RL78/G23

UART 通信とブート・スワップを使用したファームウェアアップデート

要旨

本アプリケーションノートでは、更新用プログラムが常にコード・フラッシュ・メモリ内に存在する状態を維持してファームウェアアップデートを行う方法を説明します。

コード・フラッシュ・メモリを 2 分割し、Execute エリアと Temporary エリアとして使用します。

Renesas Flash Driver RL78 Type01 を使用し、フラッシュ・メモリの書き換えとブート・スワップを行います。

動作確認デバイス

RL78/G23

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

1.1 仕様概要

本アプリケーションノートのサンプル・プログラムは、コード・フラッシュ・メモリを対象としたファームウェアアップデートを行います。

ブート領域に対してはブート・スワップ機能を使用した書き換えを行い、ブート領域以外に対しては、書き換えデータを一時保存するテンポラリ領域を設けて書き換えることにより、ユーザ・プログラム (アプリケーション) の実行と並行してファームウェアアップデートを行うことができます。

ファームウェアアッデートは、UART 通信によるコマンドで実行します。4 種類のコマンド (START, WRITE_BOOT1, WRITE_TEMP, END) を使用します。

また、アプリケーションやコマンド実行の状態は LED で表示します。

本アプリケーションノートには、サンプルプロジェクトを 2 種類同梱しており、それぞれファームウェアアップデートによる入れ替えが可能です。また各プロジェクトは ROM サイズが 128KB の製品と 768KB の製品向けに用意しています。128KB, 768KB 以外の製品を使用する場合は、1.2.8 コピーフラグ、4.3 ROM サイズ設定用の定数設定 を参照し、サンプル・プログラムを変更してください。

表 1-1　プロジェクトフォルダ構成

プロジェクトフォルダ				説明
\workspace				
	\CS+ \e2studio \IAR			
		\128KB		128KB 製品用プロジェクト
			\LED1	サンプルプロジェクト 1 (LED1 点滅)
			\LED8	サンプルプロジェクト 2 (LED8 点滅)
		\768KB		768KB 製品用プロジェクト
			\LED1	サンプルプロジェクト 1 (LED1 点滅)
			\LED8	サンプルプロジェクト 2 (LED8 点滅)

128KB 用プロジェクトと、768KB 用プロジェクトとでは、LED 出力ポートの設定が異なります。本アプリケーションノートでは、128KB 用プロジェクトの使用を例として説明するため、768KB 用プロジェクトをご使用の際は、下記表の通りにポート番号を読み替えてください。

表 1-2　LED 出力ポート番号

LED no.	128KB プロジェクト	768KB プロジェクト
LED1	P03	P33
LED2	P02	P34
LED3	P43	P145
LED4	P42	P106
LED5	P77	P105
LED6	P41	P104
LED7	P31	P103
LED8	P76	P46

表 1-3　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UART0	データ通信
P03, P02, P43, P42, P77, P41, P31, P76	LED1 - LED8への出力端子

表 1-4　アプリケーションの動作状態と LED 表示状態の関係 (サンプルプロジェクト 1 から 2 へのアップデート)

アプリケーションの動作状態	LED1～8 表示状態	実行ファームウェア
書き換え前アプリケーション動作中	LED1 が点滅	サンプルプロジェクト 1
START コマンド受信	LED2 が点灯	
WRITE_BOOT1 コマンド受信	LED3 が点灯	
WRITE_TEMP コマンド受信	LED4 が点灯	
END コマンド受信	LED5 が点灯	
Temporary エリアコピー中	LED6 が点灯	
エラー終了	LED7 のみ点灯	
書き換え後アプリケーション動作中	LED8 が点滅	サンプルプロジェクト 2

1.1.1 Renesas Flash Driver RL78 Type01 概要

Renesas Flash Driver RL78 Type01 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ェアです。

Renesas Flash Driver RL78 Type01 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。

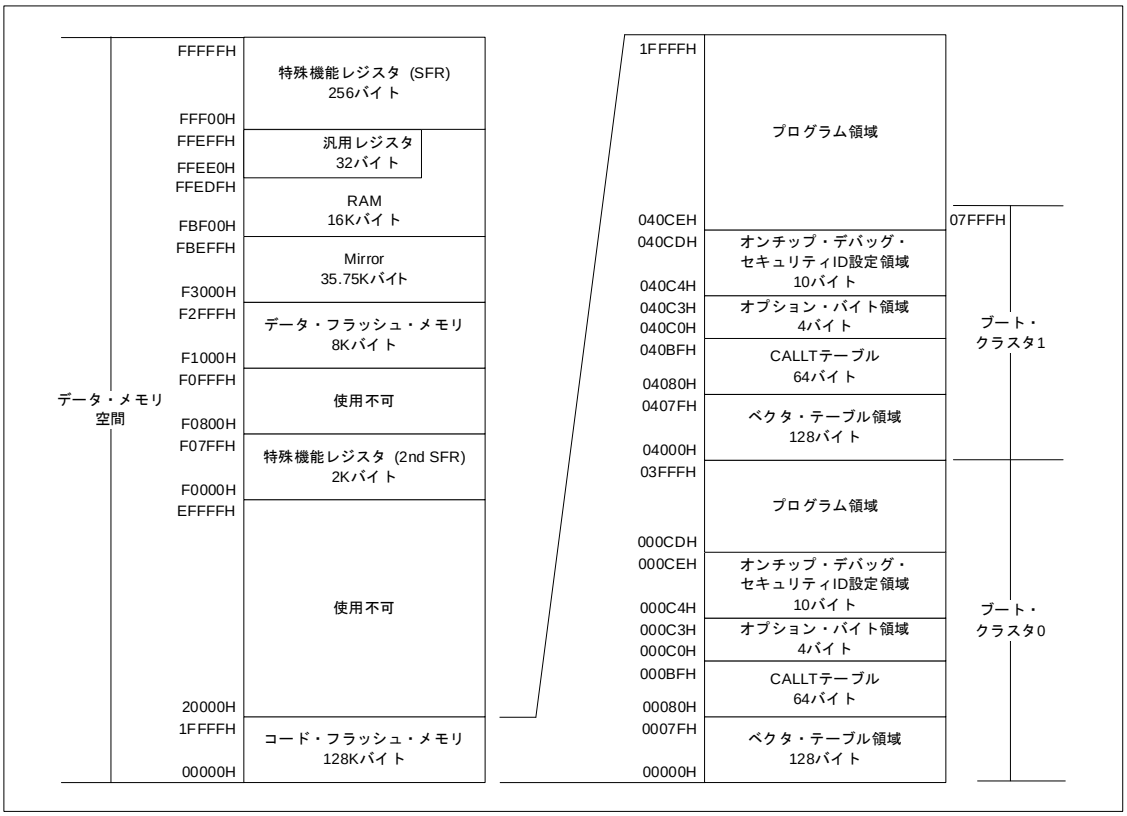
1.1.2　コード・フラッシュ・メモリ

本アプリケーションノートのサンプル・プログラムは、プログラム領域 (08000H ~ 最終アドレス) を 2 分割して使用します。2 つに分割した前半の部分を Execute エリア (08000H ~ 分割部分) 、後半の部分を Temporary エリア (分割部分 ~ 最終アドレス) とします。分割部分のアドレスと、最終アドレスは各 ROM サイズの製品で異なります。ブート・クラスタ 1 と Temporary エリアは更新用のプログラムが書き込まれる部分になるため、ユーザ・プログラムを書き込む場合はブート・クラスタ 0 と Execute エリアに収まるように書き込みを行ってください。

表 1-5　各 ROM サイズの分割部分のアドレス

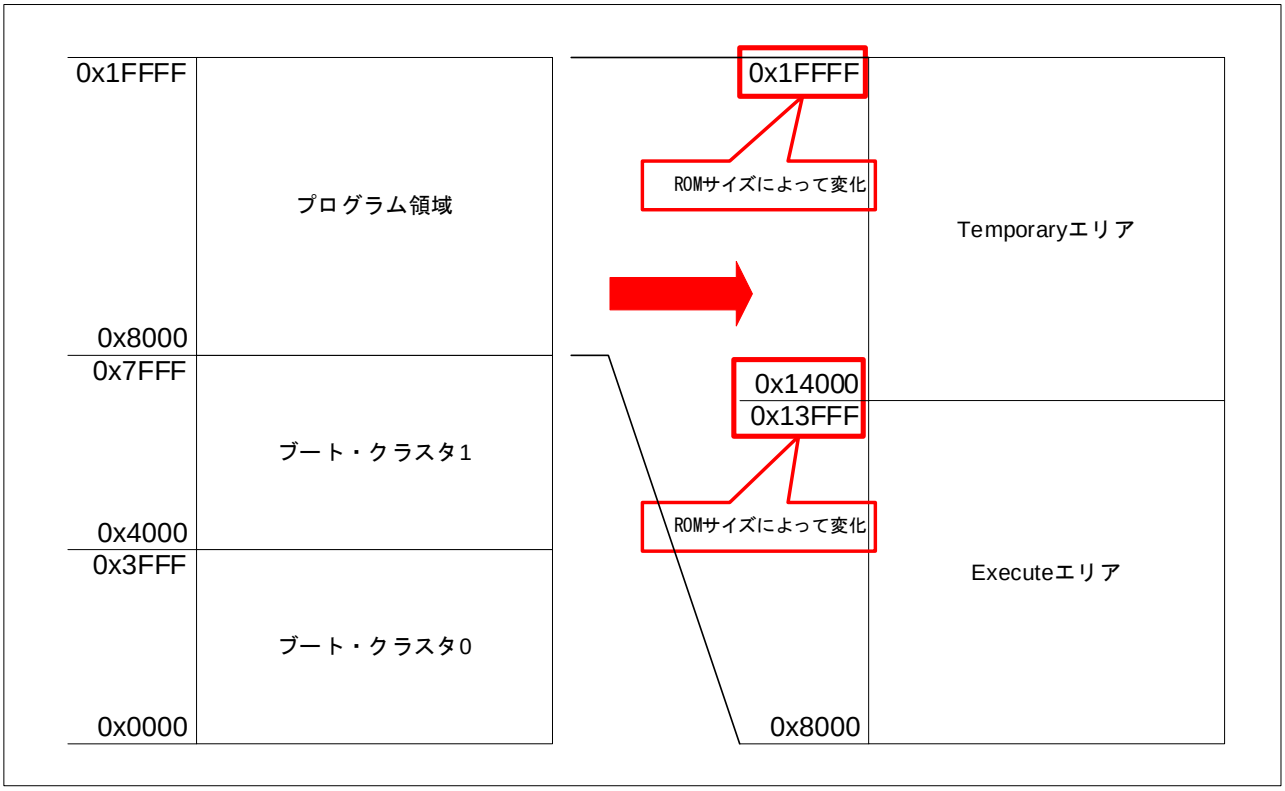
ROM サイズ	Execute エリア	Temporary エリア
96KB	08000H ~ FFFFH	10000H ~ 17FFFH
128KB	08000H ~ 13FFFH	14000H ~ 1FFFFH
192KB	08000H ~ 1BFFFH	1C000H ~ 2FFFFH
256KB	08000H ~ 23FFFH	24000H ~ 3FFFFH
384KB	08000H ~ 33FFFH	34000H ~ 5FFFFH
512KB	08000H ~ 43FFFH	44000H ~ 7FFFFH
768KB	08000H ~ 63FFFH	64000H ~ BFFFFH

図 1-1　メモリ構成



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。

図 1-2　コード・フラッシュ・メモリの構成



RL78/G23 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 1-6　コード・フラッシュ・メモリの特徴

項目	内容
消去の最小単位	1 ブロック (2048バイト)
書き込みの最小単位	1 ワード (4バイト)
ベリファイの最小単位	1 バイト
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
	Renesas Flash Driver RL78 Type01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。

1.1.3 フラッシュ・セルフ・プログラミング

RL78/G23 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 Renesas Flash Driver RL78 Type01 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/G23 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、Renesas Flash Driver RL78 Type01 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、Renesas Flash Driver RL78 Type01 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。

1.1.4 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 Renesas Flash Driver RL78 Type01 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 を切り替えます。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。

1.1.5　ファームウェアアップデート

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とプログラム領域に設定しています。

図 1-3　書き換え動作イメージ (1/2)

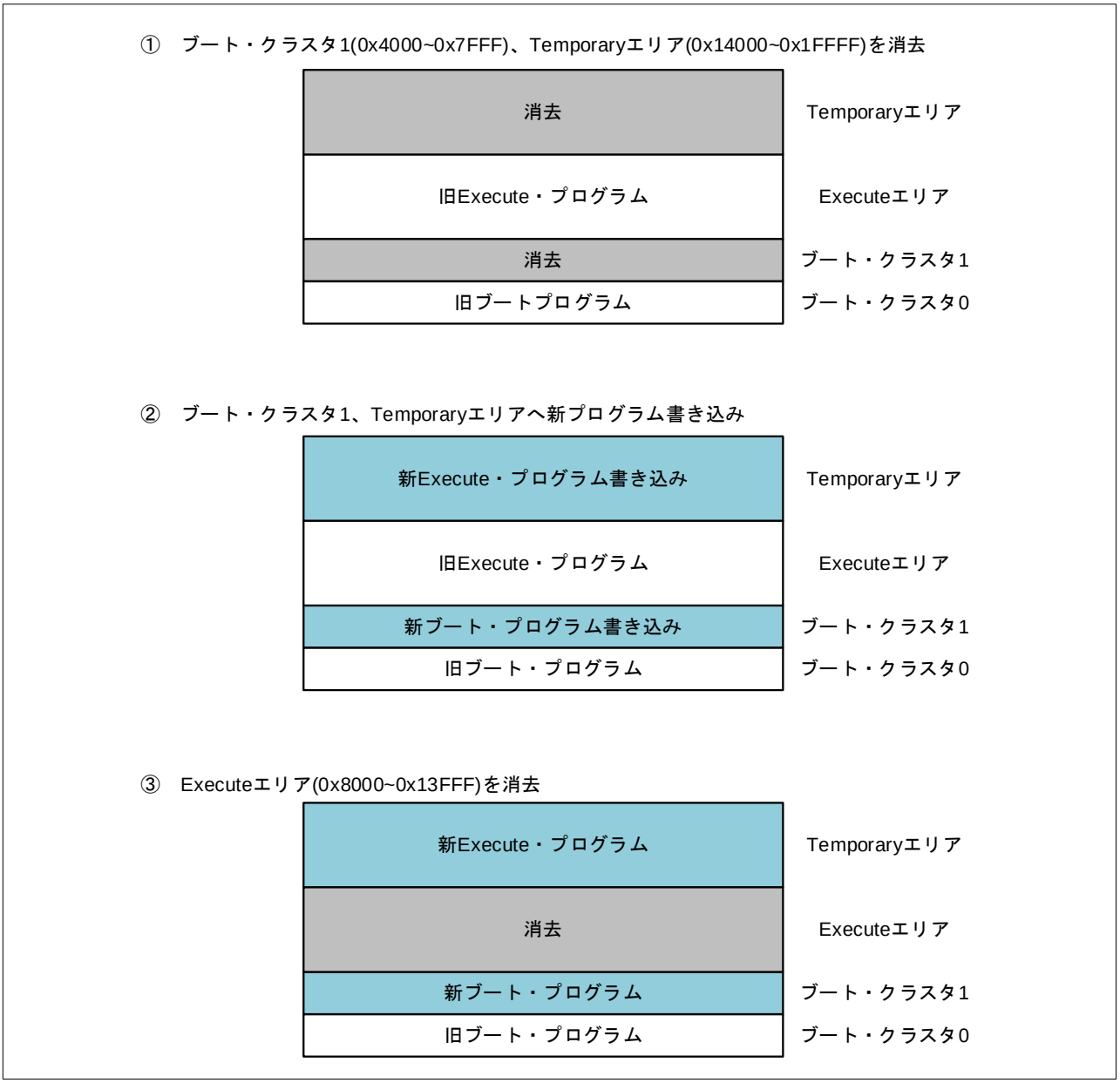
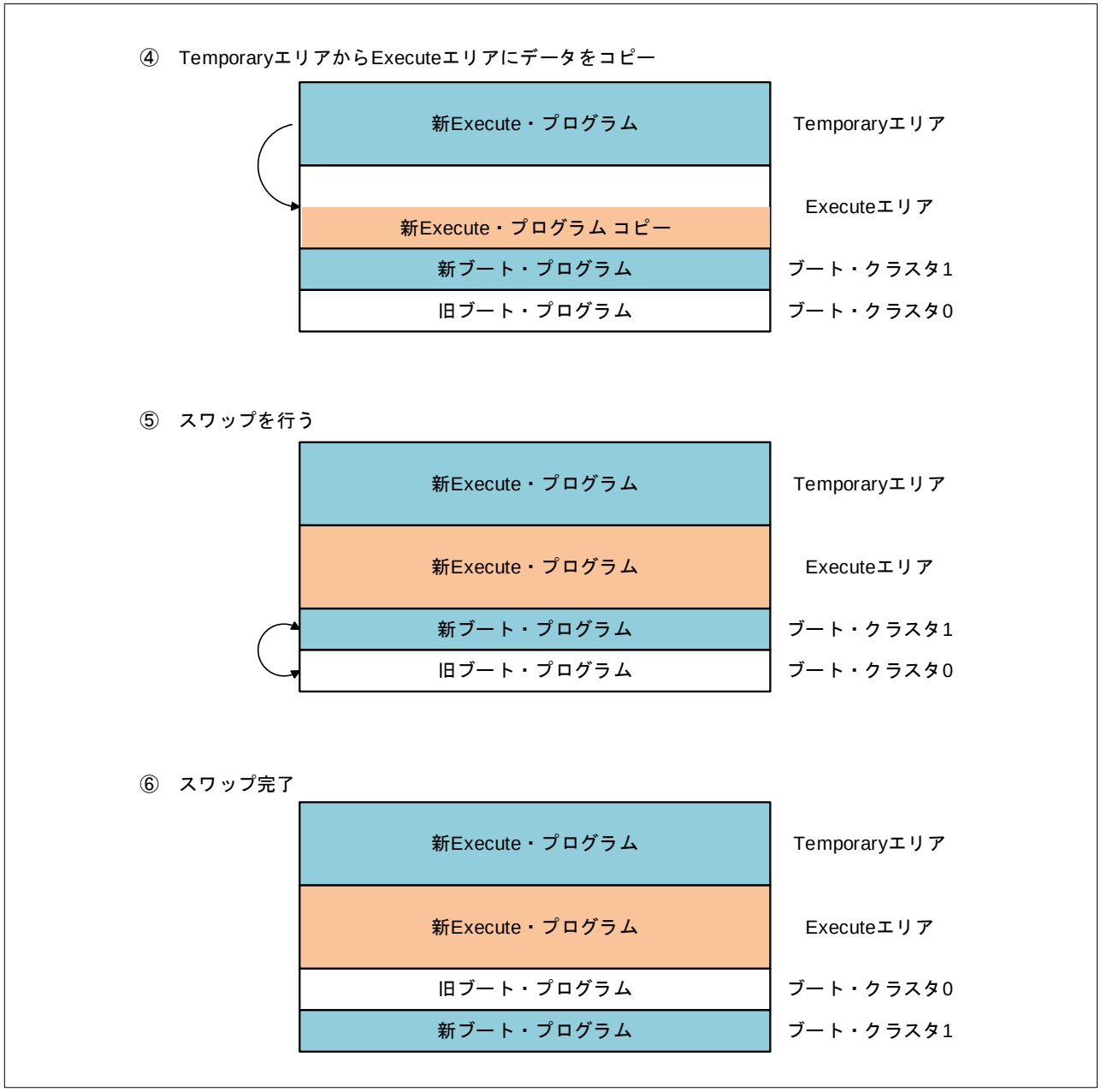


