R01AN1356JJ0110
Rev. 1.10
2016.06.01

RL78/G13
セルフ・プログラミング（UART 受信データ）

要旨

本アプリケーションノートでは、セルフ書き込みによるフラッシュ・メモリ・プログラミングの使用方法の概要を説明します。フラッシュ・セルフ・プログラミング・ライブラリ Type01 を使用し、フラッシュ・メモリの書き換えを行います。

尚、本アプリケーションノートのサンプル・プログラムは、書き換え対象をブート領域に限定しています。セルフ・プログラミングの実行方法、および、コード・フラッシュの全領域の書き換え方法の詳細については、「RL78/G13 マイクロコントローラ フラッシュ・セルフ・プログラミング実行編 アプリケーションノート（R01AN0718J）」を参照してください。

対象デバイス

RL78/G13

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

本アプリケーションノートでは、セルフ書き込みによるフラッシュ・メモリ・プログラミングの使用方法を説明します。

LCD に現在のバージョン情報を表示します。その後、送信側からデータ（書き換え用データ）を受信し、「フラッシュ・アクセス中」を示す LED を点灯後にセルフ書き込みを行ってコード・フラッシュを書き換え用データに書き換えます。書き換えが完了すると LED を消灯し、LCD にバージョン情報を表示します。

表 1.1に使用する周辺機能と用途を示します。

表 1.1 使用する周辺機能と用途

周辺機能	用途
シリアル・アレイ・ユニット 0 チャネル 2	UART でデータの受信を行う
シリアル・アレイ・ユニット 0 チャネル 3	UART でデータの送信を行う
ポート入出力	LCD に文字列を表示する LED の点灯／消灯

1.1 フラッシュ・セルフ・プログラミング・ライブラリ概要

フラッシュ・セルフ・プログラミング・ライブラリは、RL78 マイクロコントローラに搭載されたファームウェアを使用し、コード・フラッシュ・メモリ内のデータを書き換えるためのソフトウエアです。

フラッシュ・セルフ・プログラミング・ライブラリをユーザ・プログラムから呼び出すことにより、コード・フラッシュ・メモリの内容を書き換えることができます。

フラッシュ・セルフ・プログラミングを行うためにはフラッシュ・セルフ・プログラミングの初期化処理や、使用する機能に対応する関数を C 言語、アセンブリ言語のどちらかでユーザ・プログラムから実行する必要があります。

1.2 コード・フラッシュ・メモリについて

RL78/G13(R5F100LE)のコード・フラッシュ・メモリの構成を以下に記載します。

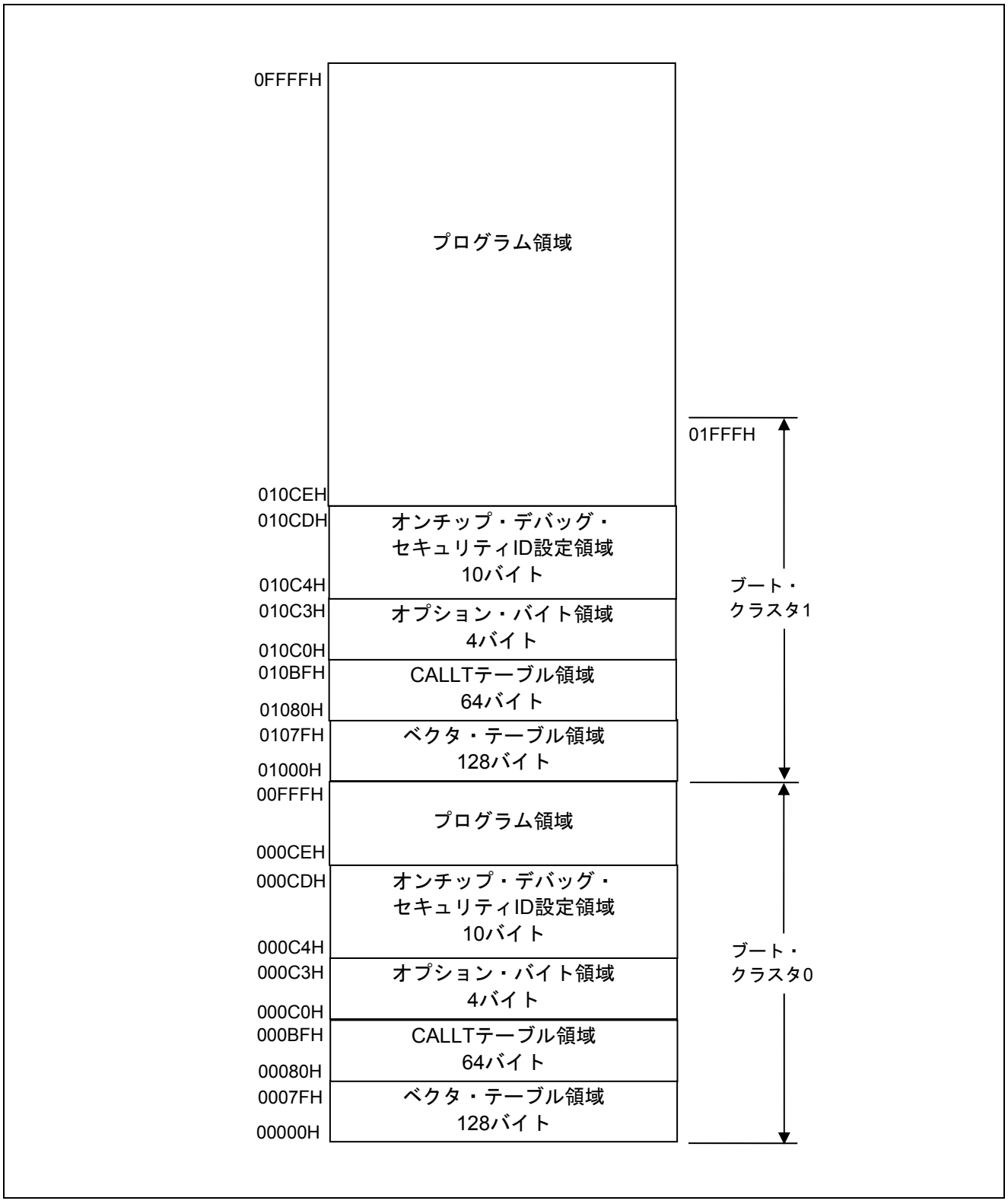


図 1.1 コード・フラッシュ・メモリの構成

注意　ブート・スワップ機能を使用する際には、ブート・クラスタ 0 のオプション・バイト領域（000C0H – 000C3H）は、ブート・クラスタ 1 のオプション・バイト領域（010C0H – 010C3H）と切り替わります。そのため、ブート・スワップ機能を使用する際には、010C0H – 010C3H に、000C0H – 000C3H と同じ値を設定してください。

RL78/G13 のコード・フラッシュ・メモリの特長を以下に記載します。

表 1.2　コード・フラッシュ・メモリの特長

項目	内容
消去、ベリファイの最小単位	1 ブロック（1024byte）
書き込みの最小単位	1 ワード（4byte）
セキュリティ機能	ブロック消去、書き込み、ブート領域の書き換え禁止設定が可能 （出荷時は全て許可）
	フラッシュ・シールド・ウィンドウにより、指定したウィンドウ範囲以外の書き込みおよび消去をフラッシュ・セルフ・プログラミング時のみ禁止にすることが可能
	フラッシュ・セルフ・プログラミング・ライブラリによりセキュリティ設定変更可能

注意　ブート領域の書き換え禁止とフラッシュ・シールド・ウィンドウ以外のセキュリティ設定は、フラッシュ・セルフ・プログラミング時は無効となります。

1.3 フラッシュ・セルフ・プログラミング

RL78/G13 には、フラッシュ・セルフ・プログラミングを行うためのライブラリが用意されています。書き換えプログラムからフラッシュ・セルフ・プログラミング・ライブラリの各関数を呼び出すことでフラッシュ・セルフ・プログラミングを行います。

RL78/G13 のフラッシュ・セルフ・プログラミングはシーケンサ（フラッシュ・メモリ制御用の専用回路）を使用してフラッシュの書き換え制御を行いますが、シーケンサの制御中はコード・フラッシュ・メモリを参照できません。そのため、シーケンサ制御中にユーザ・プログラムを動作させる必要がある場合、コード・フラッシュ・メモリの消去や書き込み、セキュリティ・フラグの設定等を行う時に、フラッシュ・セルフ・プログラミング・ライブラリの一部のセグメントや、書き換えプログラムを RAM に配置して制御を行う必要があります。シーケンサ制御中にユーザ・プログラムを動作させる必要が無い場合は、フラッシュ・セルフ・プログラミング・ライブラリや書き換えプログラムを ROM（コード・フラッシュ・メモリ）上に配置して動作させることが可能です。

1.3.1 ブート・スワップ機能

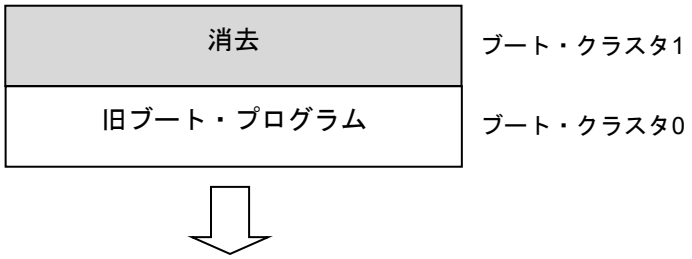
ベクタ・テーブル・データ、プログラムの基本機能、およびフラッシュ・セルフ・プログラミング・ライブラリを配置している領域の書き換え中に、電源の瞬断、外部要因によるリセットの発生などにより書き換えが失敗した場合、書き換え中のデータが破壊され、その後のリセットによるユーザ・プログラムの再スタートや再書き込みができなくなります。この問題を回避するための機能がブート・スワップ機能です。

ブート・スワップ機能では、ブート・プログラム領域であるブート・クラスタ 0 とブート・スワップ対象領域であるブート・クラスタ 1 を置換します。書き換え処理を行う前に、あらかじめ新しいブート・プログラムをブート・クラスタ 1 に書き込んでおきます。このブート・クラスタ 1 とブート・クラスタ 0 をスワップし、ブート・クラスタ 1 をブート・プログラム領域にします。これによって、ブート・プログラム領域の書き換え中に電源の瞬断が発生しても、次のリセット・スタートはブート・クラスタ 1 からブートを行うため、正常にプログラムが動作します。

以下にブート・スワップのイメージ図を記載します。

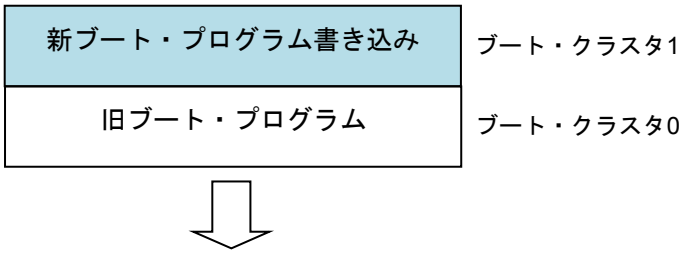
① ブート・クラスタ1の消去

FSL_Erase関数の呼び出しにより、ブート・クラスタ1（ブロック4～7）を消去します。



② ブート・クラスタ1へ新ブート・プログラム書き込み

FSL_Write関数の呼び出しにより、ブート・クラスタ1に新ブート・プログラムを書き込み、
FSL_IVerify関数の呼び出しにより、ブート・クラスタ1のベリファイを行います。
ここまでの処理で電源の瞬断、リセットの発生等により、新ブート・プログラムが正常に書き込まれなかった場合でも、旧ブート・プログラムから起動を行うため、正常にプログラムが動作します。

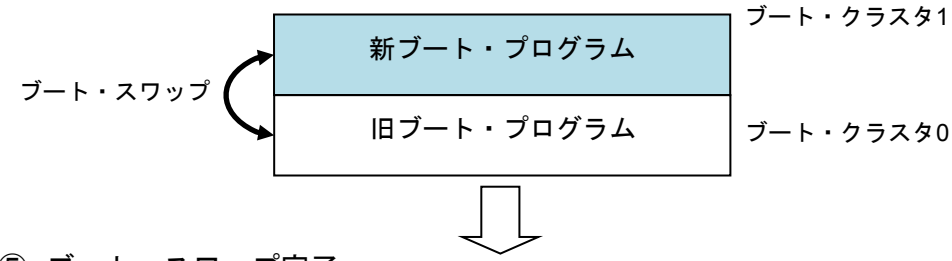


③ ブート・スワップ・ビットの設定

FSL_InvertBootFlag関数の呼び出しにより、ブート・フラグの切り替えを行います。
ブート・フラグの切り替え後に電源の瞬断、リセットが発生した場合には、書き換えが完了した新ブート・プログラムから起動を行うため、正常にプログラムが動作します。

④ リセットが発生した場合

リセットが発生すると、ブート・クラスタ0とブート・クラスタ1が入れ替わります。



⑤ ブート・スワップ完了

図 1.2 ブート・スワップのイメージ図

1.3.2 フラッシュ書き換え

フラッシュ・セルフ・プログラミングでのプログラムの書き換え動作イメージを以下に記載します。フラッシュ・セルフ・プログラミングを行うプログラムは、ブート・クラスタ 0 に配置しています。

本アプリケーションノートのサンプル・プログラムは、書き換え対象をブート領域に限定しています。セルフ・プログラミングの実行方法、および、コード・フラッシュの全領域の書き換え方法の詳細については、「RL78/G13 マイクロコントローラ フラッシュ・セルフ・プログラミング実行編 アプリケーションノート（R01AN0718J）」を参照してください。

