ/データ領域プログラム選択のレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

次に 256 バイトプログラムコマンド(50h)を、プログラムアドレスは 256 バイトでアライメントされたアドレス、プログラムデータは 256 バイト単位に指定してターゲット MCU のユーザ領域に書き込むユーザプログラムの容量分、送信します。

プログラムアドレス(ターゲット MCU の書き込み先)の範囲は以下のとおりです。

- ・ユーザプログラムの容量が 256K バイトの場合、0xFFFFC 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFFFA 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFFF8 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFFF4 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFFF0 2000 番地から 0xFFFF FFFF 番地

プログラムデータ(RSK+RX63N 上の書き込み元データ)の範囲は以下のとおりです。

- ・ユーザプログラムの容量が 256K バイトの場合、0xFFFF0 0000 番地から 0xFFFF3 FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFFF0 0000 番地から 0xFFFF5 FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFFF0 0000 番地から 0xFFFF7 FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFFF0 0000 番地から 0xFFFFB FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFFF0 0000 番地から 0xFFFF DFFF 番地

ターゲット MCU のユーザ領域に書き込むユーザプログラムの容量分、プログラムコマンドを送信すると、プログラムを終了するプログラムコマンド(50h FFh FFh FFh FFh B4h)を送信します。プログラムコマンドを送信後にプログラムのレスポンス(06h)を受信すると、1 回のプログラムを完了します。プログラムコマンドを送信後にプログラムのレスポンス(06h)以外を受信した場合、ターゲット MCU をリセットして中断します。

図 5.12 にユーザ領域をプログラムする手順を示します。

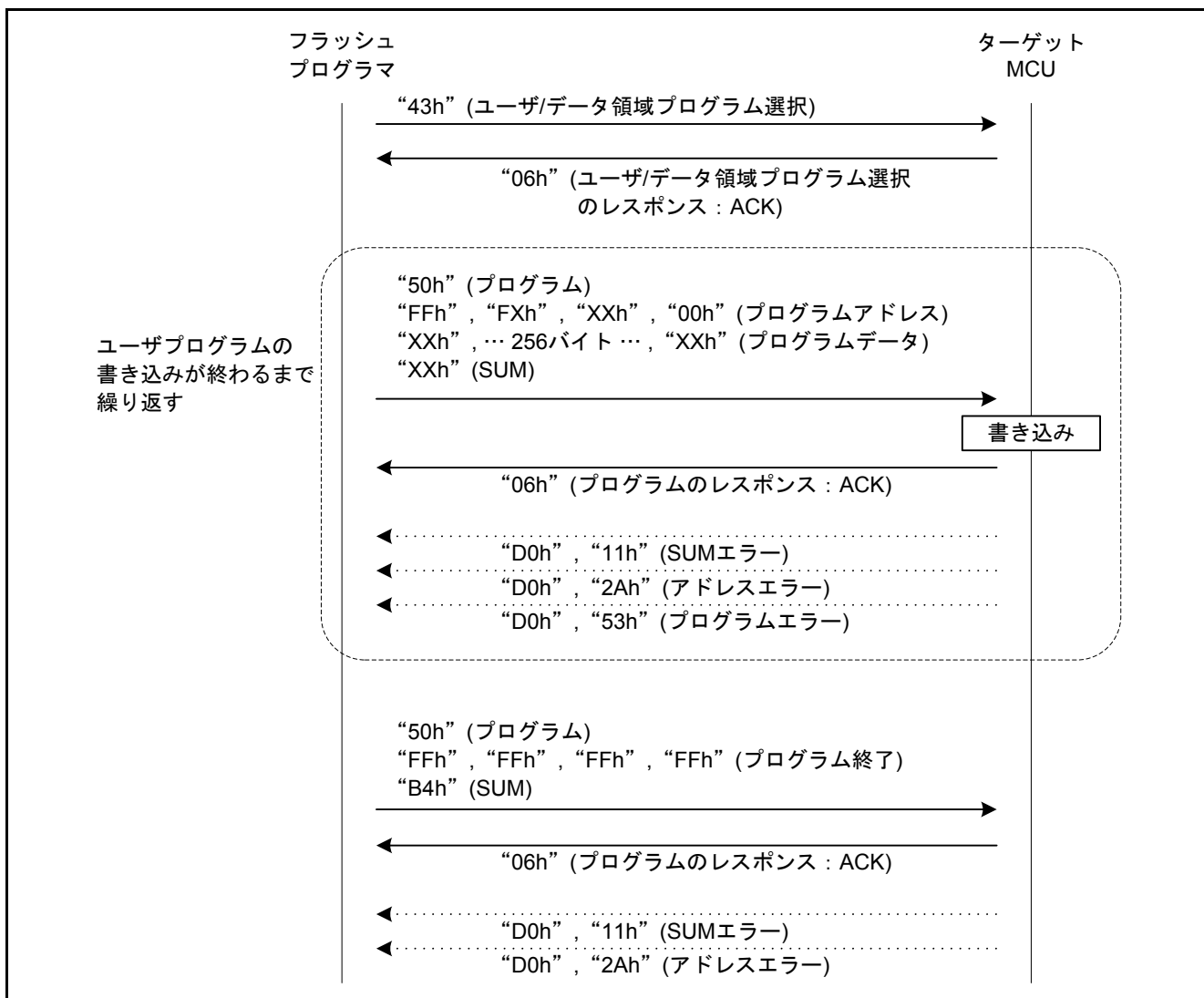


図 5.12 ユーザ領域をプログラムする手順

(2) ターゲット MCU のユーザ領域に正しく書き込んだことを確認します。

フラッシュプログラマはターゲット MCU のユーザ領域に書き込んだデータを確認するため、ターゲット MCU のユーザ領域にあるデータを読み出して書き込んだ値と比較します。

フラッシュプログラマはメモリリードコマンド(52h)を、読み出しアドレスは 256 バイトでアライメントされたアドレスを指定してターゲット MCU のユーザ領域に書き込むユーザプログラムの容量分、送信します。

読み出しアドレスの範囲は以下のとおりです。

- ・ユーザプログラムの容量が 256K バイトの場合、0xFFFFC 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFFFA 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFFF8 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFFF4 0000 番地から 0xFFFF FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFFF0 2000 番地から 0xFFFF FFFF 番地

メモリリードコマンドを送信後にメモリリードのレスポンス(52h から始まるデータ)を受信すると、RSK+RX63N のユーザ領域の書き込み元データと比較します。比較結果が一致しなかった場合や、メモリリードコマンドを送信後にメモリリードのレスポンス(52h から始まるデータ)以外を受信した場合、ターゲット MCU をリセットして中断します。

書き込み元データの範囲は以下のとおりです。

- ・ユーザプログラムの容量が 256K バイトの場合、0xFFFF0 0000 番地から 0xFFFF3 FFFF 番地
- ・ユーザプログラムの容量が 384K バイトの場合、0xFFFF0 0000 番地から 0xFFFF5 FFFF 番地
- ・ユーザプログラムの容量が 512K バイトの場合、0xFFFF0 0000 番地から 0xFFFF7 FFFF 番地
- ・ユーザプログラムの容量が 768K バイトの場合、0xFFFF0 0000 番地から 0xFFFFB FFFF 番地
- ・ユーザプログラムの容量が 1016K バイトの場合、0xFFFF0 0000 番地から 0xFFFF DFFF 番地

図 5.13にユーザ領域のデータを確認する手順を示します。

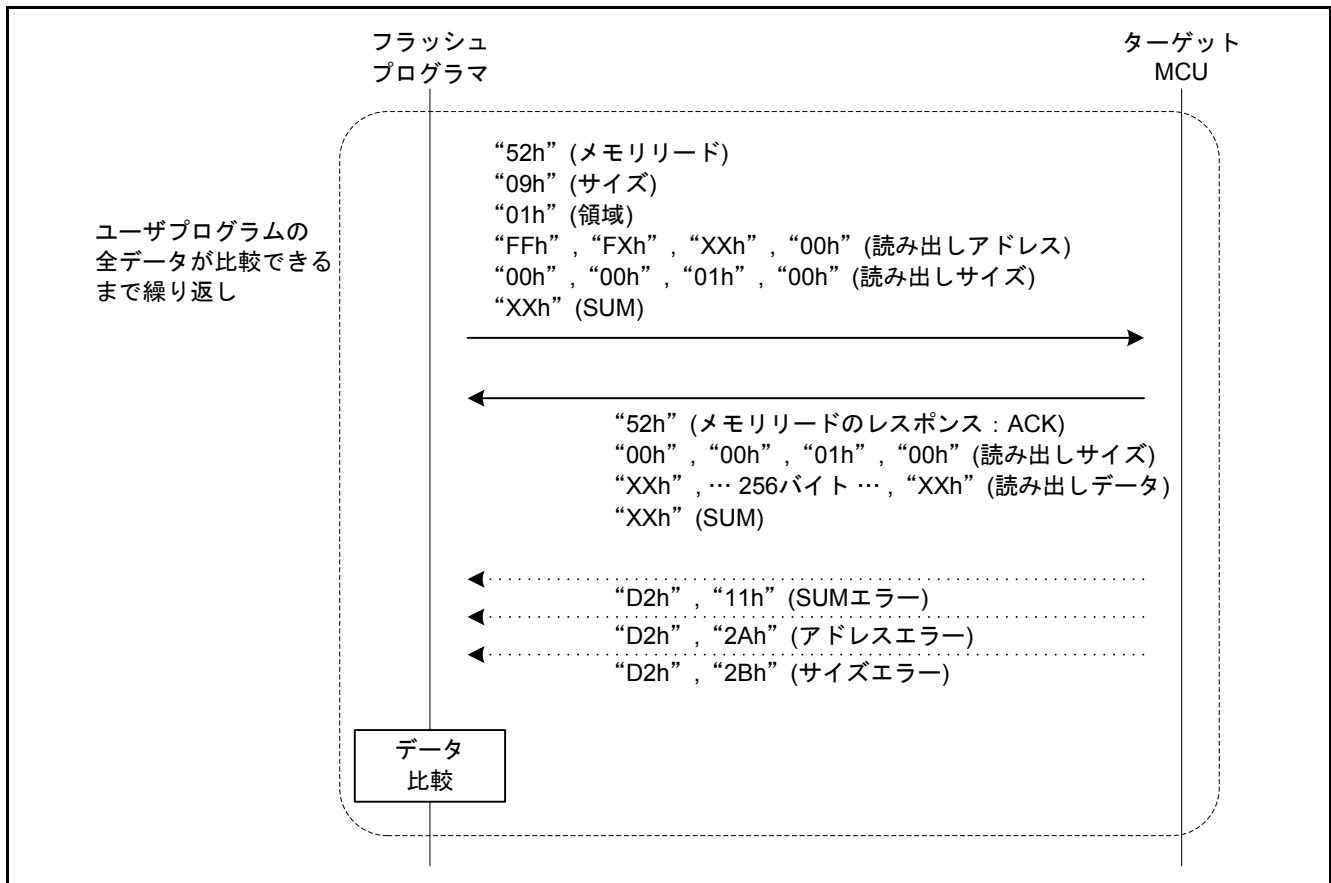


図 5.13 ユーザ領域のデータを確認する手順

## 5.2.6 ターゲット MCU のリセット

- (1) ターゲット MCU の MD 端子を “H” にします。
- (2) ターゲット MCU の RES#端子を “L” にします。
- (3) 2ms のウェイト後、ターゲット MCU の RES#端子を “H” にします。
- (4) 無限ループします。

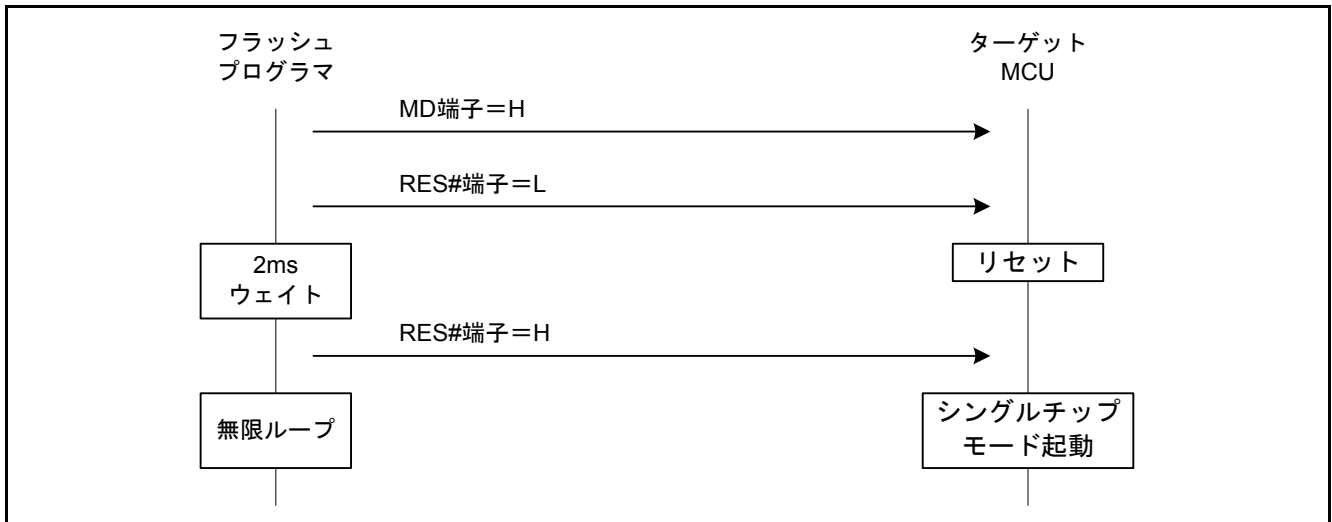


図 5.14 ターゲット MCU のリセット手順

### 5.3 ファイル構成

表 5.2にサンプルコードで使用するファイルを、表 5.3に標準インクルードファイルを、表 5.4と表 5.5に関連アプリケーションノートの関数と設定値を示します。なお、統合開発環境で自動生成されるファイルは除きます。

表5.2 サンプルコードで使用するファイル

| ファイル名      | 概要                     |
|------------|------------------------|
| main.c     | メイン処理、コマンド送信、レスポンス受信処理 |
| cmt_wait.c | CMT を使用した時間待ち処理        |
| cmt_wait.h | cmt_wait.c のヘッダファイル    |

表5.3 標準インクルードファイル

| 参照元       | 内容                         |
|-----------|----------------------------|
| stdint.h  | 指定した幅の整数型を宣言してマクロを定義します。   |
| stdbool.h | 論理型、および論理値に関するマクロを定義します。   |
| machine.h | RX ファミリー用 組み込み関数の形式を定義します。 |
| string.h  | 文字列の比較、複写等を行うライブラリです。      |

表5.4 参照する関連アプリケーションノートの関数と設定値(RX63N グループ、RX631 グループ 初期設定例)

| ファイル名                      | 関数                       | 設定値                             |
|----------------------------|--------------------------|---------------------------------|
| r_init_stop_module.c       | R_INIT_StopModule()      | -                               |
| r_init_stop_module.h       | -                        | DMAC/DTC、EXDMAC の停止指定           |
| r_init_non_existent_port.c | R_INIT_NonExistentPort() | -                               |
| r_init_non_existent_port.h | -                        | 176 ピン版の指定                      |
| r_init_clock.c             | R_INIT_Clock()           | -                               |
| r_init_clock.h             | -                        | システムクロックを PLL とし、サブクロックを使用しない指定 |

表5.5 参照する関連アプリケーションノートの関数と設定値(RX63N High-performance Embedded Workshop 用ルネサススタータキットのサンプルコード)

| ファイル名         | 関数                          | 設定値 |
|---------------|-----------------------------|-----|
| lcd.c         | Init_LCD()<br>Display_LCD() | -   |
| lcd.h         | -                           | -   |
| rskrx63ndef.h | -                           | -   |

## 5.4 オプション設定メモリ

表 5.6にサンプルコードで使用するオプション設定メモリの状態を示します。必要に応じて、お客様のシステムに最適な値を設定してください。

表5.6 サンプルコードで使用するオプション設定メモリ

| シンボル | アドレス                  | 設定値        | 内容                                      |
|------|-----------------------|------------|---|
| OFS0 | FFFF FF8Fh~FFFF FF8Ch | FFFF FFFFh | リセット後、IWDT は停止<br>リセット後、WDT は停止         |
| OFS1 | FFFF FF8Bh~FFFF FF88h | FFFF FFFFh | リセット後、電圧監視 0 リセット無効<br>リセット後、HOCO 発振が無効 |
| MDES | FFFF FF83h~FFFF FF80h | FFFF FFFFh | リトルエンディアン                               |