図 1-4　書き換え動作イメージ (2/2)

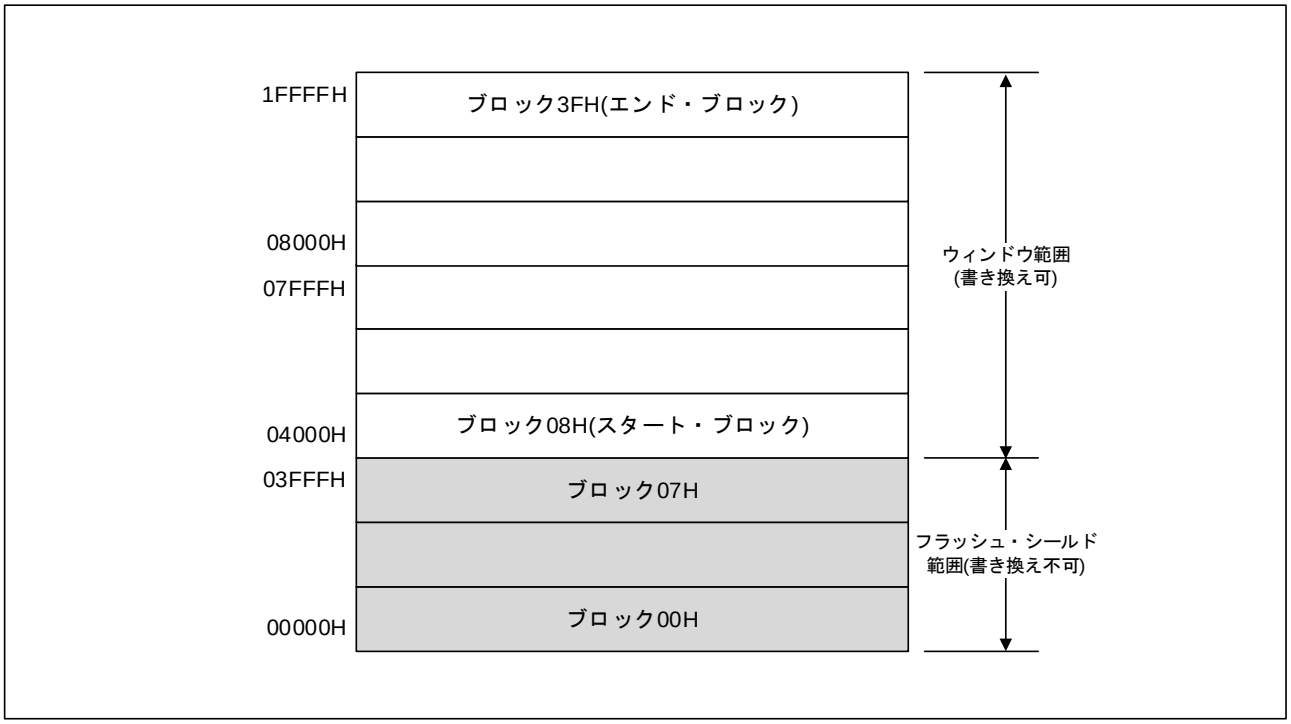


1.1.6　フラッシュ・シールド・ウィンドウ

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



1.1.7　Renesas Flash Driver RL78 Type01 の取得方法

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・コード(Renesas Flash Driver RL78 Type01)をダウンロードし、各プロジェクトの RFD フォルダ内にファイルをコピーしてください。

プロジェクトフォルダ				説明
r01an6255jj0100-rl78g23-flash				
	\src			
		\RFD		
			\include	ダウンロードした Renesas Flash Driver RL78 Type01 を配置してください
			\source	
			\userown	

Renesas Flash Driver RL78 Type01 は、下記 URL から取得することができます。

https://www.renesas.com/jp/ja/document/scd/renesas-flash-driver-rl78-type-01-rl78g23

1.2 動作概要

(1) ポートの初期設定を行います。
- P03, P02, P43, P42, P77, P41, P31, P76 を出力モードに設定

(2) シリアル・アレイ・ユニットの初期設定を行います。
- UART0 を使用 (P12 を TXD0 に、P11 を RXD0 に設定)
- 動作クロックを CK00 に、クロックソースを fCLK/2 に設定
- 転送モード設定をクロックソースに設定
- データ・ビット長設定を 8bit に設定
- データ転送方向設定を LSB に設定
- パリティ設定をパリティなしに設定
- ストップビット長設定を 1bit
- 送信データ・レベル設定を標準に設定
- ボーレート設定を 115200bps に設定

(3) コマンド通信を使用して、ブート・クラスタ 1、プログラム領域のデータを書き換え、ブート・スワップを行います。

1.2.1 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_BOOT1 コマンド、WRITE_TEMP コマンド、END コマンドの 4 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（01H）を返します。異常終了の場合には応答を返さず、異常終了用の LED を点灯して、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 1-7 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

1.2.2 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とブート・クラスタ 1 と Temporary エリアの消去を行います。正常終了すると送信側へ正常応答 (01H) を返します。異常終了の場合には応答を返さず、異常終了用の LED を点灯して、以降の処理は行いません。

START コード (01H)	データ長 (0002H)	コマンド (02H)	データ (なし)	チェックサム (1 バイト)

1.2.3 WRITE_BOOT1 コマンド

WRITE_BOOT1 コマンドを受信すると受信したデータをブート・クラスタ 1 エリア (4000H ~ 7FFFH) へ書き込み、256 バイトの書き込み毎にベリファイを行います。正常終了すると書き込み先アドレスを 256 バイト分インクリメントし、送信側へ正常応答 (01H) を返します。異常終了の場合には応答を返さず、異常終了用の LED を点灯して、以降の処理は行いません。

START コード (01H)	データ長 (0102H)	コマンド (03H)	データ (256 バイト)	チェックサム (1 バイト)

1.2.4 WRITE_TEMP コマンド

WRITE_TEMP コマンドを受信すると受信したデータを Temporary エリアへ書き込み、256 バイトの書き込み毎にベリファイを行います。正常終了すると書き込み先アドレスを 256 バイト分インクリメントし、送信側へ正常応答 (01H) を返します。異常終了の場合には応答を返さず、異常終了用の LED を点灯して、以降の処理は行いません。

(Temporary エリアの書き込み先アドレスは使用する製品によって変動します。)

START コード (01H)	データ長 (0102H)	コマンド (04H)	データ (256 バイト)	チェックサム (1 バイト)

1.2.5 END コマンド

END コマンドを受信すると Execute エリアを消去します。消去が正常終了の場合は Temporary エリアから Execute エリアにデータをコピーします。コピーが正常終了の場合は送信側へ正常応答 (01H) を返します。その後ブート・フラグを反転しリセットを発生させ、ブート・スワップを行います。

START コード (01H)	データ長 (0002H)	コマンド (05H)	データ (なし)	チェックサム (1 バイト)

1.2.6 チェックサム計算方法

チェックサムの計算方式は“32 ビット加算計算方式”を使用します。

コマンド、データを対象に、00000000H から 1 バイトずつ値を加算した結果の下位 8 ビットをチェックサムとして使用します。

1.2.7　サンプル・プログラムの動作

本サンプル・プログラムでの動作を以下に説明します。

(1)　入出力ポートの設定を行います。

(2)　SAU0 チャンネル 0 の初期設定を行います。

(3)　送信側からの送信データを待ちます。

(4)　送信側から START コマンド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。

(5)　P02 をハイ・レベル出力にし、「START コマンド受信」を示す LED2 を点灯します。

(6)　r_CF_EraseBlock 関数を呼び出しブート・クラスタ 1 のデータを消去します。

(7)　r_CF_EraseBlock 関数を呼び出し Temporary エリアのデータを消去します。

(8)　送信側へ正常応答 (01H) を送信します。

(9)　P02 をロウ・レベル出力にし、「START コマンド受信」を示す LED2 を消灯します。

(10) WRITE_BOOT1 コマンド (03H) と書き込みデータ (256 バイト) を受信します。

(11) P43 をハイ・レベル出力にし、「WRITE_BOOT1 コマンド受信」を示す LED3 を点灯します。

(12) r_CF_WriteData 関数を呼び出し、書き込み先アドレス (ブート・クラスタ 1 書き込み用ローカル変数)に受信データを書き込みます。ブート・クラスタ 1 書き込み用ローカル変数の初期値は、ブート・クラスタ 1 の開始アドレスとなります。

(13) r_CF_VerifyData 関数を呼び出し、書き込まれたデータと受信データのベリファイを行います。

(14) 書き込み先アドレス(ブート・クラスタ 1 書き込み用ローカル変数)を書き込みサイズ (256 バイト) 分加算します。

(15) 送信側へ正常応答 (01H) を送信します。

(16) P43 をロウ・レベル出力にし、「WRITE_BOOT1 コマンド受信」を示す LED3 を消灯します。

(17) 送信側から WRITE_TEMP コマンド (04H) を受信するまで (11) ~ (17) の処理を繰り返します。

(18) WRITE_TEMP コマンド (04H) と書き込みデータ (256 バイト) を受信します。

(19) P42 をハイ・レベル出力にし、「WRITE_BOOT1 コマンド受信」を示す LED4 を点灯します。

(20) r_CF_WriteData 関数を呼び出し、書き込み先アドレス (Temporary エリア書き込み用ローカル変数)に受信データを書き込みます。Temporary エリア書き込み用ローカル変数の初期値は Temporary エリアの開始アドレスとなります。

(21) r_CF_VerifyData 関数を呼び出し、書き込まれたデータと受信データのベリファイを行います。

(22) 書き込み先アドレス (Temporary エリア書き込み用ローカル変数) を書き込みサイズ (256 バイト) 分加算します。

(23) 送信側へ正常応答 (01H) を送信します。

(24) P42 をロウ・レベル出力にし、「WRITE_TEMP コマンド受信」を示す LED4 を消灯します。

(25) 送信側から END コマンド (05H) を受信するまで (19) ~ (25) の処理を繰り返します。

(26) END コマンドを受信した場合は次の処理を行います。

(27) P77 をハイ・レベル出力にし、「END コマンド受信」を示す LED5 を点灯します。

(28) r_CF_EraseBlock 関数を呼び出し Execute エリアのデータを消去します。

(29) P41 をハイ・レベル出力にし、「Temporary コピー中」を示す LED6 を点灯します。

(30) r_temp_copy 関数を呼び出し、Temporary エリアから Execute エリアにデータをコピー注します。

(31) P41 をロウ・レベル出力にし、「Temporary コピー中」を示す LED6 を消灯します。

(32) 送信側へ正常応答 (01H) を送信します。

(33) r_RequestBootSwap 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。内部リセットを発生させます。

注　Temporary エリアから Execute エリアにコピーしている最中に電源の瞬断などのリセットがかかった場合、再度 r_temp_copy 関数が呼び出され、コピーが完了した後に r_RequestBootSwap 関数が呼び出されます。

注意　上記(10) ~ (17)の間で送信側より END コマンド (05H) を受信した場合は、エラー状態でなければ Temporary エリアから Execute エリアにデータのコピーを行います。その後正常応答 (01H) を送信し、r_RequestBootSwap 関数が呼び出され、ブート・スワップを行います。ブート・クラスタ 1 の書き換え途中に END コマンド (05H) を受信した場合、正常に書き換えが終了しないままブート・スワップを行います。そのためスワップ後にプログラムを起動することができなくなります。

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、LED6 のみを点灯して以降の処理は行いません。

1.2.8　コピーフラグ

本アプリケーションノートのサンプル・プログラムは Execute エリアの最終アドレスにコピーフラグ用のセクションとコピーフラグを 4 バイトのサイズで設けてあります。

このコピーフラグは正常にプログラムが書き込まれている状態だと AAAA5555H に設定されています。Temporary エリアから Execute エリアにデータをコピーする直前に Execute エリアを消去するため、コピーフラグが初期化されます。書き込み途中に電源の瞬断が発生し、リセットがかかった場合は書き込み処理が正常に終了しないためコピーフラグの値が AAAA5555H 以外の値になります。

本サンプル・プログラムの起動時の処理にこのコピーフラグを確認し、設定値以外の値であれば書き込み処理を行ってからスワップ処理を行います。

各 ROM サイズのフラグ配置用セクションなどのアドレスを以下に記載します。

表 1-8　各 ROM サイズのフラグ配置用セクション

ROM サイズ	Execute エリア	コピーフラグの配置アドレス
96KB	08000H ~ FFFFH	FFFCH
128KB	08000H ~ 13FFFH	13FFCH
192KB	08000H ~ 1BFFFH	1BFFCH
256KB	08000H ~ 23FFFH	23FFCH
384KB	08000H ~ 33FFFH	33FFCH
512KB	08000H ~ 43FFFH	43FFCH
768KB	08000H ~ 63FFFH	63FFCH

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

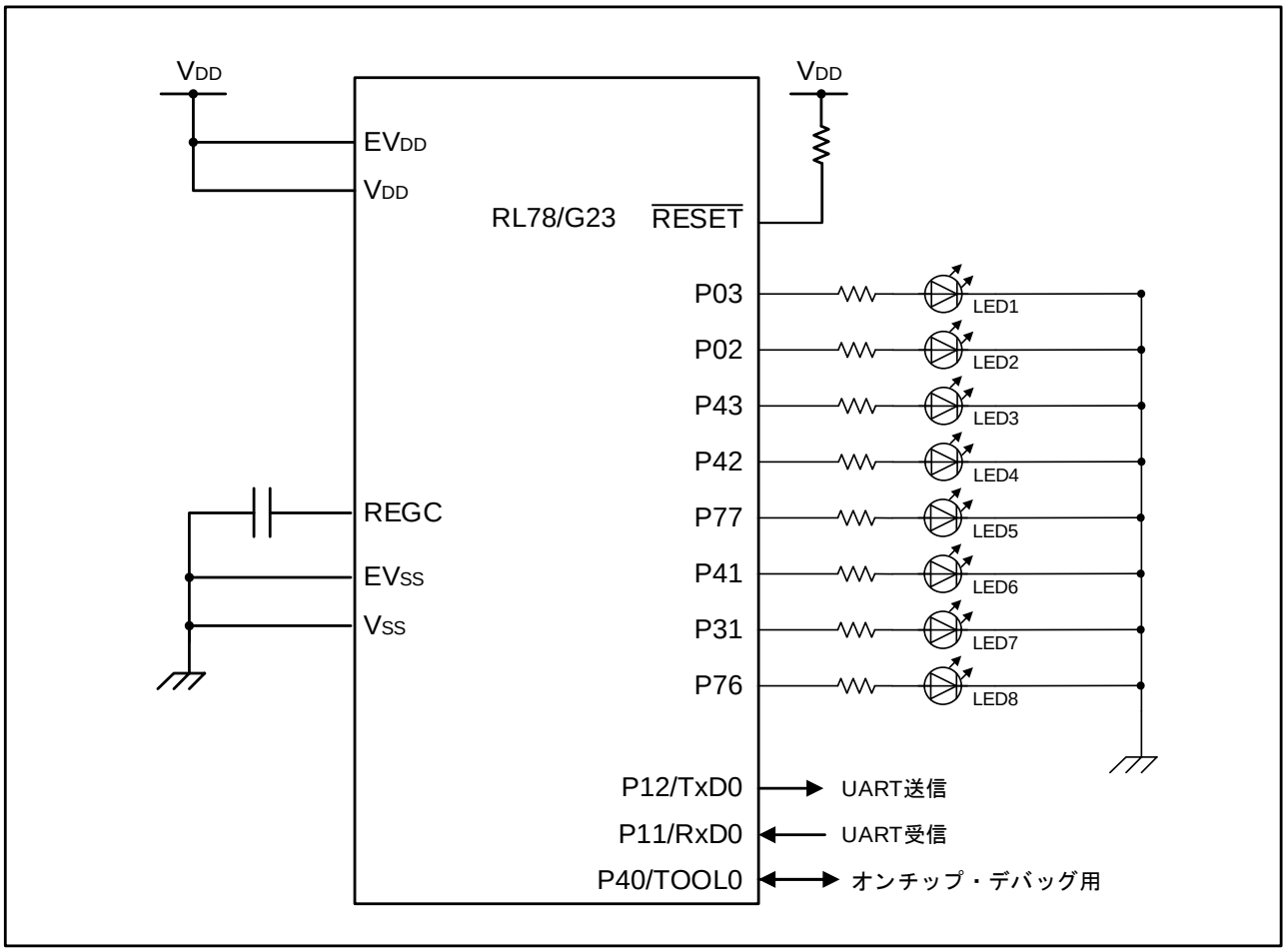
周辺機能	用途
使用マイコン	RL78/G23 (R7F100GLG)
使用ボード	[128KB 製品用プロジェクト] RL78/G23-64p Fast Prototyping Board (RTK7RLG230CLG000BJ) [768KB 製品用プロジェクト] RL78/G23-128p Fast Prototyping Board (RTK7RLG230CSN000BJ)
動作周波数	高速オンチップ・オシレータ・クロック (fIH) : 32MHz
動作電圧	3.3V (3.1V～3.5V で動作可能) LVD 動作 (V_{LVD}): リセット・モード 立ち上がり時 TYP. 1.90 V (1.84 V ～ 1.95 V) 立ち下がり時 TYP. 1.86 V (1.80 V ～ 1.91 V)
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ for CC V8.06.00
C コンパイラ (CS+)	ルネサス エレクトロニクス製 CC-RL V1.10.00
統合開発環境 (e2studio)	ルネサス エレクトロニクス製 e2studio V2021-10
C コンパイラ (e2studio)	ルネサス エレクトロニクス製 CC-RL V1.10.00
統合開発環境 (IAR)	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 V4.21.1
C コンパイラ (IAR)	IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 V 4.21.1.2409