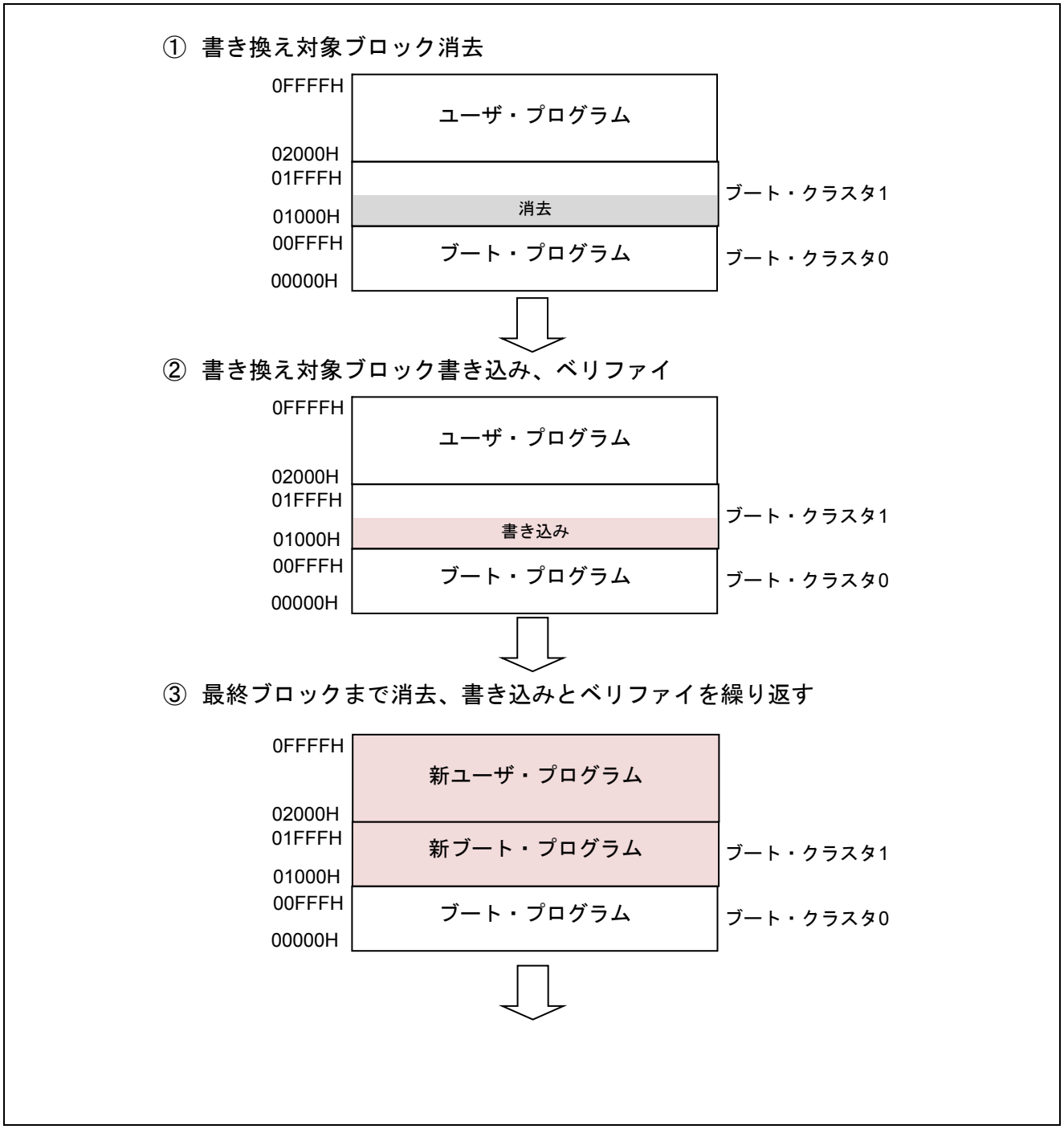


図 1.3 フラッシュ書き換えのイメージ図（1/2）

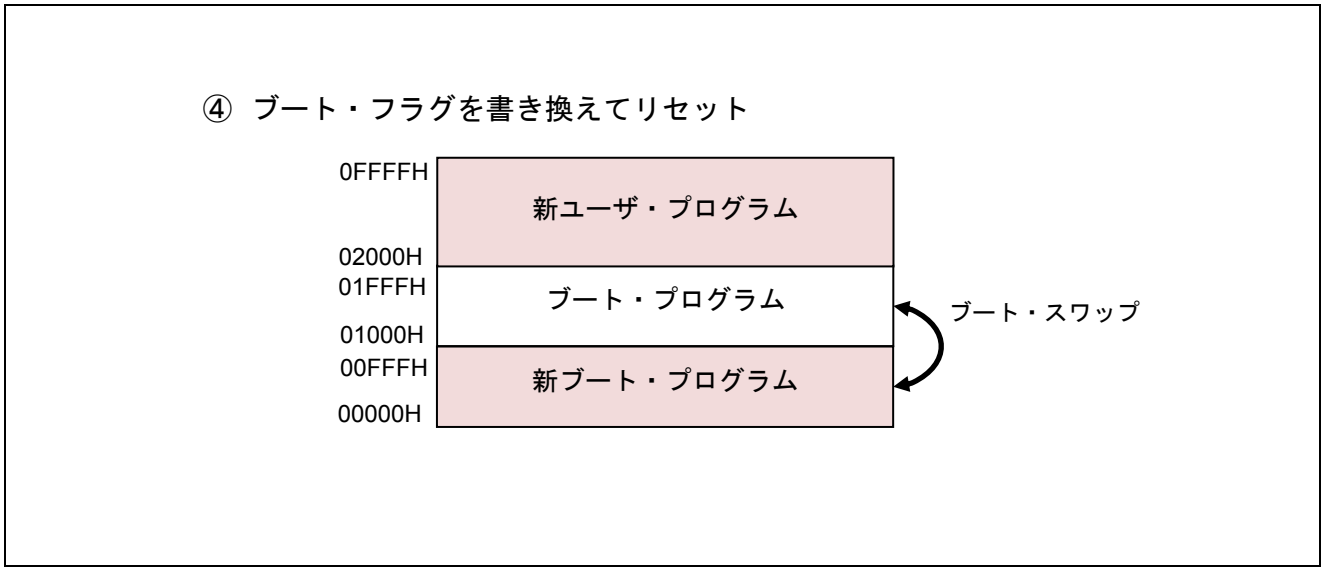


図 1.4 フラッシュ書き換えのイメージ図（2/2）

1.3.3 フラッシュ・シールド・ウィンドウ

フラッシュ・シールド・ウィンドウはフラッシュ・セルフ・プログラミング時のセキュリティ機能の一つで、指定したウィンドウ範囲以外の書き込み、及び消去をフラッシュ・セルフ・プログラミング時のみ禁止に設定する機能です。

以下に、スタート・ブロックが 08H、エンド・ブロックが 1FH の場合のイメージ図を記載します。

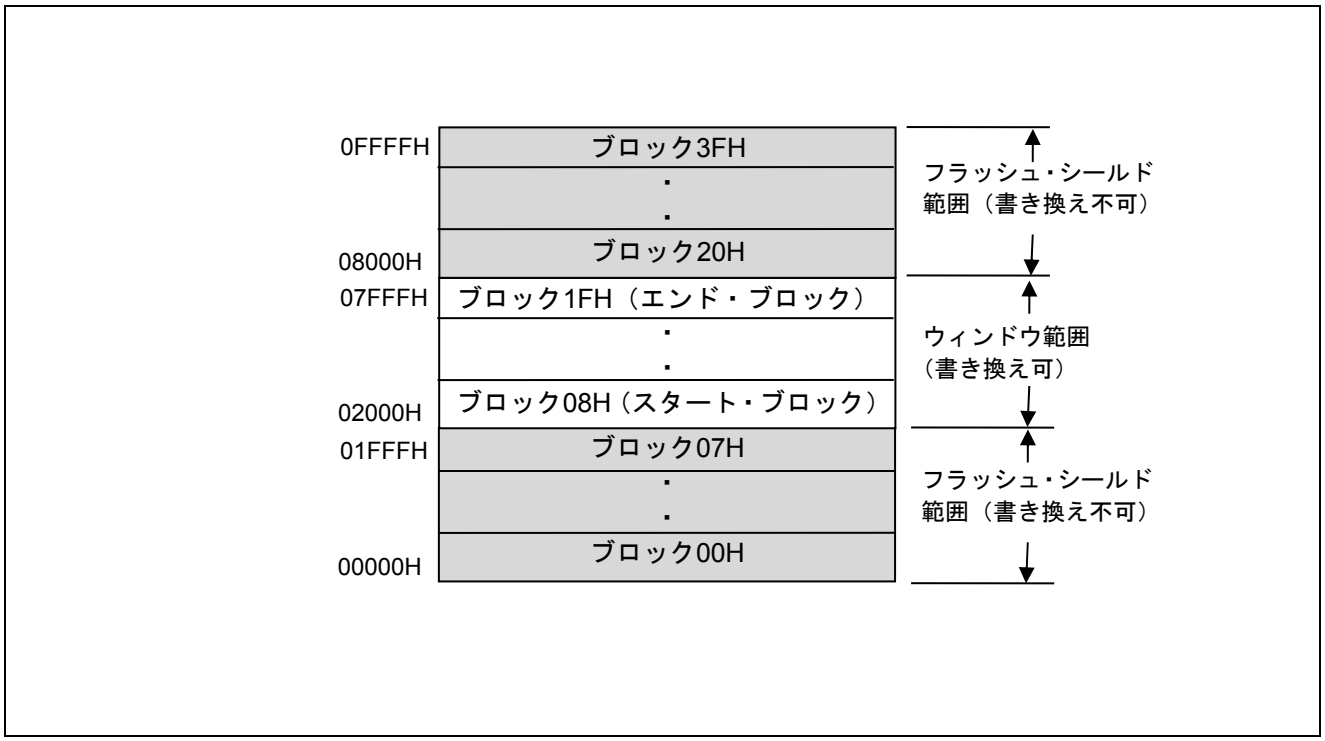


図 1.5 フラッシュ・シールド・ウィンドウのイメージ図

1.4 フラッシュ・セルフ・プログラミング・ライブラリ取得方法

コンパイルを実行する前に、最新版のフラッシュ・セルフ・プログラミング・ライブラリをダウンロードして、本サンプルコードの Workspace フォルダ内の以下のフォルダにライブラリファイルをコピーしてください。

”incrl78”フォルダに”fsl.h”、”fsl.inc”、”fsl_types.h”をコピーする。

”librl78”フォルダに”fsl.lib”をコピーする。

フラッシュ・セルフ・プログラミング・ライブラリは、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。

詳細は、最寄りのルネサス営業または特約店にお問い合わせください。

2. 動作確認条件

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

表 2.1 動作確認条件

項目	内容
使用マイコン	RL78/G13 (R5F100LEA)
動作周波数	● 高速オンチップ・オシレータ・クロック ：32MHz ● CPU／周辺ハードウエア・クロック ：32MHz
動作電圧	5.0V（2.9V～5.5V で動作可能） LVD 動作（V_{LVD}）：リセット・モード 2.81V (2.76V～2.87V)
統合開発環境	ルネサス エレクトロニクス製 CS+ V3.02.00
C コンパイラ	ルネサス エレクトロニクス製 CA78K0R V1.72
使用ボード	Renesas Starter Kit for RL78/G13（R0K50100LS000BE）
フラッシュ・セルフ・プログラミング・ライブラリ (Type, Ver)	FSLRL78 Type01, Ver2.20 注

注 最新バージョンをご使用／評価の上、ご使用ください。

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。

RL78/G13 初期設定（R01AN0451J）アプリケーションノート
RL78/G13 シリアル・アレイ・ユニット（UART 通信）（R01AN0459J）アプリケーションノート
RL78 マイクロコントローラ フラッシュ・セルフ・プログラミング・ライブラリ Type01 （R01AN0350J）アプリケーションノート

4. ハードウエア説明

4.1 ハードウエア構成例

図 4.1に本アプリケーションノートで使用する**ハードウエア構成例**を示します。

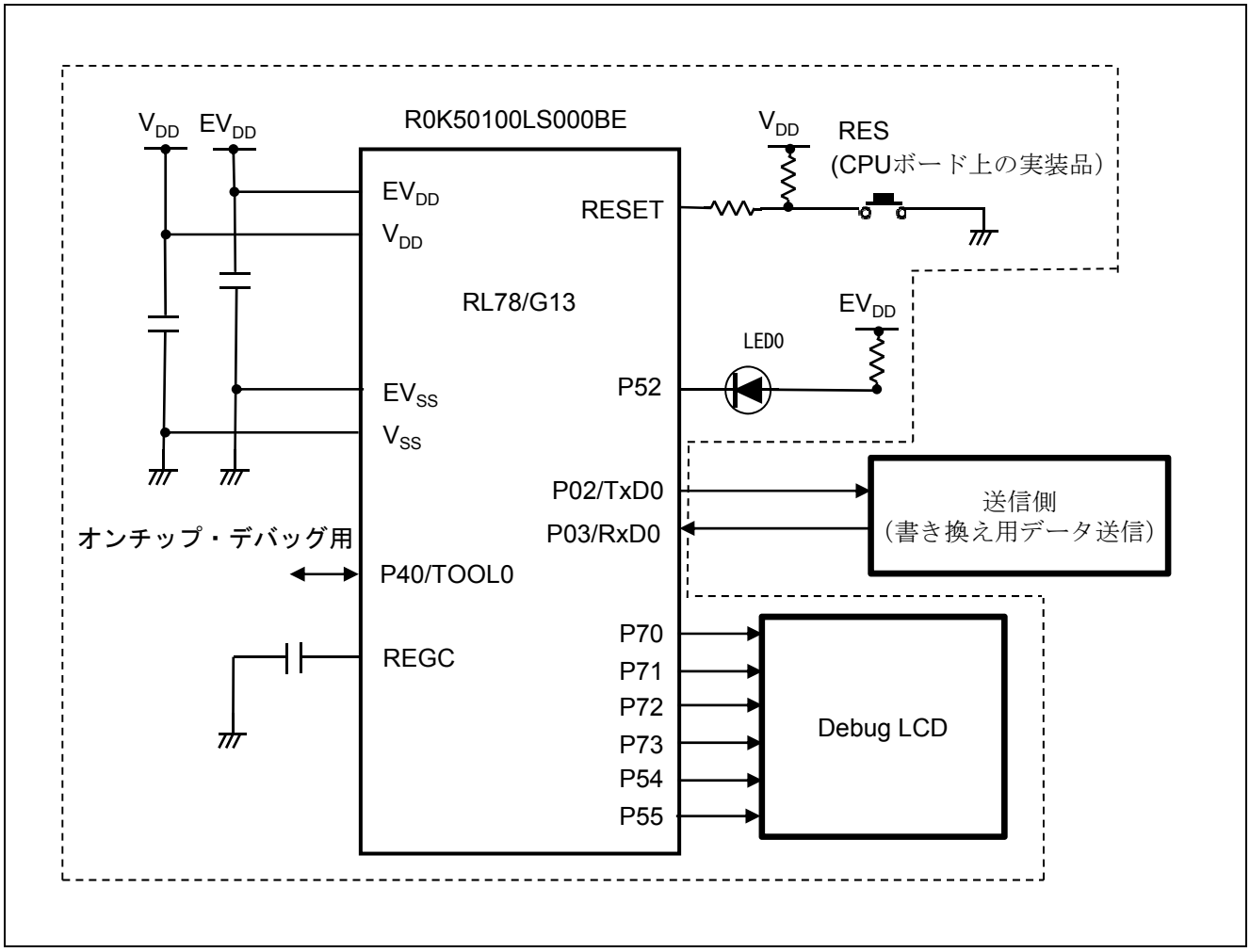


図 4.1 ハードウエア構成例

注意 1 この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続して下さい）。

2 V_{DD} は LVD にて設定したリセット解除電圧（V_{LVD}）以上にしてください。

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	機能
P02/ANI17/SO10/TxD1	出力	UART シリアル・データ送信用端子
P03/ANI16/SI10/RxD1/SDA10	入力	UART シリアル・データ受信用端子
P52	出力	LED0（フラッシュ・アクセス中を示す LED）の 点灯／消灯
P54	出力	Debug LCD 制御
P55	出力	Debug LCD 制御
P70/KR0/SCK21/SCL21	出力	Debug LCD 制御
P71/KR1/SI21/SDA21	出力	Debug LCD 制御
P72/KR2/SO21	出力	Debug LCD 制御
P73/KR3/SO01	出力	Debug LCD 制御

5. ソフトウエア説明

5.1 通信仕様

本アプリケーションノートのサンプル・プログラムは、UART で書き換えデータを受信し、フラッシュ・セルフ・プログラミングを行います。送信側からは START コマンド、WRITE コマンド、END コマンドの 3 つのコマンドのいずれかが送信されます。それぞれのコマンドに応じた処理を行い、正常終了の場合には送信側へ正常応答（0x01）を返します。異常終了の場合には応答を返さず、LCD に” ERROR! ”と表示して、以降の処理は行いません。以下に UART 通信設定と、各コマンドの仕様を記載します。

表 5.1 UART 通信設定

データ・ビット長[bit]	8
データ転送方向	LSB ファースト
パリティ設定	パリティなし
転送レート[bps]	115200

5.1.1 START コマンド

START コマンドを受信するとフラッシュ・セルフ・プログラミングの初期設定を行います。正常終了すると送信側へ正常応答（0x01）を返します。異常終了の場合には応答を返さず、LCD に” ERROR! ”と表示して、以降の処理は行いません。

START コード (0x01)	データ長 (0x0002)	コマンド (0x02)	データ (なし)	チェックサム (1byte)

5.1.2 WRITE コマンド

WRITE コマンドを受信すると受信したデータをフラッシュ・メモリへ書き込み、1 ブロックの書き込み毎にベリファイを行います。正常終了すると送信側へ正常応答（0x01）を返します。異常終了の場合には応答を返さず、LCD に” ERROR! ”と表示して、以降の処理は行いません。

START コード (0x01)	データ長 (0x0102)	コマンド (0x03)	データ (256byte)	チェックサム (1byte)

5.1.3 END コマンド

END コマンドを受信すると現在書き込んでいるブロックのベリファイを行います。ベリファイが正常終了の場合はブート・フラグを反転後にリセットを発生させ、ブート・スワップを行います。ベリファイが異常終了の場合には LCD に” ERROR! ”と表示して、以降の処理は行いません。END コマンドを受信した場合には、正常終了／異常終了に関わらず、送信側への応答は返しません。

START コード (0x01)	データ長 (0x0002)	コマンド (0x04)	データ (なし)	チェックサム (1byte)

※チェックサムは、コマンド部とデータ部のバイト単位の加算値です。

5.1.4 通信シーケンス

本サンプル・プログラムは送信側からのコマンド受信により、以下に示すシーケンスで動作を行います。

(1) 送信側：

START コマンドを送信します。

(2) 本サンプル・プログラム：

「フラッシュ・アクセス中」を示す LED1 を点灯し、フラッシュ・セルフ・プログラミングの初期設定を行い、正常終了すると応答（0x01）を返します。

(3) 送信側：

WRITE コマンドと書き換え用データ（256byte）を送信します。

(4) 本サンプル・プログラム：

受信したデータをコード・フラッシュに書き込みます。書き込みアドレスは 0x1000（ブート・クラスタ 1 の先頭）から開始され、以降は WRITE コマンドと書き換え用データを受信する度に、受信データサイズ（書き換え用データサイズ：256byte）だけ加算されていきます。

1block（1024byte）の書き換えが完了した場合にはベリファイを行います。

これらの処理が正常終了すると応答（0x01）を返します。

(5) 全データの書き換えが完了するまで(3)と(4)を繰り返します。

(6) 送信側：

END コマンドを送信します。

(7) 本サンプル・プログラム：

現在の書き換え対象ブロックのベリファイを行い、ブート・フラグを切り替えて「フラッシュ・アクセス中」を示す LED0 を消灯後にリセットを発生させます。

5.2 動作概要

本アプリケーションノートでは、セルフ書き込みによるフラッシュ・メモリ・プログラミングの使用方法を説明します。

LCD に現在のバージョン情報を表示します。その後、送信側からデータ（書き換え用データ）を受信し、「フラッシュ・アクセス中」を示す LED を点灯後にセルフ書き込みを行ってコード・フラッシュを書き換え用データに書き換えます。書き換えが完了すると LED を消灯し、LCD にバージョン情報を表示します。

(1) SAU0 のチャネル 2、3 の初期設定を行います。

<設定条件>

- SAU0 チャネル 2、3 を UART として使用します。
- データ出力は P02/TxD1 端子、データ入力は P03/RxD1 端子を使用します。
- データ長は 8 ビットを使用します。
- データ転送方向設定は LSB ファーストを使用します。
- パリティ設定はパリティビットなしを使用します。
- 受信データ・レベル設定は標準を使用します。
- 転送レートは 115200bps を使用します。