## 5.5 定数一覧

表 5.7から表 5.15にサンプルコードで使用する定数を示します。

表5.7 サンプルコードで使用する定数 (1/2)

| 定数名               | 設定値                              | 内容                                   |
|-------------------|----------------------------------|--------------------------------------|
| ROMVOL_256KB      | (256 * 1024)                     | ターゲット MCU のユーザ領域容量(256K バイト)選択値      |
| ROMVOL_384KB      | (384 * 1024)                     | ターゲット MCU のユーザ領域容量(384K バイト)選択値      |
| ROMVOL_512KB      | (512 * 1024)                     | ターゲット MCU のユーザ領域容量(512K バイト)選択値      |
| ROMVOL_768KB      | (768 * 1024)                     | ターゲット MCU のユーザ領域容量(768K バイト)選択値      |
| ROMVOL_1MB        | (1024 * 1024)                    | ターゲット MCU のユーザ領域容量(1M バイト)選択値        |
| TARGET_ROMVOL     | ROMVOL_256KB                     | ターゲット MCU のユーザ領域容量(256K バイト選択)       |
| TARGET_DATA_ADD   | 0xFFFF0000                       | ターゲット MCU のユーザ領域に書き込むデータを格納する先頭アドレス  |
| FLASH_PRGRMA_SIZE | (8 * 1024)                       | フラッシュプログラマプログラムのサイズ(8K バイト)          |
| READING_HEAD_ADD  | WRITING_HEAD_ADD                 | ターゲット MCU の読み出し開始アドレス(書き込み開始アドレスと同一) |
| MDES_ADD          | 0xFFFFFFFF80                     | MDES 判定アドレス                          |
| WRITING_TIME      | (TARGET_ROMVOL / 256)            | ターゲット MCU の書き込み回数(256 バイト単位に書き込む回数)  |
| READING_TIME      | WRITING_TIME                     | ターゲット MCU の読み出し回数(書き込み回数と同一)         |
| RES_BUF_SIZE      | (262)                            | 受信データ格納バッファサイズ                       |
| OK                | (0)                              | 真値                                   |
| NG                | (1)                              | 偽値                                   |
| ERRLOOP_ON        | (1)                              | 受信時にエラー検出した場合、エラー処理(無限ループ)する選択値      |
| ERRLOOP_OFF       | (0)                              | 受信時にエラー検出した場合、エラー処理(無限ループ)しない選択値     |
| INTERVAL_ON       | (1)                              | 送信時に間隔を設ける選択値                        |
| INTERVAL_OFF      | (0)                              | 送信時に間隔を設けない選択値                       |
| RES_ACK_NORMAL    | (0x06)                           | 通常 ACK 判定値                           |
| RES_ID_DISABLED   | (0x26)                           | ID コードプロテクト無効 ACK 判定値                |
| RES_ID_ENABLED    | (0x16)                           | ID コードプロテクト有効 ACK 判定値                |
| ARRAY_SIZE_OF(a)  | ( sizeof( a ) / sizeof( a[0] ) ) | コマンド送信データのバイト数取得マクロ関数                |
| WT_BASE_US        | (1000000L)                       | 1 $\mu$ s 単位の待ち時間計算値                 |
| WT_BASE_MS        | (1000L)                          | 1ms 単位の待ち時間計算値                       |
| WT_CMT_CLOCK      | (48L * WT_BASE_US)               | CMT カウントソース周波数(PCLKB:48MHz)          |
| WT_CMT_DIVIDE     | (512L)                           | CMT カウントソースの分周比                      |

表5.8 サンプルコードで使用する定数 (2/2)

| 定数名        | 設定値   | 内容                    |
|------------|---|-----------------------|
| WAIT_1MS   | $((1 * (WT\_CMT\_CLOCK / WT\_CMT\_DIVIDE)) / WT\_BASE\_MS + 0.5)$   | CMTによる待ち時間<br>(1ms)   |
| WAIT_2MS   | $((2 * (WT\_CMT\_CLOCK / WT\_CMT\_DIVIDE)) / WT\_BASE\_MS + 0.5)$   | CMTによる待ち時間<br>(2ms)   |
| WAIT_25MS  | $((25 * (WT\_CMT\_CLOCK / WT\_CMT\_DIVIDE)) / WT\_BASE\_MS + 0.5)$  | CMTによる待ち時間<br>(25ms)  |
| WAIT_100MS | $((100 * (WT\_CMT\_CLOCK / WT\_CMT\_DIVIDE)) / WT\_BASE\_MS + 0.5)$ | CMTによる待ち時間<br>(100ms) |
| WAIT_400MS | $((400 * (WT\_CMT\_CLOCK / WT\_CMT\_DIVIDE)) / WT\_BASE\_MS + 0.5)$ | CMTによる待ち時間<br>(400ms) |

表5.9 サンプルコードで使用する定数(TARGET\_ROMVOLにROMVOL\_256KBを選択した場合)

| 定数名              | 設定値         | 内容  |
|------------------|-------------|---|
| TARGET_ID1_ADD   | 0xFFFF3FFA0 | ターゲット MCU に書き込む 制御コードと ID コード 1 から ID コード 3 の参照アドレス |
| TARGET_ID2_ADD   | 0xFFFF3FFA4 | ターゲット MCU に書き込む ID コード 4 から ID コード 7 の参照アドレス        |
| TARGET_ID3_ADD   | 0xFFFF3FFA8 | ターゲット MCU に書き込む ID コード 8 から ID コード 11 の参照アドレス       |
| TARGET_ID4_ADD   | 0xFFFF3FFAC | ターゲット MCU に書き込む ID コード 12 から ID コード 15 の参照アドレス      |
| WRITING_HEAD_ADD | 0xFFFFC0000 | ターゲット MCU の書き込み開始アドレス                               |
| MAX_BLK_NUMBER   | 0x15        | ターゲット MCU の最大ブロック番号                                 |

表 5.10 サンプルコードで使用する定数(TARGET\_ROMVOLにROMVOL\_384KBを選択した場合)

| 定数名              | 設定値         | 内容  |
|------------------|-------------|---|
| TARGET_ID1_ADD   | 0xFFFF5FFA0 | ターゲット MCU に書き込む 制御コードと ID コード 1 から ID コード 3 の参照アドレス |
| TARGET_ID2_ADD   | 0xFFFF5FFA4 | ターゲット MCU に書き込む ID コード 4 から ID コード 7 の参照アドレス        |
| TARGET_ID3_ADD   | 0xFFFF5FFA8 | ターゲット MCU に書き込む ID コード 8 から ID コード 11 の参照アドレス       |
| TARGET_ID4_ADD   | 0xFFFF5FFAC | ターゲット MCU に書き込む ID コード 12 から ID コード 15 の参照アドレス      |
| WRITING_HEAD_ADD | 0xFFFFA0000 | ターゲット MCU の書き込み開始アドレス                               |
| MAX_BLK_NUMBER   | 0x1D        | ターゲット MCU の最大ブロック番号                                 |

表5.11 サンプルコードで使用する定数(TARGET\_ROMVOL に ROMVOL\_512KB を選択した場合)

| 定数名              | 設定値         | 内容  |
|------------------|-------------|---|
| TARGET_ID1_ADD   | 0xFFFF7FFA0 | ターゲット MCU に書き込む 制御コードと ID コード 1 から ID コード 3 の参照アドレス |
| TARGET_ID2_ADD   | 0xFFFF7FFA4 | ターゲット MCU に書き込む ID コード 4 から ID コード 7 の参照アドレス        |
| TARGET_ID3_ADD   | 0xFFFF7FFA8 | ターゲット MCU に書き込む ID コード 8 から ID コード 11 の参照アドレス       |
| TARGET_ID4_ADD   | 0xFFFF7FFAC | ターゲット MCU に書き込む ID コード 12 から ID コード 15 の参照アドレス      |
| WRITING_HEAD_ADD | 0xFFFF80000 | ターゲット MCU の書き込み開始アドレス                               |
| MAX_BLK_NUMBER   | 0x25        | ターゲット MCU の最大ブロック番号                                 |

表 5.12 サンプルコードで使用する定数(TARGET\_ROMVOL に ROMVOL\_768KB を選択した場合)

| 定数名              | 設定値         | 内容  |
|------------------|-------------|---|
| TARGET_ID1_ADD   | 0xFFFFBFFA0 | ターゲット MCU に書き込む 制御コードと ID コード 1 から ID コード 3 の参照アドレス |
| TARGET_ID2_ADD   | 0xFFFFBFFA4 | ターゲット MCU に書き込む ID コード 4 から ID コード 7 の参照アドレス        |
| TARGET_ID3_ADD   | 0xFFFFBFFA8 | ターゲット MCU に書き込む ID コード 8 から ID コード 11 の参照アドレス       |
| TARGET_ID4_ADD   | 0xFFFFBFFAC | ターゲット MCU に書き込む ID コード 12 から ID コード 15 の参照アドレス      |
| WRITING_HEAD_ADD | 0xFFFF40000 | ターゲット MCU の書き込み開始アドレス                               |
| MAX_BLK_NUMBER   | 0x2D        | ターゲット MCU の最大ブロック番号                                 |

表 5.13 サンプルコードで使用する定数(TARGET\_ROMVOL に ROMVOL\_1MB を選択した場合)

| 定数名              | 設定値         | 内容  |
|------------------|-------------|---|
| TARGET_ID1_ADD   | 0xFFFFDFA0  | ターゲット MCU に書き込む 制御コードと ID コード 1 から ID コード 3 の参照アドレス |
| TARGET_ID2_ADD   | 0xFFFFDFA4  | ターゲット MCU に書き込む ID コード 4 から ID コード 7 の参照アドレス        |
| TARGET_ID3_ADD   | 0xFFFFDFA8  | ターゲット MCU に書き込む ID コード 8 から ID コード 11 の参照アドレス       |
| TARGET_ID4_ADD   | 0xFFFFDFAC  | ターゲット MCU に書き込む ID コード 12 から ID コード 15 の参照アドレス      |
| WRITING_HEAD_ADD | 0xFFFF02000 | ターゲット MCU の書き込み開始アドレス                               |
| MAX_BLK_NUMBER   | 0x35        | ターゲット MCU の最大ブロック番号                                 |

表5.14 サンプルコードで使用する定数(ブートモードエントリに使用する定義)

| 定数名            | 設定値                 | 内容  |
|----------------|---------------------|---|
| BTMD_PMR       | (PORTE.PMR.BYTE)    | ターゲット MCU の PC7 端子、MD 端子、RES#端子への出力端子割り付け(ポートモードレジスタ)   |
| BTMD_PODR      | (PORTE.PODR.BYTE)   | ターゲット MCU の PC7 端子、MD 端子、RES#端子への出力端子割り付け(ポート出力データレジスタ) |
| BTMD_PDR       | (PORTE.PDR.BYTE)    | ターゲット MCU の PC7 端子、MD 端子、RES#端子への出力端子割り付け(ポート方向レジスタ)    |
| UB_PIN         | (PORTE.PODR.BIT.B2) | ターゲット MCU の PC7 端子への出力割り当て                              |
| MD_PIN         | (PORTE.PODR.BIT.B1) | ターゲット MCU の MD 端子への出力割り当て                               |
| RES_PIN        | (PORTE.PODR.BIT.B0) | ターゲット MCU の RES#端子への出力割り当て                              |
| BTMD_PDR_INIT  | (0x07)              | ターゲット MCU の PC7 端子、MD 端子、RES#端子出力初期設定値                  |
| BTMD_PODR_INIT | (0x00)              | ターゲット MCU の PC7 端子 High レベル出力初期設定値                      |

表5.15 サンプルコードで使用する定数(調歩同期式シリアルの定義)

| 定数名             | 設定値                                   | 内容                          |
|-----------------|---------------------------------------|-----------------------------|
| SCIn            | SCI0                                  | SCI チャンネル: SCI0             |
| MSTP_SCIn       | MSTP(SCI0)                            | SCI0 モジュールストップ設定ビット         |
| IR_SCIn_RXIn    | IR(SCI0,RXI0)                         | SCI0.RXI0 割り込みステータスフラグ      |
| IR_SCIn_TXIn    | IR(SCI0,TXI0)                         | SCI0.TXI0 割り込みステータスフラグ      |
| RXDn_PDR        | (PORT3.PDR.BIT.B3)                    | SCI0.RXI0 端子方向制御ビット         |
| RXDn_PMR        | (PORT3.PMR.BIT.B3)                    | SCI0.RXI0 端子モード制御ビット        |
| RXDnPFS         | P33PFS                                | SCI0.RXI0 端子機能制御レジスタ        |
| RXDnPFS_SELECT  | (0x0B)                                | RXD0 端子機能選択ビット設定値           |
| TXDn_PODR       | (PORT3.PODR.BIT.B2)                   | SCI0.TXI0 端子出力データ格納ビット      |
| TXDn_PDR        | (PORT3.PDR.BIT.B2)                    | SCI0.TXI0 端子方向制御ビット         |
| TXDn_PMR        | (PORT3.PMR.BIT.B2)                    | SCI0.TXI0 端子モード制御ビット        |
| TXDnPFS         | P32PFS                                | SCI0.TXI0 端子機能制御レジスタ        |
| TXDnPFS_SELECT  | (0x0B)                                | TXD0 端子機能選択ビット設定値           |
| SSR_ERROR_FLAGS | (0x38)                                | SCI.SSR レジスタのエラーフラグのビットパターン |
| BRR_SET(bps)    | (WT_CMT_CLOCK/(32*(0.5)*(bps))-1+0.5) | SCI.BRR レジスタの設定値計算マクロ関数     |

## 5.6 構造体/共用体一覧

図 5.15にサンプルコードで使用する構造体/共用体を示します。

```
typedef struct{
    uint32_t TrnSize;      /* expected value of the transmit size of command */
    uint32_t RecSize;     /* expected value of the receive size of response */
    uint8_t ACKRes;      /* ACK value of response */
    uint8_t *Command;    /* boot command sequence data pointer */
} boot_cmd_t;
```

図 5.15 サンプルコードで使用する構造体/共用体

## 5.7 変数一覧

表 5.16にグローバル変数を、表 5.17にstatic 型変数を示します。

表5.16 グローバル変数

| 型                | 変数名               | 内容        | 使用関数  |
|------------------|-------------------|-----------|---|
| volatile uint8_t | CMT_InterruptFlag | 待ち時間有効フラグ | CMT_WaitSet<br>CMT_Wait<br>Excep_CMT0_CMIO<br>ReceiveResponse |

表5.17 static 型変数

| 型        | 変数名                          | 内容                          | 使用関数                    |
|----------|------------------------------|-----------------------------|-------------------------|
| uint8_t  | ResponseBuffer[RES_BUF_SIZE] | 受信データ格納バッファ                 | main<br>ReceiveResponse |
| uint8_t  | TransferMode                 | 送信モードフラグ                    | main<br>TransferCommand |
| uint8_t  | ReceiveMode                  | 受信モードフラグ                    | main<br>ReceiveResponse |
| uint8_t  | IDProtectMode                | ID コードプロテクト状態判定<br>バッファ     | main                    |
| uint32_t | BufferIndex                  | 受信データ格納バッファ<br>インデックス       | ReceiveResponse         |
| uint32_t | DeviceCode                   | デバイスコード格納バッファ               | main                    |
| uint8_t  | CMD_BitRateAdjustment_1st[]  | ビットレート自動調整コマンド<br>データ       | -                       |
| uint8_t  | CMD_BitRateAdjustment_2nd[]  | ビットレート自動調整確認<br>コマンドデータ     | -                       |
| uint8_t  | CMD_EnquiryDevice[]          | サポートデバイス問い合わせ<br>コマンドデータ    | -                       |
| uint8_t  | CMD_SelectDevice[]           | デバイス選択コマンドデータ               | -                       |
| uint8_t  | CMD_SelectClockMode[]        | クロックモード選択コマンド<br>データ        | -                       |
| uint8_t  | CMD_OperatingFreqSel_1st[]   | 新ビットレート選択コマンド<br>データ        | -                       |
| uint8_t  | CMD_OperatingFreqSel_2nd[]   | 新ビットレート選択確認<br>コマンドデータ      | -                       |
| uint8_t  | CMD_PEstatusTransition[]     | P/E ステータス遷移コマンド<br>データ      | -                       |
| uint8_t  | CMD_IDCodeCheck[]            | ID コードチェックコマンドデー<br>タ       | -                       |
| uint8_t  | CMD_EraseSelection[]         | イレーズ選択コマンドデータ               | -                       |
| uint8_t  | CMD_BlockErase[]             | ブロックイレーズ<br>コマンドデータ         | -                       |
| uint8_t  | CMD_ProgramSelection[]       | ユーザ/データ領域プログラム選<br>択コマンドデータ | -                       |
| uint8_t  | CMD_Program[]                | プログラムコマンドデータ                | -                       |

| 型          | 変数名                      | 内容                      | 使用関数 |
|------------|--------------------------|-------------------------|------|
| uint8_t    | CMD_ProgramTermination[] | プログラム終了コマンドデータ          | -    |
| uint8_t    | CMD_MemoryRead[]         | メモリアドレスコマンドデータ          | -    |
| boot_cmd_t | BitRateAdjustment_1st    | ビットレート自動調整コマンド構造体       | main |
| boot_cmd_t | BitRateAdjustment_2nd    | ビットレート自動調整確認コマンド構造体     | main |
| boot_cmd_t | EnquiryDevice            | サポートデバイス問い合わせコマンド構造体    | main |
| boot_cmd_t | SelectDevice             | デバイス選択コマンド構造体           | main |
| boot_cmd_t | SelectClockMode          | クロックモード選択コマンド構造体        | main |
| boot_cmd_t | OperatingFreqSel_1st     | 新ビットレート選択コマンド構造体        | main |
| boot_cmd_t | OperatingFreqSel_2nd     | 新ビットレート選択確認コマンド構造体      | main |
| boot_cmd_t | PEstatusTransition       | P/E ステータス遷移コマンド構造体      | main |
| boot_cmd_t | IDCodeCheck              | ID コードチェックコマンド構造体       | main |
| boot_cmd_t | EraseSelection           | イレーズ選択コマンド構造体           | main |
| boot_cmd_t | BlockErase               | ブロックイレーズコマンド構造体         | main |
| boot_cmd_t | ProgramSelection         | ユーザ/データ領域プログラム選択コマンド構造体 | main |
| boot_cmd_t | Program                  | プログラムコマンド構造体            | main |
| boot_cmd_t | ProgramTermination       | プログラム終了コマンド構造体          | main |
| boot_cmd_t | MemoryRead               | メモリアドレスコマンド構造体          | main |