3. ハードウェア説明

3.1 ハードウェア構成例

図 3-1 に本アプリケーションノートで使用するハードウェア構成例を示します。

図 3-1 ハードウェア構成



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) 。

注意 2. EV_{SS} で始まる名前の端子がある場合には V_{SS} に、EV_{DD} で始まる名前の端子がある場合には V_{DD} にそれぞれ接続してください。

注意 3. V_{DD} は LVD0 にて設定したリセット解除電圧 (V_{LVD0}) 以上にしてください。

3.2 使用端子一覧

表 3-1 にサンプル・プログラムで使用する端子と機能を示します。

表 3-1 使用端子一覧

端子名	入出力	内容
P12//TxD0	出力	UART シリアル・データ送信用端子
P11/ RxD0	入力	UART シリアル・データ受信用端子
P03, P02, P43, P42, P77, P41, P31, P76	出力	LED1-LED8 への出力端子

注意 本アプリケーションノートは、使用端子のみを端子処理しています。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。

4. ソフトウェア説明

4.1 オプション・バイトの設定一覧

表 4-1 にサンプル・プログラムで使用する、オプション・バイトの設定を示します。

表 4-1 オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	11101111B	ウォッチドッグ・タイマ動作停止 (リセット解除後、カウント開始)
000C1H/040C1H	11111110B	LVD 動作 (V_{LVD}): リセット・モード 立ち上がり時 TYP. 1.90 V (1.84 V ～ 1.95 V) 立ち下がり時 TYP. 1.86 V (1.80 V ～ 1.91 V)
000C2H/040C2H	11101000B	HS モード、 高速オンチップ・オシレータ・クロック : 32MHz
000C3H/040C3H	10000101B	オンチップ・デバッグ許可

RL78/G23 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、040C0H - 040C3H にも 000C0H - 000C3H と同じ値を設定する必要があります。

4.2　スタートアップ・ルーチンの設定

4.2.1　スタック領域用セクション (.stack_bss) の定義

スタック領域用セクション (.stack_bss) の定義を行います。

スタートアップ・ルーチンを設定する cstart.asm の下記内容を修正してください。

```
;$IF (__RENESAS_VERSION__ < 0x01010000)                先頭行に ';' を追加しコメントアウト
;--------------------------------------------------------------------------
;
;--------------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
.DS     0x800                                         16 進数で任意のスタックサイズを入力
_stacktop:
;$ENDIF                                                先頭行に ';' を追加しコメントアウト


;------------------------------------------------
; setting the stack pointer
;------------------------------------------------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    SP,#LOWW(__STACK_ADDR_START)                  先頭行に ';' を追加しコメントアウト
;$ELSE   ; for CC-RL V1.00                             先頭行に ';' を追加しコメントアウト
MOVW    SP,#LOWW(_stacktop)
$ENDIF

;------------------------------------------------
; initializing stack area
;------------------------------------------------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    AX,#LOWW(__STACK_ADDR_END)                    先頭行に ';' を追加しコメントアウト
;$ELSE   ; for CC-RL V1.00                             先頭行に ';' を追加しコメントアウト
MOVW    AX,#LOWW(_stackend)
$ENDIF
CALL    !!_stkinit
```

4.2.2　書き換え用プログラムの RAM 領域への配置

ファームウェアの書き換えに使用するプログラムの RAM 領域へ配置します。

ファームウェアの書き換えに使用するプログラムは表 4-2 に記載されるセクションに配置されています。

表 4-2　セクション情報

セクション名	配置先セクション名	配置内容
RFD_CMN_f	RFD_CMN_fR	共通フラッシュ・メモリ制御 API 関数のプログラム・セクション
RFD_CF_f	RFD_CF_fR	コード・フラッシュ・メモリ API 関数のプログラム・セクション
RFD_EX_f	RFD_EX_fR	エクストラ領域制御 API 関数のプログラム・セクション
SMP_CMN_f	SMP_CMN_fR	共通フラッシュ・メモリ制御 サンプル関数のプログラム・セクション
SMP_CF_f	SMP_CF_fR	コード・フラッシュ・メモリ制御サンプル関数のプログラム・セクション

上記セクションを RAM 領域に配置するために cstart.asm に処理を追加する必要があります。

cstart.asm 内の下記記述の後に処理を追加します。

```
;---------------------------------------------------
; ROM data copy
;---------------------------------------------------
```

追記する内容は以下の通りです。

```
    ; copy .text to RAM (セクション名)
    MOV     C,#HIGHW(STARTOF(セクション名))
    MOVW    HL,#LOWW(STARTOF(セクション名))
    MOVW    DE,#LOWW(STARTOF(配置先セクション名))
    BR      $.Lm2_TEXT
.Lm1_TEXT:
    MOV     A,C
    MOV     ES,A
    MOV     A,ES:[HL]
    MOV     [DE],A
    INCW    DE
    INCW    HL
    CLRW    AX
    CMPW    AX,HL
    SKNZ
    INC
.Lm2_TEXT:
    MOVW    AX,HL
    CMPW    AX,#LOWW(STARTOF(セクション名) + SIZEOF(セクション名))
    BNZ     $.Lm1_TEXT
```

注意 1.　**セクション名**には配置対象となるセクション名を 1 つ記載してください。

注意 2.　配置が必要なセクションの数だけ上記記述を追加してください。

注意 3.　m は任意の数値を設定してください。セクションごとに異なる数値を設定してください。

4.3　ROM サイズ設定用の定数設定

本サンプル・プログラムは条件付きコンパイルによって RL78/G23 の各 ROM サイズに対応しています。

r_cg_userdefine.h に記述されている以下の定数のコメントアウトを削除することで該当の ROM サイズに対応できます。

表 4-3　各 ROM サイズの定数

定数名	対応 ROM サイズ
ROM_SIZE_96KB	96KB 製品
ROM_SIZE_128KB	128KB 製品
ROM_SIZE_192KB	192KB 製品
ROM_SIZE_256KB	256KB 製品
ROM_SIZE_384KB	384KB 製品
ROM_SIZE_512KB	512KB 製品
ROM_SIZE_768KB	768KB 製品

4.4　オンチップ・デバッグ・セキュリティ ID

RL78/G23 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、040C4H - 040CDH にも 000C4H - 000CDH と同じ値を設定する必要があります。

4.5 サンプル・プログラム使用リソース

4.5.1 ROM 領域セクション一覧

表 4-4 にサンプル・プログラムで使用する ROM 領域のセクション一覧を示します。

表 4-4 ROM 領域セクション一覧

セクション名	配置内容
RFD_DATA_n	RFD RL78 Type01 のデータ・セクション
RFD_CMN_f	共通フラッシュ・メモリ制御 API 関数のプログラム・セクション
RFD_CF_f	コード・フラッシュ・メモリ制御 API 関数のプログラム・セクション
RFD_EX_f	エクストラ領域制御 API 関数のプログラム・セクション
RFD_DF_f	データ・フラッシュ・メモリ制御 API 関数のプログラム・セクション
SMP_CMN_f	共通フラッシュ・メモリ制御 サンプル関数のプログラム・セクション
SMP_CF_f	コード・フラッシュ・メモリ制御サンプル関数のプログラム・セクション
BOOT_AREA1	ブート・クラスタ 1 のプログラム・セクション
USER_APPLICATION	ユーザーアプリケーションのプログラム・セクション
COPY_FLAG_f	コピー完了フラグ格納用のプログラム・セクション
TEMPORARY_AREA	受信データ格納用のプログラム・セクション

4.5.2 RAM 領域セクション一覧

表 4-5 にサンプル・プログラムで使用する RAM 領域セクションの一覧を示します。

表 4-5 RAM 領域セクション一覧

セクション名	配置内容
RFD_DATA_nR	RFD RL78 Type01 のデータ・セクション
RFD_CMN_fR	共通フラッシュ・メモリ制御 API 関数のプログラム・セクション
RFD_CF_fR	コード・フラッシュ・メモリ制御 API 関数のプログラム・セクション
RFD_EX_fR	エクストラ領域制御 API 関数のプログラム・セクション
SMP_CMN_fR	共通フラッシュ制・メモリ御 サンプル関数のプログラム・セクション
SMP_CF_fR	コード・フラッシュ・メモリ制御サンプル関数のプログラム・セクション

4.6 定数一覧

表 4-6、表 4-7 にサンプル・プログラムで使用する定数を示します。

表 4-6 定数一覧 (1/2)

定数名	設定値	内容
ROM_SIZE_96KB	01H	ROM サイズを 96KB に設定する値
ROM_SIZE_128KB	01H	ROM サイズを 128KB に設定する値
ROM_SIZE_192KB	01H	ROM サイズを 192KB に設定する値
ROM_SIZE_256KB	01H	ROM サイズを 256KB に設定する値
ROM_SIZE_384KB	01H	ROM サイズを 384KB に設定する値
ROM_SIZE_512KB	01H	ROM サイズを 512KB に設定する値
ROM_SIZE_768KB	01H	ROM サイズを 768KB に設定する値
LED_ON	01H	LED ON
LED_OFF	00H	LED OFF
WRITE_DATA_SIZE	0100H	コード・フラッシュ・メモリ書き込みサイズ (256 バイト)
CF_BLOCK_SIZE	0800H	コード・フラッシュ・メモリブロックサイズ (2048 バイト)
BT1_START_ADDRESS	00004000H	ブート・クラスタ 1 開始アドレス
BT1_END_ADDRESS	00007FFFH	ブート・クラスタ 1 終了アドレス
EXECUTE_START_ADDRESS	00008000H	EXECUTE エリア開始アドレス
EXECUTE_END_ADDRESS 注	00013FFFH	EXECUTE エリア終了アドレス
TEMPORARY_START_ADDRESS 注	00014000H	TEMPORARY エリア開始アドレス
TEMPORARY_END_ADDRESS 注	0001FFFFH	TEMPORARY エリア終了アドレス
CPU_FREQUENCY	32	CPU 動作周波数
COMMAND_START	02H	コマンドコード: START
COMMAND_WRITE_BOOT1	03H	コマンドコード: WRITE_BOOT1
COMMAND_WRITE_TEMP	04H	コマンドコード: WRITE_TEMP
COMMAND_END	05H	コマンドコード: END
VALUE_U08_MASK1_FSQ_STATUS_ERR_ERASE	01H	フラッシュ・メモリ・シーケンサ実行結果のエラー・ステータス・マスク値 0 ビット目 : 消去コマンド・エラー
VALUE_U08_MASK1_FSQ_STATUS_ERR_WRITE	02H	フラッシュ・メモリ・シーケンサ実行結果のエラー・ステータス・マスク値 1 ビット目 : 書き込みコマンド・エラー
VALUE_U08_MASK1_FSQ_STATUS_ERR_BLANKCHECK	08H	フラッシュ・メモリ・シーケンサ実行結果のエラー・ステータス・マスク値 3 ビット目 : ブランク・チェック・コマンド・エラー
VALUE_U08_MASK1_FSQ_STATUS_ERR_CFDF_SEQUENCER	10H	フラッシュ・メモリ・シーケンサ実行結果のエラー・ステータス・マスク値 4 ビット目 : コード／データ・フラッシュ領域シーケンサ・エラー
VALUE_U08_MASK1_FSQ_STATUS_ERR_EXTRA_SEQUENCER	20H	フラッシュ・メモリ・シーケンサ実行結果のエラー・ステータス・マスク値 5 ビット目 : エクストラ領域シーケンサ・エラー

注　使用する製品によりアドレスが変化します。

表 4-7　定数一覧 (2/2)

VALUE_U08_SHIFT_ADDR_TO_BLOCK_CF	11	コード・フラッシュ・メモリのブロック番号算出時に行うビットシフト用定数
VALUE_U01_MASK0_1BIT	0	演算用定数(0)
VALUE_U01_MASK1_1BIT	1	演算用定数(1)
VALUE_U08_MASK0_8BIT	00H	演算用定数(00H)
VALUE_U08_MASK1_8BIT	FFH	演算用定数(FFH)
COPY_FLAG_USUAL	AAAA5555H	コピーフラグ用の設定値

4.7　列挙型

表 4-8 にサンプル・プログラムで使用する列挙型の定義を示します。

表 4-8　enum e_ret (列挙変数名: e_ret_t)

Symbol Name	値	内容
ENUM_RET_STS_OK	00H	ステータス正常
ENUM_RET_STS_RECEIVING	01H	コマンド受信待ち/受信中
ENUM_RET_ERR_CFDF_SEQUENCER	02H	コード/データ・フラッシュ領域シーケンサ・エラー
ENUM_RET_ERR_EXTRA_SEQUENCER	03H	エクストラ領域シーケンサ・エラー
ENUM_RET_ERR_ERASE	04H	消去エラー
ENUM_RET_ERR_WRITE	05H	書き込みエラー
ENUM_RET_ERR_BLANKCHECK	06H	ブランク・エラー
ENUM_RET_ERR_CHECK_WRITE_DATA	07H	書き込みデータのリード値比較エラー
ENUM_RET_ERR_MODE_MISMATCHED	08H	モード不一致エラー
ENUM_RET_ERR_PARAMETER	09H	パラメータ・エラー
ENUM_RET_ERR_CONFIGURATION	0AH	デバイス構成・エラー
ENUM_RET_ERR_PACKET	0BH	パケット受信エラー

4.8 変数一覧

表 4-9 にサンプル・プログラムで使用するグローバル変数を示します。

表 4-9 グローバル変数一覧

型	変数名	内容	使用関数
uint8_t	f_UART0_sendend	UART0 にてデータの送信が完了したことを示すグラグ	r_Send_nByte r_Config_UART0_callback_sendend
uint32_t	g_copy_end	データのコピーが正常に終了していることを示すフラグ	main
uint8_t	g_recv_data [261]	受信データ用バッファ	R_Config_UART0_Receive r_AsyncRecvPacketData
uint8_t	g_soft_recv_overrun	受信データ用バッファの大きさを超えるデータを受信したことを示すフラグ	r_Config_UART0_callback_softwareoverrun r_ClearUARTRecvBuff r_AsyncRecvPacketData

4.9 関数一覧

表 4-10 にサンプル・プログラムで使用する関数を示します。

表 4-10 関数一覧

関数名	概要
r_rfd_initialize	RFD RL78 Type01 初期化処理
r_cmd_start	STARTコマンド処理
r_cmd_end	ENDコマンド処理
r_CF_RangeErase	コード・フラッシュ・メモリ 範囲消去処理
r_CF_EraseBlock	コード・フラッシュ・メモリ ブロック消去処理
r_CF_WriteVerifySequence	コード・フラッシュ・メモリ 書き込み・ベリファイ処理
r_CF_WriteData	コード・フラッシュ・メモリ 書き込み処理
r_CF_VerifyData	コード・フラッシュ・メモリ ベリファイ処理
r_CheckCFDFSequencerEnd	コード・フラッシュ・メモリ シーケンス終了処理
r_CheckExtraSequencerEnd	エクストラ領域 シーケンス終了処理
r_RequestBootSwap	ブート・スワップ 実行処理
r_Config_UART0_callback_sendend	UART0 送信完了割り込み時のコールバック処理
r_Send_nByte	UART0 データ送信処理
r_SendACK	UART0 正常応答送信処理
r_CF_TempCopy	Temporaryエリアからデータコピー処理
r_CF_MemoryWrite	コード・フラッシュ・メモリ 書き換え処理
r_AsyncRecvPacketData	非同期コマンドパケット受信処理
r_GetUARTRecvSize	受信データサイズ取得処理
r_ClearUARTRecvBuff	受信バッファクリア処理
userApplicationLoop	ユーザーアプリケーション実装用関数
updateLoop	ファームウェアアップデート用コマンド受信/実行処理
errorLedOn	エラーLED点灯処理

4.10 関数仕様

サンプルコードの関数仕様を示します。

r_rfd_initialize	
概　要	RFD RL78 Type01 初期化処理
ヘッダ	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_rfd_initialize(void);
説　明	RFD RL78 Type01 の初期化処理を行います。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_CONFIGURATION: クロック構成エラー ENUM_RET_ERR_PARAMETER: 周波数設定エラー

r_cmd_start	
概　要	START コマンド処理
ヘッダ	r_rfd_common_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_cmd_start(void);
説　明	START コマンド受信時の処理を行います。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

r_cmd_end	
概　要	END コマンド処理
ヘッダ	r_rfd_common_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_cmd_end(void);
説　明	END コマンド受信時の処理を行います。
引　数	なし
リターン値	ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー

r_CF_RangeErase	
概　要	コード・フラッシュ・メモリ 範囲消去処理
ヘッダ	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_RangeErase(uint32_t start_addr, uint32_t end_addr);
説　明	コード・フラッシュ・メモリに対してデータ消去を行います。 引数で指定されたアドレスが含まれるブロックを対象とし、ブロック単位で消去処理が実施されます。
引　数	uint32_t start_addr: 消去開始アドレス uint32_t end_addr: 消去終了アドレス
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

r_CF_EraseBlock	
概　要	コード・フラッシュ・メモリ ブロック消去処理
ヘッダ	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_EraseBlock(uint32_t start_addr);
説　明	コード・フラッシュ・メモリに対してデータ消去を行います。 引数で指定されたアドレスが含まれるブロックを対象とし、1 ブロック分の消去処理が実施されます。
引　数	uint32_t start_addr: 消去開始アドレス
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

r_CF_WriteVerifySequence	
概　要	コード・フラッシュ・メモリ 書き込み・ベリファイ処理
ヘッダ	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_WriteVerifySequence(uint32_t write_start_addr, uint16_t write_data_length, uint8_t __near *write_data);
説　明	コード・フラッシュ・メモリに対してデータ書き込み・ベリファイ処理を行います。
引　数	uint32_t start_addr,: 書き込み開始アドレス uint16_t write_data_length: 書き込みサイズ uint8_t __near *write_data: 書き込みデータ
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

r_CF_WriteData	
概　要	コード・フラッシュ・メモリ 書き込み処理
ヘッダ	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_WriteData(uint32_t start_addr, uint16_t write_data_length, uint8_t __near *write_data);
説　明	コード・フラッシュ・メモリに対して書き込みを行います。
引　数	uint32_t start_addr,: 書き込み開始アドレス uint16_t write_data_length: 書き込みサイズ uint8_t __near *write_data: 書き込みデータ
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_WRITE: 書き込みエラー

r_CF_VerifyData	
概　要	コード・フラッシュ・メモリ ベリファイ処理
ヘッダ	r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_VerifyData(uint32_t start_addr, uint16_t data_length, uint8_t __near * write_data);
説　明	コード・フラッシュ・メモリに書き込まれたデータに対してベリファイ処理を行います。
引　数	uint32_t start_addr: ベリファイ開始アドレス uint16_t data_length: データサイズ uint8_t __near * write_data: 比較データ
リターン値	ENUM_RET_STS_OK: 正常終了 (一致) ENUM_RET_ERR_CHECK_WRITE_DATA: 書き込みデータのリード値比較エラー (不一致)