(2) 入出力ポートを設定します。

<設定条件>

- LED 点灯制御ポート（LED0）：P52 を出力ポートに設定します。

(3) 割り込みを禁止します。

(4) UART1 の動作を開始します。

(5) LCD の初期設定を行い、LCD に定数 LCD_STRING で設定された文字列を表示します。

(6) HALT モードに移行して、送信側からの送信データを待ちます。

UART 受信完了割り込み要求で HALT モードから通常動作に移行します。

(7) 送信側から START コマンド（0x02）を受信したら、セルフ・プログラミングの初期設定を行います。

- P52 を Low レベル出力にし、「フラッシュ・アクセス中」を示す LED0 を点灯します。
- FSL_Init 関数を呼び出し、フラッシュ・セルフ・プログラミング環境の初期化を行い、以下のように設定します。

  電圧モード　：フルスピードモード

  CPU の動作周波数　：32[MHz]

  ステータス・チェック・モード　：ステータス・チェック・インターナル・モード
- FSL_Open 関数を呼び出し、フラッシュ・セルフ・プログラミングを開始（フラッシュ環境の開始）します。
- FSL_PrepareFunctions 関数を呼び出し、RAM 実行が必要なフラッシュ関数（標準書き換え関数）を使用できる状態にします。
- FSL_PrepareExtFunctions 関数を呼び出し、RAM 実行が必要なフラッシュ関数（拡張機能関数）を使用できる状態にします。
- FSL_GetFlashShieldWindow 関数を呼び出し、フラッシュ・シールド・ウィンドウの開始ブロックと終了ブロックを取得します。
- フラッシュ・シールド・ウィンドウの開始ブロックが 0 以外、または終了ブロックが 63 以外の場合は、FSL_SetFlashShieldWindow 関数を呼び出し、フラッシュ・シールド・ウィンドウの開始ブロックを 0、終了ブロックを 63 に設定します。

(8) 書き込み先アドレスを 0x1000（ブート・クラスタ 1 の先頭）に設定します。

(9) 送信側へ正常応答（0x01）を送信します。

(10) WRITE コマンド（0x03）と書き込みデータ（256byte）を受信します。

(11) 書き込み先アドレスから、書き換え対象ブロックを算出します。

(12) FSL_BlankCheck 関数を呼び出し、書き換え対象ブロックが書込み済みかどうかを確認します。

(13) 書き換え対象ブロックが書込み済みの場合は、FSL_Erase 関数を呼び出し、書き換え対象ブロックを消去します。

(14) FSL_Write 関数を呼び出し、書き込み先アドレスに受信データを書き込みます。

(15) 書き込み先アドレスを書き込みサイズ分加算します。

(16) 送信側へ正常応答（0x01）を送信します。

(17) WRITE コマンドと書き込みデータ（256byte）、または END コマンド（0x04）を受信します。

(18) 1 ブロック（1024byte）の書き込みが完了するか、送信側から END コマンド（0x04）を受信するまで(14)～(17)の処理を繰り返します。1 ブロック（1024byte）の書き込みが完了するか、送信側から END コマンド（0x04）を受信した場合は次の処理を行います。

(19) FSL_IVerify 関数を呼び出し、書き換え対象ブロックをベリファイします。

(20) 送信側から END コマンド（0x04）を受信していない場合は(11)～(19)の処理を繰り返します。END コマンドを受信した場合は次の処理を行います。

(21) FSL_InvertBootFlag 関数を呼び出し、ブート・フラグの値を反転します。リセット時に、ブート・クラスタ 0 とブート・クラスタ 1 が入れ替わります。

(22) P52 を High レベル出力にし、「フラッシュ・アクセス中」を示す LED0 を消灯後、FSL_ForceReset 関数を呼び出して内部リセットを発生させます。

注意 フラッシュ・セルフ・プログラミングを正常終了することができなかった場合（処理中にエラーが発生した場合）は、LCD に” ERROR! ”と表示し、以降の処理は行いません。

5.3 ファイル構成

表 5.2に統合開発環境で自動生成されるファイルへの追加関数、追加ファイル一覧を示します。

表 5.2 追加関数、追加ファイル一覧

ファイル名	概要	備考
r_main.c	メインモジュール	追加関数： R_MAIN_PacketAnalyze R_MAIN_SelfExecute R_MAIN_SelfInitialize R_MAIN_WriteExecute
r_cg_serial_user.c	SAU モジュール	追加関数： R_UART1_ReceiveStart R_UART1_SendStart
lcd.c	DebugLCD モジュール	Renesas Starter Kit for RL78/G13 付属の DebugLCD 制御処理

5.4 オプション・バイトの設定一覧

表 5.3にオプション・バイト設定一覧を示します。

表 5.3 オプション・バイト設定一覧

アドレス	設定値	内容
000C0H/010C0H	11101111B	ウォッチドッグ・タイマ 動作停止 (リセット解除後、カウント停止)
000C1H/010C1H	01111111B	LVD リセット・モード 2.81V (2.76V～2.87V)
000C2H/010C2H	11101000B	HS モード、HOCO クロック : 32MHz
000C3H/010C3H	10000100B	オンチップ・デバッグ許可 オンチップ・デバッグ・セキュリティ ID 認証失敗時にフラッシュ・メモリのデータを消去する

RL78/G13 のオプション・バイトは、ユーザ・オプション・バイト（000C0H – 000C2H）とオンチップ・デバッグ・オプション・バイト（000C3H）で構成されています。

電源投入時、またはリセット解除後、自動的にオプション・バイトを参照して、指定された機能の設定が行われます。セルフ・プログラミング時にブート・スワップを使用する場合は、000C0H – 000C3H は 010C0H – 010C3H と切り替わるので、010C0H – 010C3H にも 000C0H – 000C3H と同じ値を設定する必要があります。

5.5 オンチップ・デバッグ・セキュリティ ID

RL78/G13 は、第三者からメモリの内容を読み取られないようにするために、フラッシュ・メモリの 000C4H – 000CDH にオンチップ・デバッグ・セキュリティ ID 設定領域を用意しています。

セルフ・プログラミング時にブート・スワップを使用する場合は、000C4H – 000CDH と 010C4H – 010CDH が切り替わるので、010C4H – 010CDH にも 000C4H – 000CDH と同じ値を設定する必要があります。

5.6 リンク・ディレクティブ・ファイル

リンク・ディレクティブ・ファイルによって、フラッシュ・セルフ・プログラミングを行う書き換えプログラム、フラッシュ・セルフ・ライブラリはブロック 0～3（ブート・クラスタ 0）に配置されます。また、フラッシュ・セルフ・ライブラリで使用する RAM 領域を使用しないように設定を行います。

本サンプル・プログラムで使用するリンク・ディレクティブ・ファイルの概要を以下に記載します。

```
;*************************************************************
; Redefined ROM area
;*************************************************************
; ---------------------------------------------------------
; Define new memory entry for boot cluster 0
; ---------------------------------------------------------
MEMORY ROM          : ( 000000H, 001000H )
; ---------------------------------------------------------
; Define new memory entry for write-data area
; ---------------------------------------------------------
MEMORY OCDROM       : ( 00FE00H, 000200H )

;*************************************************************
; Library(fsl.lib) segment
;*************************************************************
; ---------------------------------------------------------
; Merge FSL_FCD segment
; ---------------------------------------------------------
MERGE FSL_FCD    := ROM
; ---------------------------------------------------------
; Merge FSL_FECD segment
; ---------------------------------------------------------
MERGE FSL_FECD := ROM
; ---------------------------------------------------------
; Merge FSL_RCD segment
; ---------------------------------------------------------
MERGE FSL_RCD    := ROM
; ---------------------------------------------------------
; Merge FSL_BCD segment
; ---------------------------------------------------------
MERGE FSL_BCD    := ROM
; ---------------------------------------------------------
; Merge FSL_BECD segment
; ---------------------------------------------------------
MERGE FSL_BECD := ROM

;*************************************************************
; Redefined RAM area
;*************************************************************
; ---------------------------------------------------------
; Define new memory entry for self-RAM
; ---------------------------------------------------------
MEMORY SELFRAM      : ( 0FEF00H, 00040AH )
; ---------------------------------------------------------
; Redefined default data segment RAM
; ---------------------------------------------------------
MEMORY RAM          : ( 0FF30AH, 000B16H )
; ---------------------------------------------------------
; Define new memory entry for saddr area
; ---------------------------------------------------------
MEMORY RAM_SADDR : ( 0FFE20H, 0001E0H )

;*************************************************************
; run-time library segment ( 0000H - FFFFH )
;*************************************************************
; ---------------------------------------------------------
; Merge @@LCODE,@@LCODEL(run-time library) segment
; ---------------------------------------------------------
MERGE @@LCODE    := ROM
MERGE @@LCODEL := ROM
;*************************************************************
; const segment
;*************************************************************
MERGE CNST_SEG := ROM
```

コード領域の定義
コードをブート領域に配置

OCD モニター領域の定義

フラッシュ・セルフ・ライブラリを
ブート領域に配置

フラッシュ・セルフ・ライブラリの使用
領域を標準 RAM 領域として使用しない
よう定義

標準 RAM 領域の定義

フラッシュ・セルフ・ライブラリの使用
領域を標準 RAM 領域として使用しない
よう定義

ランタイム・ライブラリをブート領域に
配置

const 変数をブート領域に配置

5.7 定数一覧

表 5.4にサンプルコードで使用する定数を示します。

表 5.4 サンプルコードで使用する定数

定数名	設定値	内容
LCD_DISPLAY	“Ver 1.0 ”	LCD に表示する文字列（バージョン情報） LCD に表示可能な文字数は 8 文字以内
ERR_DISPLAY	“ ERROR! “	エラー発生時に LCD に表示する文字列
NORMAL_END	0x00	正常終了
ERROR	0xFF	異常終了
NO_RECIEVE	0x00	コマンド受信状態：未受信
START_CODE	0x01	コマンド受信状態：START コード受信済
PACKET_SIZE	0x02	コマンド受信状態：データ長受信済
START	0x02	START コマンド
WRITE	0x03	WRITE コマンド
END	0x04	END コマンド
FULL_SPEED_MODE	0x00	フラッシュ・セルフ・ライブラリ初期化関数の引数：動作モードをフルスピードモードに設定
FREQUENCY_32M	0x20	フラッシュ・セルフ・ライブラリ初期化関数の引数： RL78/G13 の動作周波数 = 32MHz
INTERNAL_MODE	0x01	フラッシュ・セルフ・ライブラリ初期関数の引数： ステータス・チェック・インターナル・モードに設定
START_BLOCK_NUM	0x00	フラッシュ・シールド・ウィンドウの開始ブロック番号
END_BLOCK_NUM	0x3F	フラッシュ・シールド・ウィンドウの終了ブロック番号
BLOCK_SIZE	0x400	コード・フラッシュの 1 ブロックのサイズ（1024byte）
TXSIZE	0x01	送信側へ送信する応答データのサイズ
RXSIZE	0x102	受信バッファのサイズ

5.8 関数一覧

表 5.5に関数一覧を示します。

表 5.5 関数一覧

関数名	概要
R_UART1_Start	UART1 動作開始
R_UART1_ReceiveStart	UART1 データ受信
R_MAIN_PacketAnalyze	受信データ解析
R_MAIN_SelfExecute	フラッシュ・セルフ・プログラミング実行
R_MAIN_SelfInitialize	フラッシュ・セルフ・プログラミング初期設定
R_MAIN_WriteExecute	フラッシュ書き換え実行
R_UART1_SendStart	UART1 データ送信

5.9 関数仕様

サンプルコードの関数仕様を示します。

[関数名] R_ UART1_Start

概要	UART1 動作開始	
ヘッダ	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
宣言	void R_ UART1_Start(void)	
説明	シリアル・アレイ・ユニット 0 のチャネル 2、チャネル 3 の動作を開始し、通信待機状態にします。	
引数	なし	
リターン値	なし	
備考	なし	

[関数名] R_UART1_ReceiveStart

概要	UART1 データ受信	
ヘッダ	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
宣言	uint8_t R_UART1_ReceiveStart(uint16_t *rxlength, uint8_t *rxbuf)	
説明	受信データを受信バッファ rxbuf に格納し、受信データ長[byte]を rxlength に格納します。	
引数	rxlength	受信データ長[byte]格納アドレス
	rxbuf	受信データ・バッファのアドレス
リターン値	正常終了：NORMAL_END 異常終了：ERROR	
備考	なし	

[関数名] R_MAIN_PacketAnalyze

概要	受信データ解析	
ヘッダ	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h	
宣言	uint8_t R_MAIN_PacketAnalyze(uint16_t rxlength, uint8_t *rxbuf)	
説明	受信したコマンドのパラメータ・チェック、チェックサムの計算、比較を行い、受信したデータが正しいかどうかを判定します。	
引数	rxlength	受信データ長[byte]
	rxbuf	受信データ・バッファのアドレス
リターン値	START コマンド受信：START WRITE コマンド受信：WRITE END コマンド受信：END コマンドのパラメータ・エラー、チェックサム・エラー：ERROR	
備考	なし	

[関数名] R_MAIN_SelfExecute

概要	フラッシュ・セルフ・プログラミング実行
ヘッダ	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h
宣言	void R_MAIN_SelfExecute(void)
説明	フラッシュ・セルフ・プログラミングを行います。
引数	なし
リターン値	なし
備考	なし

[関数名] R_MAIN_SelfInitialize

概要	フラッシュ・セルフ・プログラミング初期設定
ヘッダ	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h
宣言	uint8_t R_MAIN_SelfExecute(void)
説明	フラッシュ・セルフ・プログラミング実行前の初期設定を行います。
引数	なし
リターン値	正常終了：FSL_OK パラメータ・エラー：FSL_ERR_PARAMETER 消去エラー：FSL_ERR_ERASE 内部ベリファイ・エラー：FSL_ERR_IVERIFY 書き込みエラー：FSL_ERR_WRITE フローエラー：FSL_ERR_FLOW
備考	なし

[関数名] R_MAIN_WriteExecute

概要	フラッシュ書き換え実行
ヘッダ	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h
宣言	uint8_t R_MAIN_SelfExecute(uint32_t WriteAddr)
説明	フラッシュ書き換えを行います。
引数	WriteAddr　　書き換え開始アドレス
リターン値	正常終了：FSL_OK 異常終了：ERROR
備考	なし

[関数名] R_UART1_SendStart

概要	UART1 データ送信	
ヘッダ	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
宣言	uint8_t R_UART1_SendStart(uint16_t txlength, uint8_t *txbuf)	
説明	txbuf に格納されているデータを txlength[byte]送信します。	
引数	txlength	送信データ長[byte]
	txbuf	送信データ・バッファのアドレス
リターン値	正常終了：NORMAL_END	
	パラメータ・エラー(txlength が 0 以下)：ERROR	
備考	なし	