## 5.8 関数一覧

表 5.18に関数を示します。

表5.18 関数

| 関数名             | 概要                    |
|-----------------|-----------------------|
| main            | メイン処理、通信プロトコル制御       |
| peripheral_init | 周辺機能初期設定              |
| CMT_WaitInit    | CMT による待ち時間用タイマ初期設定   |
| CMT_WaitSet     | CMT による待ち時間設定         |
| CMT_Wait        | CMT による時間待ち処理         |
| Excep_CMT0_CMIO | CMT0,CMIO 割り込み処理      |
| SCI_Init        | SCI 初期設定              |
| SCI_change      | SCI ビットレート変更処理        |
| CalcSumData     | “SUM” データ計算処理         |
| BootModeEntry   | ターゲット MCU のブートモード起動処理 |
| BootModeRelease | ターゲット MCU のリセット処理     |
| TransferCommand | コマンド送信処理              |
| ReceiveResponse | レスポンス受信処理             |
| U4memcpy        | 符号なし 4 バイトデータの複写      |

## 5.9 関数仕様

サンプルコードの関数仕様を示します。

| main  |   |
|-------|---|
| 概要    | メイン処理                                     |
| ヘッダ   | lcd.h, cmt_wait.h                         |
| 宣言    | void main(void)                           |
| 説明    | 初期設定後、ターゲット MCU をブートモード起動して、ユーザ領域を書き換えます。 |
| 引数    | なし  |
| リターン値 | なし  |

| peripheral_init |                            |
|-----------------|----------------------------|
| 概要              | 周辺機能初期設定                   |
| ヘッダ             | lcd.h, cmt_wait.h          |
| 宣言              | void peripheral_init(void) |
| 説明              | 使用する周辺機能の初期設定を行います。        |
| 引数              | なし                         |
| リターン値           | なし                         |

| CMT_WaitInit |                           |
|--------------|---------------------------|
| 概要           | CMT による待ち時間用タイマ初期設定       |
| ヘッダ          | cmt_wait.h                |
| 宣言           | void CMT_WaitInit(void)   |
| 説明           | 待ち時間用タイマ(CMT0)の初期設定を行います。 |
| 引数           | なし                        |
| リターン値        | なし                        |

| CMT_WaitSet |   |
|-------------|---|
| 概要          | CMT による待ち時間設定   |
| ヘッダ         | cmt_wait.h  |
| 宣言          | void CMT_WaitSet(uint16_t cnt)  |
| 説明          | 引数で指定した時間( $\mu$ s)を CMCOR レジスタに設定し、CMCNT レジスタをカウント開始します。                   |
| 引数          | uint16_t cnt                      待ち時間                                      |
| リターン値       | なし  |
| 備考          | 待ち時間の最小単位は $1 / (\text{PCLKB}[\text{MHz}] / 512) \approx 10.67 \mu\text{s}$ |



---

**CMT\_Wait**

---

|       |  |
|-------|--|
| 概要    | CMT による時間待ち処理  |
| ヘッダ   | cmt_wait.h   |
| 宣言    | void CMT_Wait(uint16_t cnt)  |
| 説明    | 引数で指定した時間( $\mu$ s)待ちます。   |
| 引数    | uint16_t cnt                      待ち時間                                     |
| リターン値 | なし   |
| 備考    | 待ち時間の最小単位は $1 / (\text{PCLKB}[\text{MHz}] / 512) \approx 10.67\mu\text{s}$ |

---

**Excep\_CMT0\_CMI0**

---

|       |  |
|-------|--|
| 概要    | CMT0,CMI0 割り込み処理                             |
| ヘッダ   | cmt_wait.h                                   |
| 宣言    | void Excep_CMT0_CMI0(void)                   |
| 説明    | CMT0.CMCNT と CMT0.CMCOR のコンペアマッチ割り込み処理を行います。 |
| 引数    | なし   |
| リターン値 | なし   |

---

**SCI\_Init**

---

|       |                     |
|-------|---------------------|
| 概要    | SCI 初期設定            |
| ヘッダ   | なし                  |
| 宣言    | void SCI_Init(void) |
| 説明    | SCI の初期設定を行います。     |
| 引数    | なし                  |
| リターン値 | なし                  |

---

**SCI\_change**

---

|       |  |
|-------|--|
| 概要    | SCI ビットレート変更処理                           |
| ヘッダ   | なし                                       |
| 宣言    | void SCI_change(void)                    |
| 説明    | SCI のビットレートを 19200bps から 1.5Mbps に変更します。 |
| 引数    | なし                                       |
| リターン値 | なし                                       |

---

**CalcSumData**

---

|       |  |
|-------|--|
| 概要    | “SUM” データ計算処理  |
| ヘッダ   | なし   |
| 宣言    | uint8_t CalcSumData(uint8_t *pData, uint32_t Length)   |
| 説明    | ブート通信プロトコルの “SUM” データを計算します。   |
| 引数    | uint8_t *pData                      “SUM” 対象データアドレス<br>uint32_t Length                      “SUM” 対象データ数 |
| リターン値 | “SUM” データ  |

---

**BootModeEntry**

---

|       |  |
|-------|--|
| 概要    | ターゲット MCU のブートモード起動処理                                |
| ヘッダ   | なし   |
| 宣言    | void BootModeEntry(void)                             |
| 説明    | ターゲット MCU をブートモードで起動するように、MD 端子、PC7 端子、RES#端子を制御します。 |
| 引数    | なし   |
| リターン値 | なし   |

---

**BootModeRelease**

---

|       |   |
|-------|---|
| 概要    | ターゲット MCU のリセット処理   |
| ヘッダ   | なし  |
| 宣言    | void BootModeRelease(uint8_t mode)                          |
| 説明    | ターゲット MCU をリセットします。   |
| 引数    | uint8_t mode                      Debug LCD ライン 2 の出力パターン選択 |
| リターン値 | なし  |

---

**TransferCommand**

---

|       |   |
|-------|---|
| 概要    | コマンド送信処理  |
| ヘッダ   | なし  |
| 宣言    | void TransferCommand(boot_cmd_t *pCmd)                    |
| 説明    | 引数で指定されたコマンド構造体のコマンドデータを送信します。                            |
| 引数    | boot_cmd_t *pCmd                  送信するコマンド構造体のアドレス        |
| リターン値 | なし  |
| 備考    | TransferMode 変数が INTERVAL_ON の場合、CMT_Wait(WAIT_1MS)を呼び出す。 |

---

**ReceiveResponse**

---

|       |  |
|-------|--|
| 概要    | レスポンス受信処理  |
| ヘッダ   | なし   |
| 宣言    | uint8_t ReceiveResponse(boot_cmd_t *pCmd)  |
| 説明    | 引数のコマンド構造体内にあるレスポンスサイズ期待値のバイト数分受信します。  |
| 引数    | boot_cmd_t *pCmd                  受信するコマンド構造体のアドレス                                 |
| リターン値 | OK : 正常に受信完了<br>NG : タイムアウト(5s)、エラーレスポンスを受信  |
| 備考    | ReceiveMode 変数が ERRLOOP_ON かつリターン値が NG の場合、BootModeRelease(NG)関数を呼び出してこの関数から復帰しない。 |

---

**U4memcpy**

---

|       |   |              |
|-------|---|--------------|
| 概要    | 符号なし 4 バイトデータの複写  |              |
| ヘッダ   | なし  |              |
| 宣言    | void *U4memcpy(void *pS1, const void *pS2)  |              |
| 説明    | 複写元の記憶域の内容を、4 バイト分、複写先の記憶域に複写します。<br>データ配置がリトルエンディアンの場合、複写先の 4 バイトデータをバイトリバース<br>します。 |              |
| 引数    | void *pS1   | 複写先の記憶域のアドレス |
|       | const void *pS2   | 複写元の記憶域のアドレス |
| リターン値 | pS1 の値  |              |