r_CheckCFDFSequencerEnd	
概　要	コード・フラッシュ・メモリ シーケンス終了処理
ヘッダ	r_rfd_common_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CheckCFDFSequencerEnd(void);
説　明	コード・フラッシュ・メモリ シーケンスの動作終了を確認します。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_CFDF_SEQUENCER: コード/データ・フラッシュ・メモリ シーケンサ・エラー ENUM_RET_ERR_ERASE: 消去エラー ENUM_RET_ERR_WRITE: 書き込みエラー ENUM_RET_ERR_BLANKCHECK: ブランク・エラー

r_CheckExtraSequencerEnd	
概　要	エクストラ・メモリ シーケンス終了処理
ヘッダ	r_rfd_common_api.h、r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CheckExtraSequencerEnd (void);
説　明	エクストラ・メモリ シーケンスの動作終了を確認します。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_EXTRA_SEQUENCER: エクストラ・メモリ シーケンサ・エラー ENUM_RET_ERR_ERASE: 消去エラー ENUM_RET_ERR_WRITE: 書き込みエラー ENUM_RET_ERR_BLANKCHECK: ブランク・エラー

r_RequestBootSwap	
概　要	ブート・スワップ 実行処理
ヘッダ	r_rfd_common_api.h、r_rfd_extra_area_api.h 、r_cg_userdefine.h
宣　言	e_ret_t r_RequestBootSwap(void);
説　明	リセット後ブート・スワップ設定を有効にし、内部リセットを発生させ再起動を行います。
引　数	なし
リターン値	ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー

r_Config_UART0_callback_sendend()	
概　要	UART0 送信完了割込み時の処理
ヘッダ	r_cg_macrodriver.h
宣　言	static void r_Config_UART0_callback_sendend(void);
説　明	UART0 の送信完了割込み時に呼ばれるコールバック関数です。
引　数	なし
リターン値	なし

r_Send_nByte	
概　要	UART0 データ送信処理
ヘッダ	Config_UART0.h、Config_WDT.h
宣　言	MD_STATUS r_Send_nByte(uint8_t *tx_buff, const uint16_t tx_num);
説　明	UART0 の送信処理を行います。 引数で指定された文字数の送信が完了するまで、送信完了待ちを行います。
引　数	uint8_t *rx_buff: 送信データ格納先バッファのポインタ const uint16_t rx_num: 送信文字数
リターン値	MD_OK: 正常終了 (送信完了) MD_ARGERROR: パラメータ・エラー

r_SendACK	
概　要	UART0 正常応答送信処理
ヘッダ	Config_UART0.h、Config_WDT.h
宣　言	MD_STATUS r_SendACK (void);
説　明	UART0 を使用して正常応答 (01H) の送信処理を行います。
引　数	なし
リターン値	MD_OK: 正常終了 (送信完了) MD_ARGERROR: パラメータ・エラー

r_CF_TempCopy	
概　要	Temporaryエリアからデータコピー処理
ヘッダ	r_cg_userdefine.h、string.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_TempCopy(void);
説　明	Temporary エリアからデータをバッファにコピーします。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了

r_CF_MemoryWrite	
概　要	コード・フラッシュ・メモリ 書き換え処理
ヘッダ	r_cg_userdefine.h
宣　言	R_RFD_FAR_FUNC e_ret_t r_CF_MemoryWrite(uint32_t* write_start_addr, uint32_t write_end_addr, uint8_t __near * write_data);
説　明	メモリにデータを書き込みます。
引　数	uint32_t* write_start_addr: 書き込み開始アドレス uint32_t write_end_addr: 書き込み終了アドレス uint8_t __near * write_data: 書き込みデータ
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

r_AsyncRecvPacketData	
概　要	非同期コマンドパケット受信処理
ヘッダ	r_cg_userdefine.h
宣　言	__far uint8_t r_AsyncRecvPacketData(uint8_t *p_cmd_type, uint8_t rdata[]);
説　明	非同期で受信したデータを解析し、状態を返します。
引　数	uint8_t *p_cmd_type: コマンド情報 uint8_t rdata[]: 受信データ用バッファ
リターン値	ENUM_PACKET_STATUS_OK: 正常終了 ENUM_PACKET_STATUS_ERROR: パケット受信エラー ENUM_PACKET_STATUS_RECEIVING: パケット受信中

r_GetUARTRecvSize	
概　要	受信データサイズ取得処理
ヘッダ	r_cg_userdefine.h Config_UART0.h
宣　言	uint16_t r_GetUARTRecvSize(void);
説　明	現在受信したデータの長さを返します。
引　数	なし
リターン値	Size: 受信したデータの長さ

r_ClearUARTRecvBuff	
概　要	受信バッファクリア処理
ヘッダ	r_cg_userdefine.h Config_UART0.h
宣　言	void r_ClearUARTRecvBuff(void);
説　明	受信したデータを格納するバッファをクリアします。
引　数	なし
リターン値	なし

userApplicationLoop	
概　要	ユーザーアプリケーション実装用関数
ヘッダ	r_cg_userdefine.h
宣　言	void userApplicationLoop(void);
説　明	LED1/LED8 を点滅させるサンプルアプリケーションを実装しています。
引　数	なし
リターン値	なし

updateLoop	
概　要	ファームウェアアップデート用コマンド受信/実行処理
ヘッダ	r_cg_userdefine.h
宣　言	e_ret_t updateLoop(void);
説　明	ファームウェアアップデート用コマンドの受信と実行を行います。
引　数	なし
リターン値	ENUM_RET_STS_OK: 正常終了 ENUM_RET_ERR_CONFIGURATION: クロック構成エラー ENUM_RET_ERR_PARAMETER: 周波数設定エラー ENUM_PACKET_STATUS_ERROR: パケット受信エラー ENUM_PACKET_STATUS_RECEIVING: パケット受信中 ENUM_RET_ERR_MODE_MISMATCHED: モード不一致エラー ENUM_RET_ERR_ERASE: 消去エラー

errorLedOn	
概　要	エラーLED点灯処理
ヘッダ	r_cg_userdefine.h
宣　言	void errorLedOn(void);
説　明	エラー発生時に LED7 を点灯し、ほかの LED をすべて消灯します。
引　数	なし
リターン値	なし

4.11 フローチャート

4.11.1 メイン処理

図 4-1、図 4-2 にメイン処理のフローチャートを示します。

図 4-1　メイン処理 (1/2)

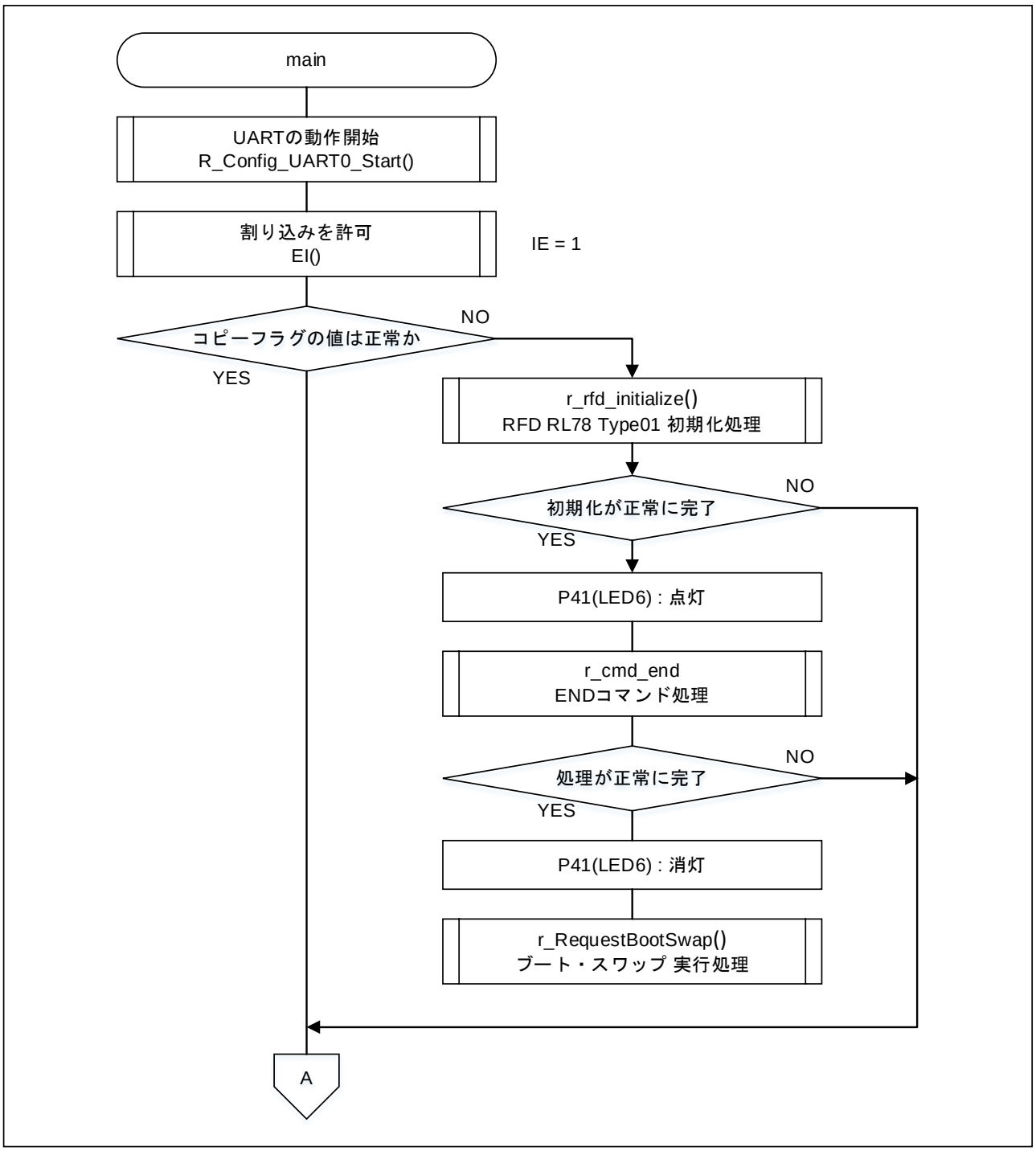
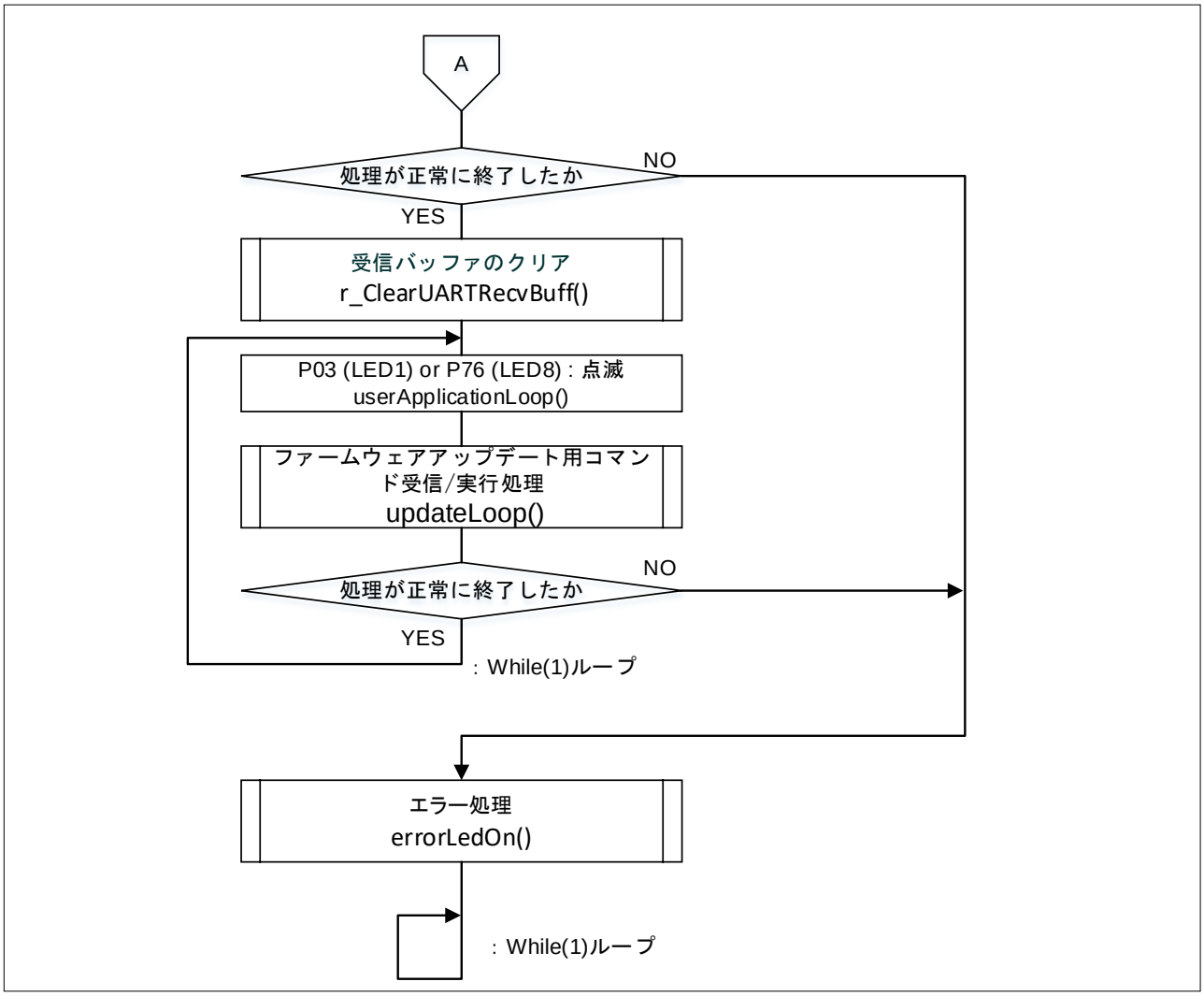


図 4-2　メイン処理 (2/2)


A
処理が正常に終了したか
NO
YES
受信バッファのクリア
r_ClearUARTRecvBuff()
P03 (LED1) or P76 (LED8) : 点滅
userApplicationLoop()
ファームウェアアップデート用コマンド受信/実行処理
updateLoop()
処理が正常に終了したか
NO
YES
: While(1)ループ
エラー処理
errorLedOn()
: While(1)ループ

4.11.2 ファームウェアアップデート用コマンド受信/実行処理

図 4-3、図 4-4、図 4-5 にファームウェアアップデート用コマンド受信/実行処理のフローチャートを示します。

図 4-3　ファームウェアアップデート用コマンド受信/実行処理 (1/3)

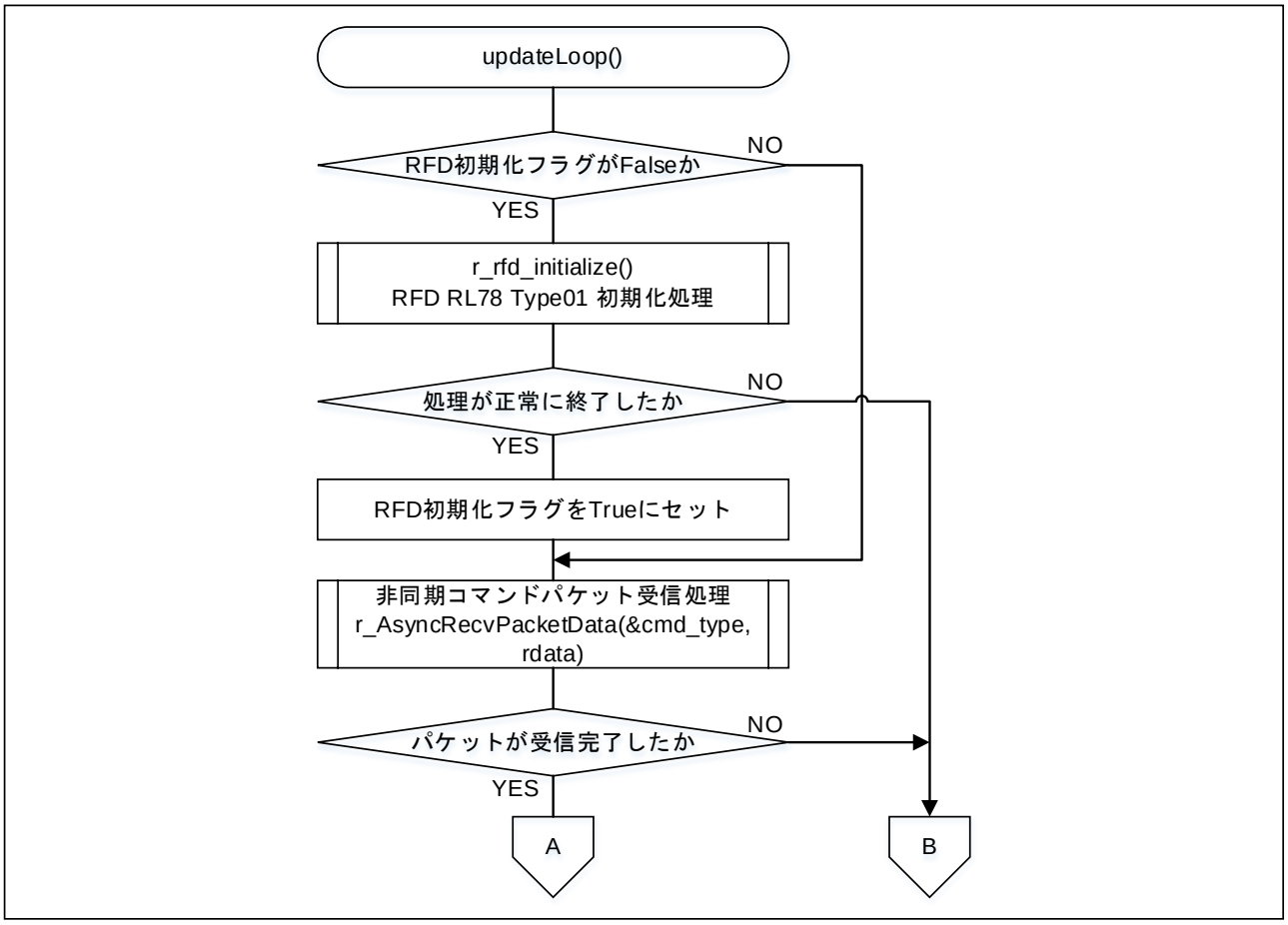


図 4-4　ファームウェアアップデート用コマンド受信/実行処理 (2/3)