5.10 フローチャート

図 5.1にサンプルコードの**全体フロー**を示します。

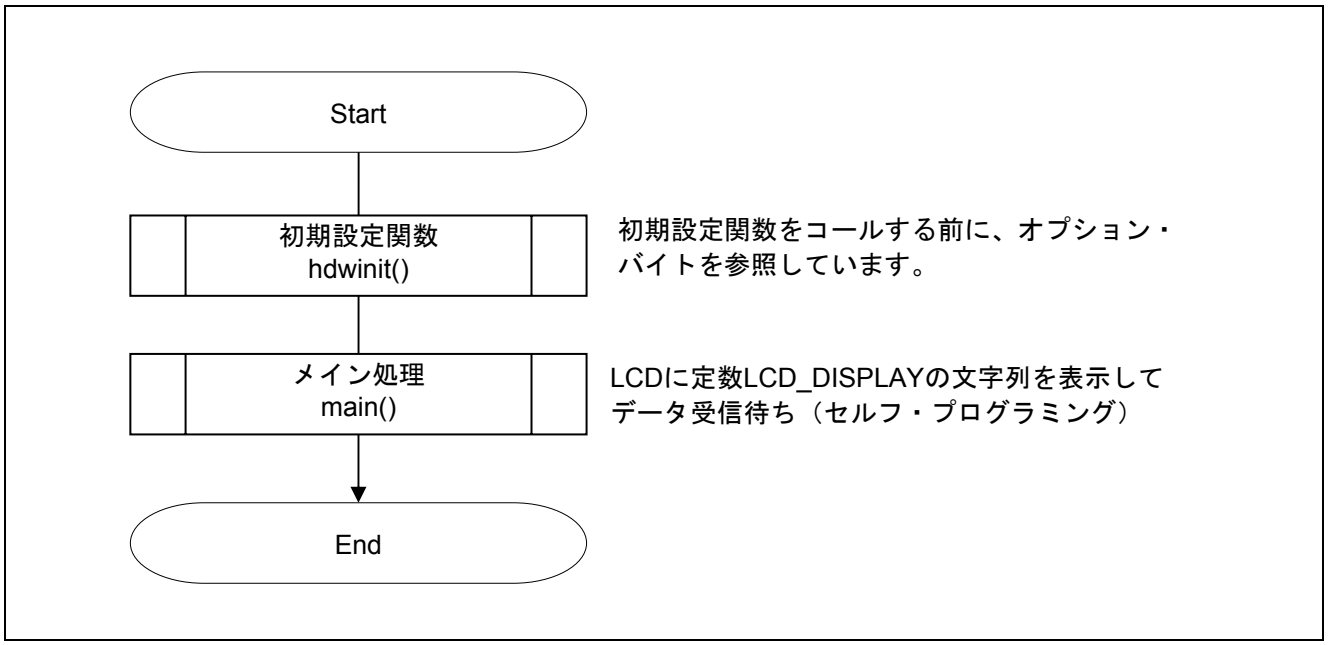


図 5.1 全体フロー

5.10.1 初期設定関数

図 5.2に**初期設定関数**のフローチャートを示します。

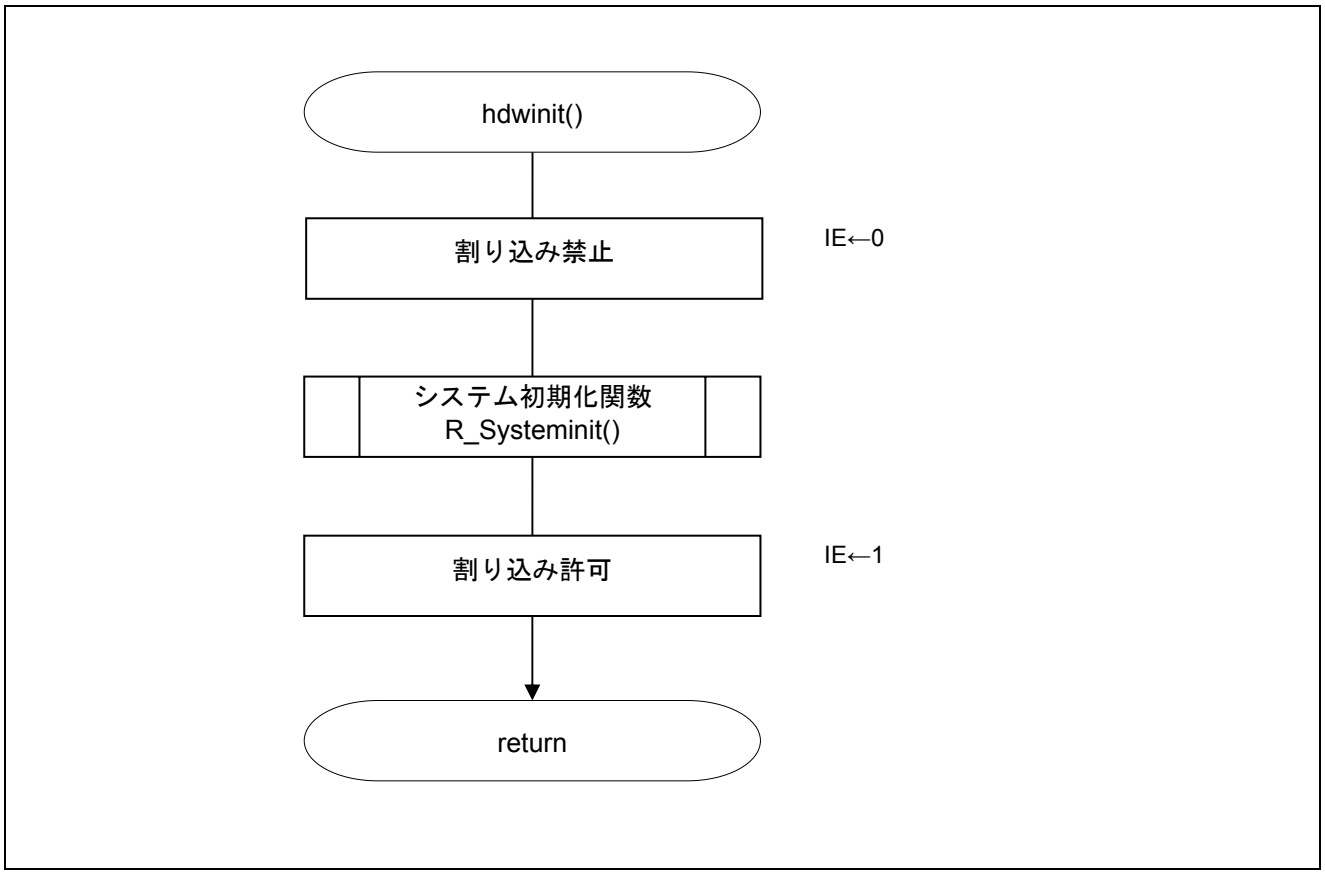


図 5.2 初期設定関数

5.10.2 システム初期化関数

図 5.3にシステム初期化関数のフローチャートを示します。

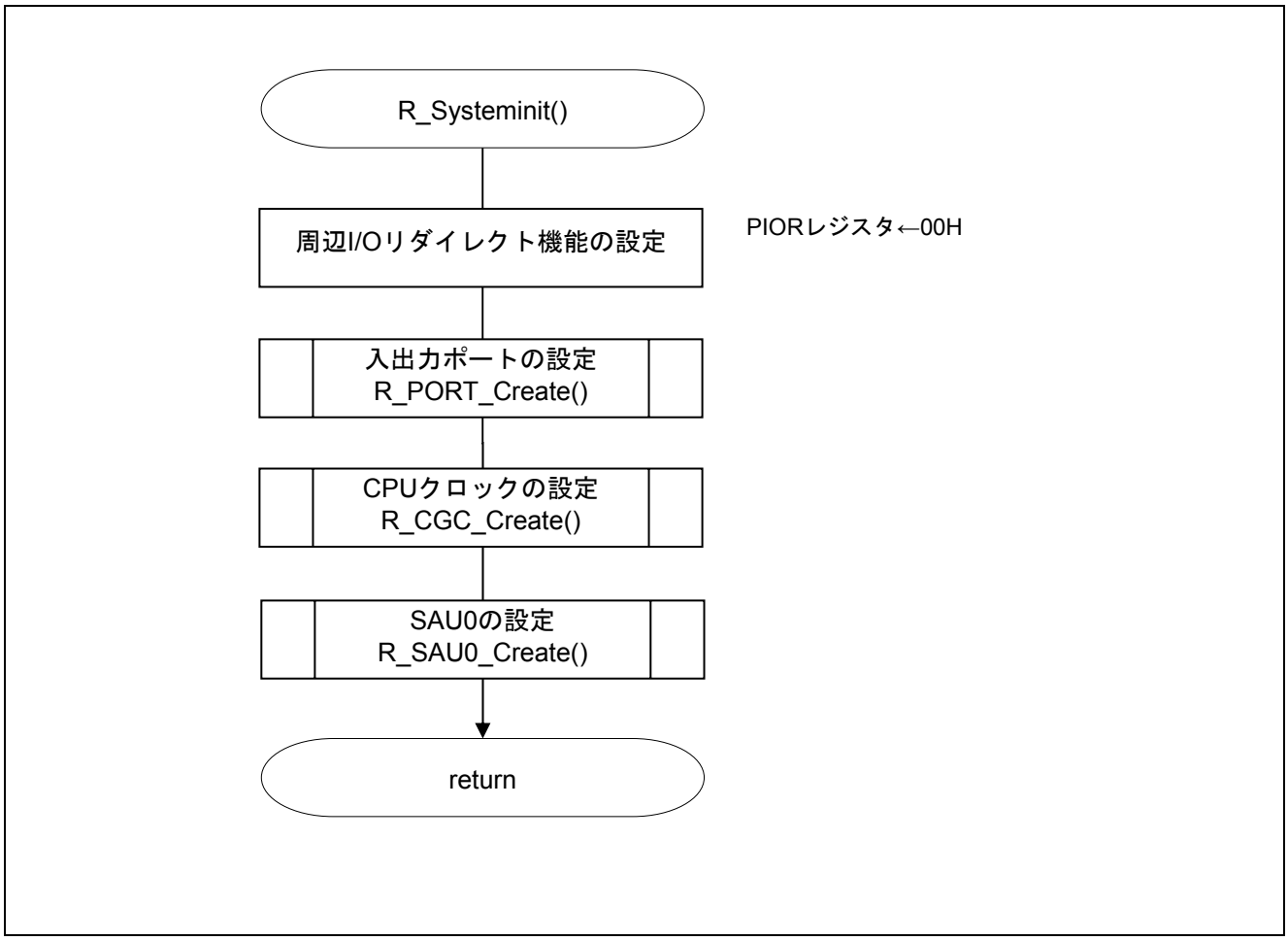


図 5.3 システム初期化関数

5.10.3 入出力ポートの設定

図 5.4に入出力ポートの設定のフローチャートを示します。

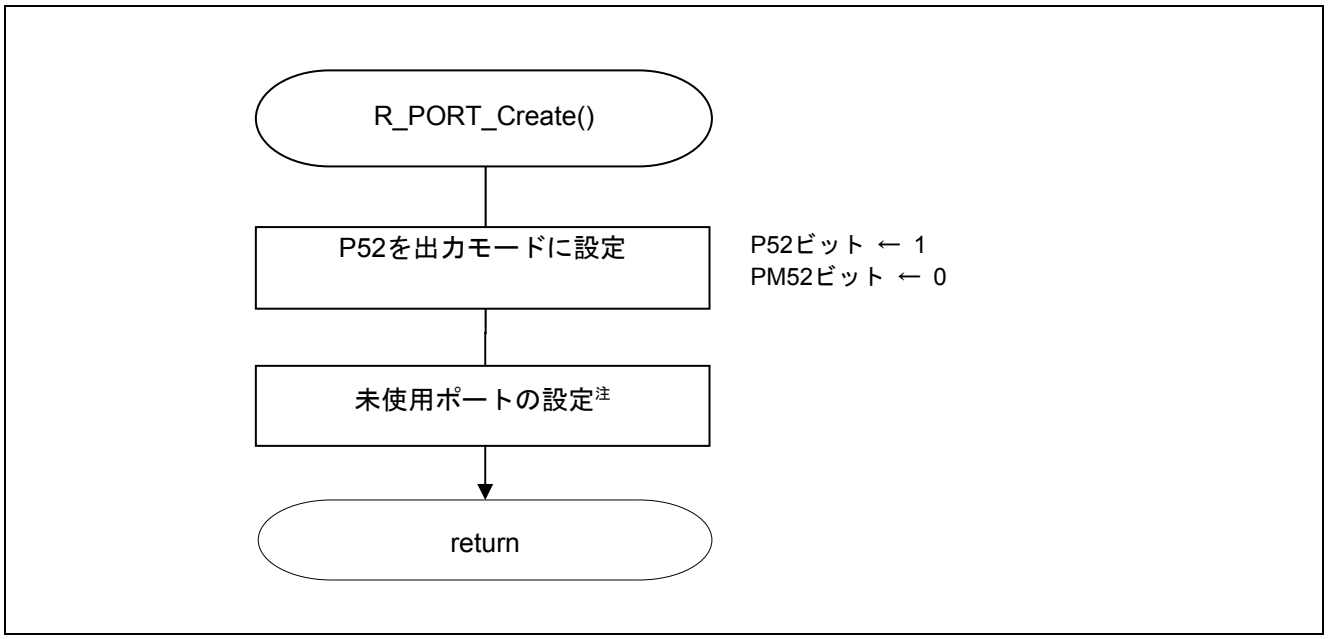


図 5.4 入出力ポートの設定

注 未使用ポートの設定については、RL78/G13 初期設定（R01AN0451J）アプリケーションノート“フローチャート”を参照して下さい。

注意 未使用のポートは、端子処理などを適切に行い、電気的特性を満たすように設計してください。
また、未使用の入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続して下さい。

5.10.4 CPU クロックの設定

図 5.5にCPU クロックの設定のフローチャートを示します。

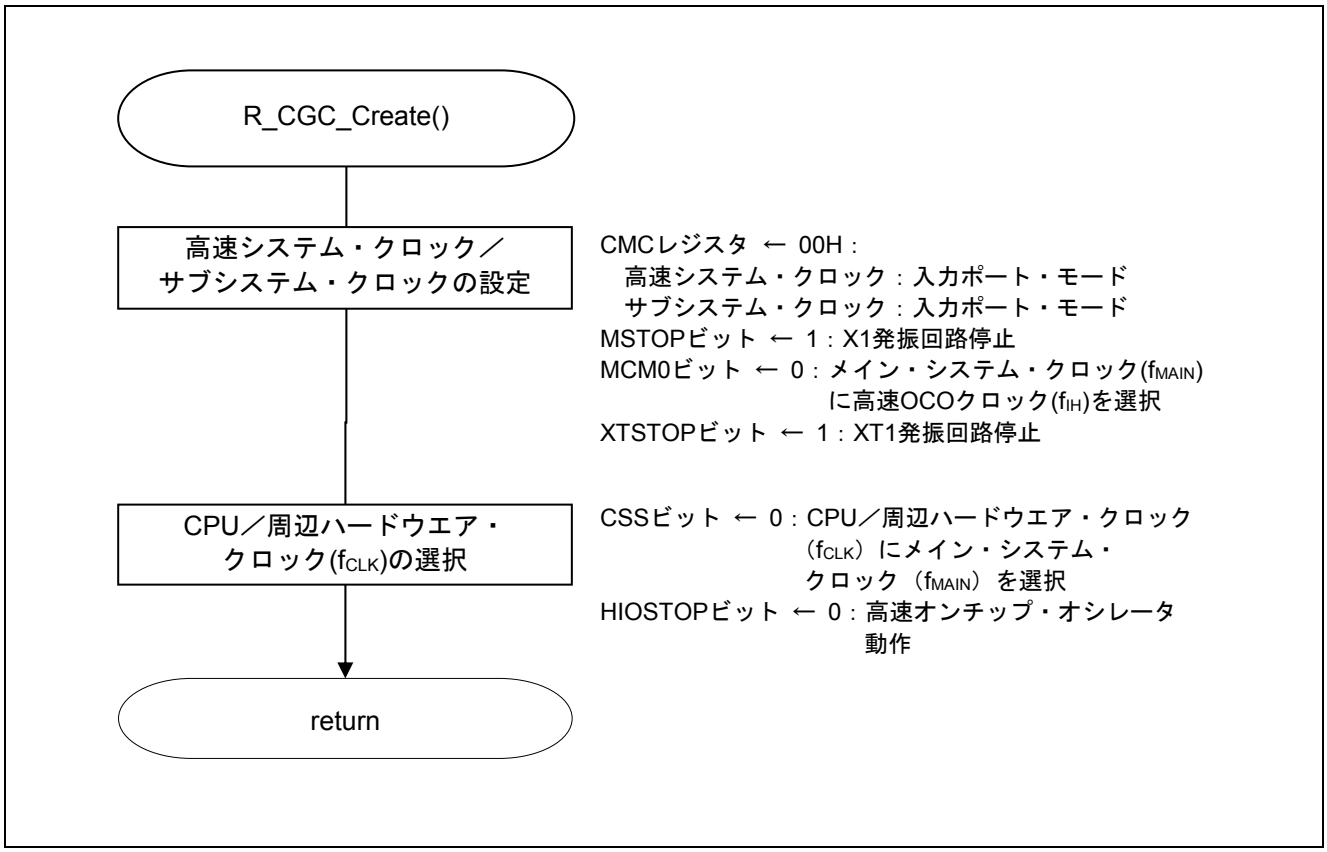


図 5.5 CPU クロックの設定

注意 CPU クロックの設定（R_CGC_Create()）については、RL78/G13 初期設定（R01AN0451J）アプリケーションノート"フローチャート"を参照して下さい。