## 5.10 フローチャート

## 5.10.1 メイン処理、通信プロトコル制御

図 5.16から図 5.28にメイン処理、通信プロトコル制御のフローチャートを示します。

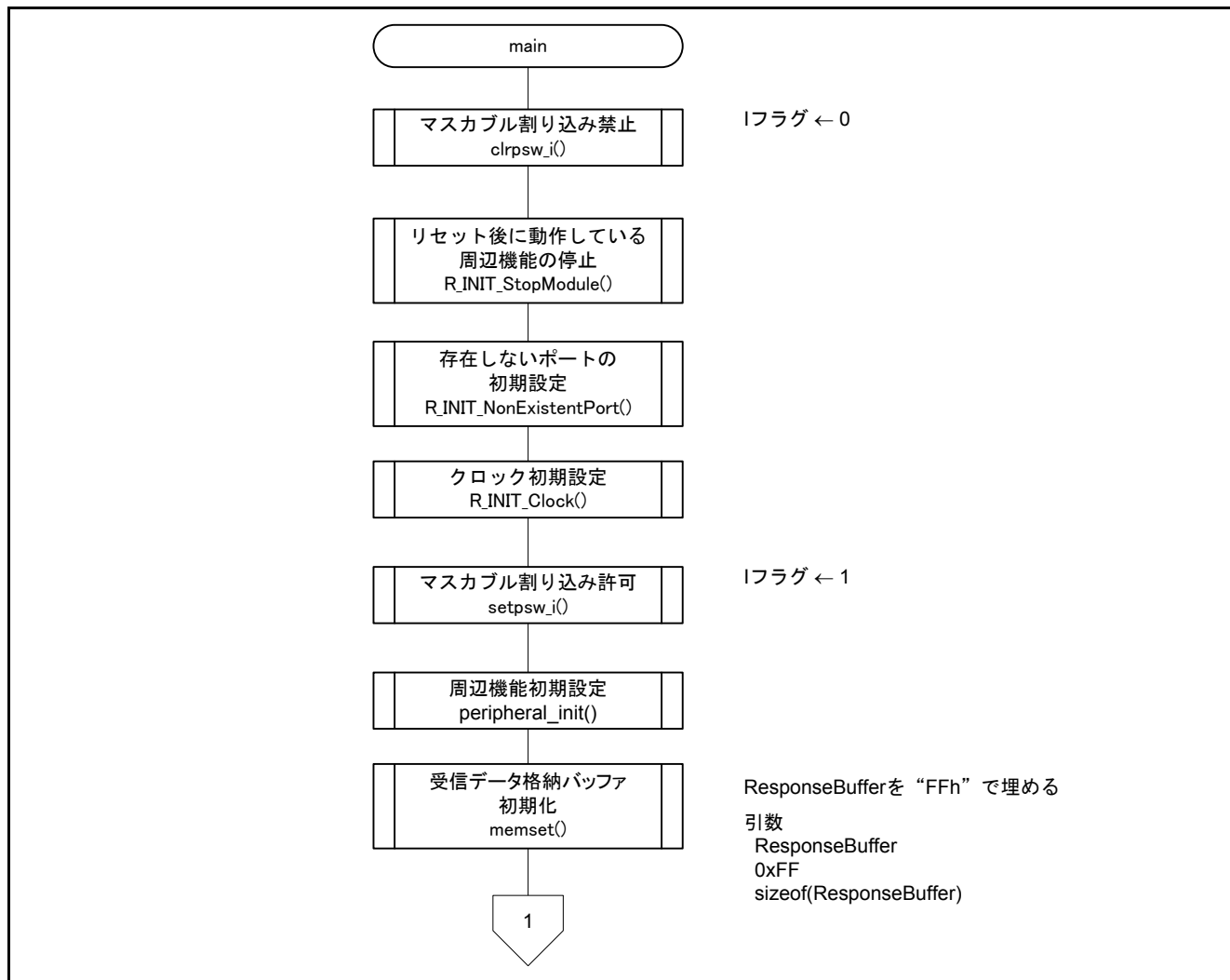


図 5.16 メイン処理、通信プロトコル制御

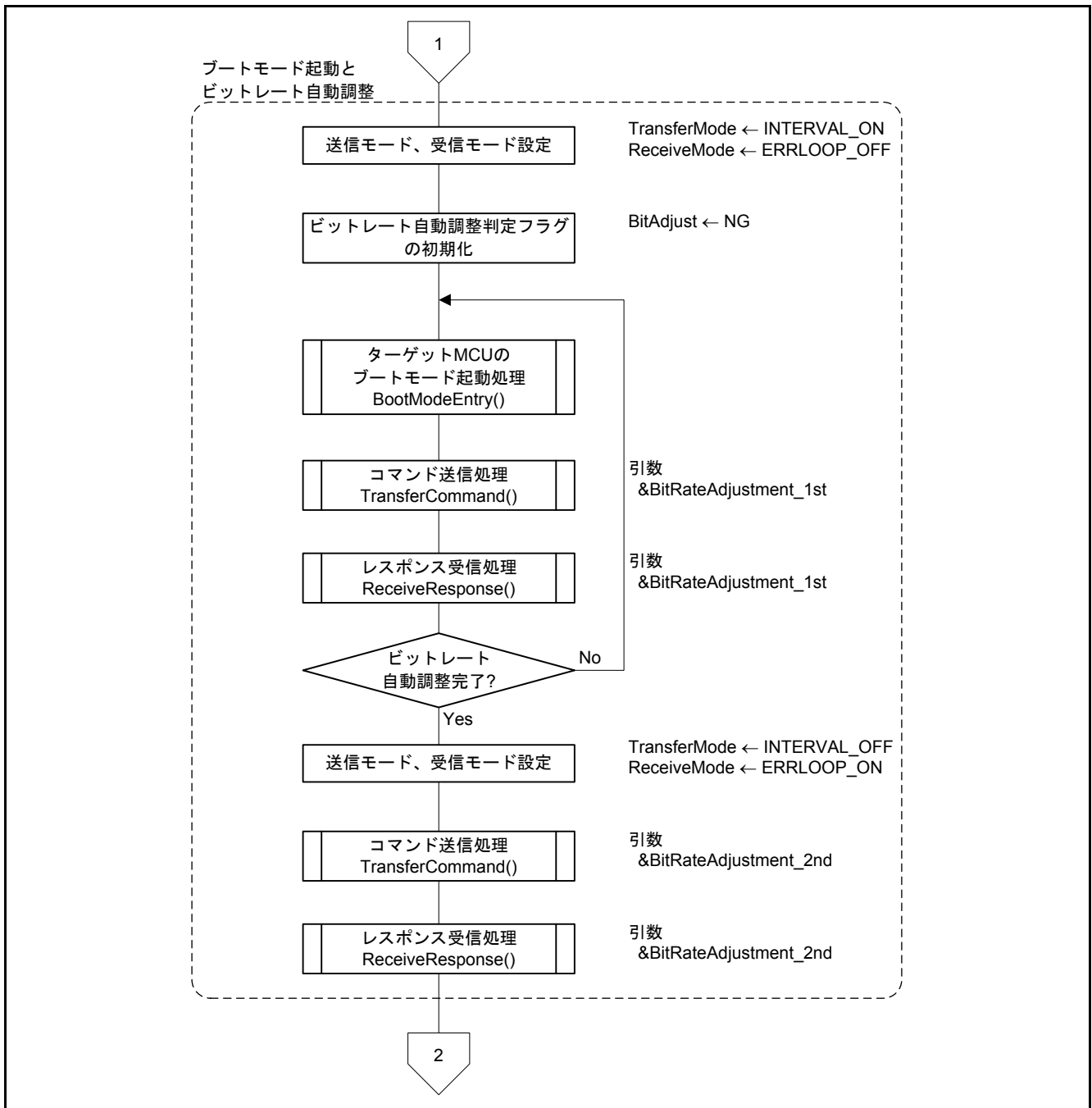


図 5.17 メイン処理、通信プロトコル制御

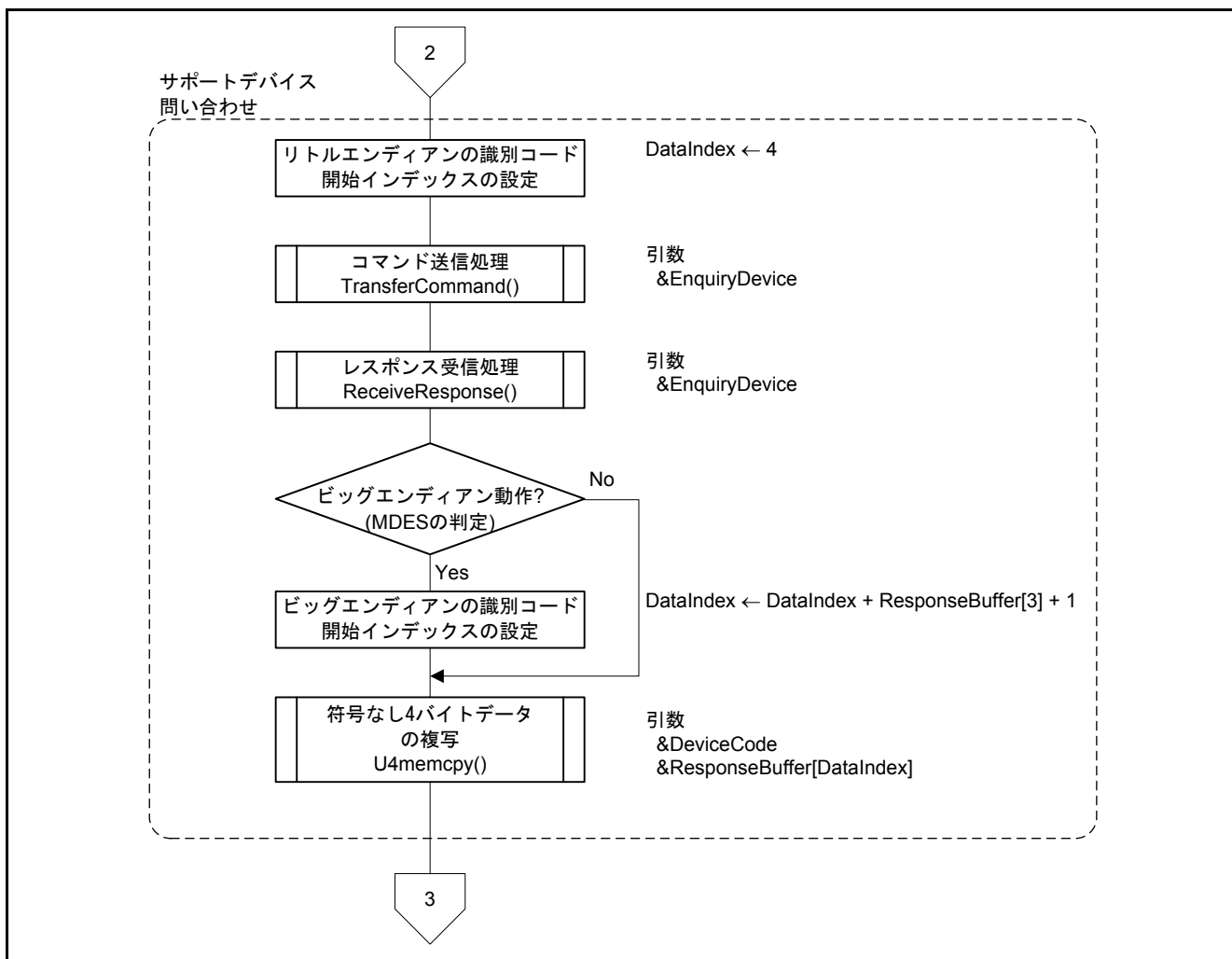


図 5.18 メイン処理、通信プロトコル制御

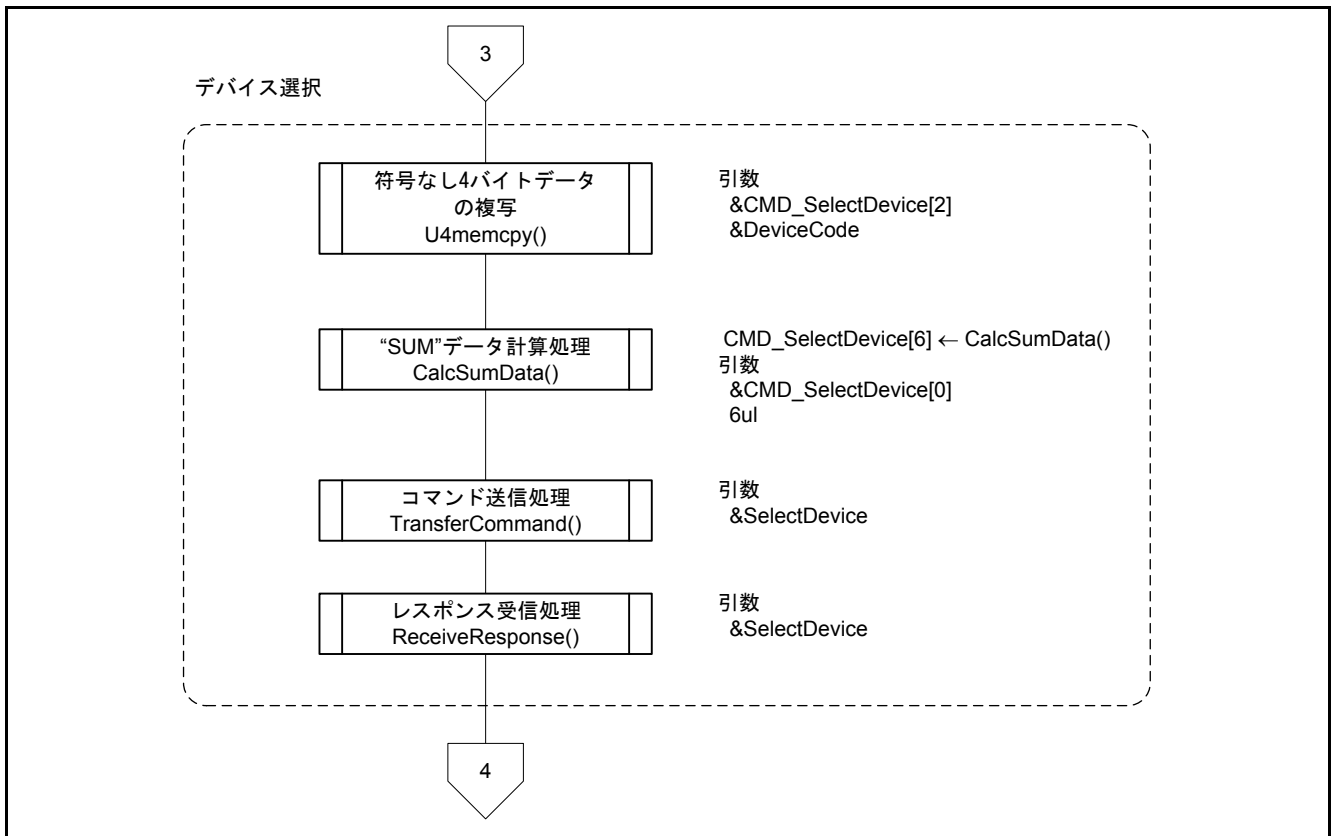


図 5.19 メイン処理、通信プロトコル制御

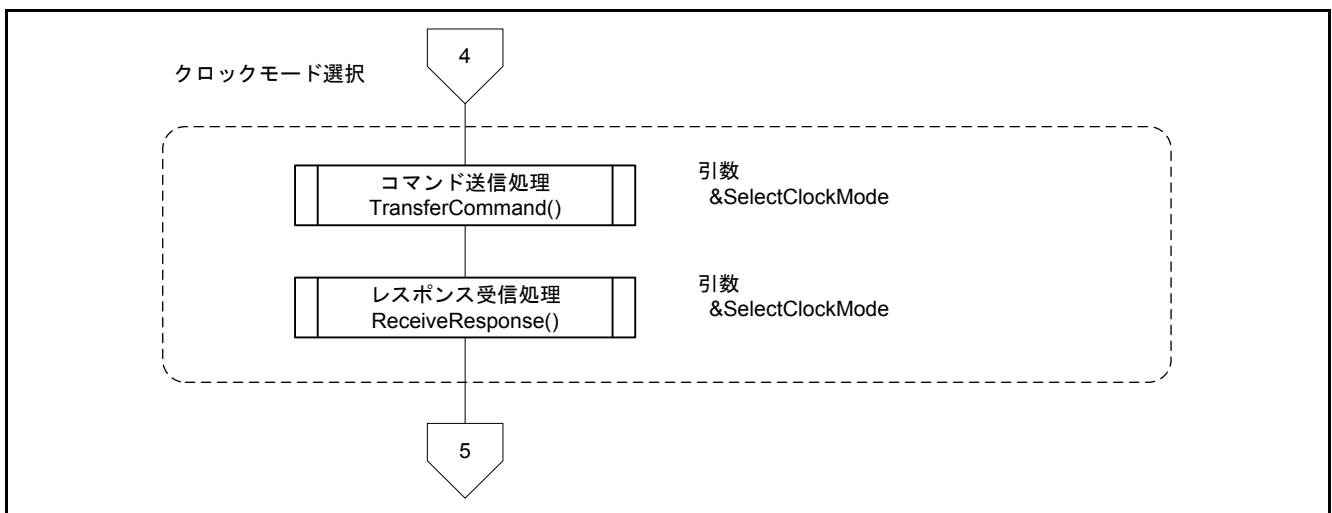


図 5.20 メイン処理、通信プロトコル制御

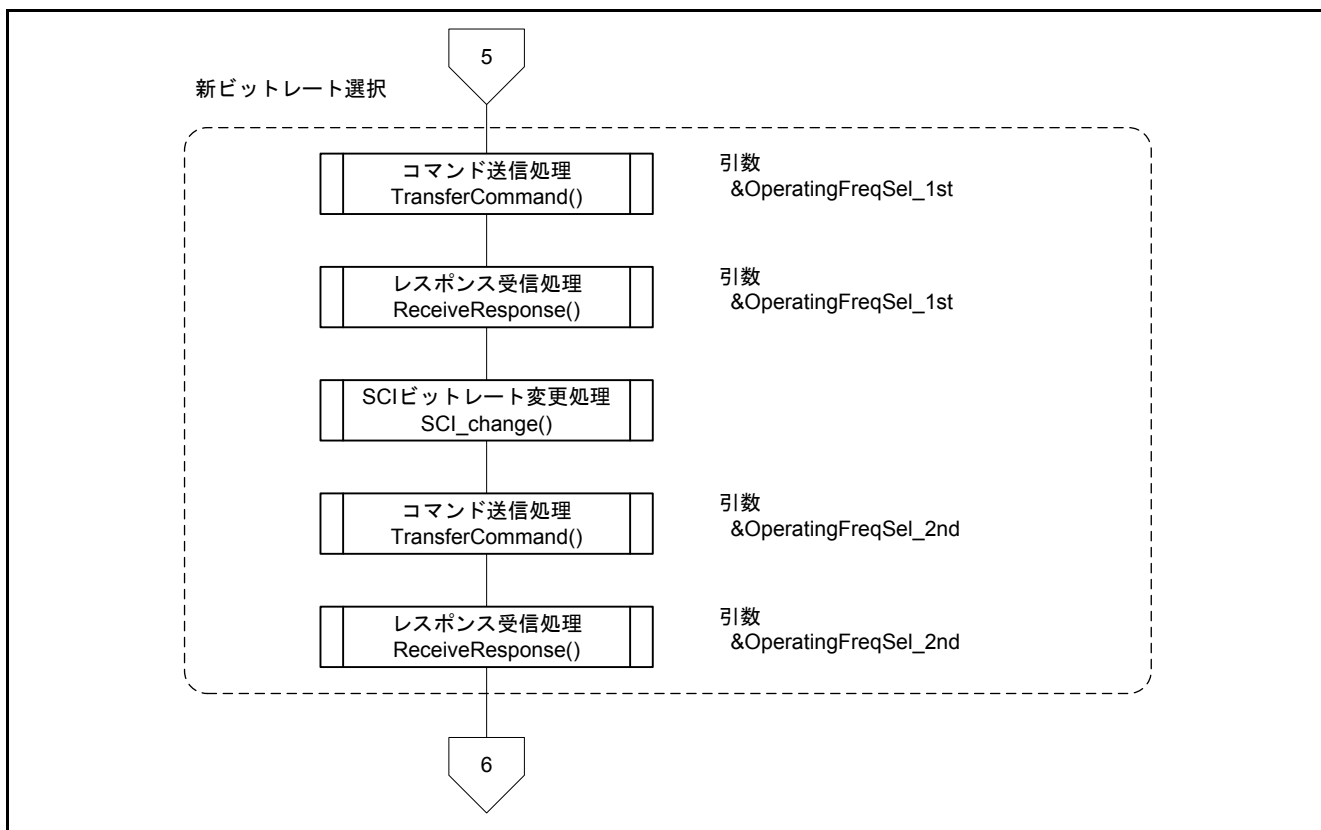


図 5.21 メイン処理、通信プロトコル制御





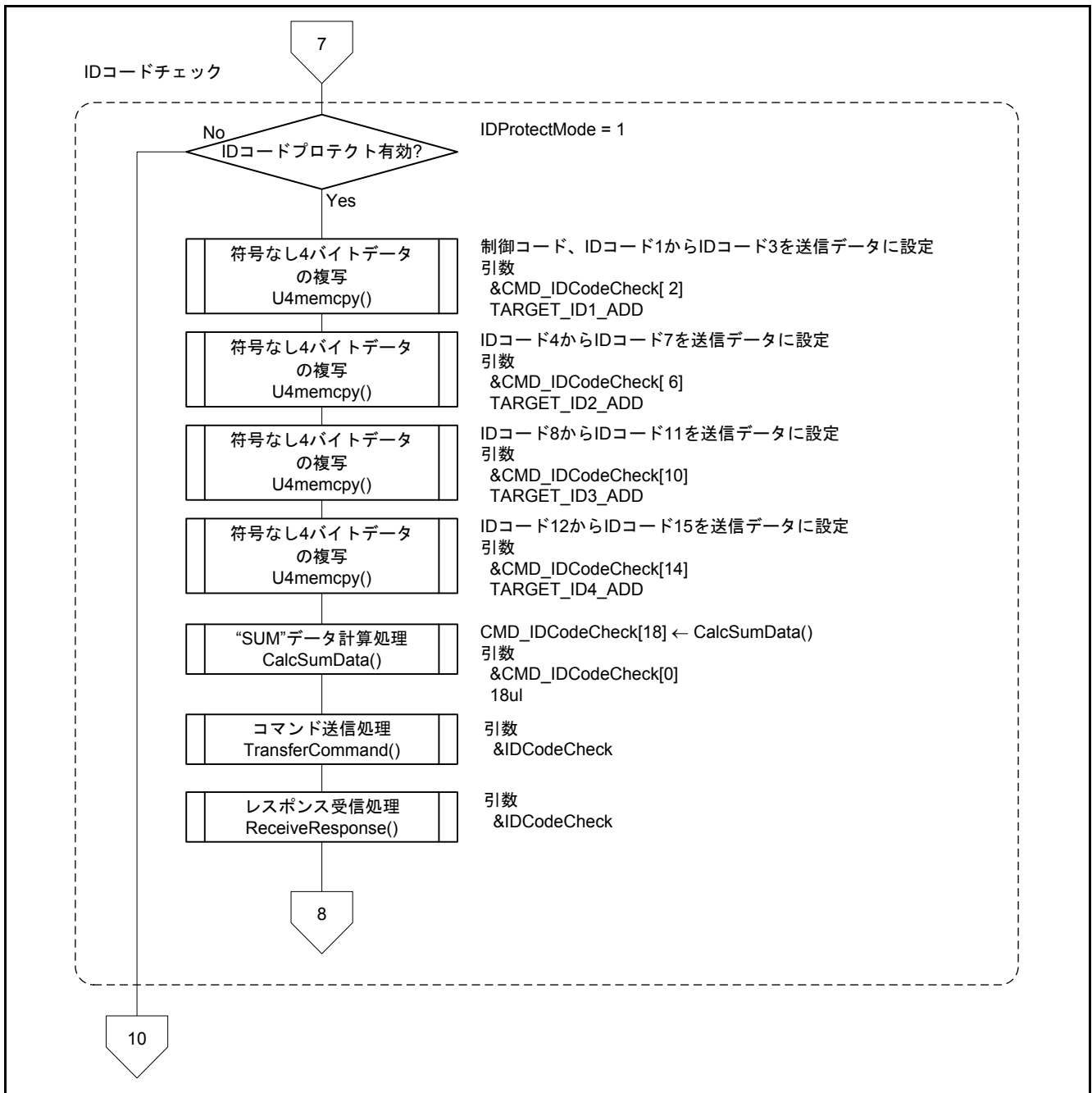


図 5.23 メイン処理、通信プロトコル制御

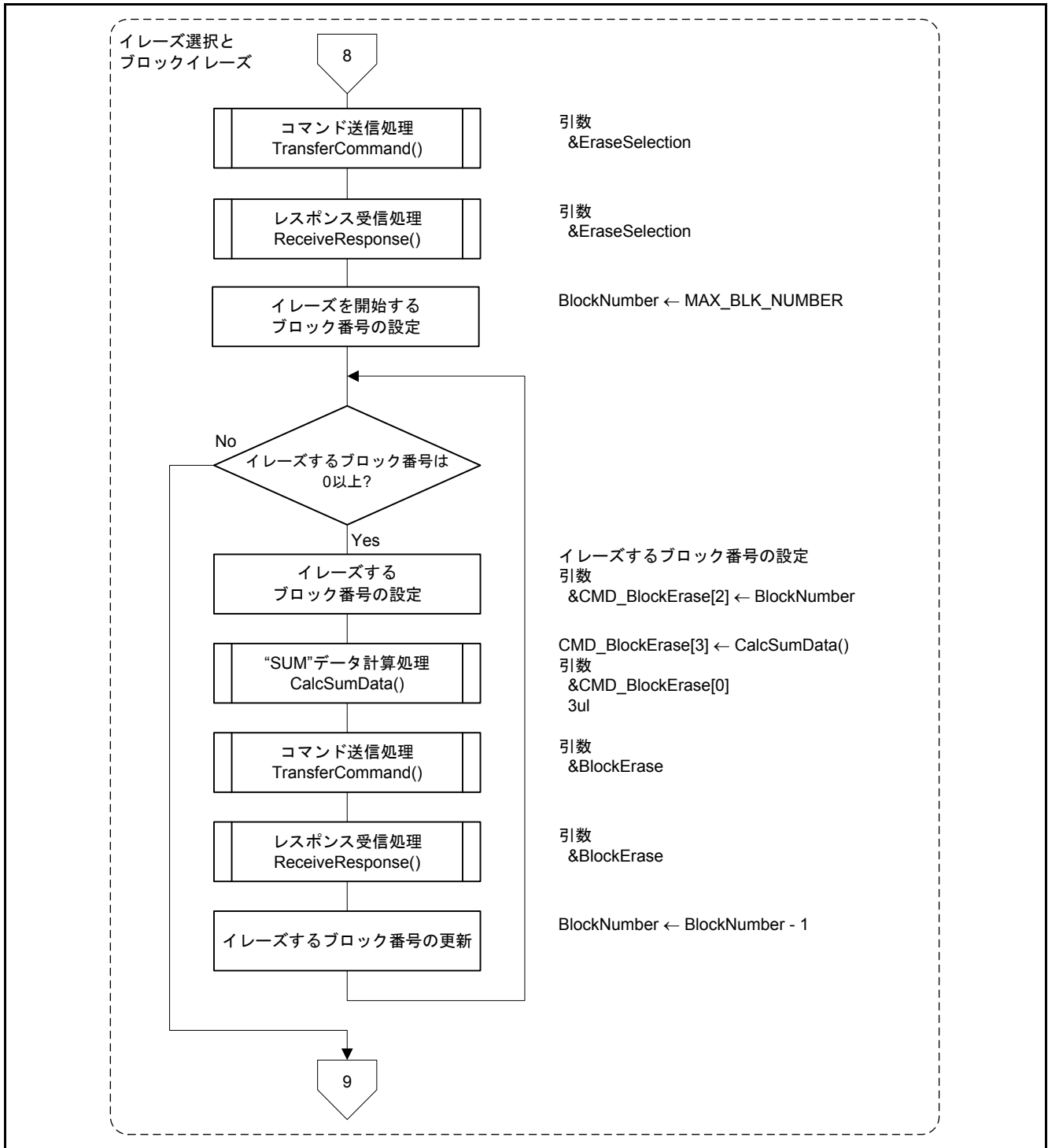


図 5.24 メイン処理、通信プロトコル制御

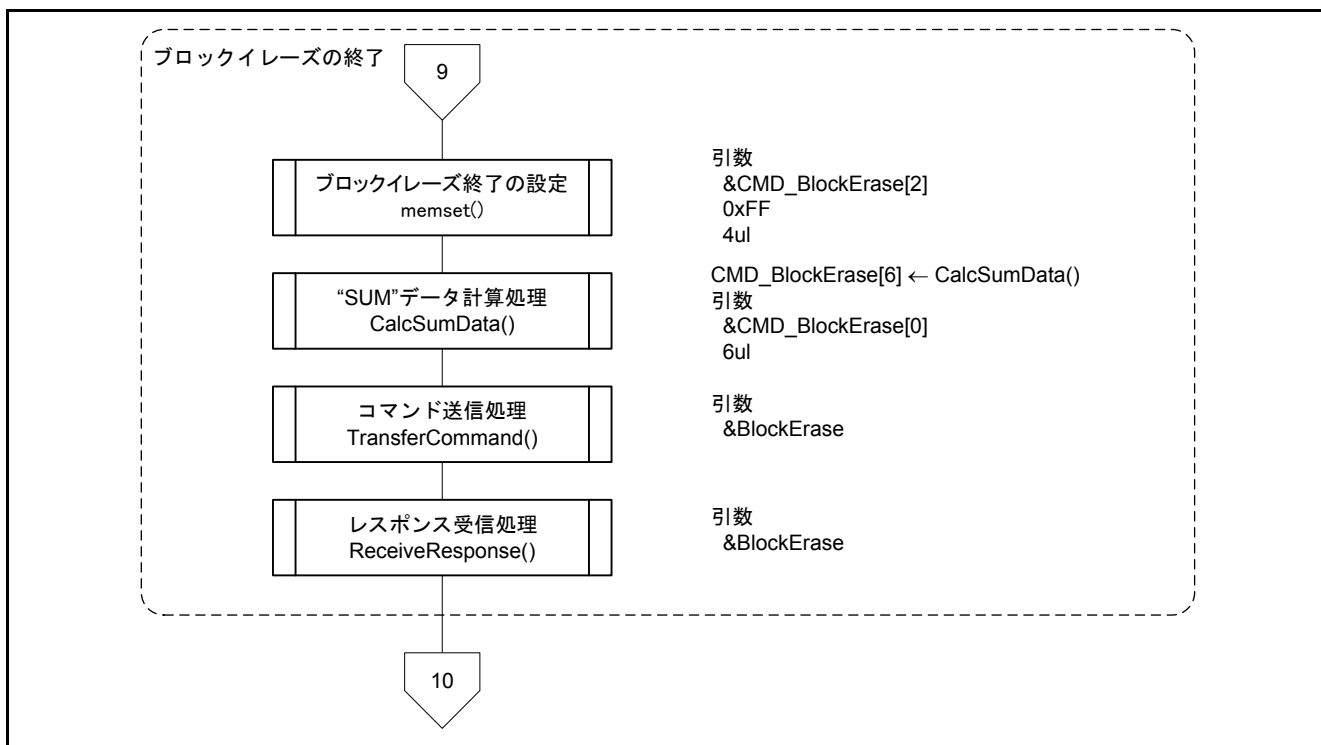


図 5.25 メイン処理、通信プロトコル制御

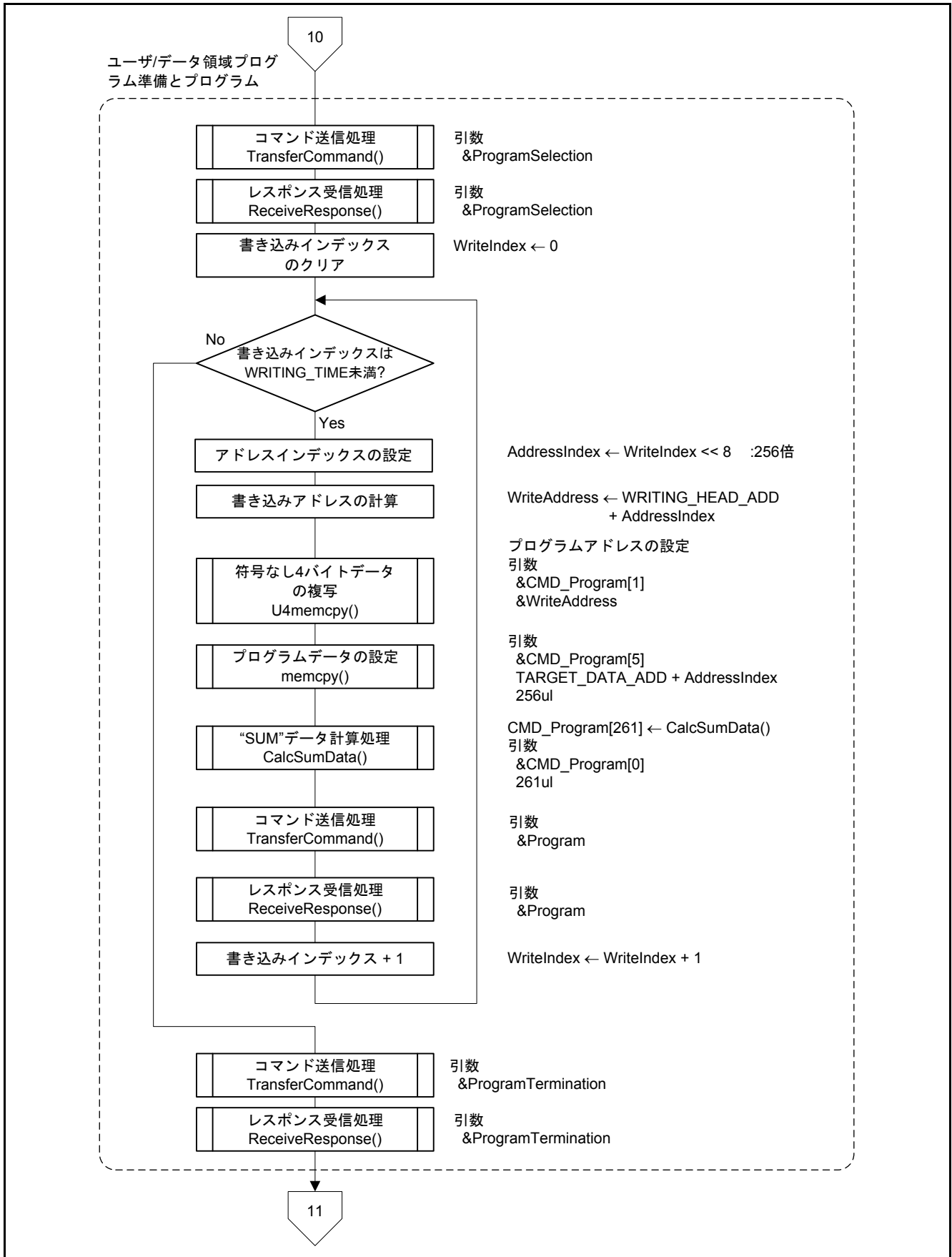


図 5.26 メイン処理、通信プロトコル制御

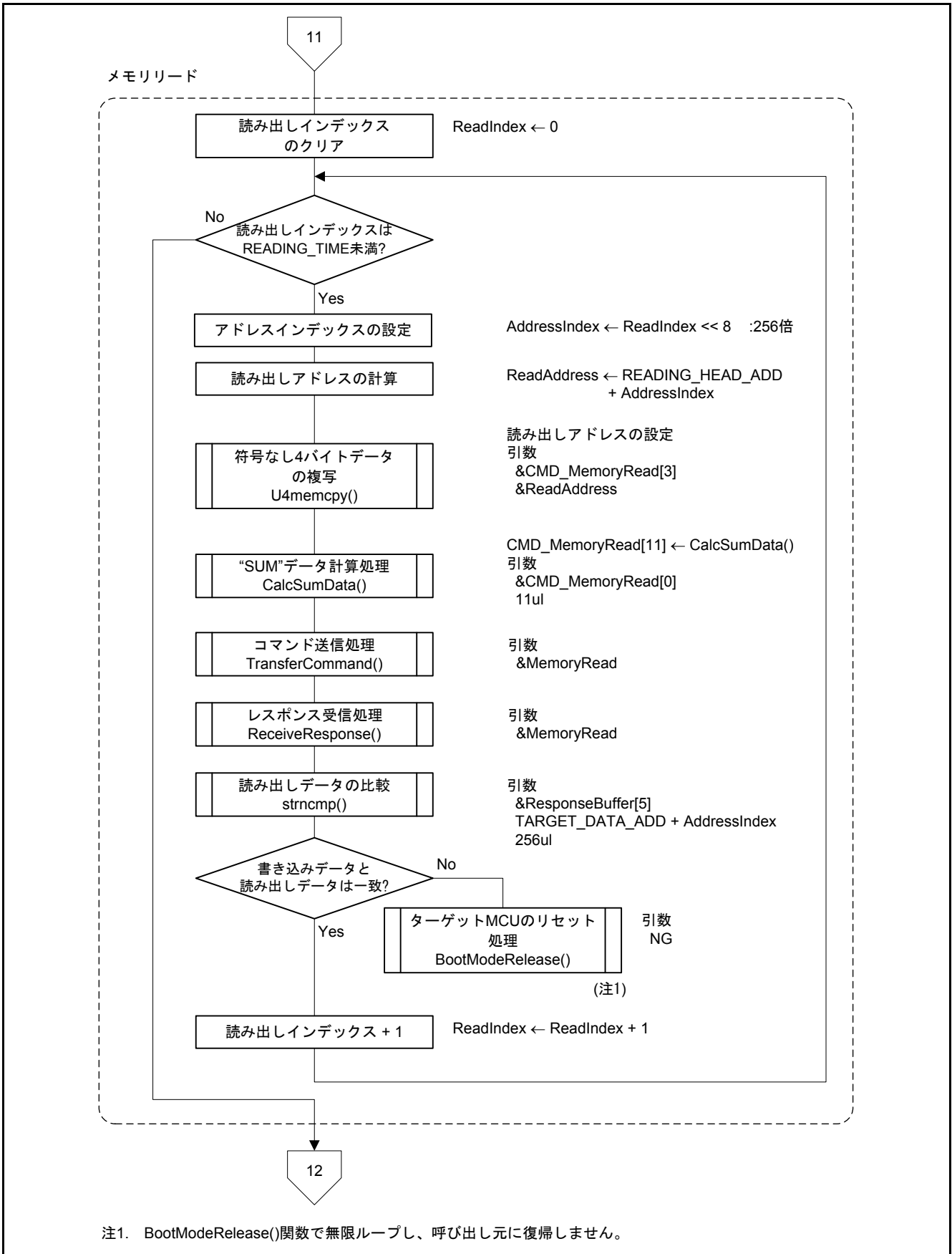


図 5.27 メイン処理、通信プロトコル制御

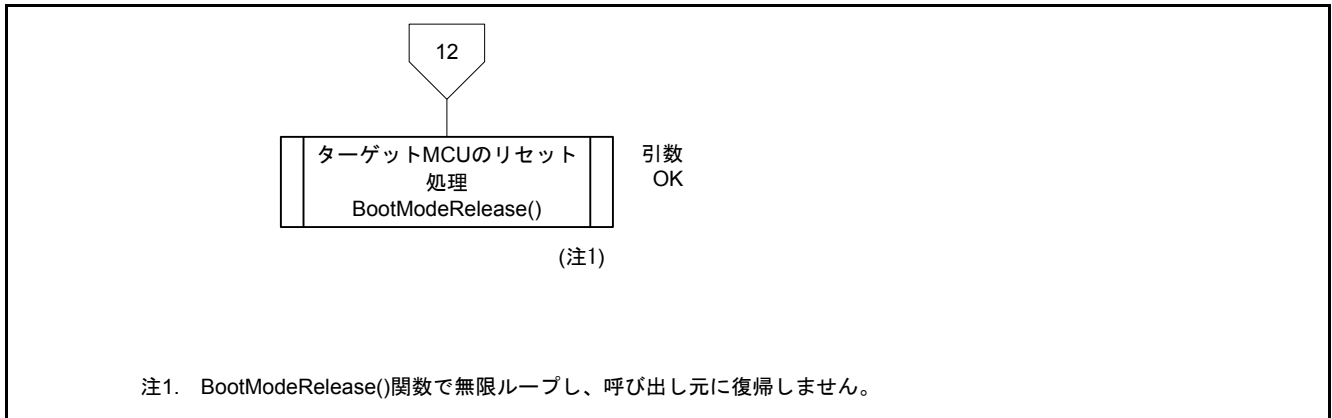


図 5.28 メイン処理、通信プロトコル制御

5.10.2 周辺機能初期設定