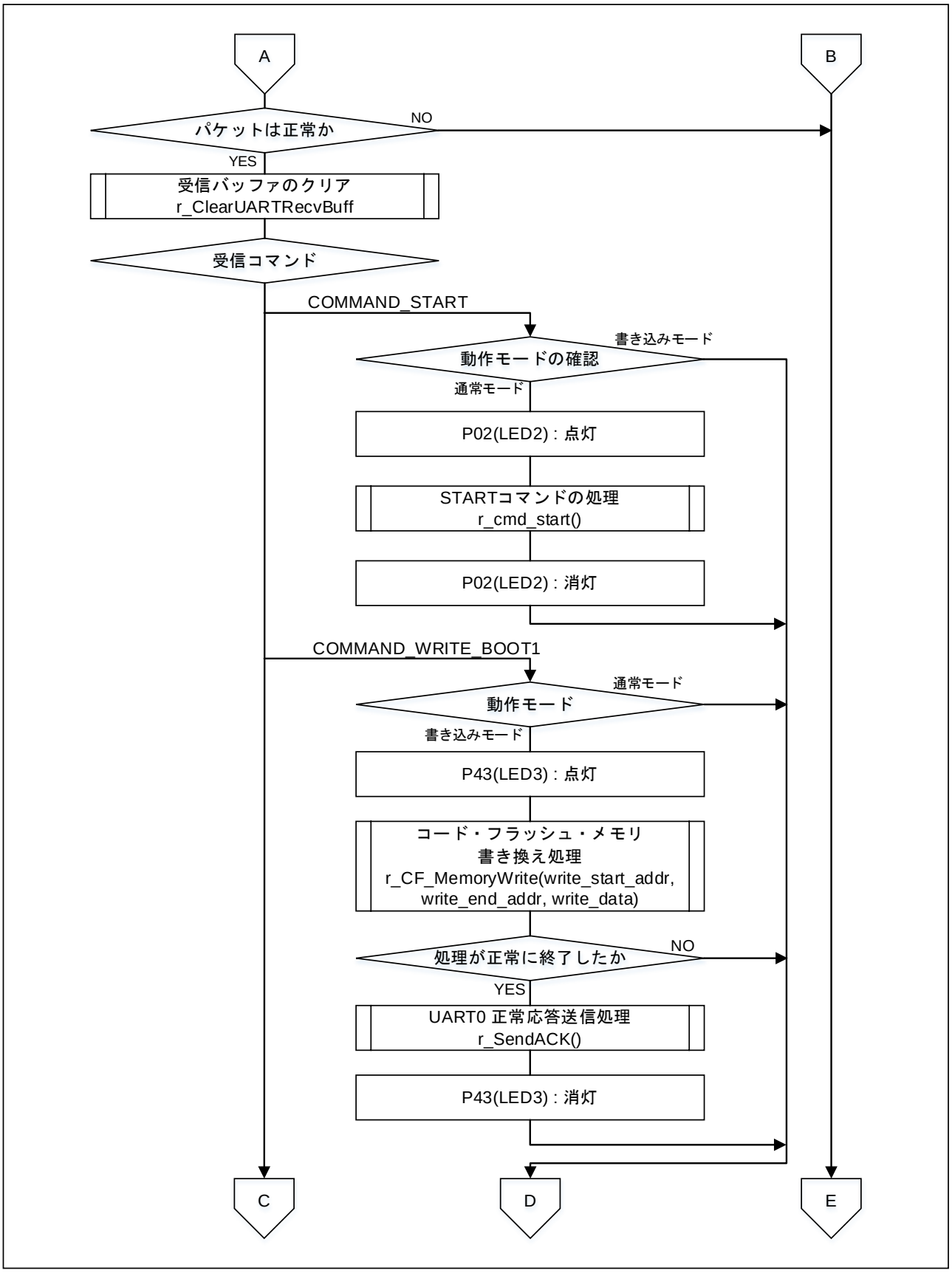
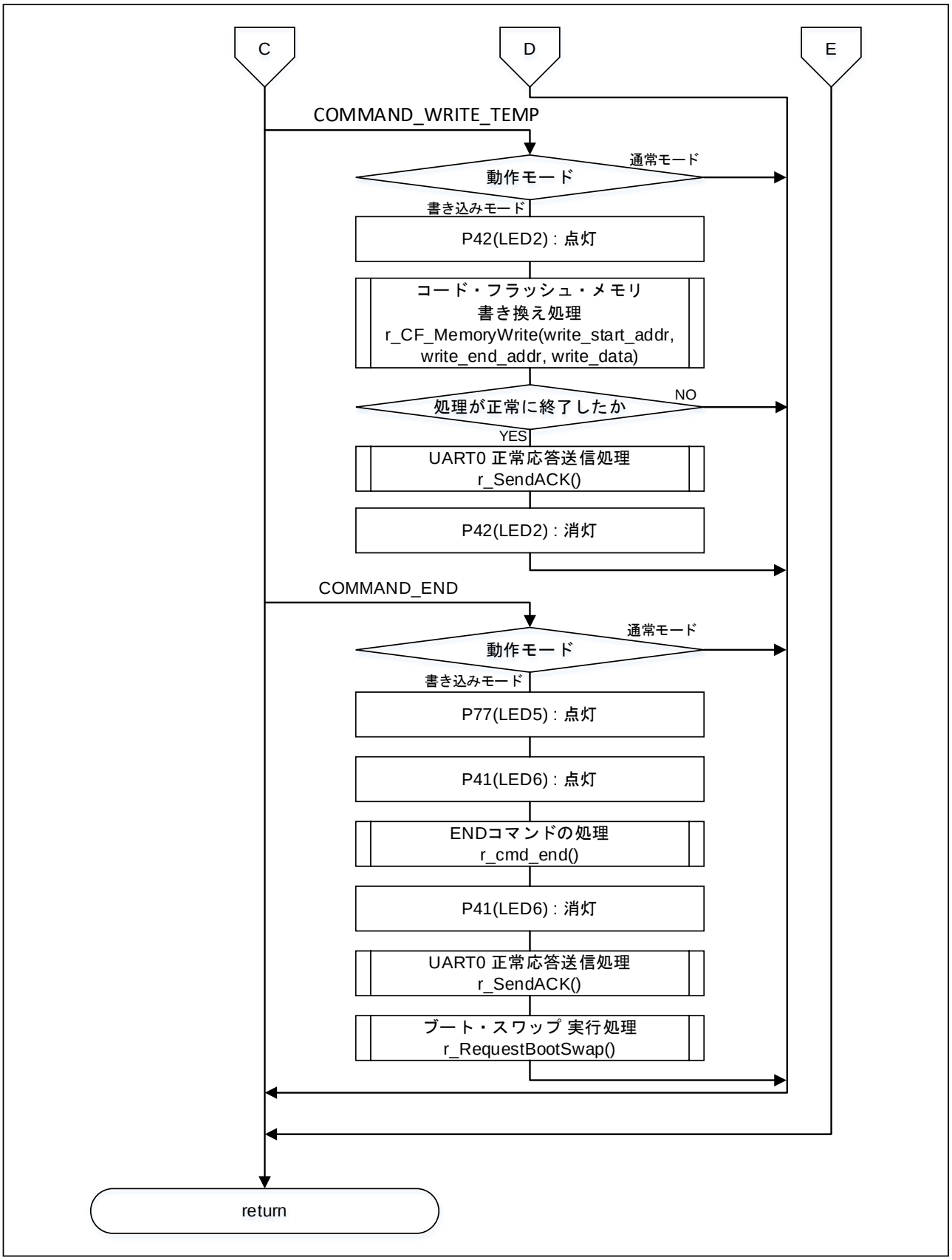


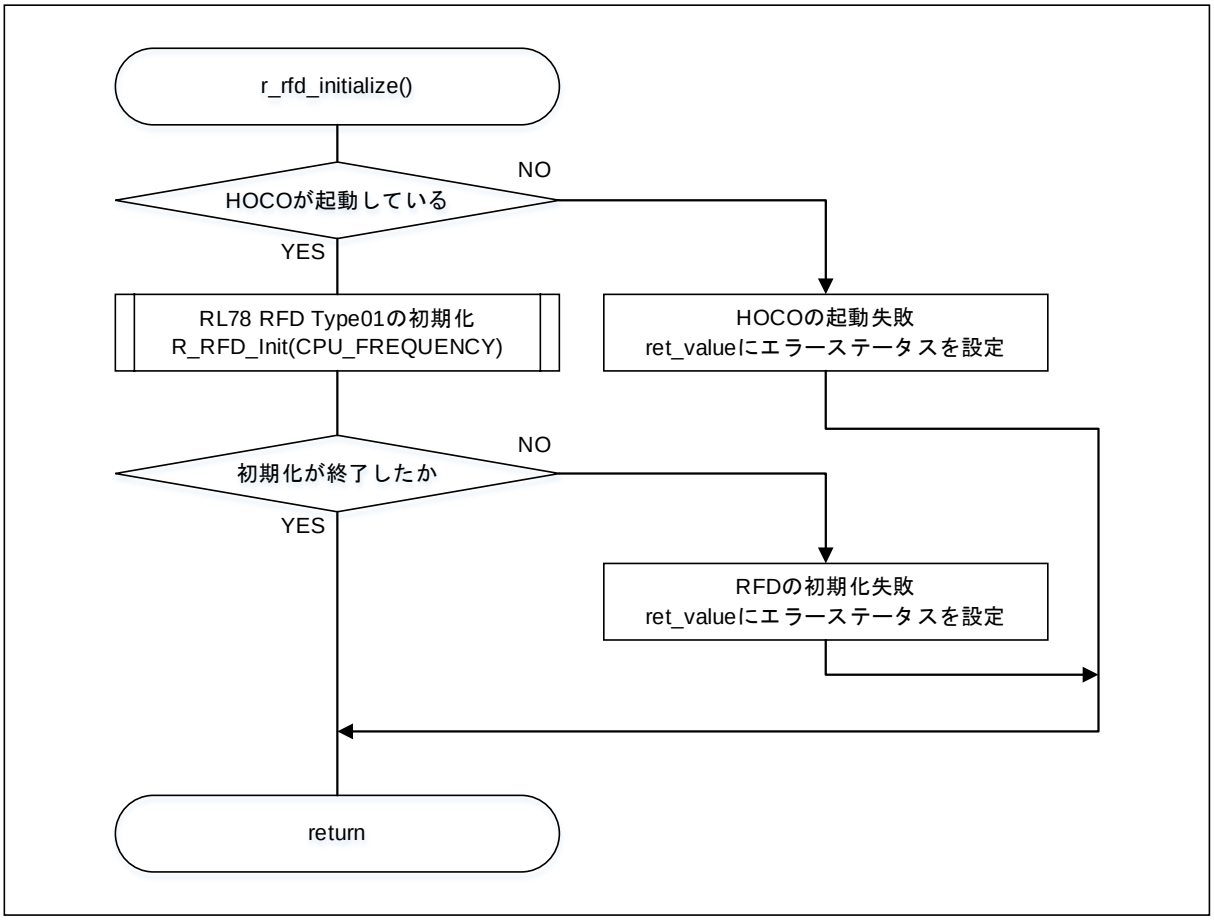
図 4-5　ファームウェアアップデート用コマンド受信/実行処理 (3/3)