5.10.5 SAU0 の設定

図 5.6にSAU0 の設定のフローチャートを示します。

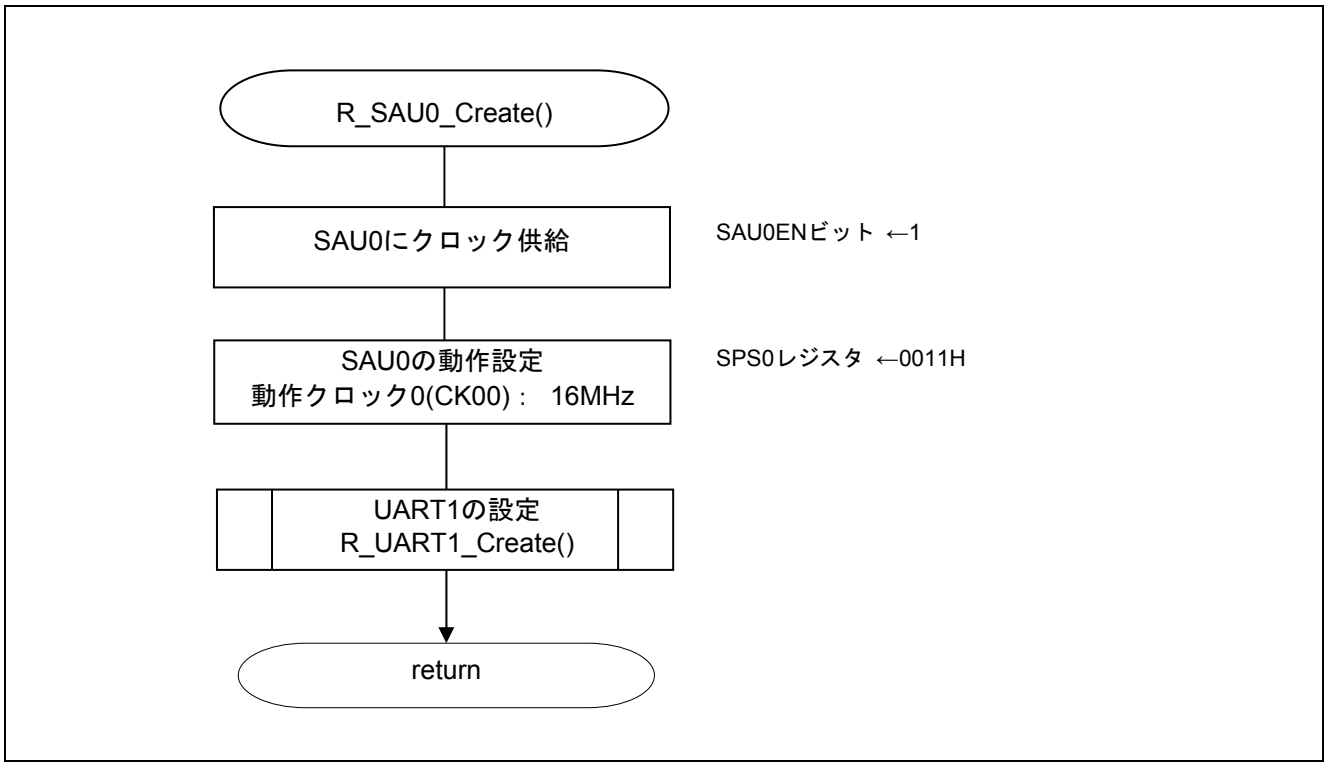


図 5.6 SAU0 の設定

5.10.6 UART1 の設定

図 5.7にUART1 の設定のフローチャートを示します。

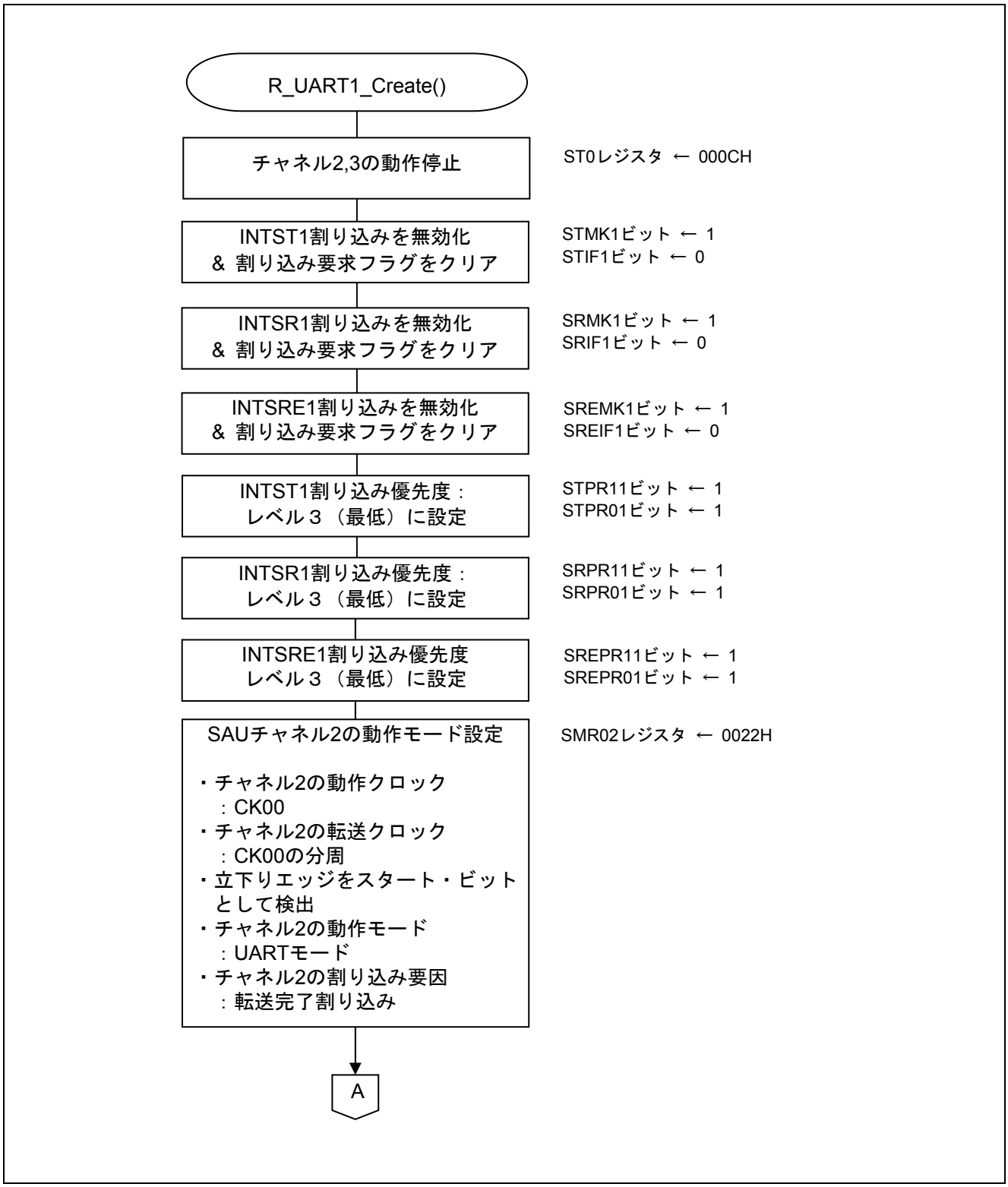


図 5.7 UART1 の設定（1/3）

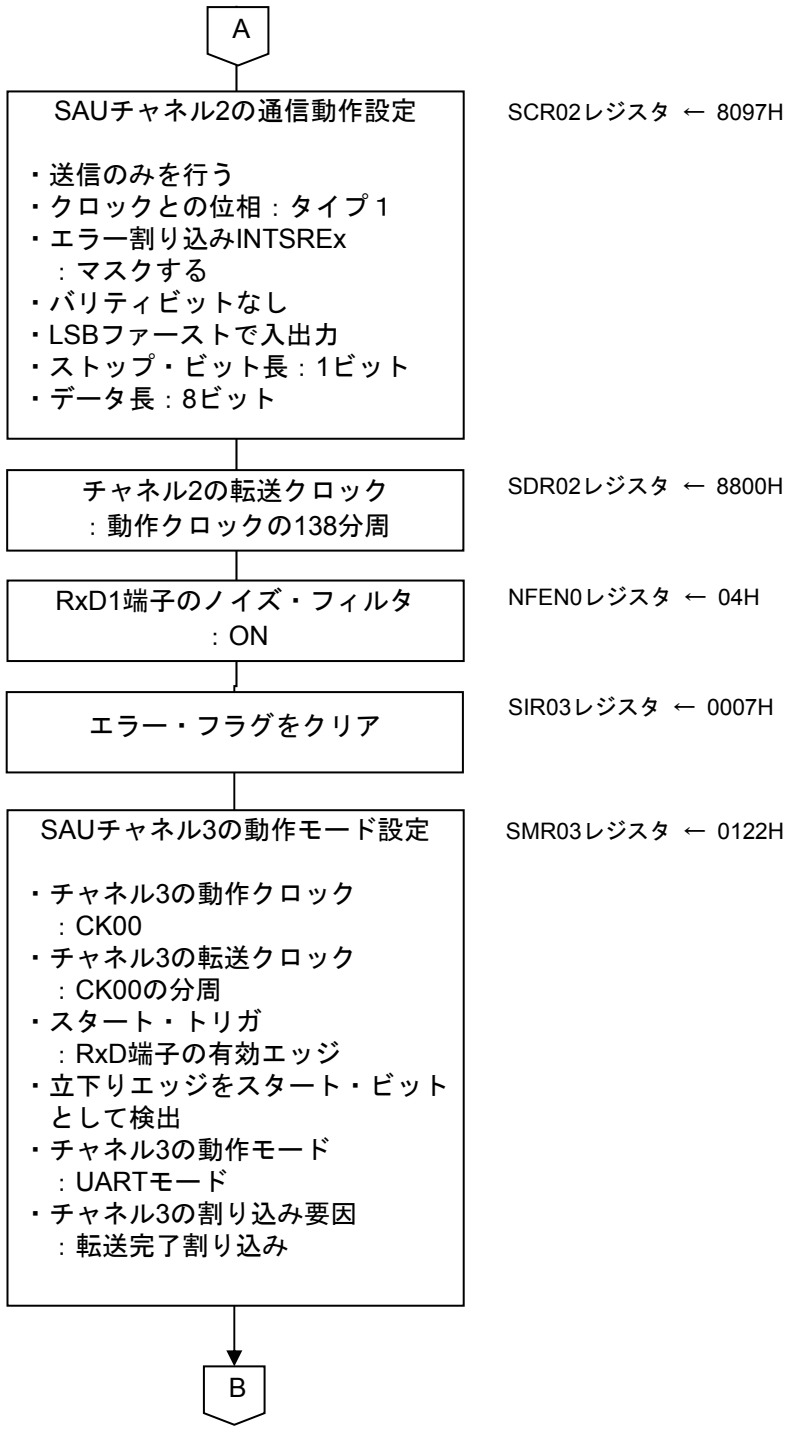


図 5.8 UART1 の設定（2/3）

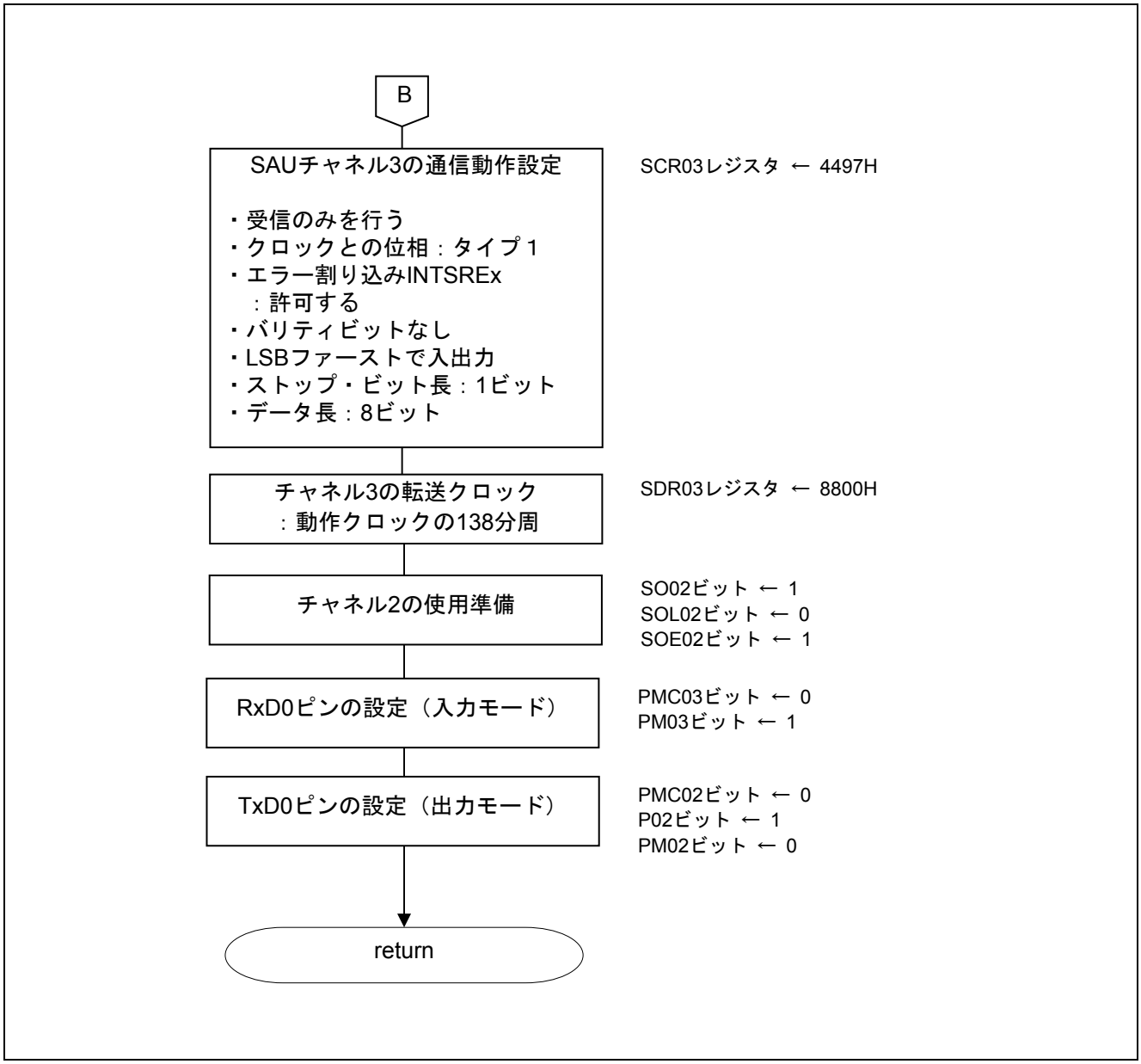


図 5.9 UART1 の設定（3/3）

5.10.7 メイン処理

図 5.10にメイン処理(1/2)、図 5.11にメイン処理(2/2)のフローチャートを示します。

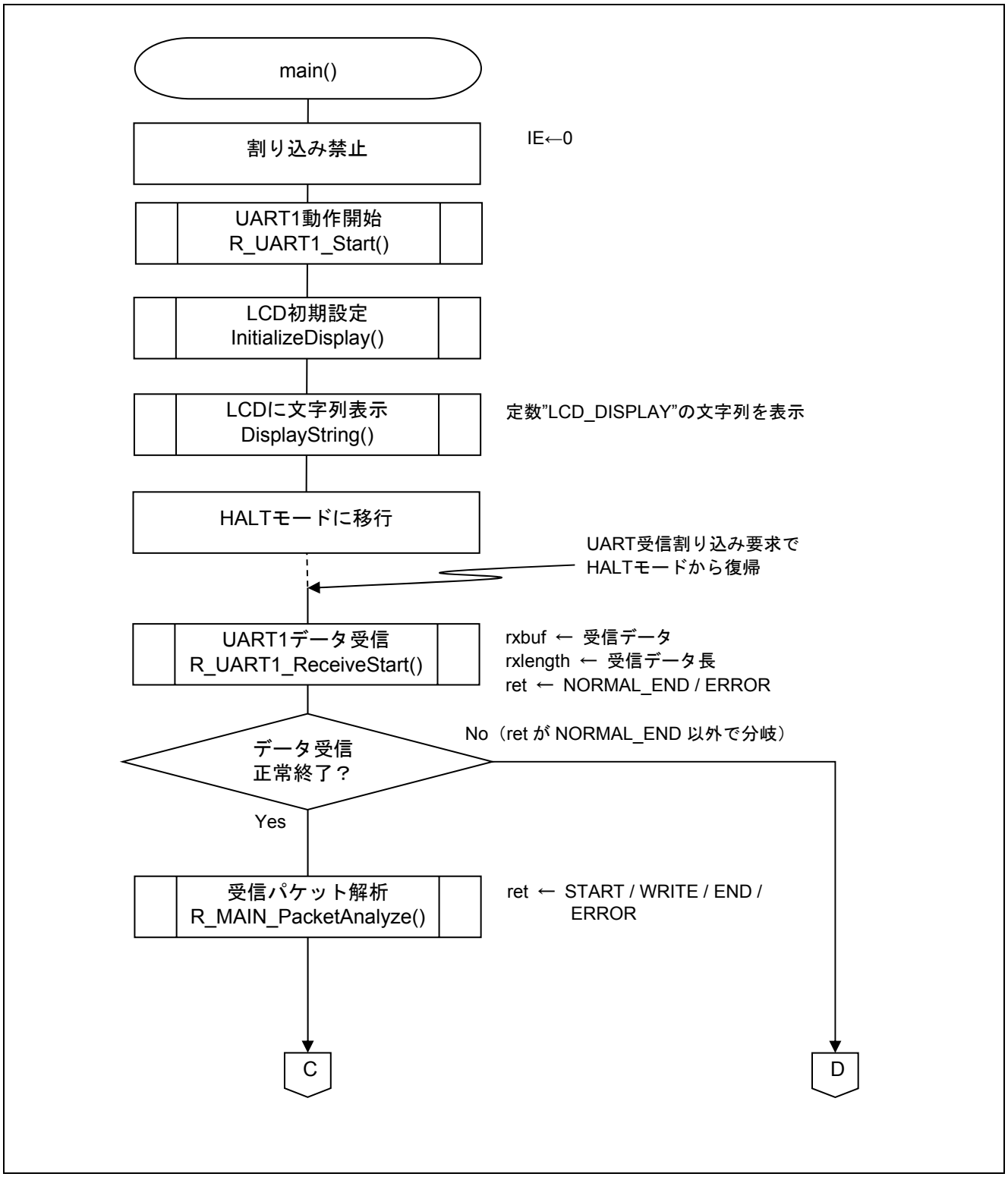


図 5.10 メイン処理(1/2)