図 5.29に周辺機能初期設定のフローチャートを示します。

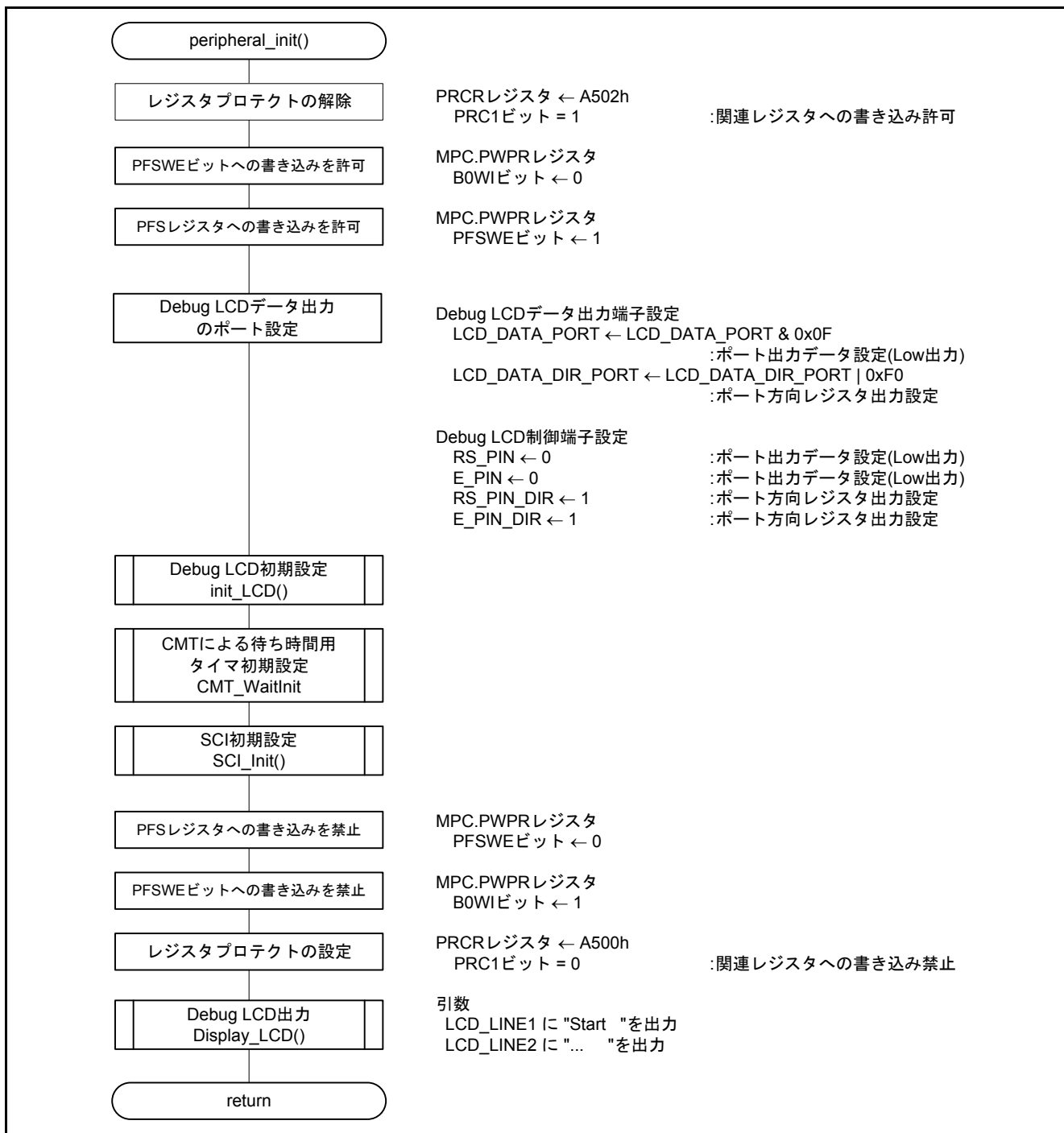


図 5.29 周辺機能初期設定



## 5.10.3 CMT による待ち時間用タイマ初期設定

図 5.30にCMT による待ち時間用タイマ初期設定のフローチャートを示します。

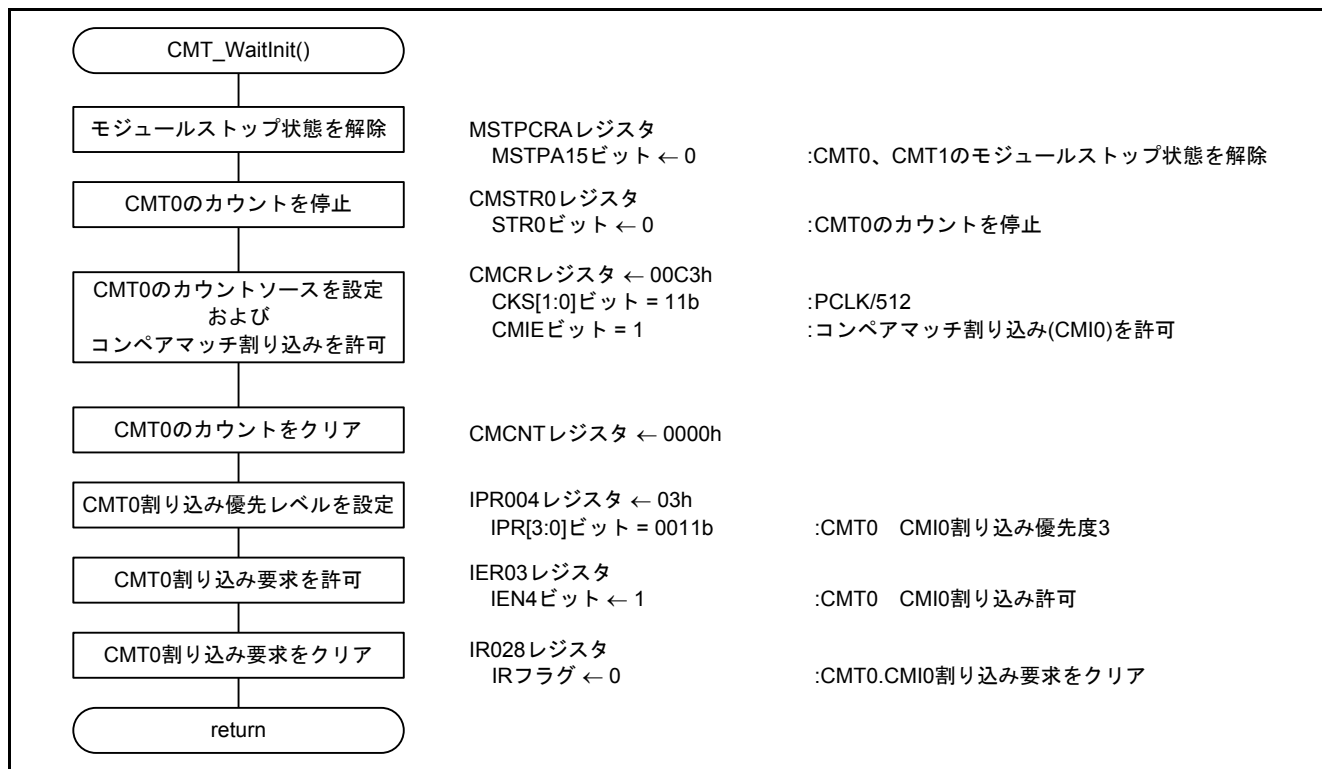


図 5.30 CMT による待ち時間用タイマ初期設定

## 5.10.4 CMT による待ち時間設定

図 5.31にCMT による待ち時間設定のフローチャートを示します。

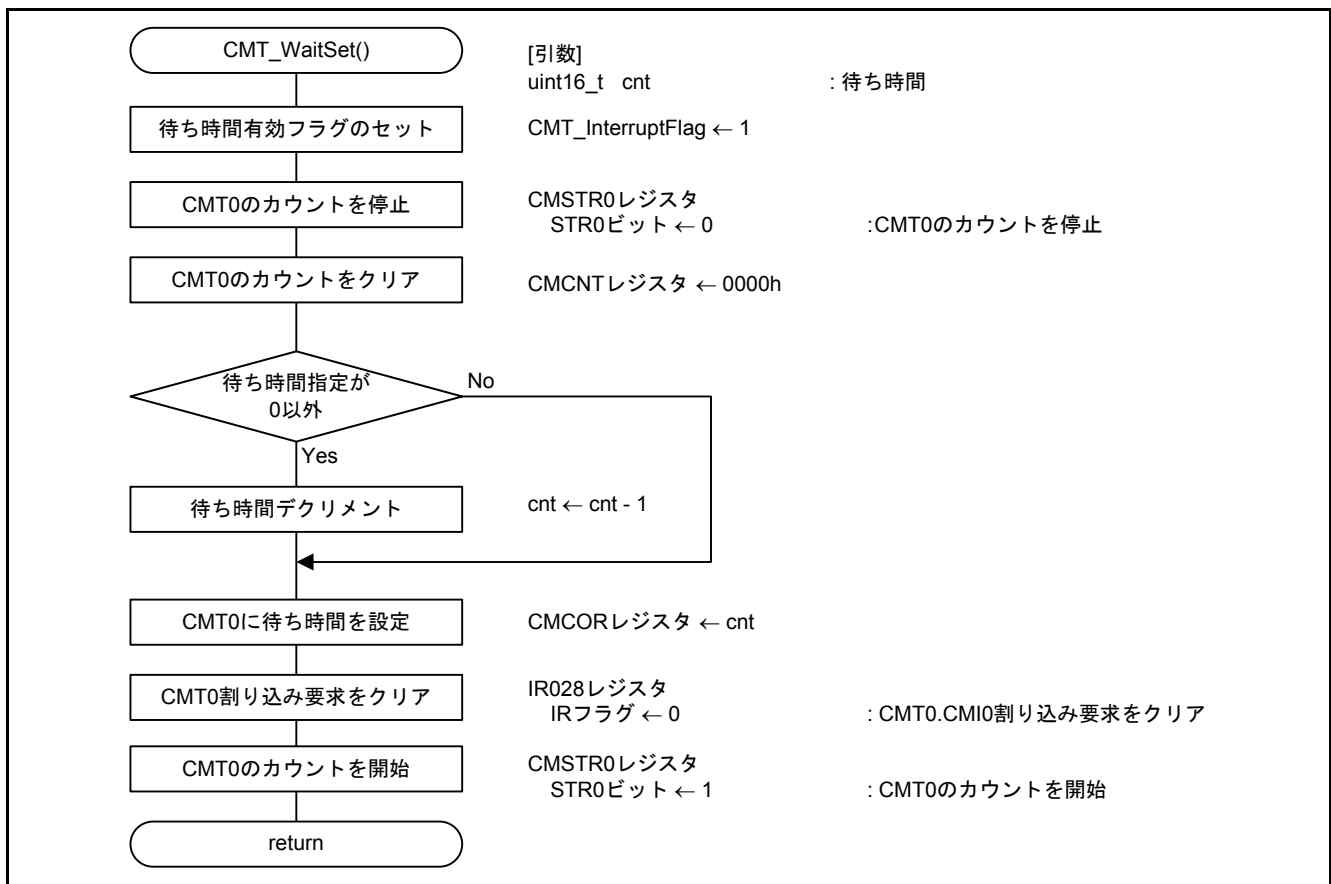


図 5.31 CMT による待ち時間設定

5.10.5 CMT による時間待ち処理

図 5.32にCMT による時間待ち処理のフローチャートを示します。

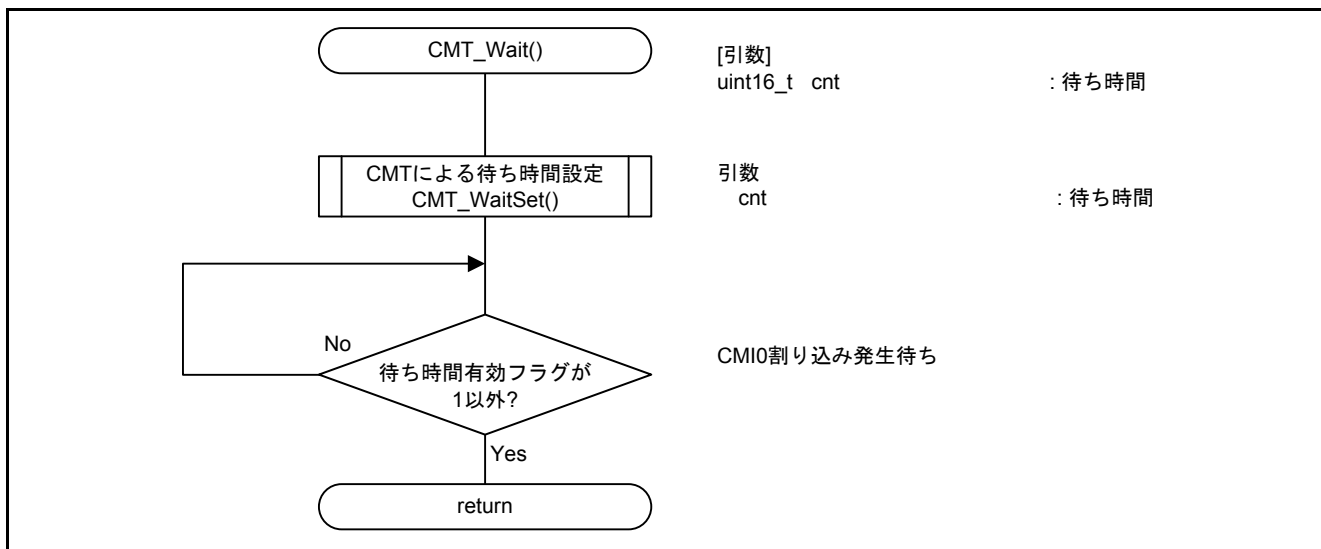


図 5.32 CMT による時間待ち処理

5.10.6 CMT0,CMIO 割り込み処理

図 5.33にCMT0,CMIO 割り込み処理のフローチャートを示します。

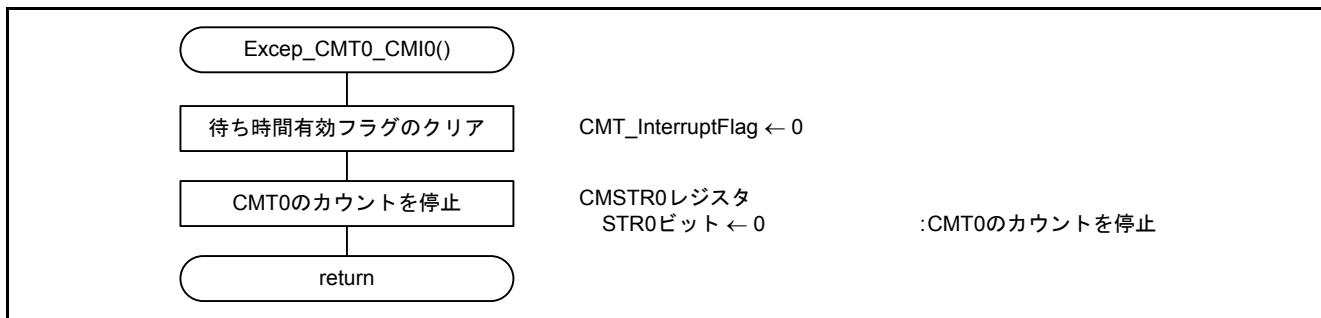


図 5.33 CMT0,CMIO 割り込み処理

## 5.10.7 SCI 初期設定

図 5.34にSCI 初期設定のフローチャートを示します。



図 5.34 SCI 初期設定

## 5.10.8 SCI ビットレート変更処理

図 5.35にSCI ビットレート変更処理のフローチャートを示します。

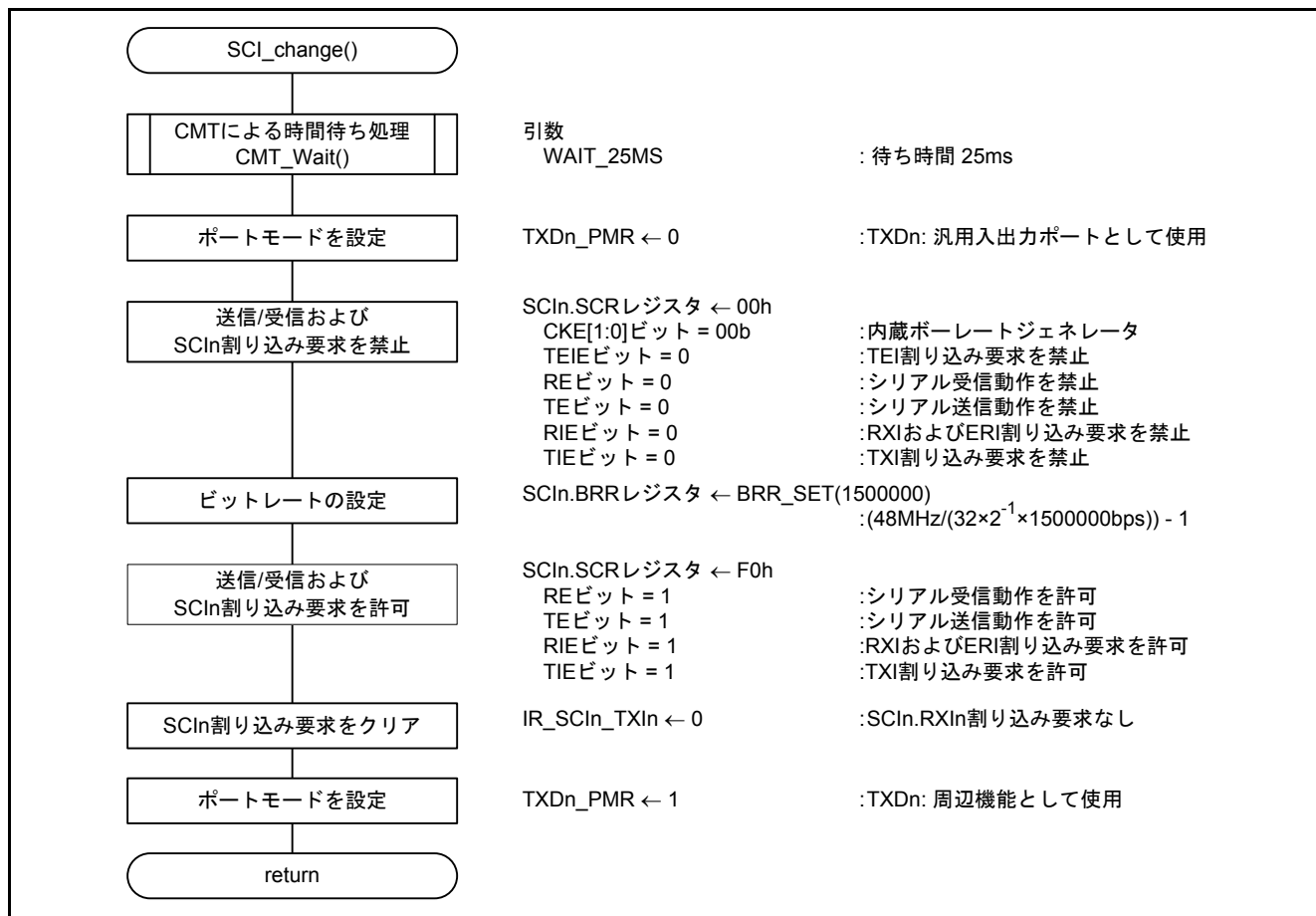


図 5.35 SCI ビットレート変更処理

## 5.10.9 “SUM” データ計算処理

図 5.36に “SUM” データ計算処理のフローチャートを示します。

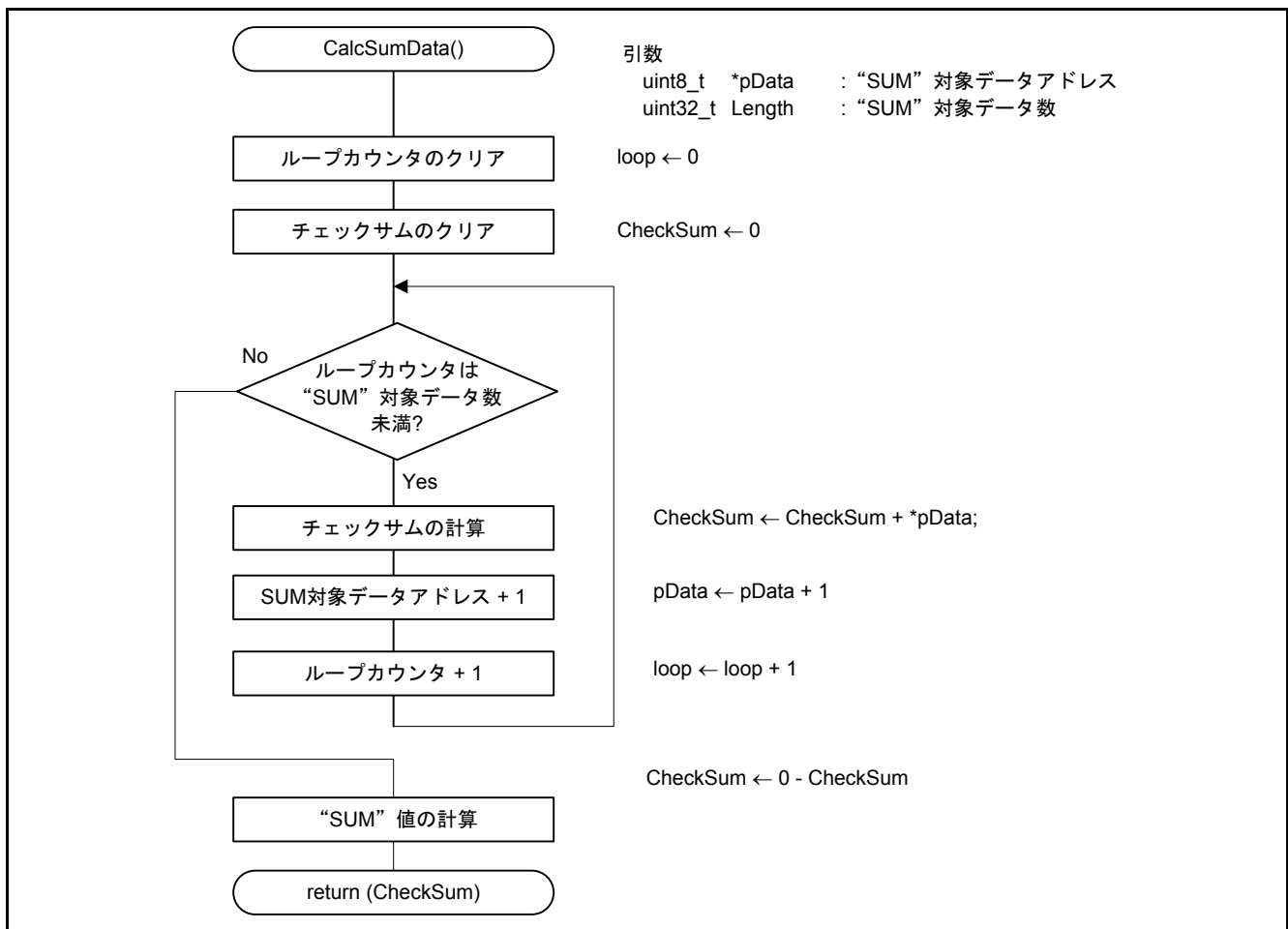