4.11.3 RFD RL78 Type01 初期化処理

図 4-6 に RFD RL78 Type01 初期化処理のフローチャートを示します。

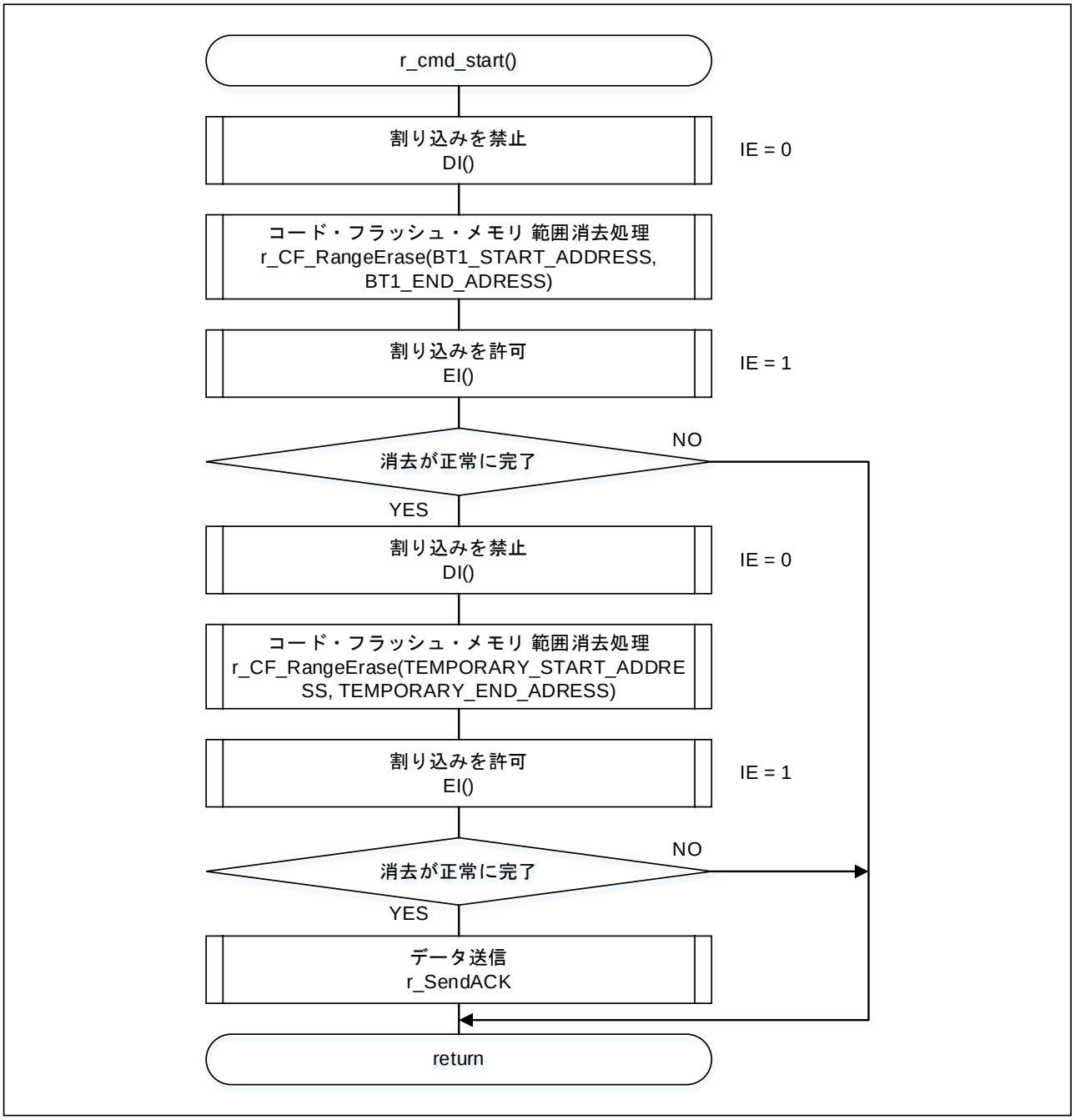
図 4-6　RFD RL78 Type01 初期化処理



4.11.4 START コマンド処理

図 4-7 にSTARTコマンド処理のフローチャートを示します。

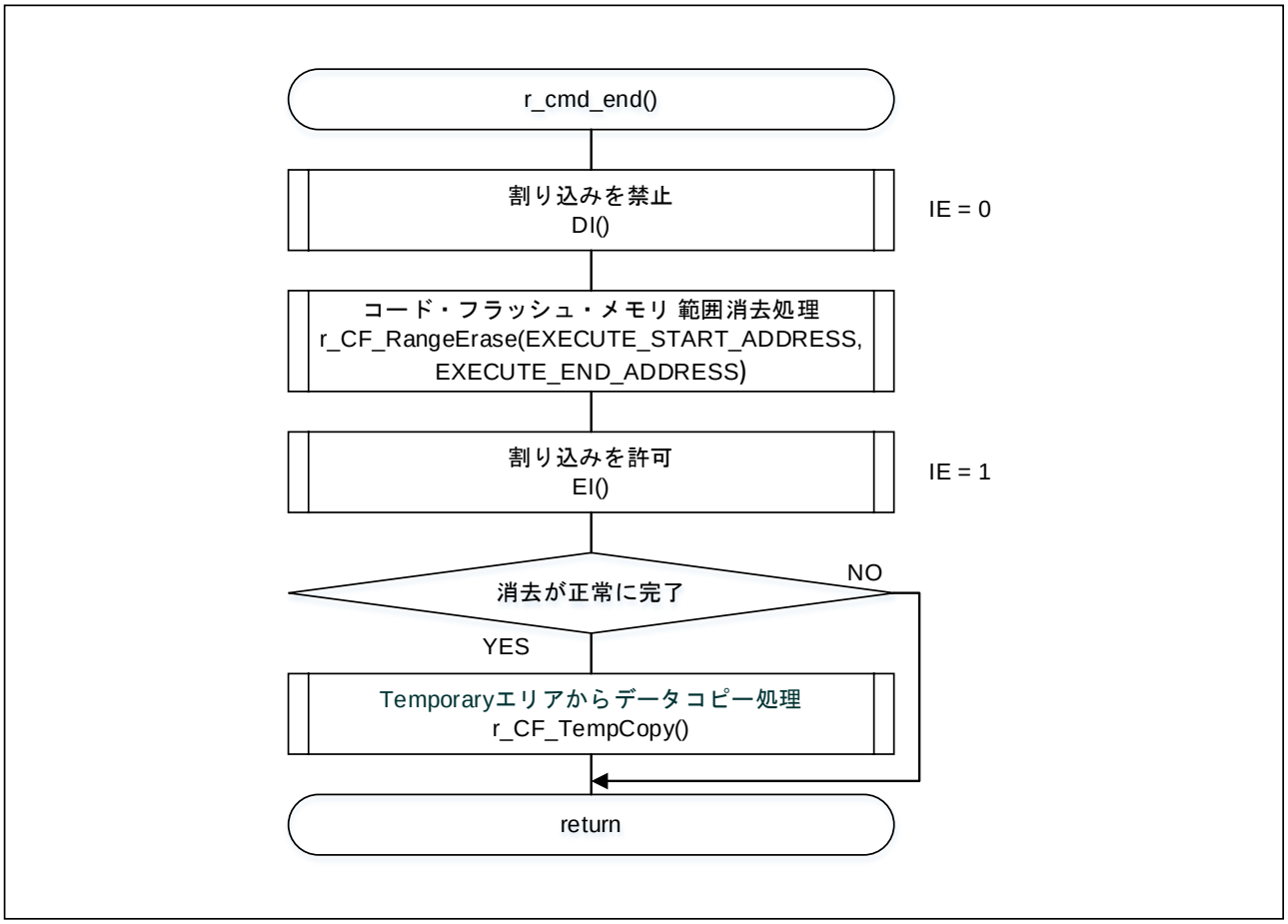
図 4-7　START コマンド処理



4.11.5 END コマンド処理

図 4-8 にENDコマンド処理のフローチャートを示します。

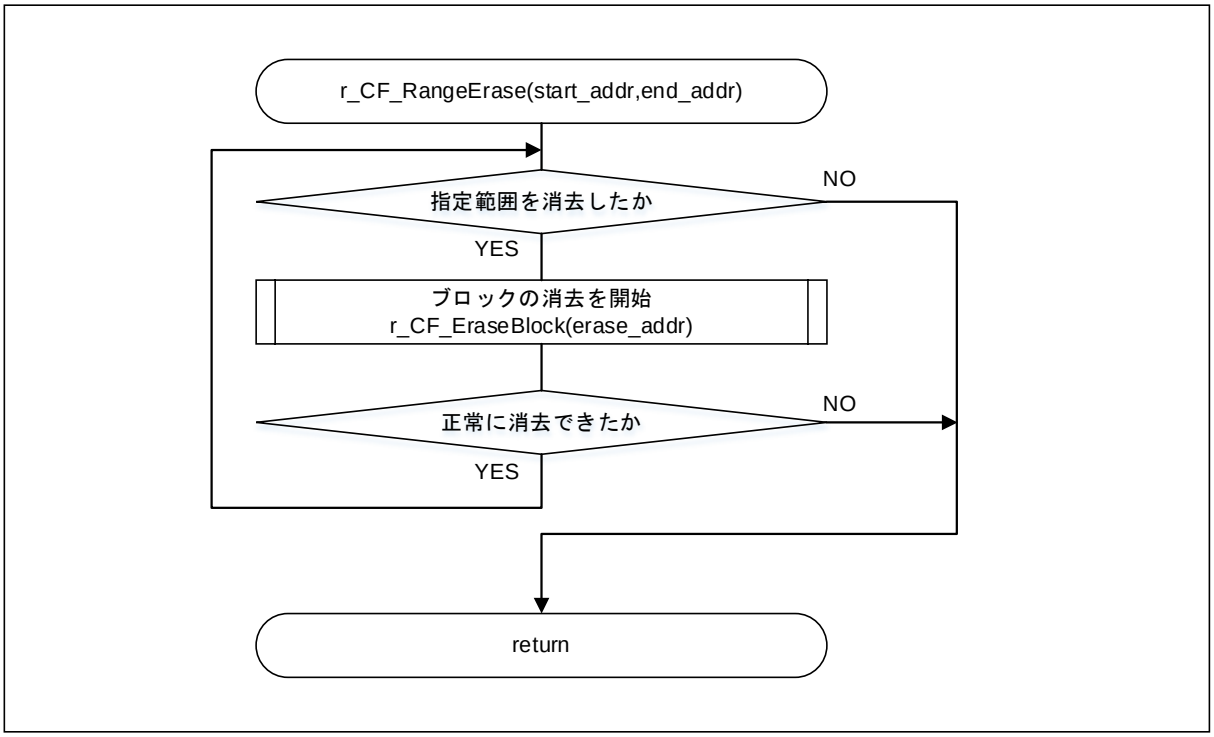
図 4-8　END コマンド処理



4.11.6 コード・フラッシュ・メモリ 範囲消去処理

図 4-9 にコード・フラッシュ・メモリ 範囲消去処理のフローチャートを示します。

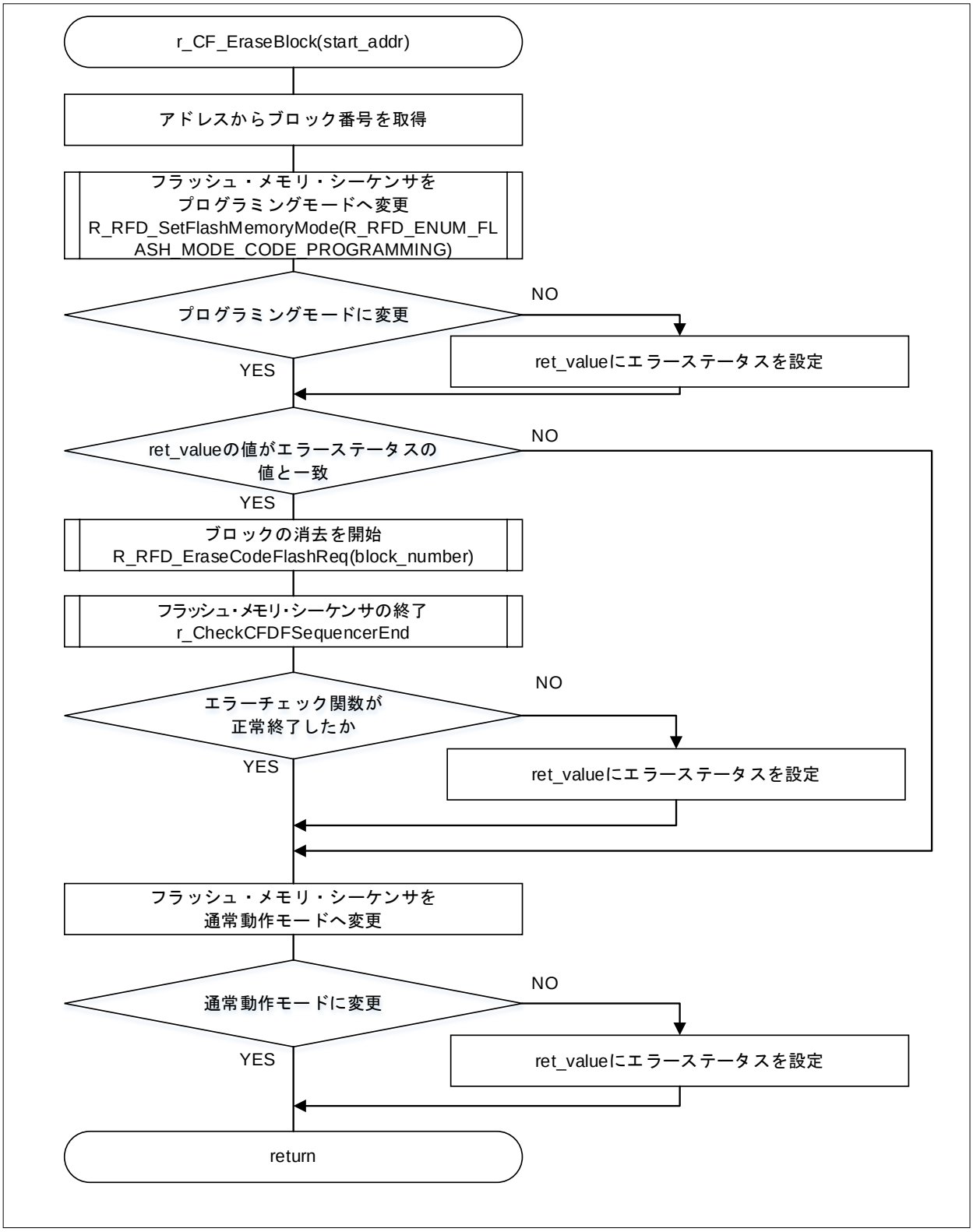
図 4-9　コード・フラッシュ・メモリ 範囲消去処理



4.11.7 コード・フラッシュ・メモリ ブロック消去処理

図 4-10 にコード・フラッシュ・メモリ ブロック消去処理のフローチャートを示します。

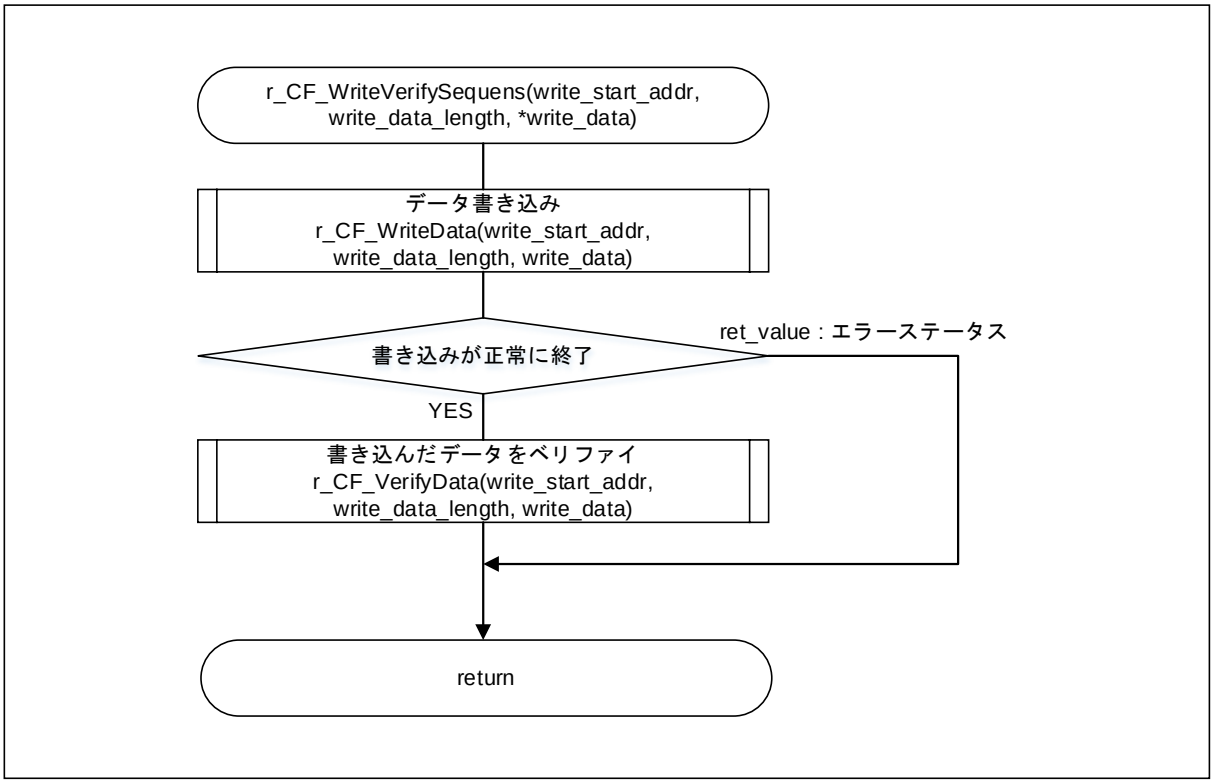
図 4-10　コード・フラッシュ・メモリ ブロック消去処理



4.11.8 コード・フラッシュ・メモリ 書き込み・ベリファイ処理

図 4-11 にコード・フラッシュ・メモリ 書き込み・ベリファイ処理のフローチャートを示します。

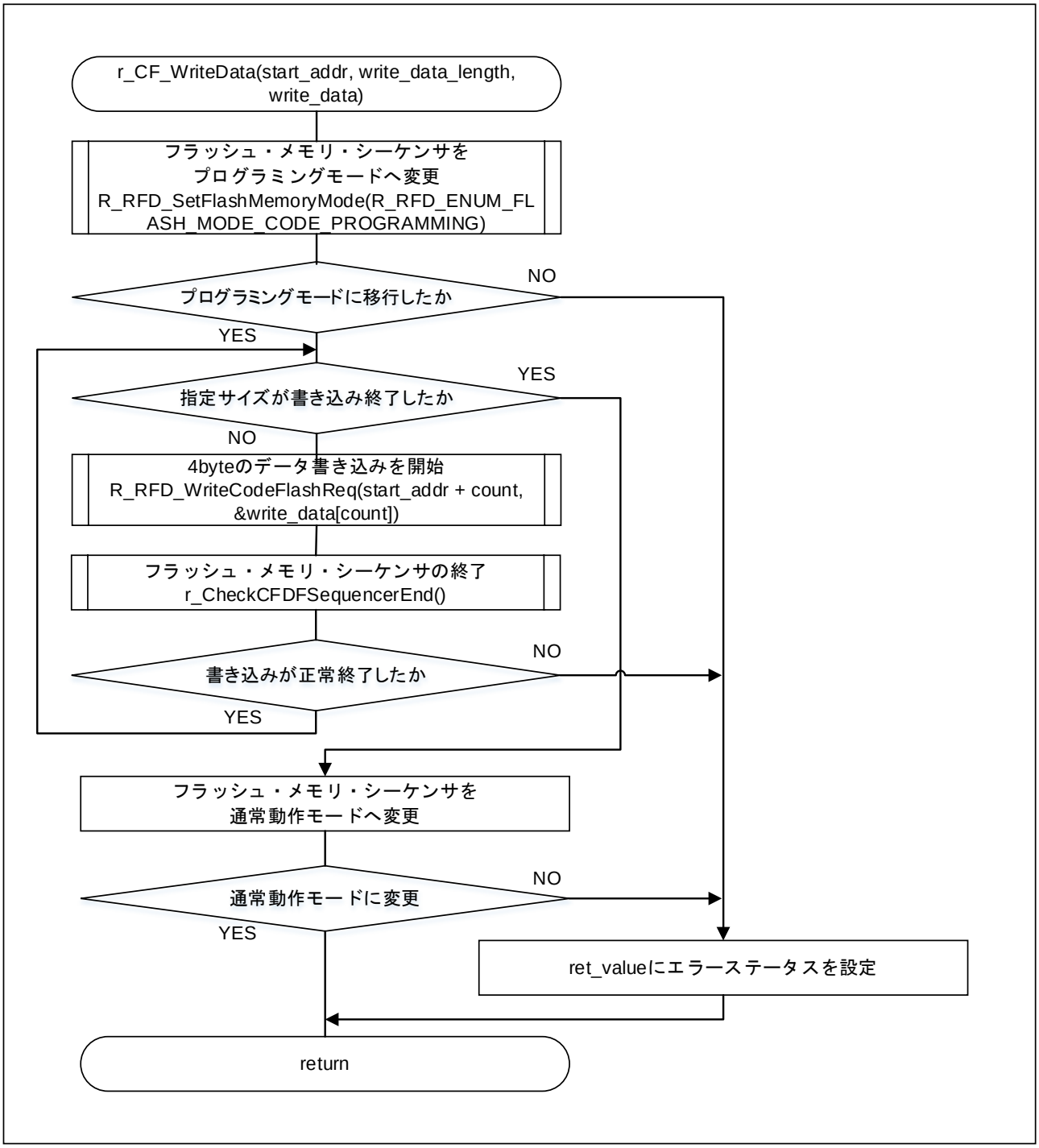
図 4-11　コード・フラッシュ・メモリ 書き込み・ベリファイ処理



4.11.9 コード・フラッシュ・メモリ 書き込み処理

図 4-12 にコード・フラッシュ・メモリ 書き込み処理のフローチャートを示します。

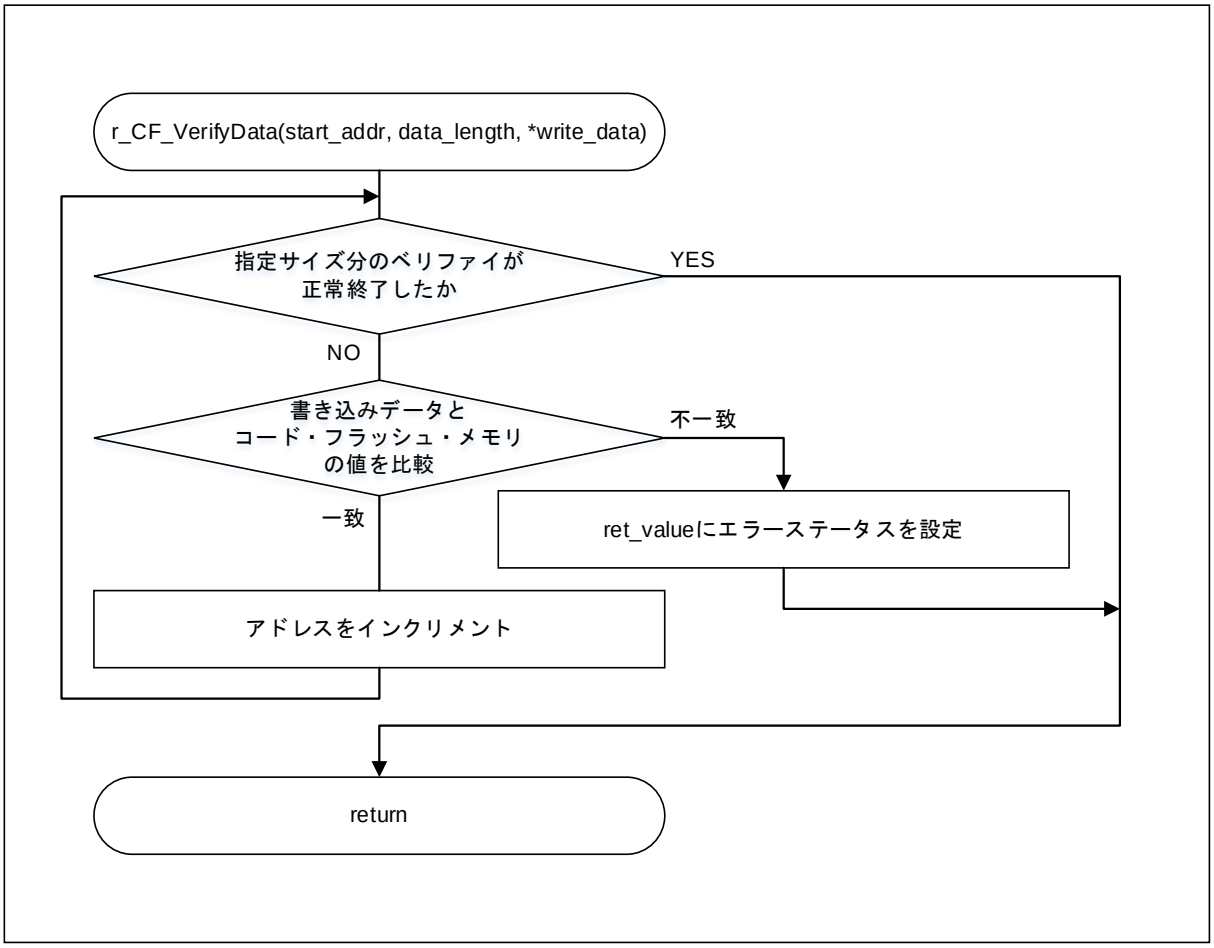
図 4-12　コード・フラッシュ・メモリ 書き込み処理



4.11.10 コード・フラッシュ・メモリ ベリファイ処理

図 4-13 にコード・フラッシュ・メモリ ベリファイ処理のフローチャートを示します。

図 4-13　コード・フラッシュ・メモリ ベリファイ処理



4.11.11 コード・フラッシュ・メモリ シーケンス終了処理

図 4-14、図 4-15 にコード・フラッシュ・メモリ シーケンス終了処理のフローチャートを示します。

図 4-14　コード・フラッシュ・メモリ シーケンス終了処理 (1/2)