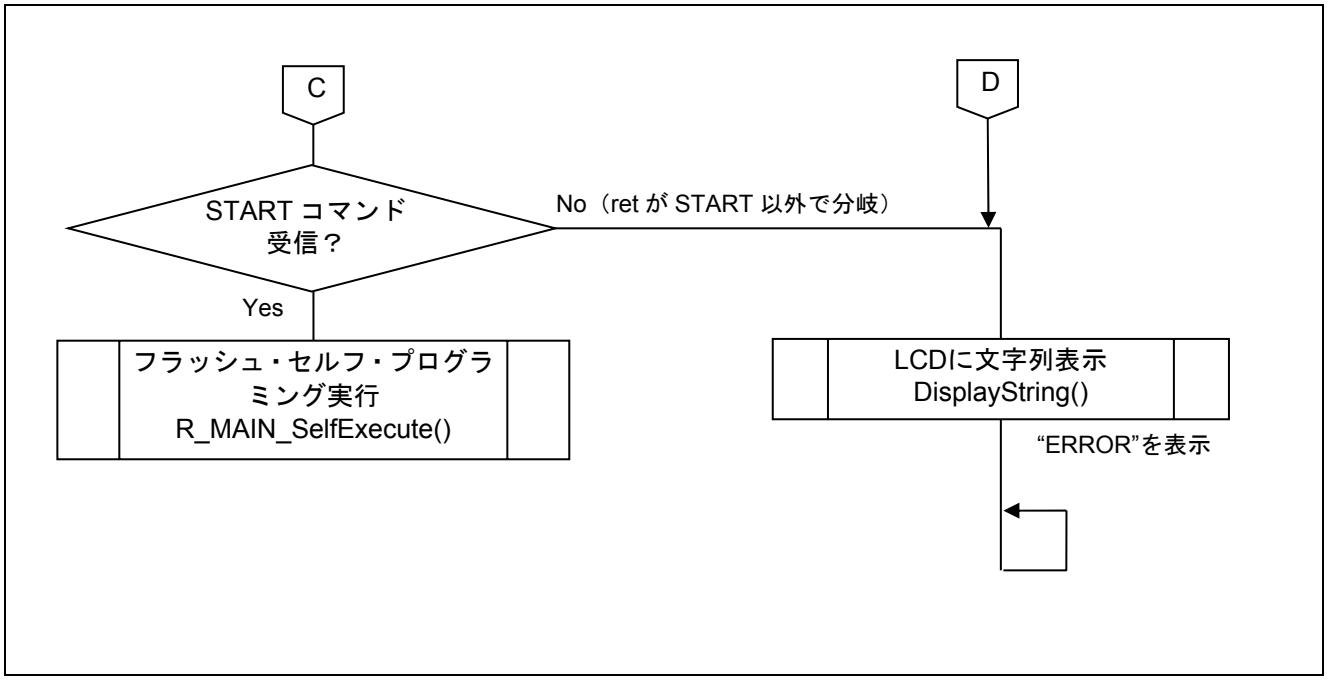


図 5.11 メイン処理(2/2)

5.10.8 UART1 動作開始

図 5.12にUART1 動作開始のフローチャートを示します。

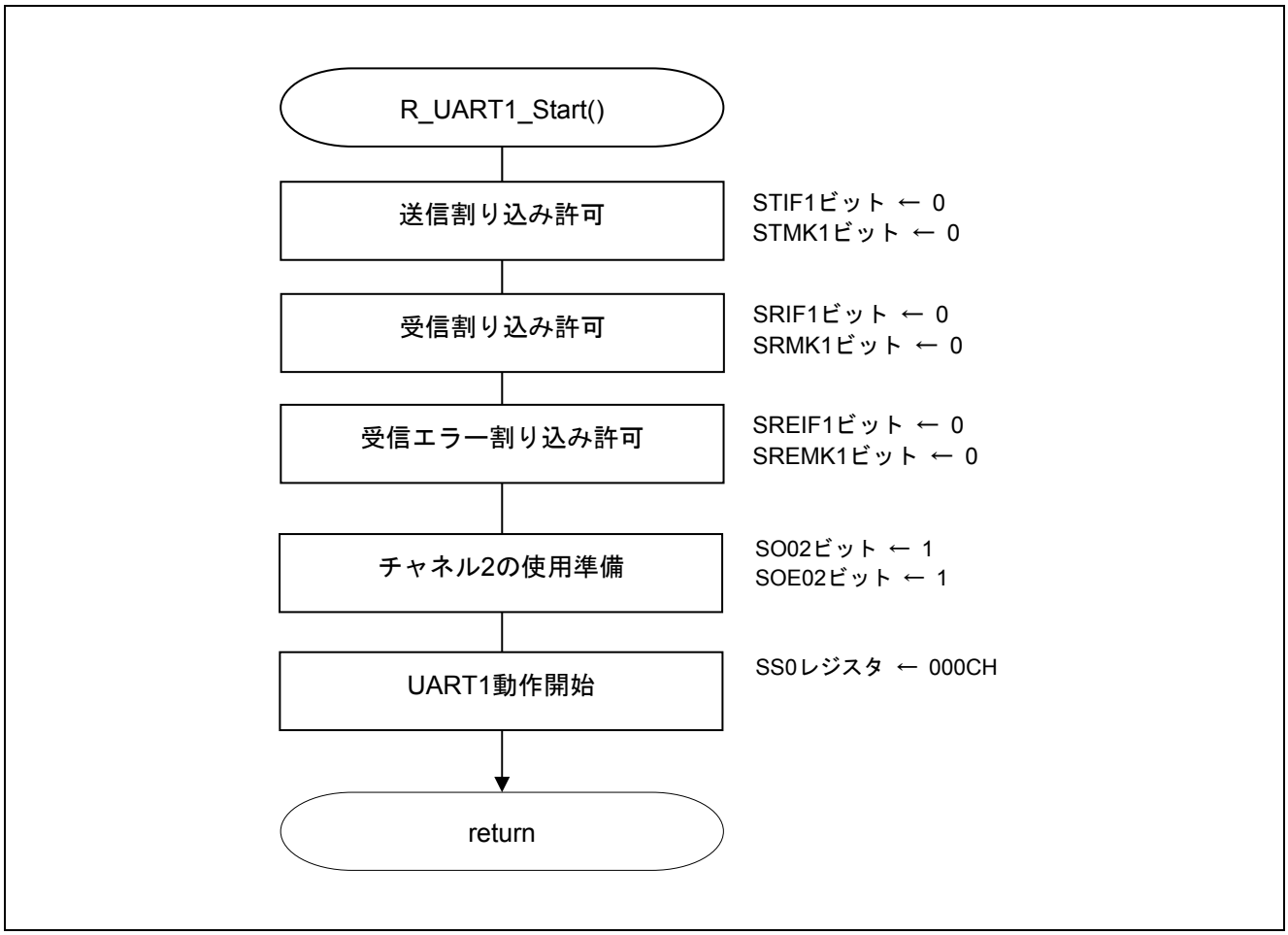


図 5.12 UART1 動作開始

5.10.9 UART1 データ受信

図 5.13にUART1 データ受信(1/2)、図 5.14にUART1 データ受信(2/2)のフローチャートを示します。

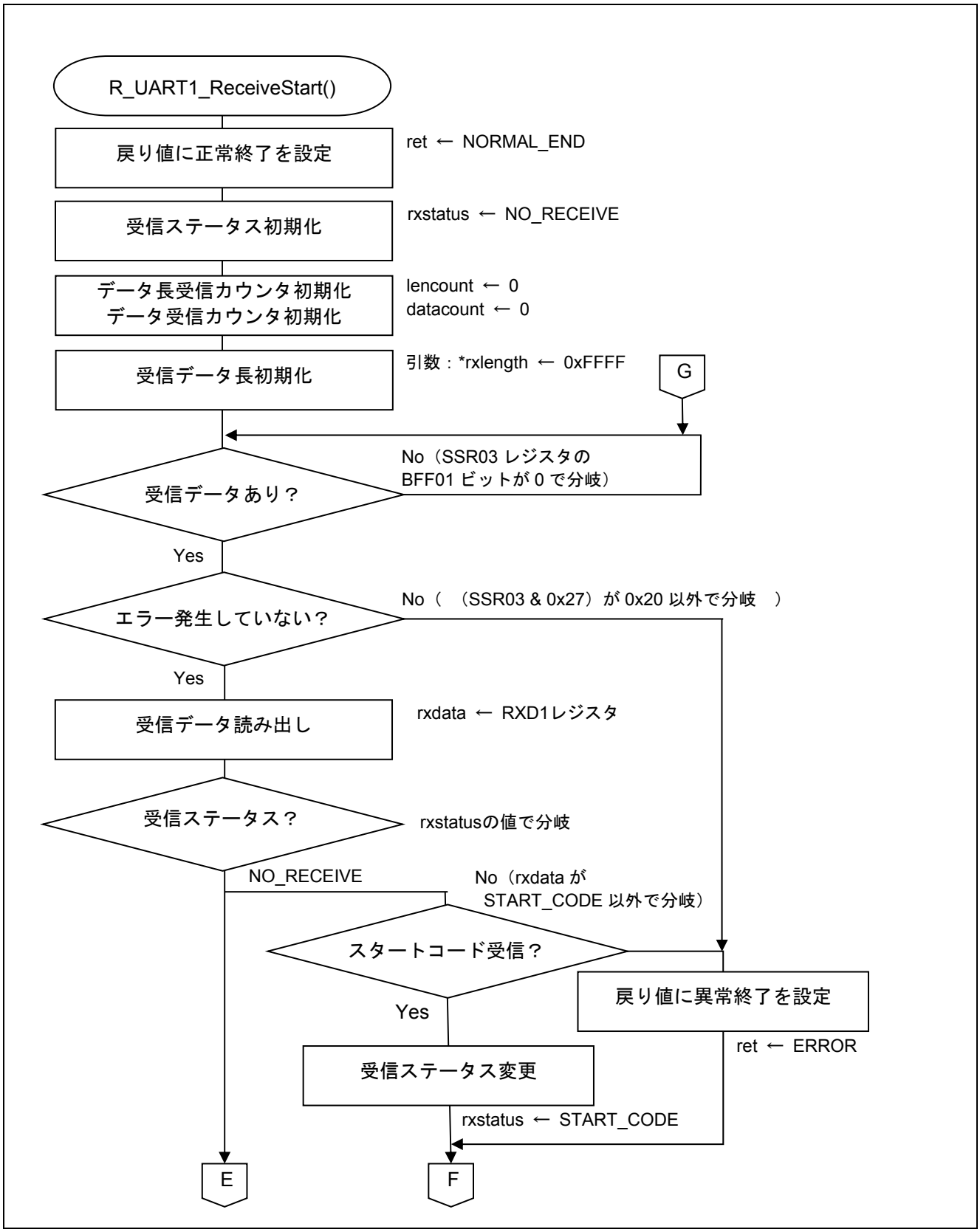


図 5.13 UART1 データ受信(1/2)

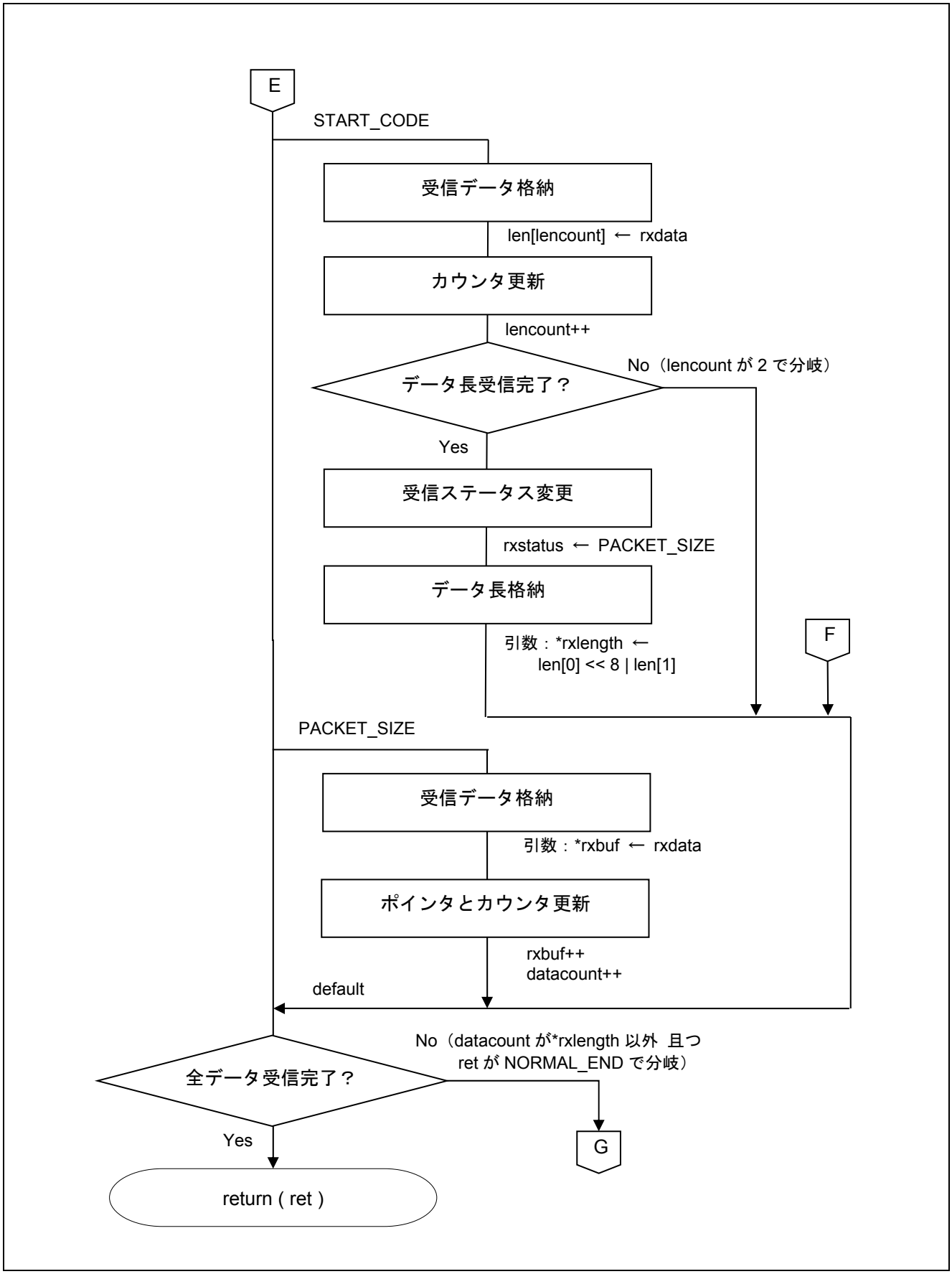


図 5.14 UART1 データ受信(2/2)