図 5.36 “SUM” データ計算処理

## 5.10.10 ターゲット MCU のブートモード起動処理

図 5.37にターゲット MCU のブートモード起動処理のフローチャートを示します。

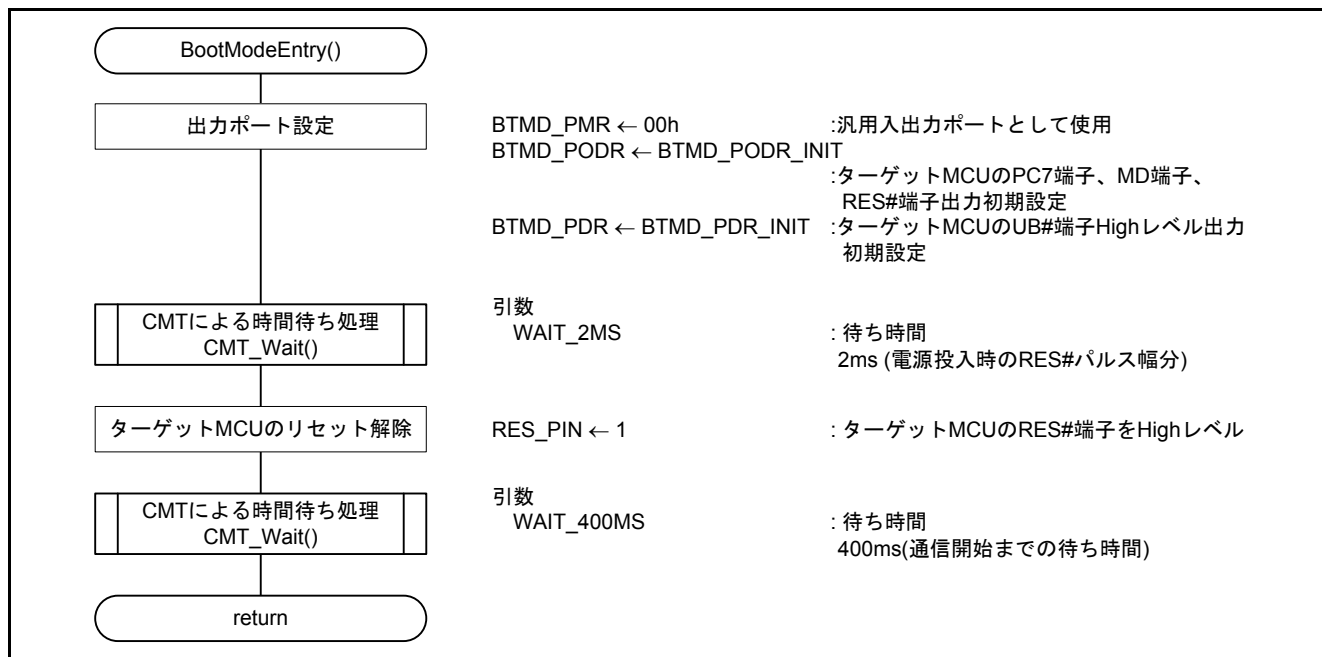


図 5.37 ターゲット MCU のブートモード起動処理

5.10.11 ターゲット MCU のリセット処理

図 5.38にターゲット MCU のリセット処理のフローチャートを示します。

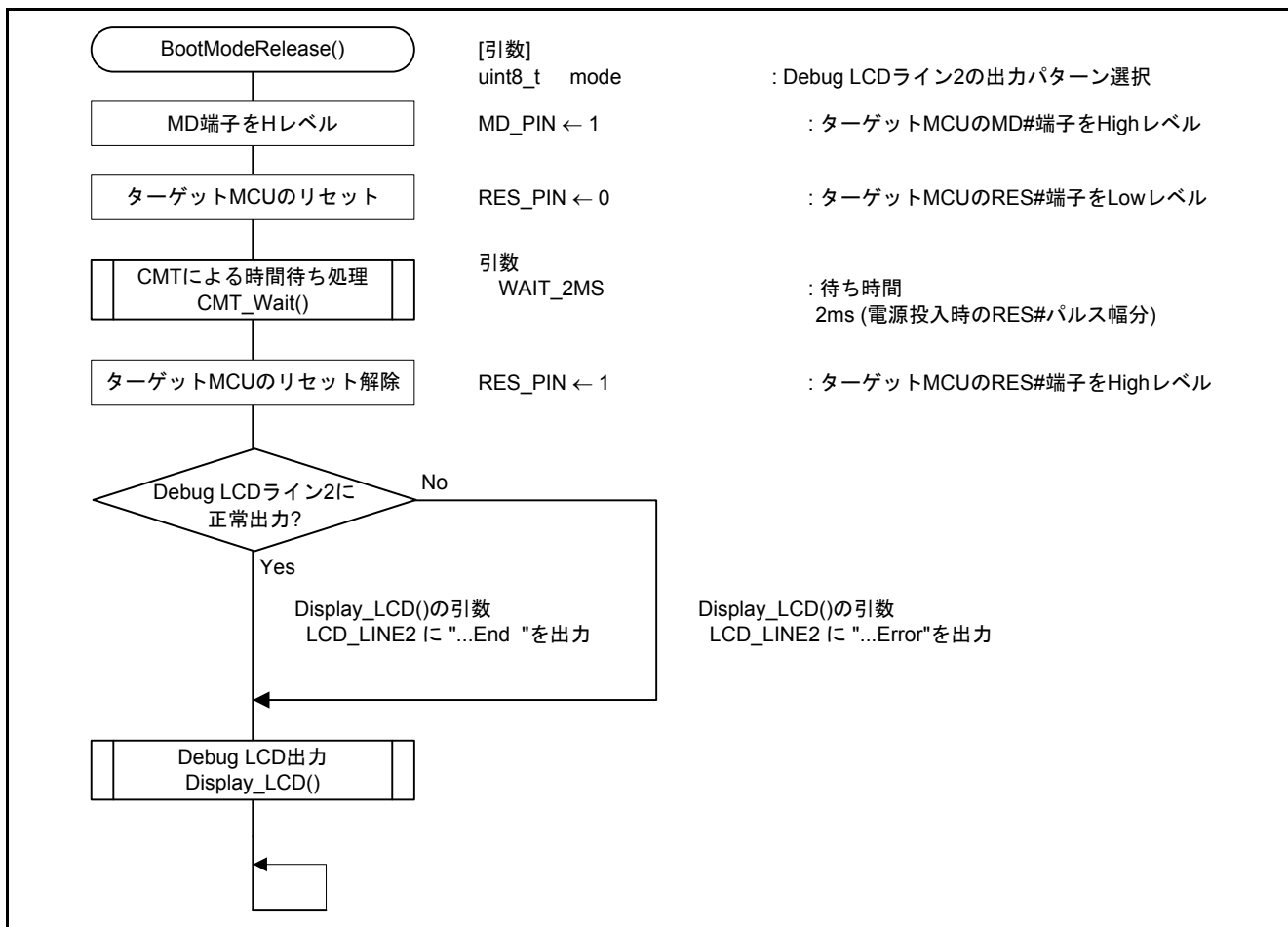


図 5.38 ターゲット MCU のリセット処理



5.10.12 コマンド送信処理

図 5.39にコマンド送信処理のフローチャートを示します。

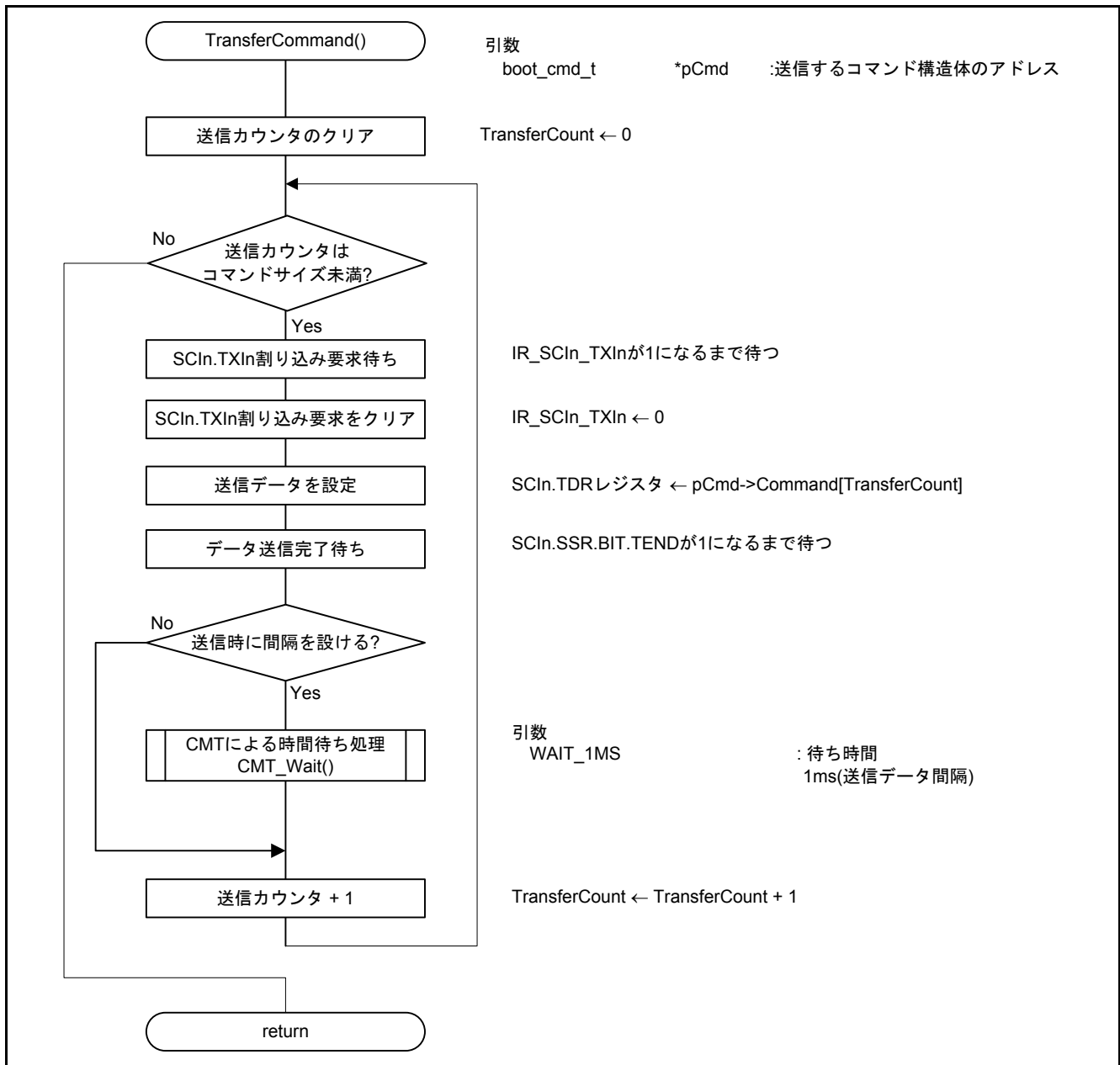


図 5.39 コマンド送信処理

5.10.13 レスポンス受信処理

図 5.40から図 5.43にレスポンス受信処理のフローチャートを示します。

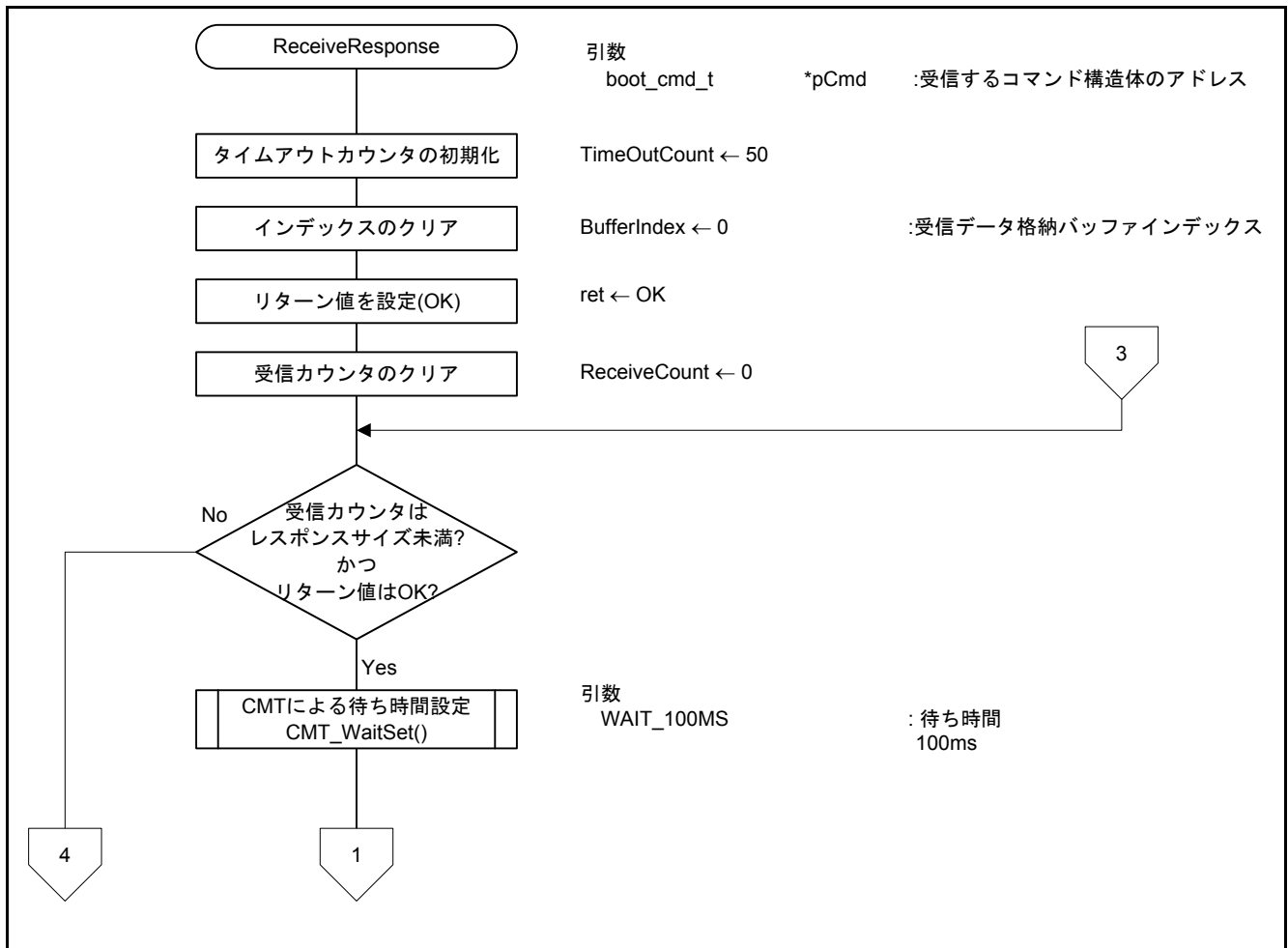


図 5.40 レスポンス受信処理

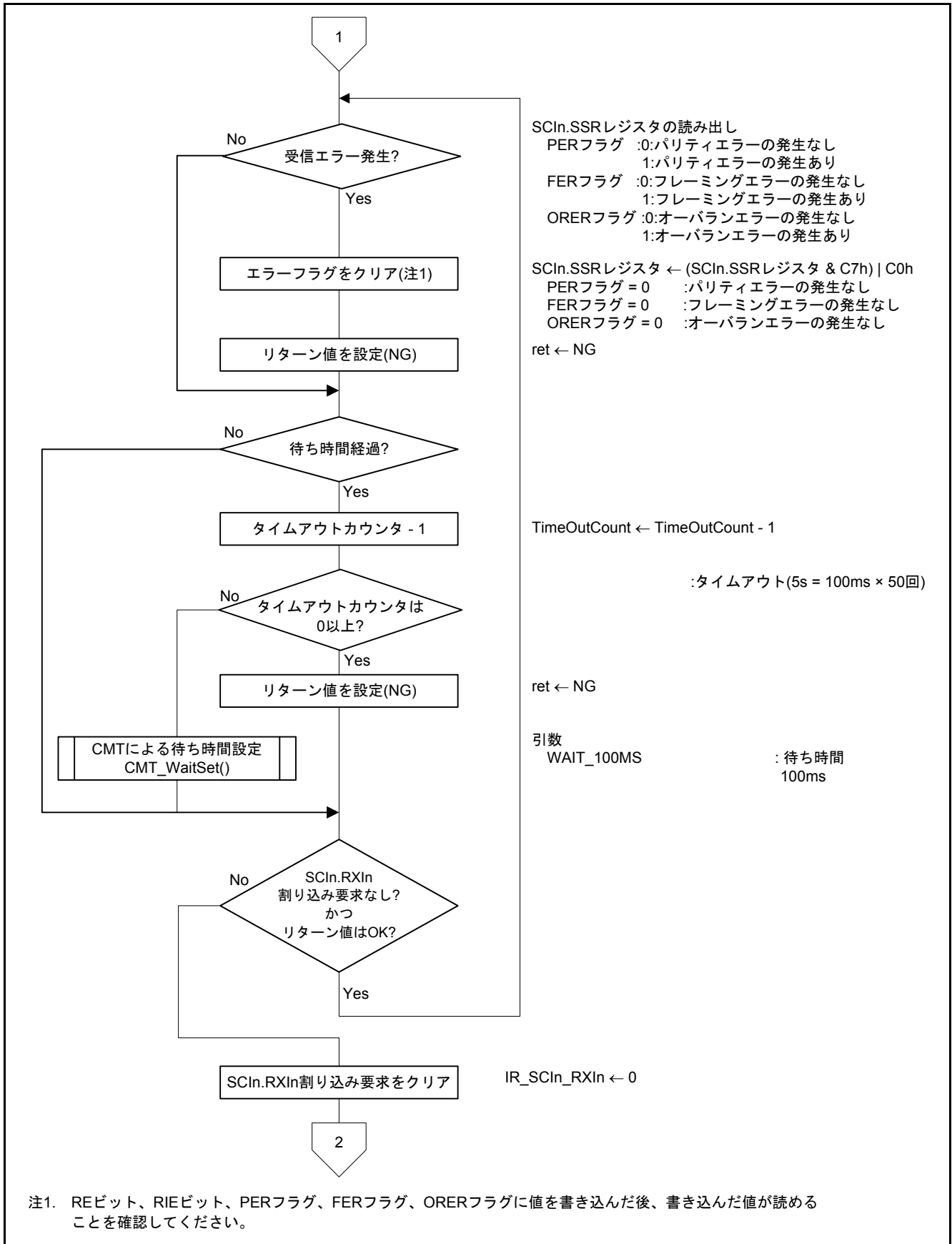


図 5.41 レスポンス受信処理

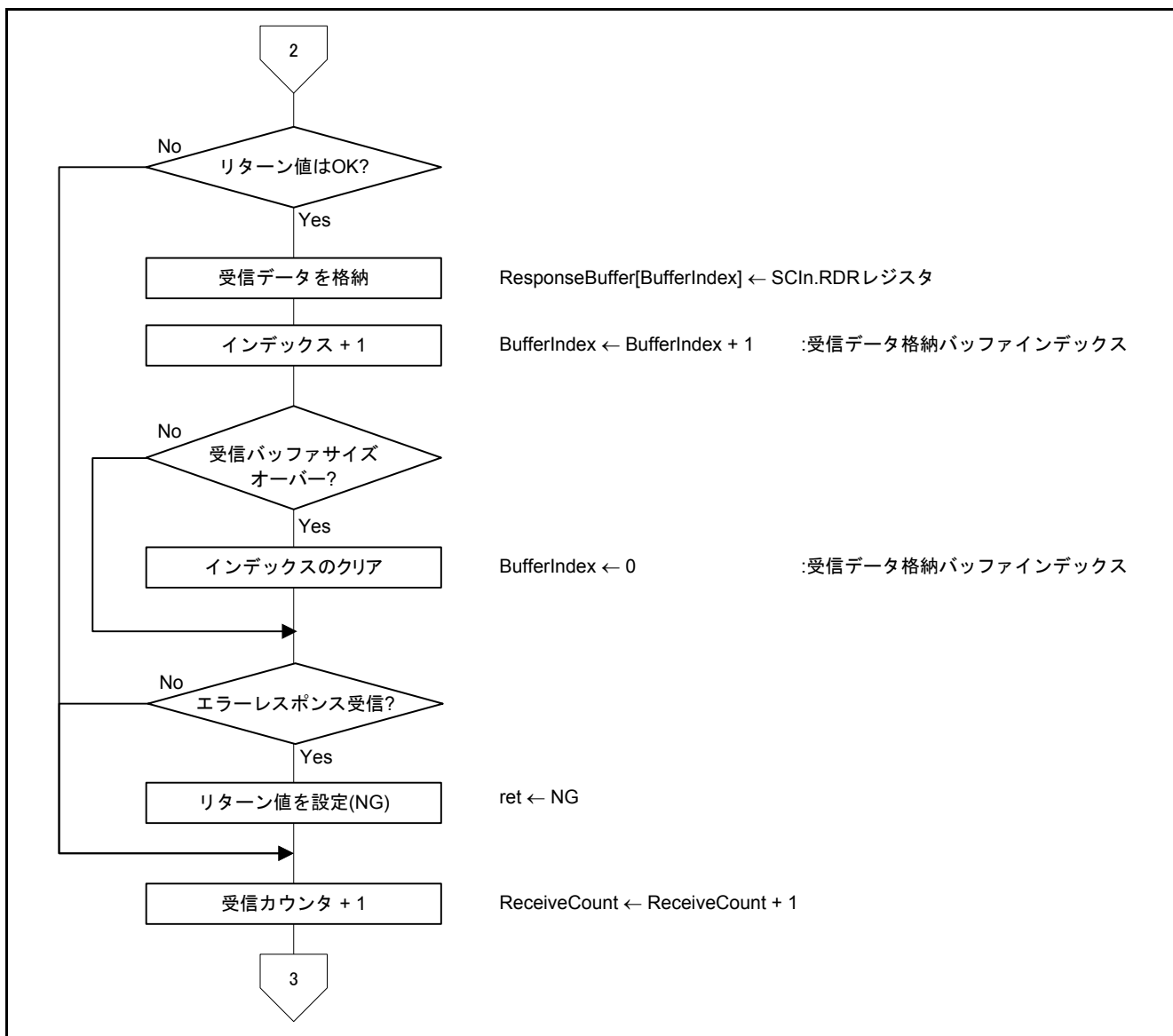


図 5.42 レスポンス受信処理

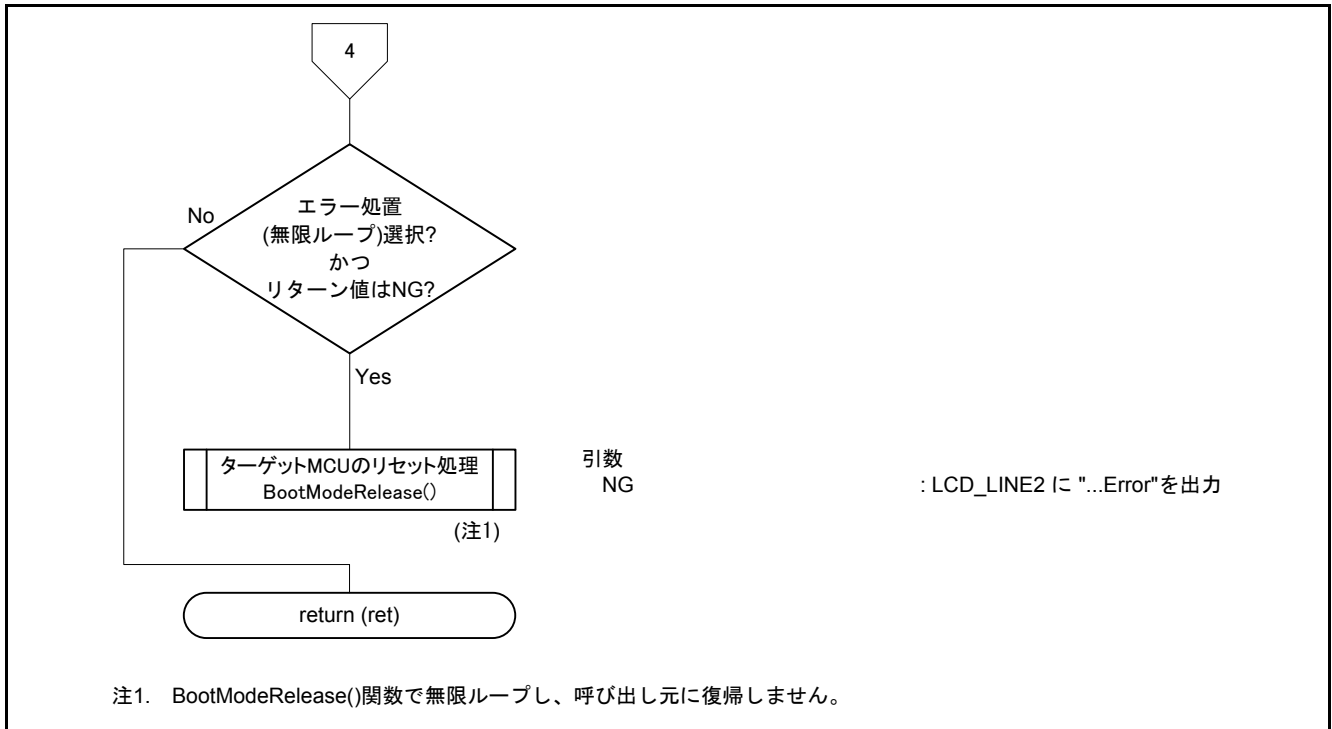


図 5.43 レスポンス受信処理

5.10.14 符号なし 4 バイトデータの複写

図 5.44に符号なし 4 バイトデータの複写のフローチャートを示します。

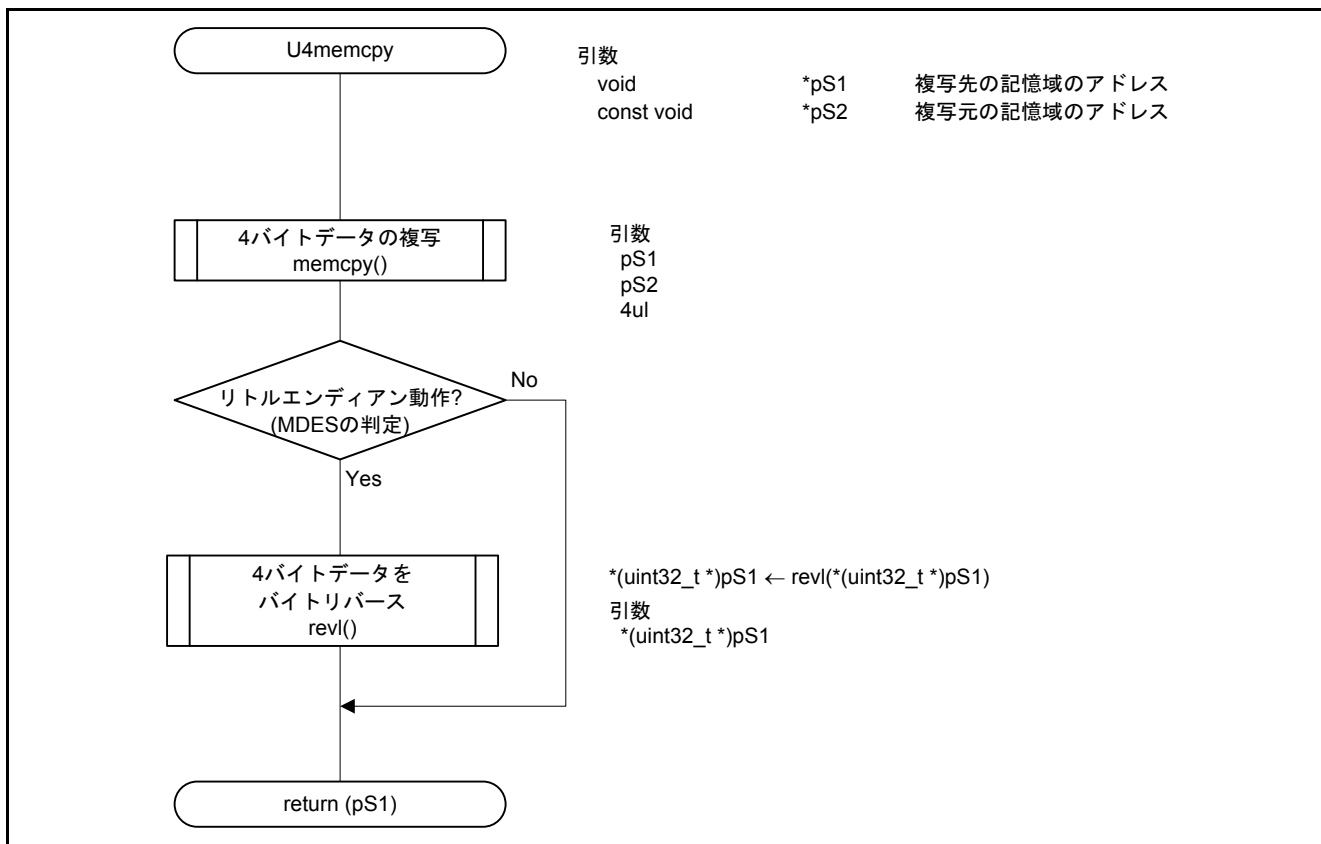


図 5.44 符号なし 4 バイトデータの複写

## 6. サンプルコード

サンプルコードは、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。