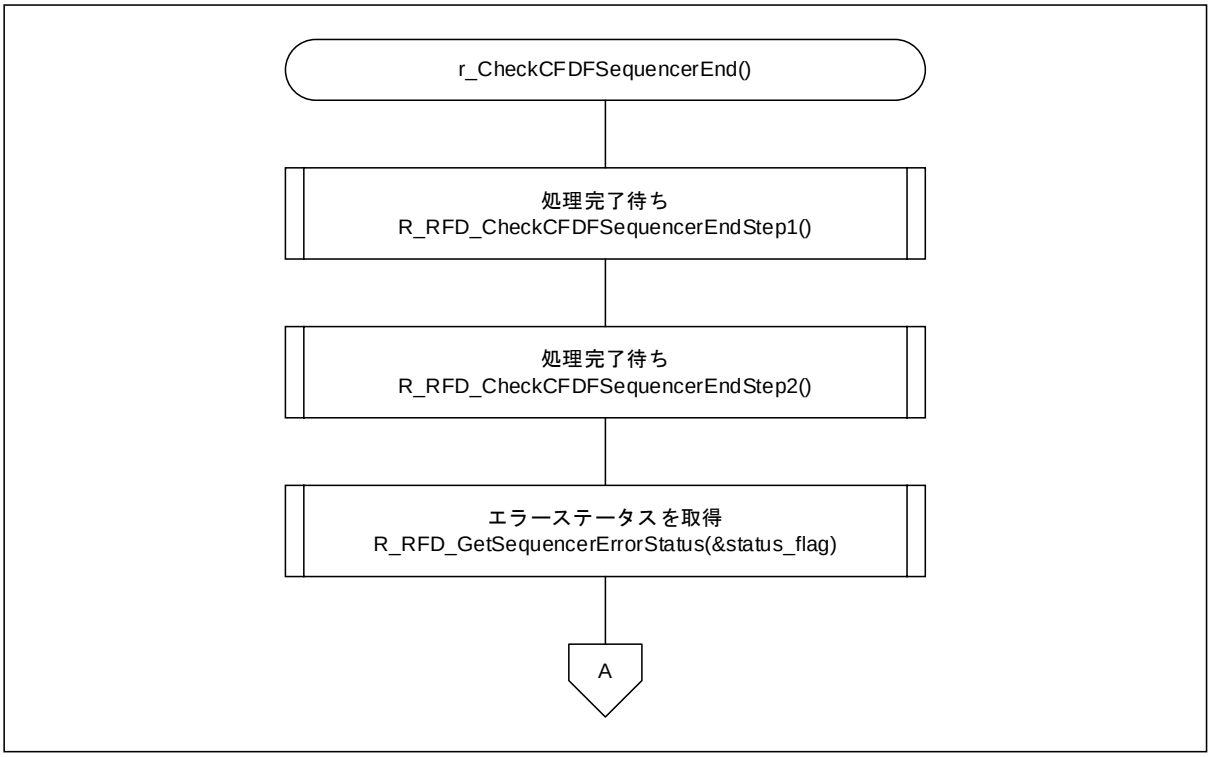
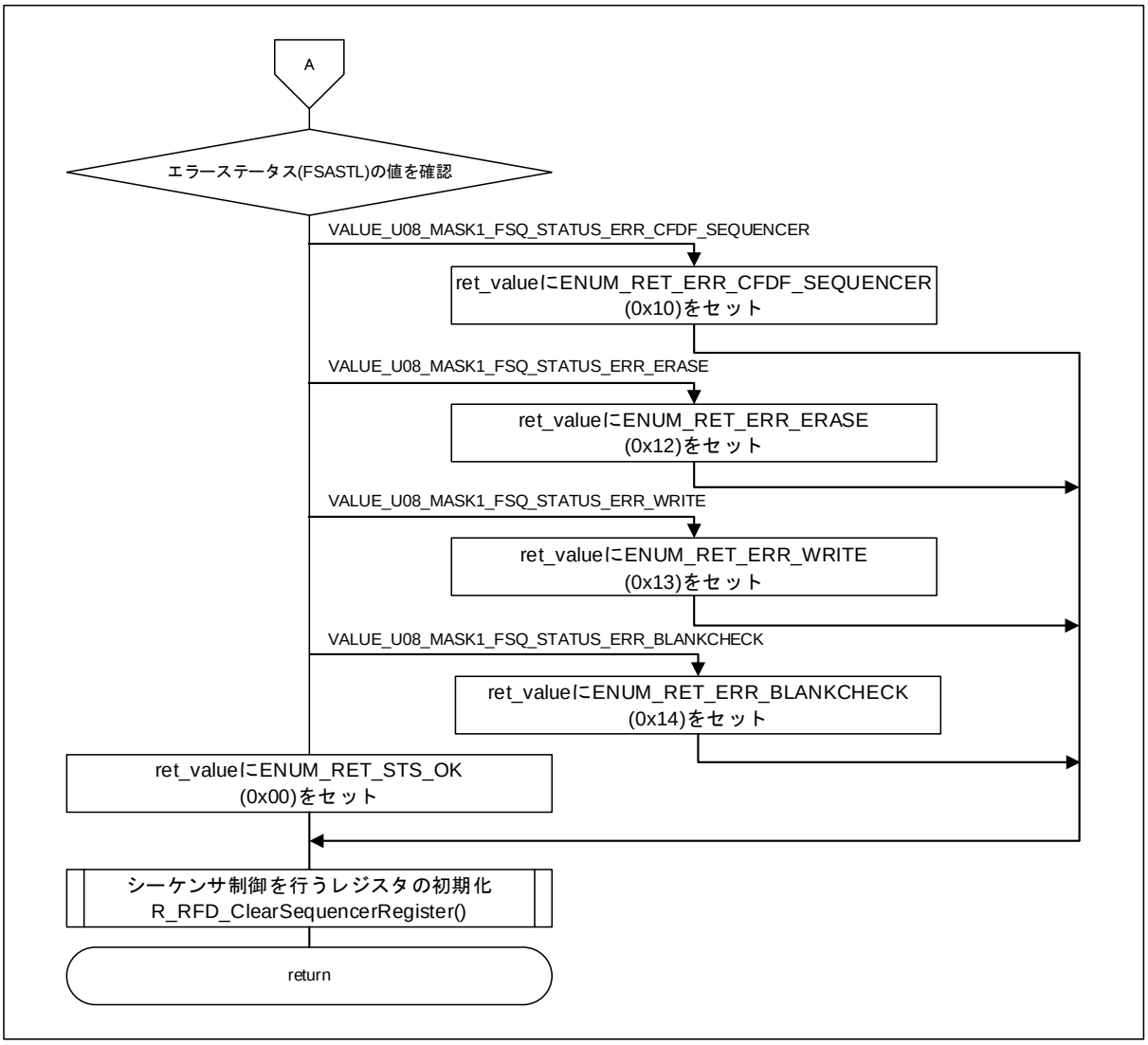


図 4-15　コード・フラッシュ・メモリ シーケンス終了処理 (2/2)



4.11.12 エクストラ・メモリ シーケンス終了処理

図 4-16、図 4-17 にエクストラ・メモリ シーケンス終了処理のフローチャートを示します。

図 4-16　エクストラ・メモリ シーケンス終了処理 (1/2)

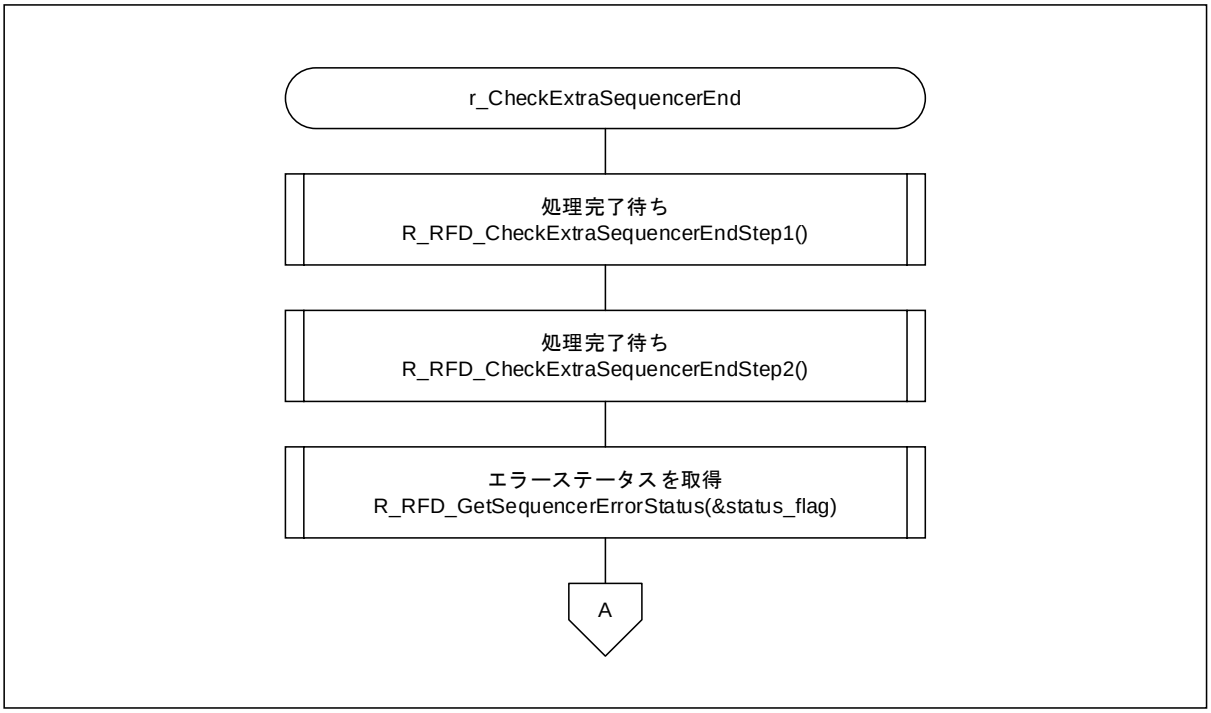
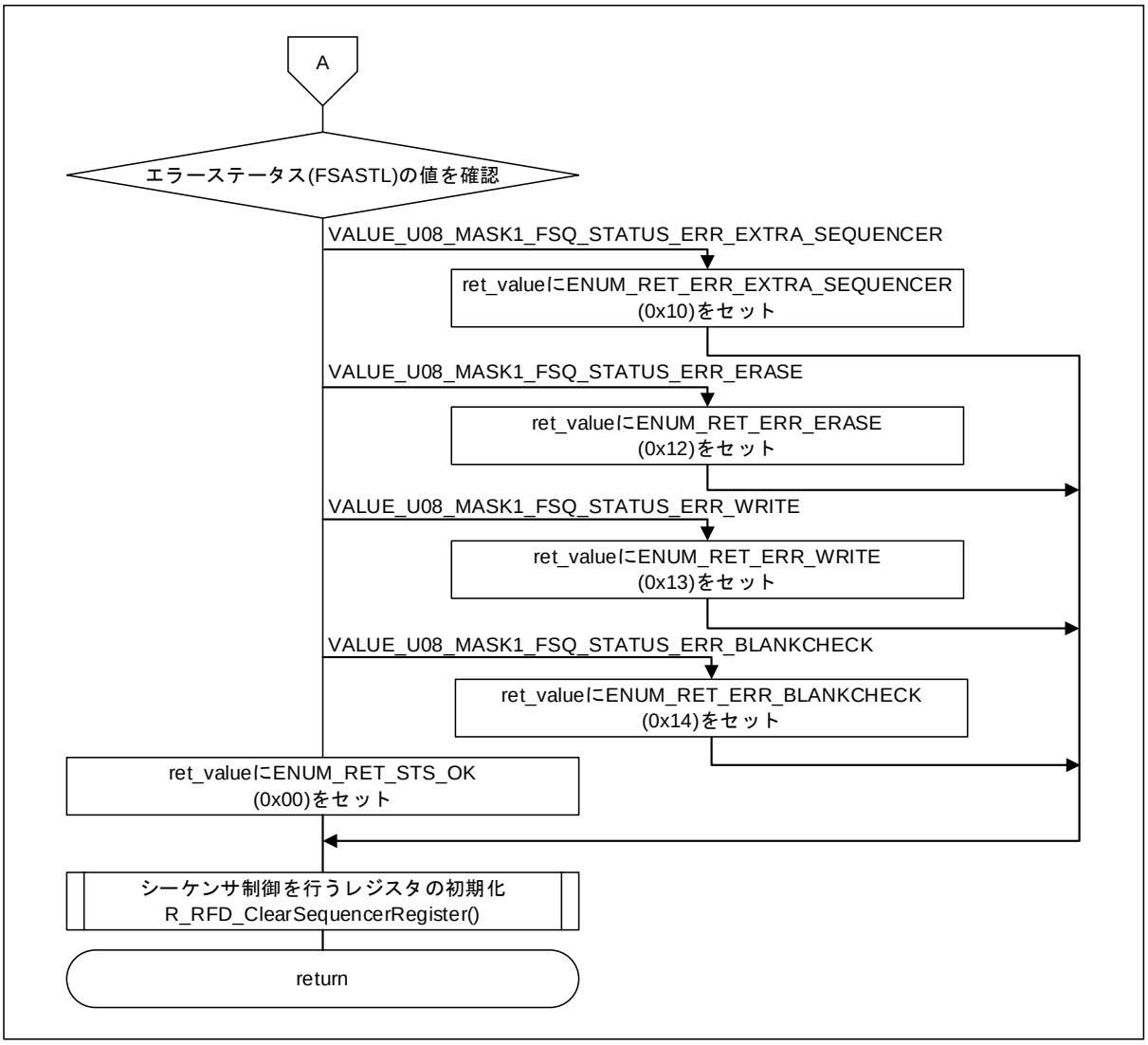


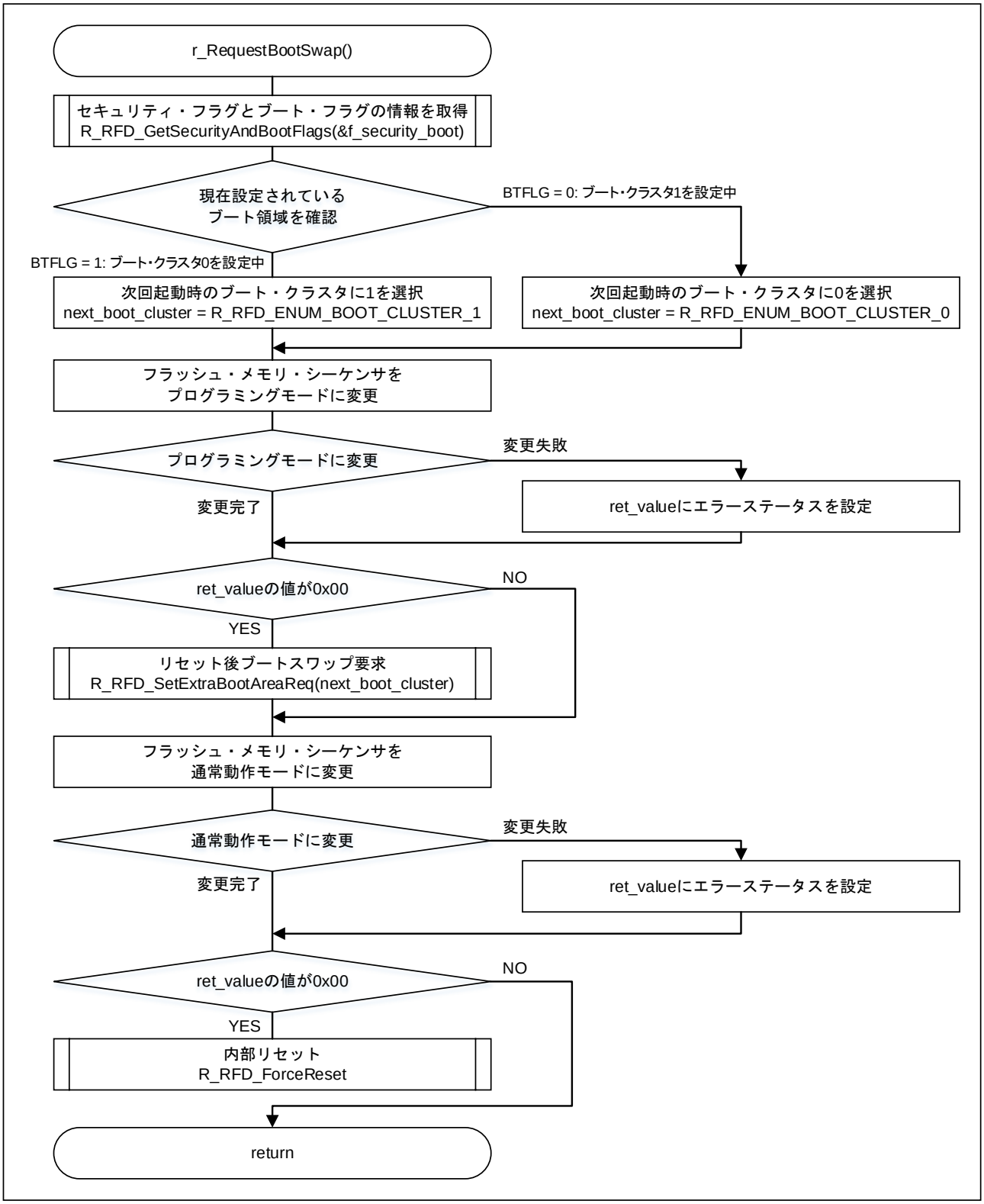
図 4-17　エクストラ・メモリ シーケンス終了処理 (2/2)