5.10.10 受信パケット解析

図 5.15に受信パケット解析のフローチャートを示します。

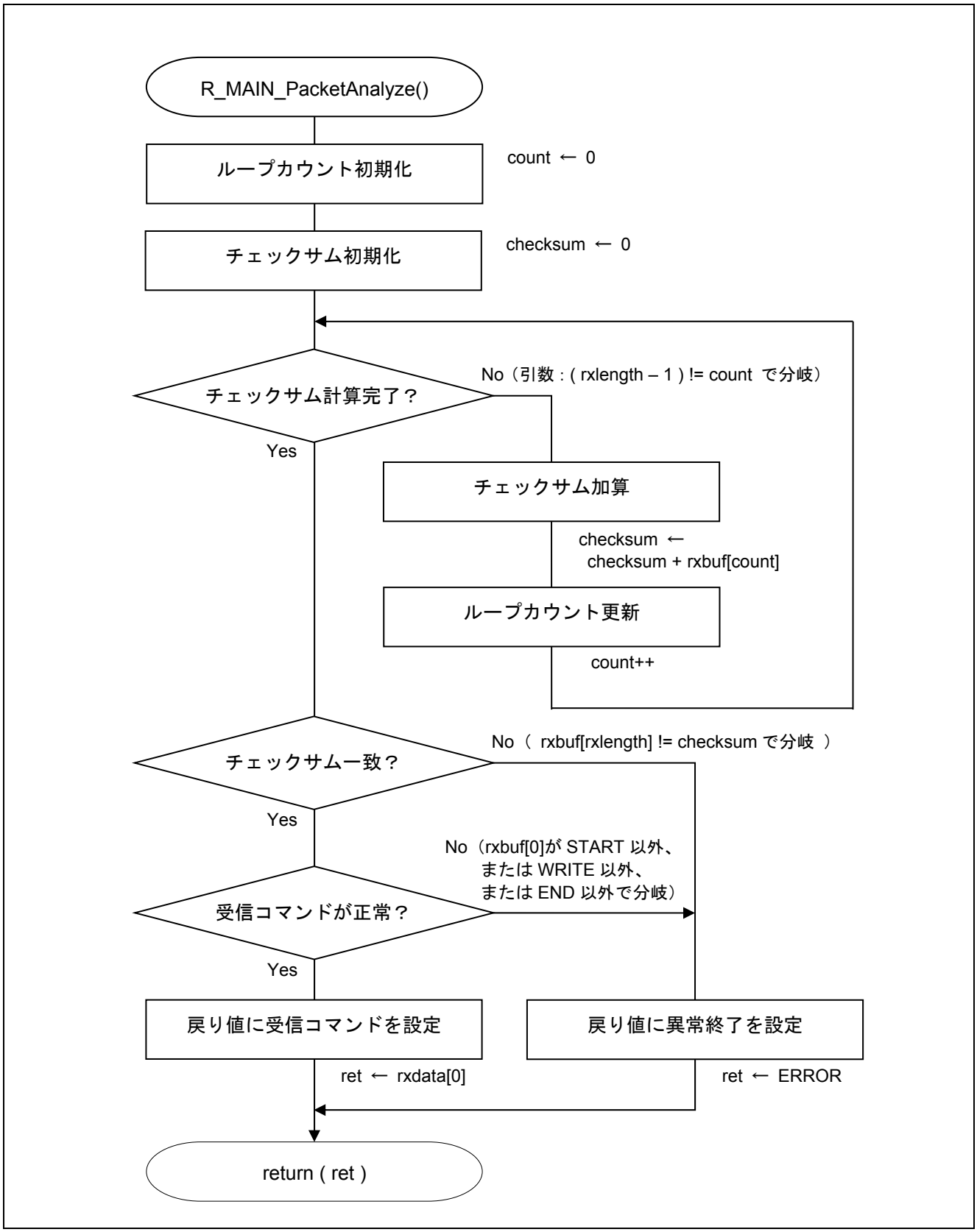


図 5.15 受信パケット解析

5.10.11 フラッシュ・セルフ・プログラミング実行

図 5.16にフラッシュ・セルフ・プログラミング実行のフローチャートを示します。

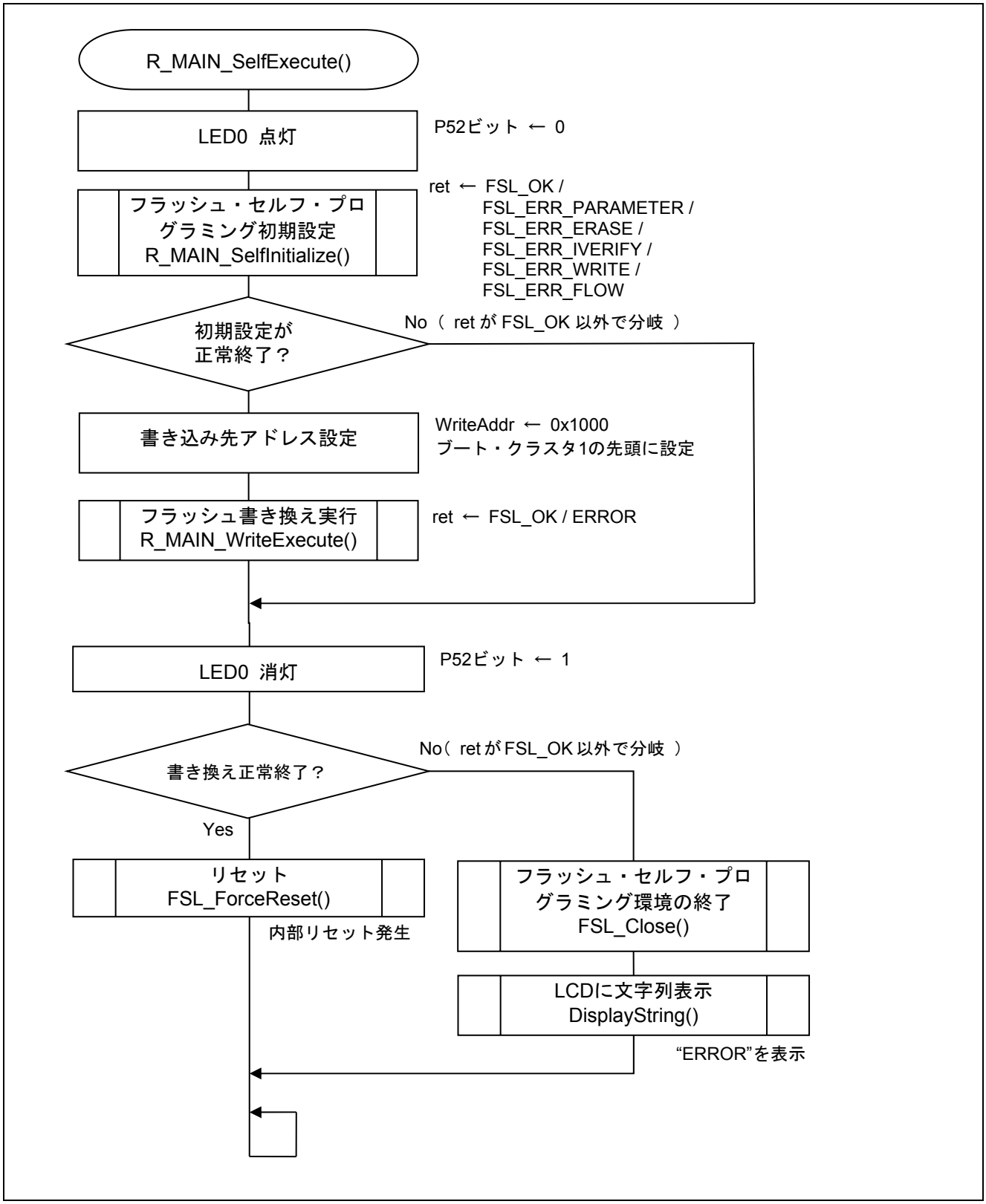


図 5.16 フラッシュ・セルフ・プログラミング実行

5.10.12 フラッシュ・セルフ・プログラミング初期設定

図 5.17にフラッシュ・セルフ・プログラミング初期設定(1/2)、図 5.18にフラッシュ・セルフ・プログラミング初期設定(2/2)のフローチャートを示します。

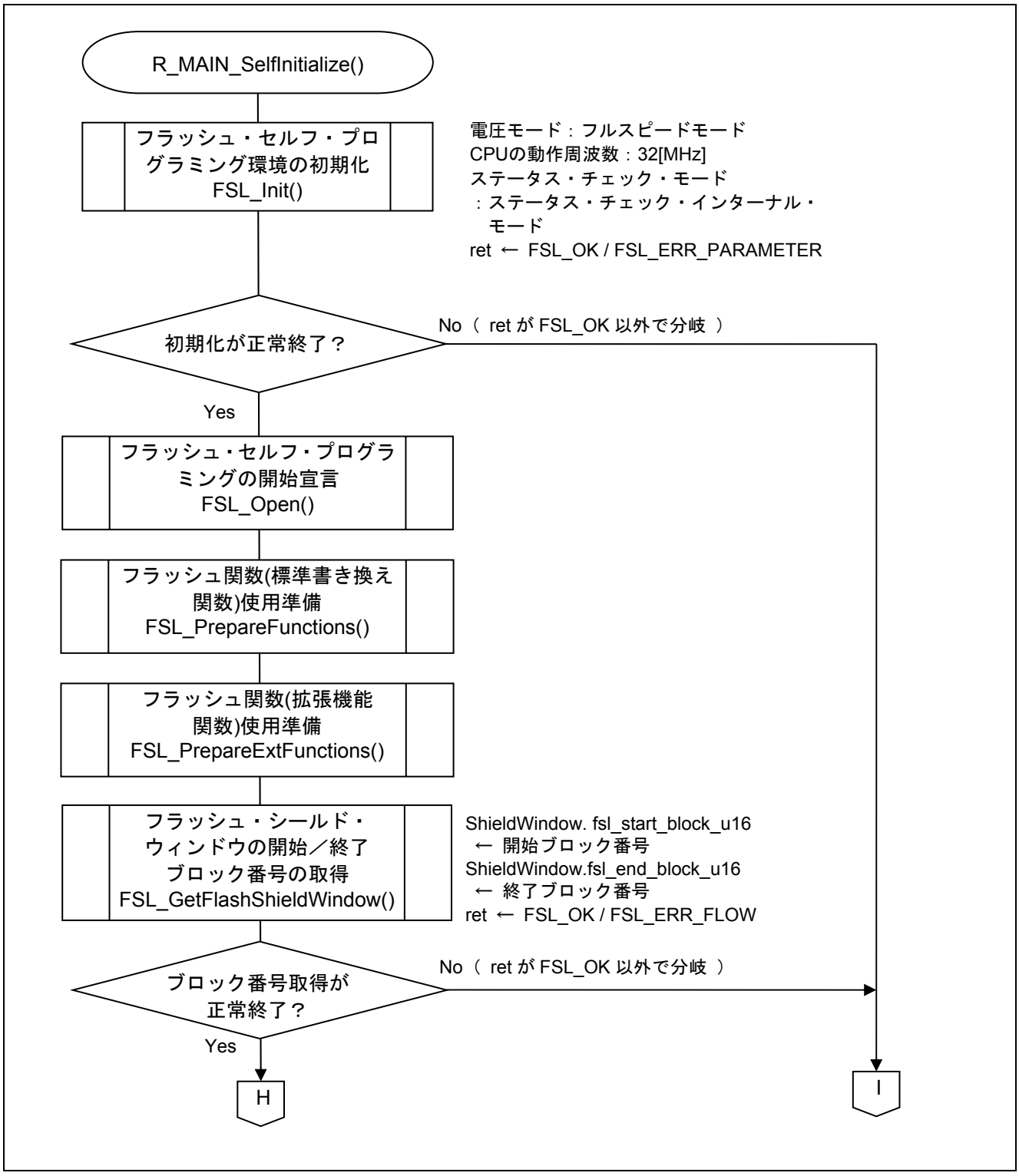


図 5.17 フラッシュ・セルフ・プログラミング初期設定(1/2)

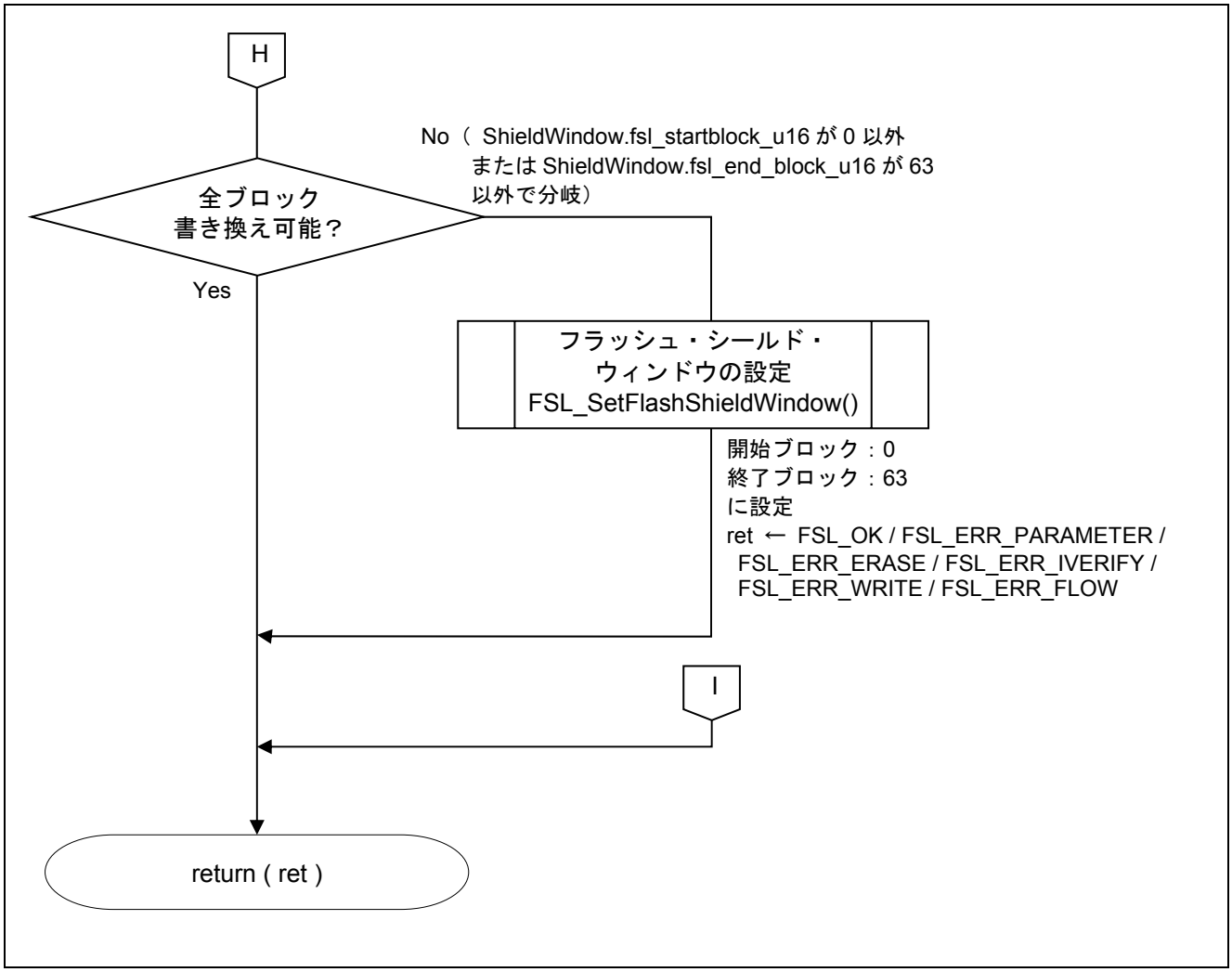


図 5.18 フラッシュ・セルフ・プログラミング初期設定(2/2)

5.10.13 フラッシュ書き換え実行

図 5.19にフラッシュ書き換え実行(1/3)、図 5.20にフラッシュ書き換え実行(2/3)、図 5.21にフラッシュ書き換え実行(3/3)のフローチャートを示します。

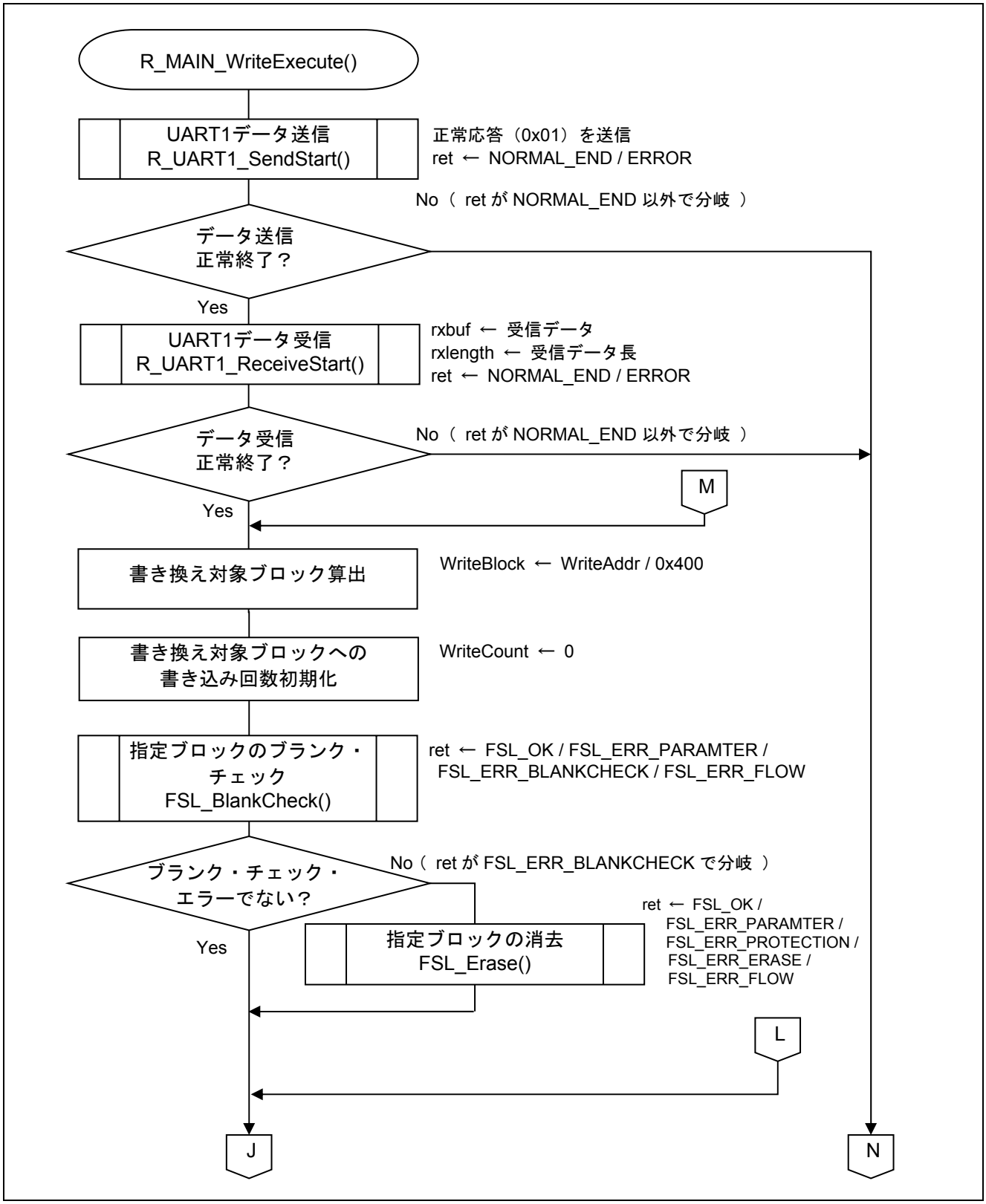


図 5.19 フラッシュ書き換え実行(1/3)