## 7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RX63N グループ、RX631 グループ ユーザーズマニュアル ハードウェア編 Rev.1.80 (R01UH0041JJ)  
(最新版をルネサス エレクトロニクスホームページから入手してください。)

テクニカルアップデート／テクニカルニュース

(最新の情報をルネサス エレクトロニクスホームページから入手してください。)

ユーザーズマニュアル：開発環境

RX ファミリー C/C++コンパイラパッケージ V.1.01 ユーザーズマニュアル Rev.1.00 (R20UT0570JJ)  
(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。)

## 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 | RX63N グループシリーズ<br>Renesas Starter Kit+ for RX63N を使用したRX63N グループ用<br>フラッシュプログラマ(ブートモード) |
|------|---|

| Rev. | 発行日        | 改訂内容 |      |
|------|------------|------|------|
|      |            | ページ  | ポイント |
| 1.00 | 2014.07.01 | —    | 初版発行 |
|      |            |      |      |

すべての商標および登録商標は、それぞれの所有者に帰属します。



## 製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本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. リザーブアドレスのアクセス禁止

【注意】リザーブアドレスのアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

### 4. クロックについて

【注意】リセット時は、クロックが安定した後、リセットを解除してください。

プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

### 5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

## ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、  
各品質水準は、以下に示す用途に製品が使用されることを意図しております。  
標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、  
家電、工作機械、パーソナル機器、産業用ロボット等  
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、  
防災・防犯装置、各種安全装置等  
当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問い合わせください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍用用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社とその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。



ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

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

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

■技術的なお問合せおよび資料のご請求は下記へどうぞ。

総合お問合せ窓口：<http://japan.renesas.com/contact/>