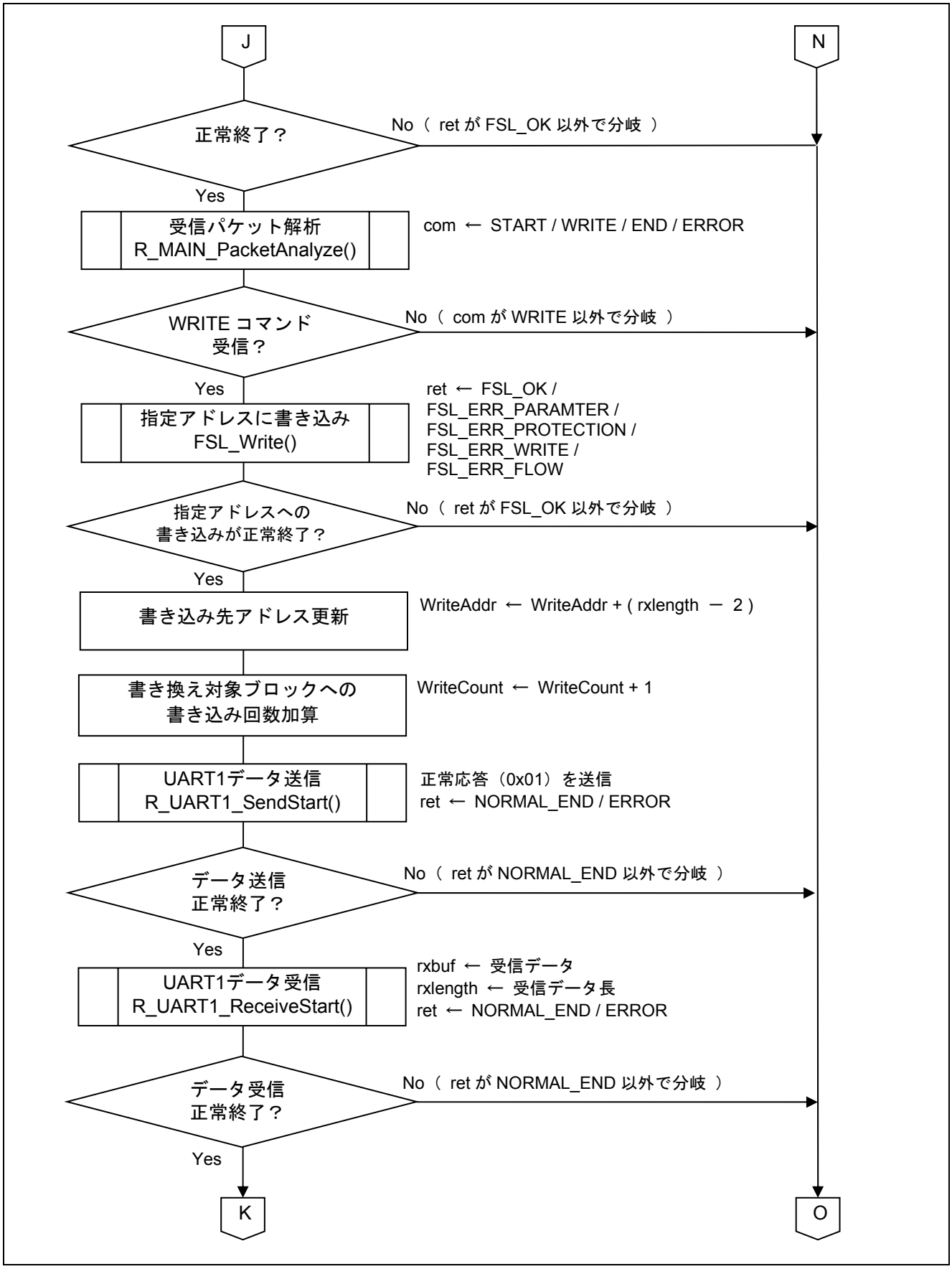


図 5.20 フラッシュ書き換え実行(2/3)

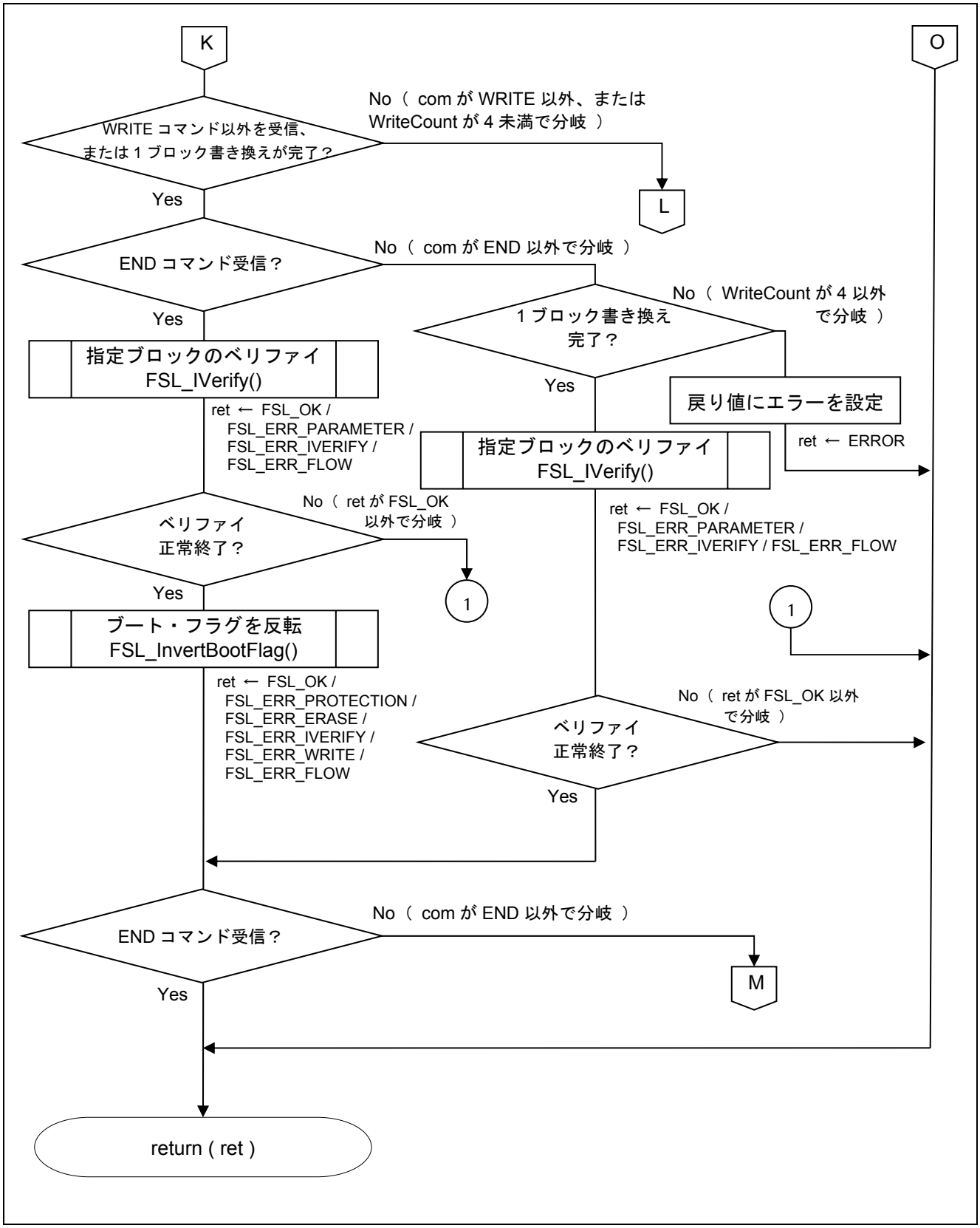


図 5.21 フラッシュ書き換え実行(3/3)

5.10.14 UART1 データ送信

図 5.22にUART1 データ送信のフローチャートを示します。

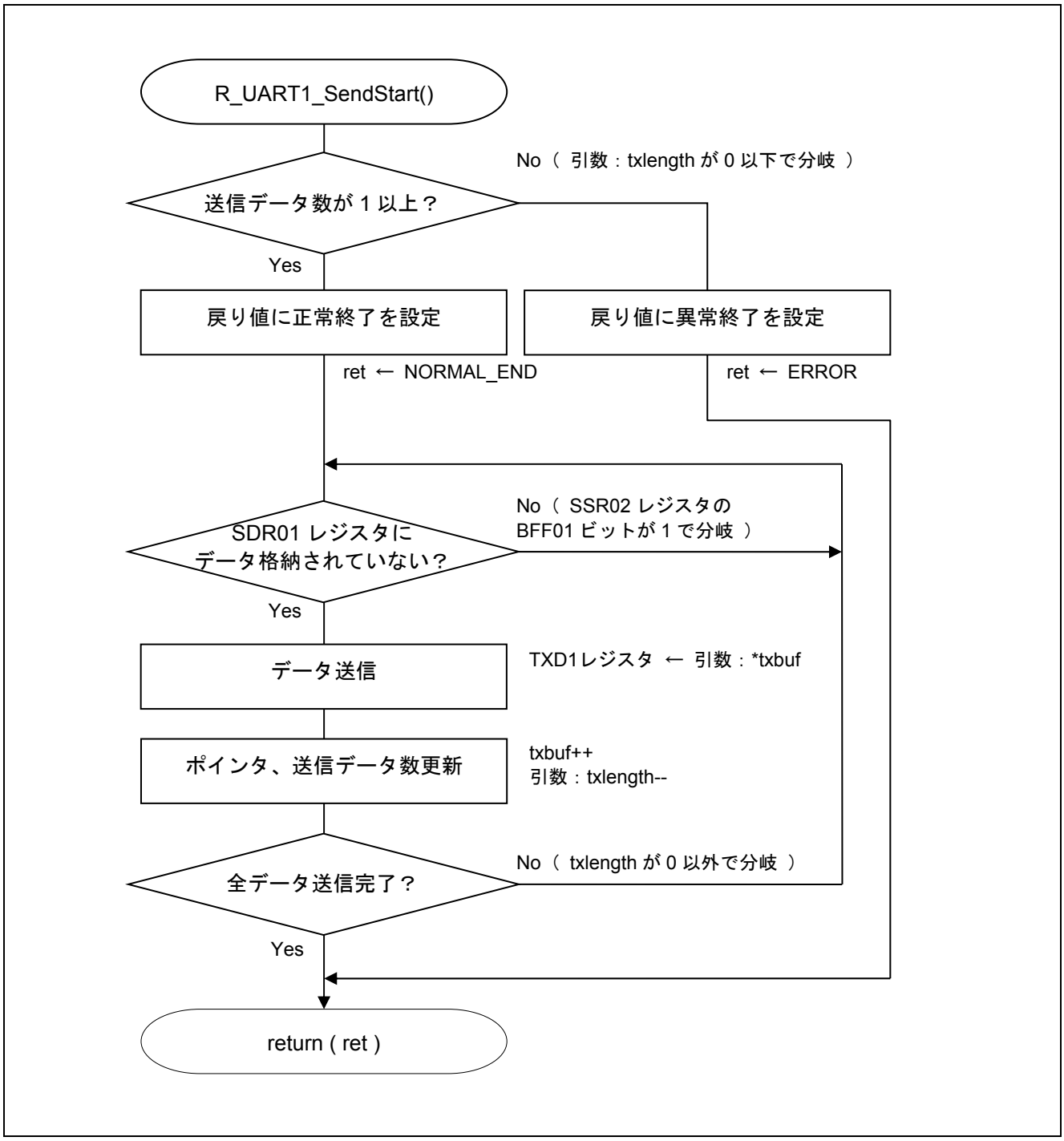


図 5.22 UART1 データ送信

5.11 動作確認方法

サンプル・プログラムの r_cg_userdefine.h で定義されている定数 LCD_DISPLAY の文字列を変更し、プロジェクトをリビルドします。LCD_DISPLAY に設定する文字列は、8 文字以内としてください。作成された HEX ファイルを書き換え用データとして送信側から送信することで、フラッシュ・セルフ・プログラミングが行われます。送信側と本サンプル・プログラムの通信仕様は、「5.1 通信仕様」を参照してください。

例として、定数 LCD_DISPLAY を”Ver 2.0 “に変更した場合の動作は以下のようになります。

(1) LCD に”Ver 1.0”と表示されます。

本サンプル・プログラムでは、定数 LCD_DISPLAY に”Ver 1.0 “を定義しています。

(2) 送信側から START コマンドを送信し、通信動作を開始します。

START コマンド送信以降は、「4.1 通信仕様」の通りに送信側と本サンプル・プログラムで通信を行います。

(3) WRITE コマンドと書き換え用データを受信し、フラッシュ・セルフ・プログラミングを開始すると、RSK ボード上の LED0 が点灯します。

(4) END コマンドを受信すると、LED0 が消灯します。

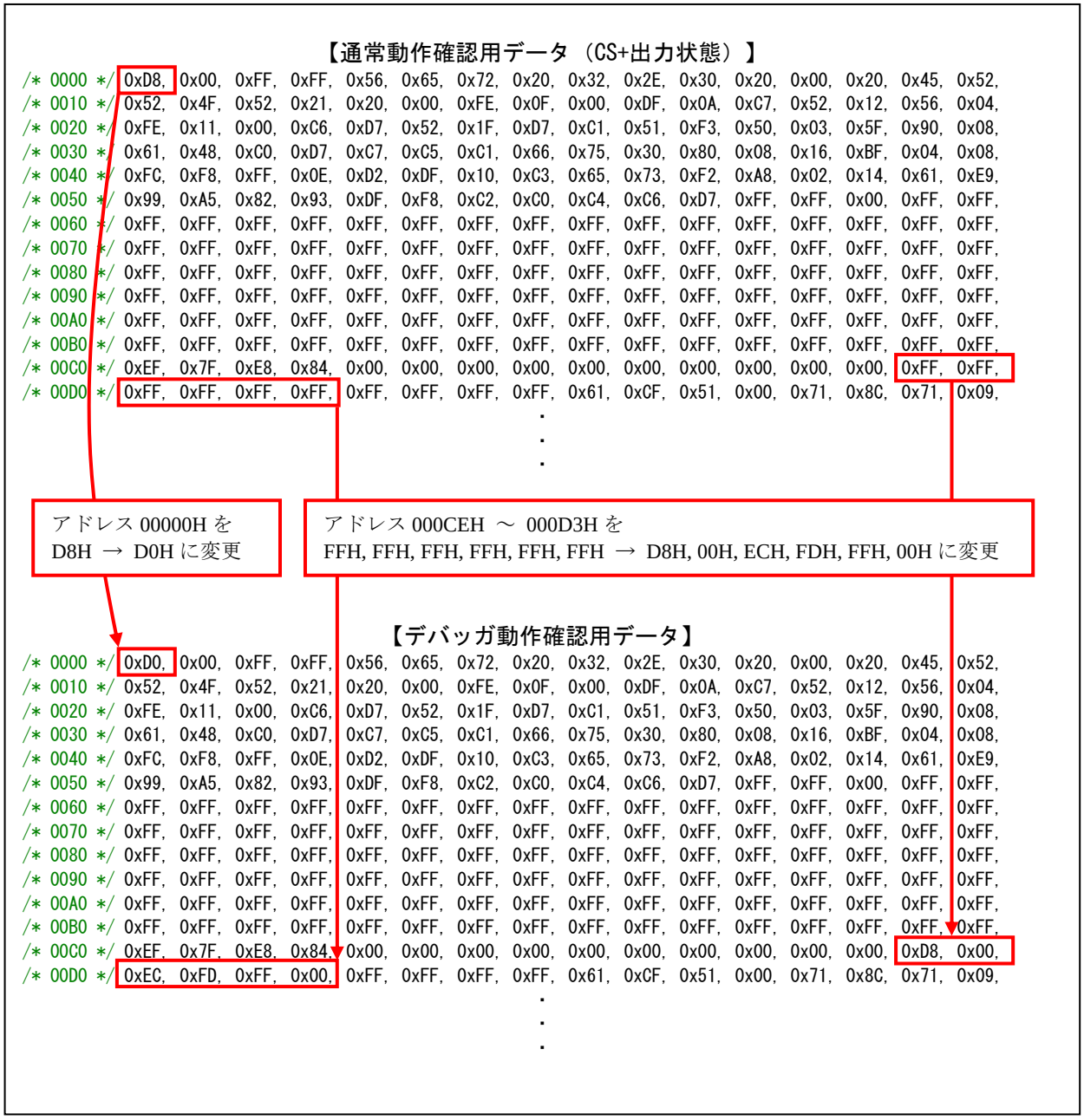
(5) リセットが発生し、LCD に”Ver 2.0 “と表示されます。

5.11.1 デバッガで動作確認を行う場合

デバッガ（E1 エミュレータ）接続状態でフラッシュ・セルフ・プログラミングを実行した場合、書き換え後はデバッガでプログラムの実行を正常に確認できなくなります。書き換え後もデバッガでプログラムの実行を確認する場合には、書き換え用データとして用いる HEX ファイルを、CS+から出力された状態から変更を加える必要があります。

具体的には、以下のようにリセット・ベクタ（アドレス 0x00000）をモニタ・プログラムの配置アドレスに書き換え、モニタ・プログラム（アドレス 0x000CE ～ 0x0000D3）部分も変更を加える必要があります。

アドレス	CS+出力状態	変更内容
0x00000 （リセット・ベクタ）	0xD8	0xD0
0x000CE	0xFF	0xD8
0x000CF	0xFF	0x00
0x000D0	0xFF	0xEC
0x000D1	0xFF	0xFD
0x000D2	0xFF	0xFF
0x000D3	0xFF	0x00



また、本サンプル・プログラムでは、ブート・クラスタ 1 の書き換えを行った後にブート・フラグを切り替えてリセットを発生させ、ブート・スワップを行います。リセットの発生はフラッシュ・セルフ・プログラミング・ライブラリの FSL_ForceReset 関数を使用しますが、デバッガ（E1 エミュレータ）接続時にこの関数を実行した場合には Break が発生して処理が停止します。Break 発生以降は、デバッガから手動でリセットを実行後に再度プログラムを実行する必要があります。

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

RL78/G13 ユーザーズマニュアル ハードウエア編（R01UH0146J）

RL78 ファミリ ユーザーズマニュアル ソフトウエア編（R01US0015J）

RL78 ファミリ フラッシュ・セルフ・プログラミング・ライブラリ Type01 ユーザーズマニュアル（R01US0050J）

（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。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/inquiry

改訂記録	RL78/G13 セルフ・プログラミング（UART）

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.03.01	—	初版発行
1.10	2016.06.01	10	1.4 フラッシュ・セルフ・プログラミング・ライブラリ取得方法を修正
		49	参考ドキュメントを追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

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

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

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

ルネサス エレクトロニクス株式会社　〒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：http://japan.renesas.com/contact/