4.11.13 ブート・スワップ処理

図 4-18 にブート・スワップ処理のフローチャートを示します。

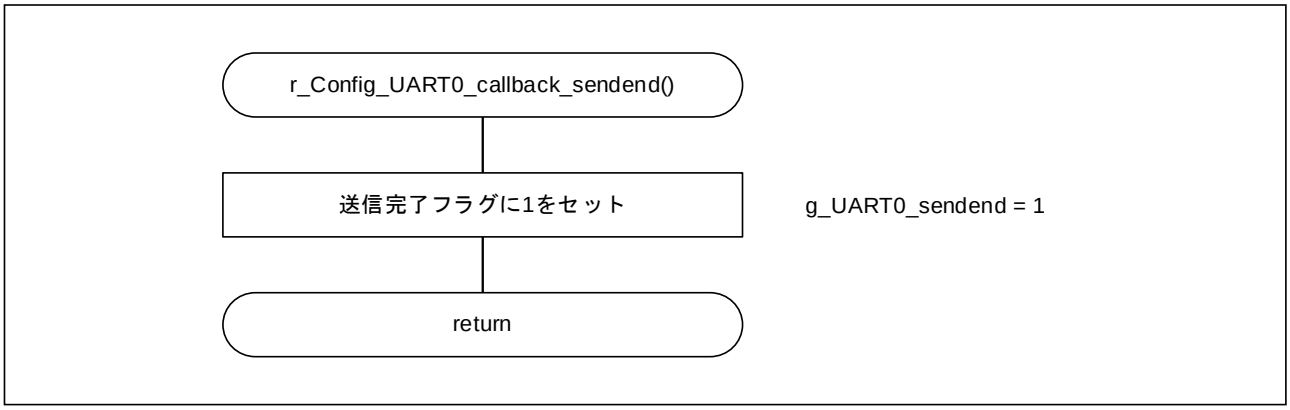
図 4-18 ブート・スワップ処理



4.11.14 UART0 送信完了割込み時の処理

図 4-19 にUART0 送信完了割込み時の処理のフローチャートを示します。

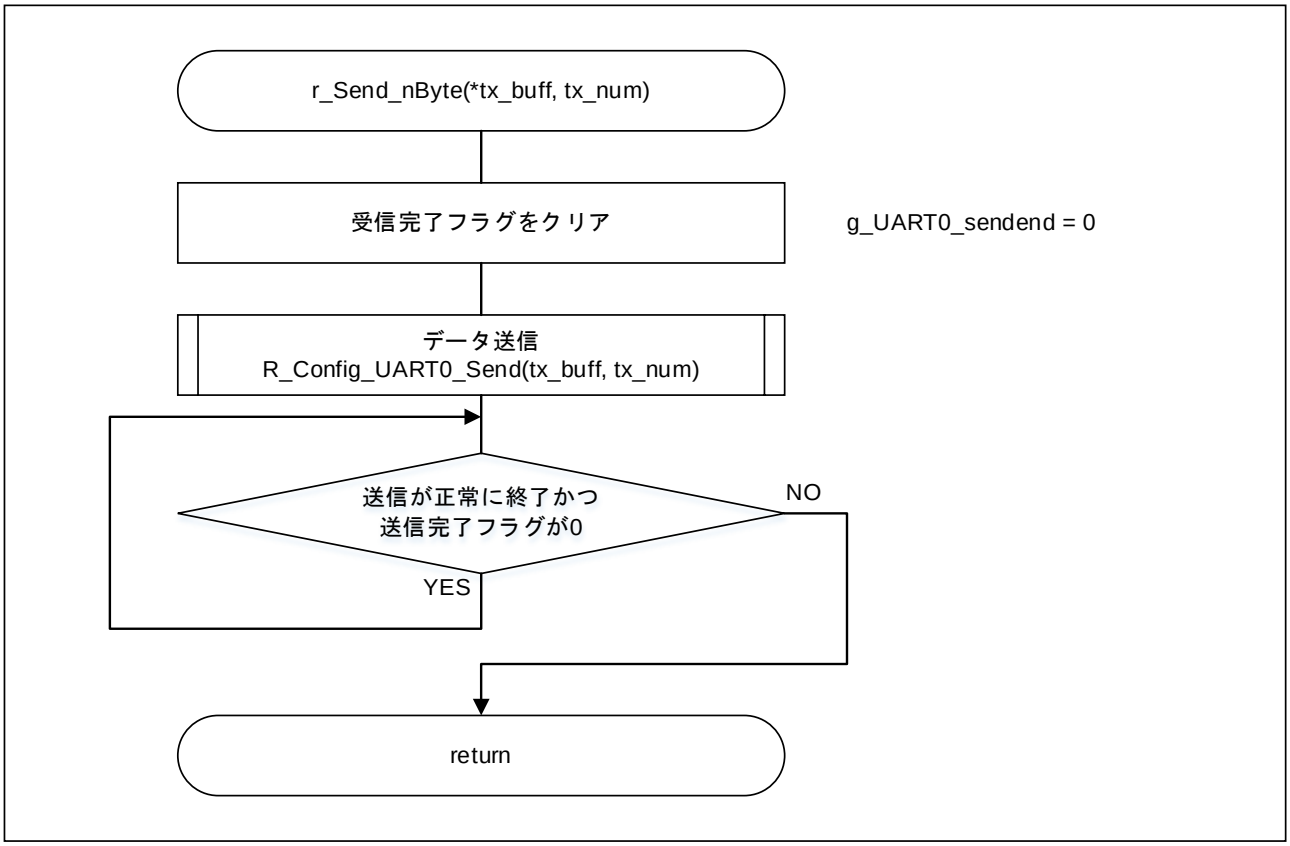
図 4-19　UART0 送信完了割込み時の処理



4.11.15 UART0 データ送信処理

図 4-20 にUART0 データ送信処理のフローチャートを示します。

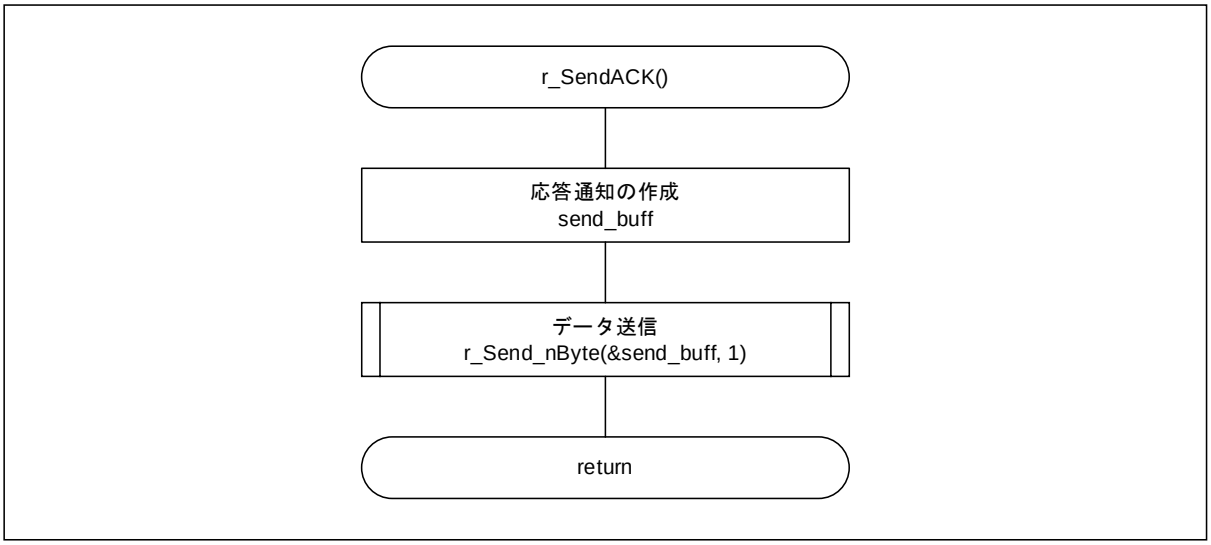
図 4-20　UART0 データ送信処理



4.11.16 UART0 正常応答送信処理

図 4-21　UART0 正常応答送信処理にUART0 正常応答送信処理のフローチャートを示します。

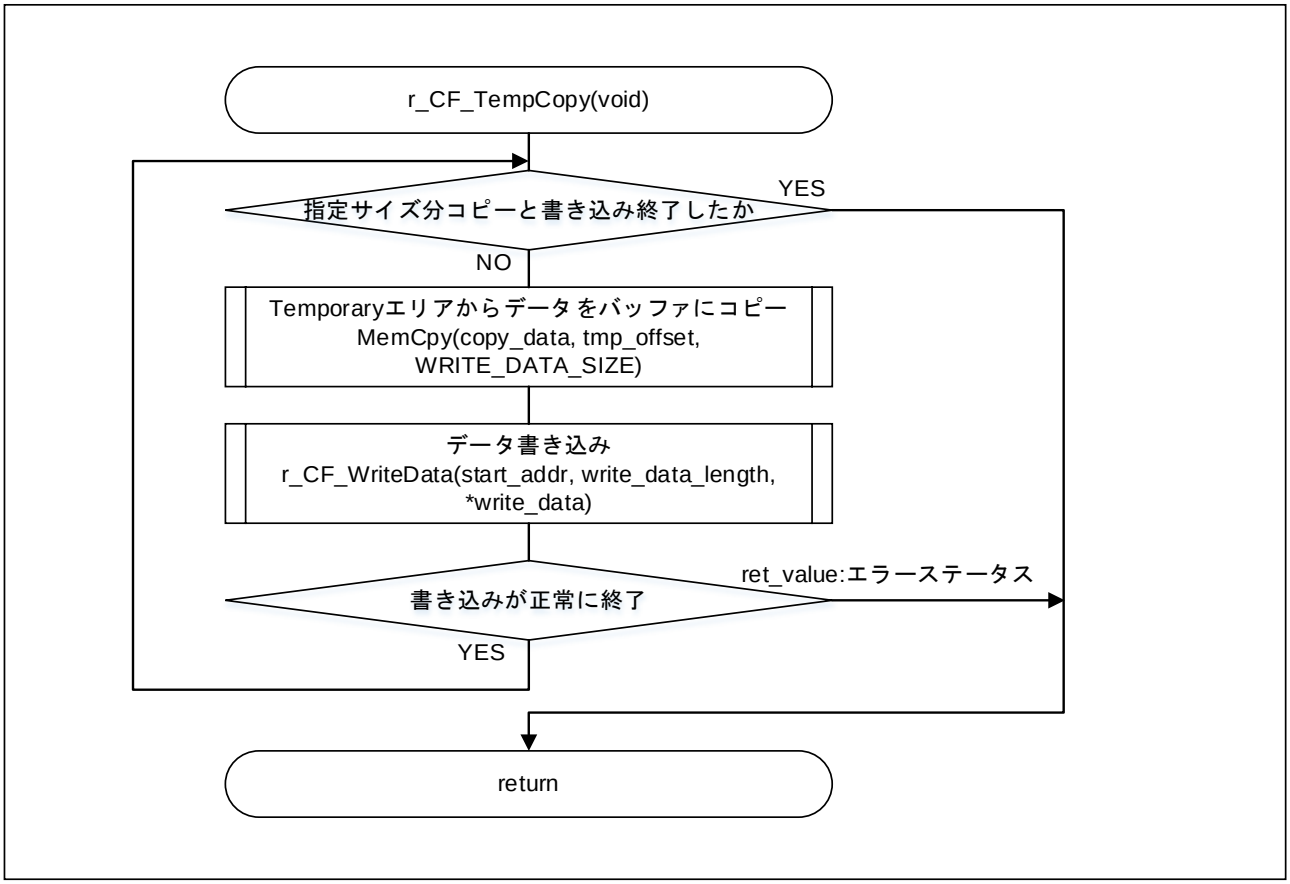
図 4-21　UART0 正常応答送信処理



4.11.17 Temporary エリアからデータコピー処理

図 4-22 に Temporary エリアからデータコピー処理のフローチャートを示します。

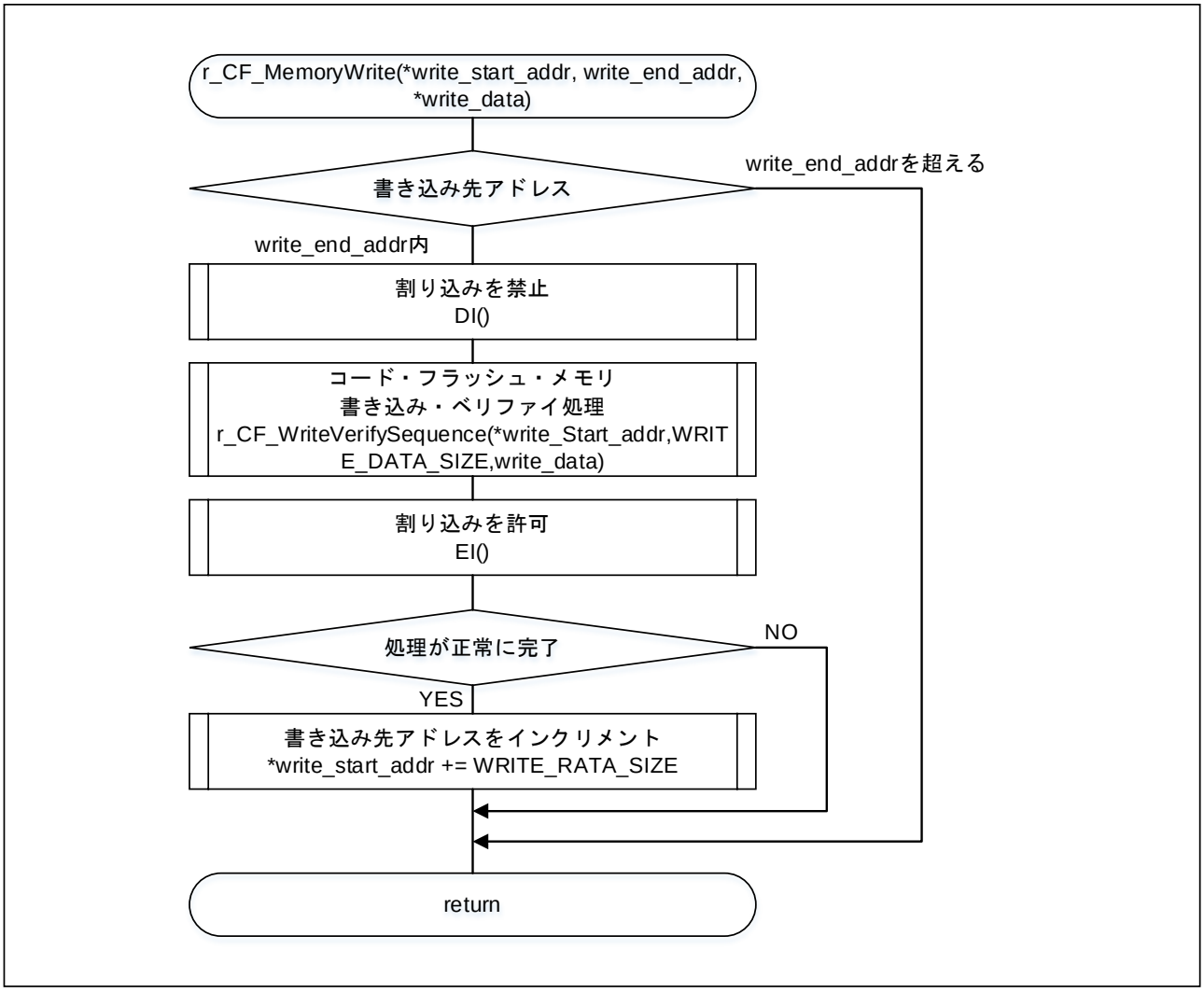
図 4-22　Temporary エリアからデータコピー処理



4.11.18 コード・フラッシュ・メモリ 書き換え処理

図 4-23 にコード・フラッシュ・メモリ 書き換え処理のフローチャートを示します。

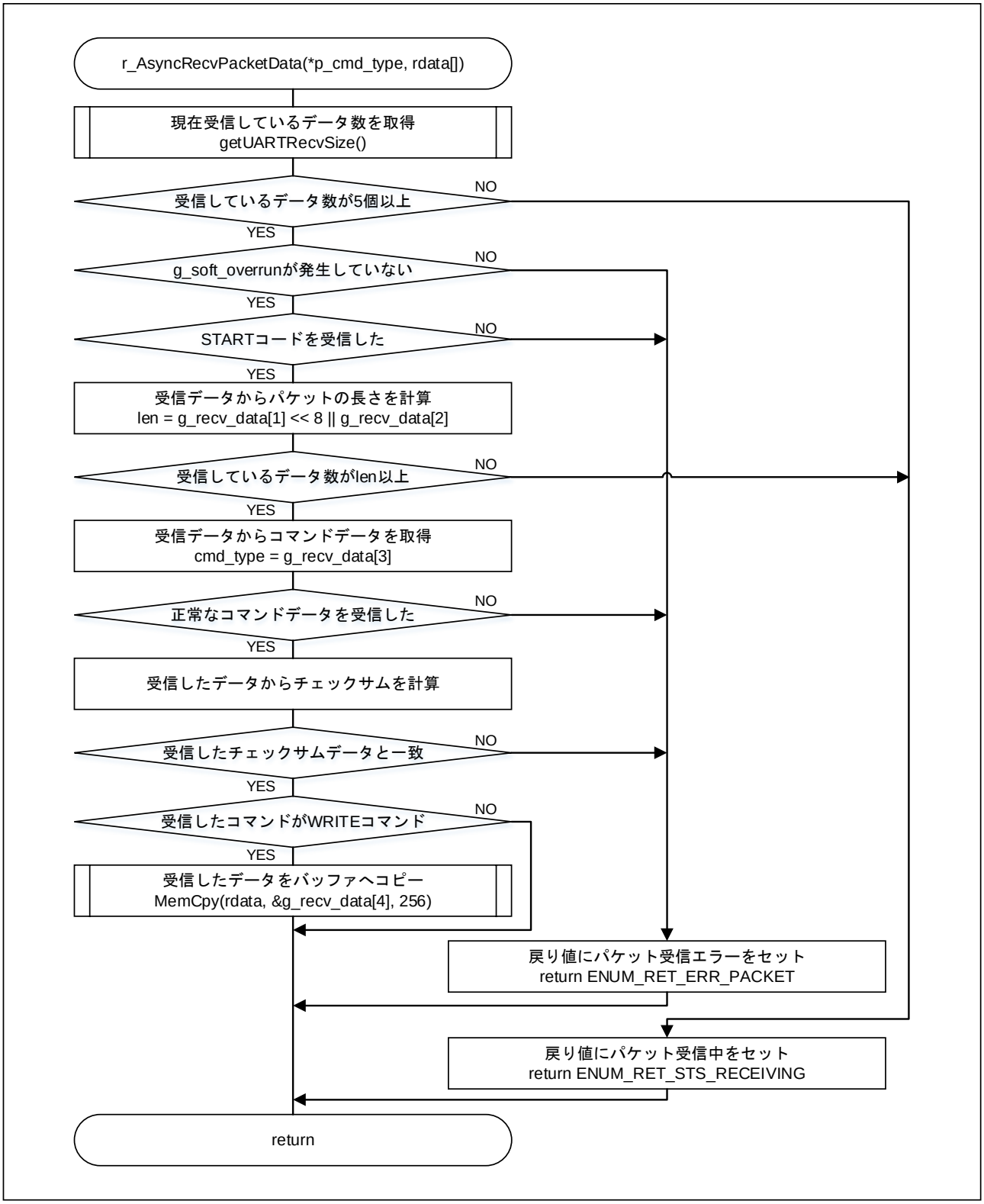
図 4-23 コード・フラッシュ・メモリ 書き換え処理



4.11.19 非同期コマンドパケット受信処理

図 4-24 に非同期コマンドパケット受信処理のフローチャートを示します。

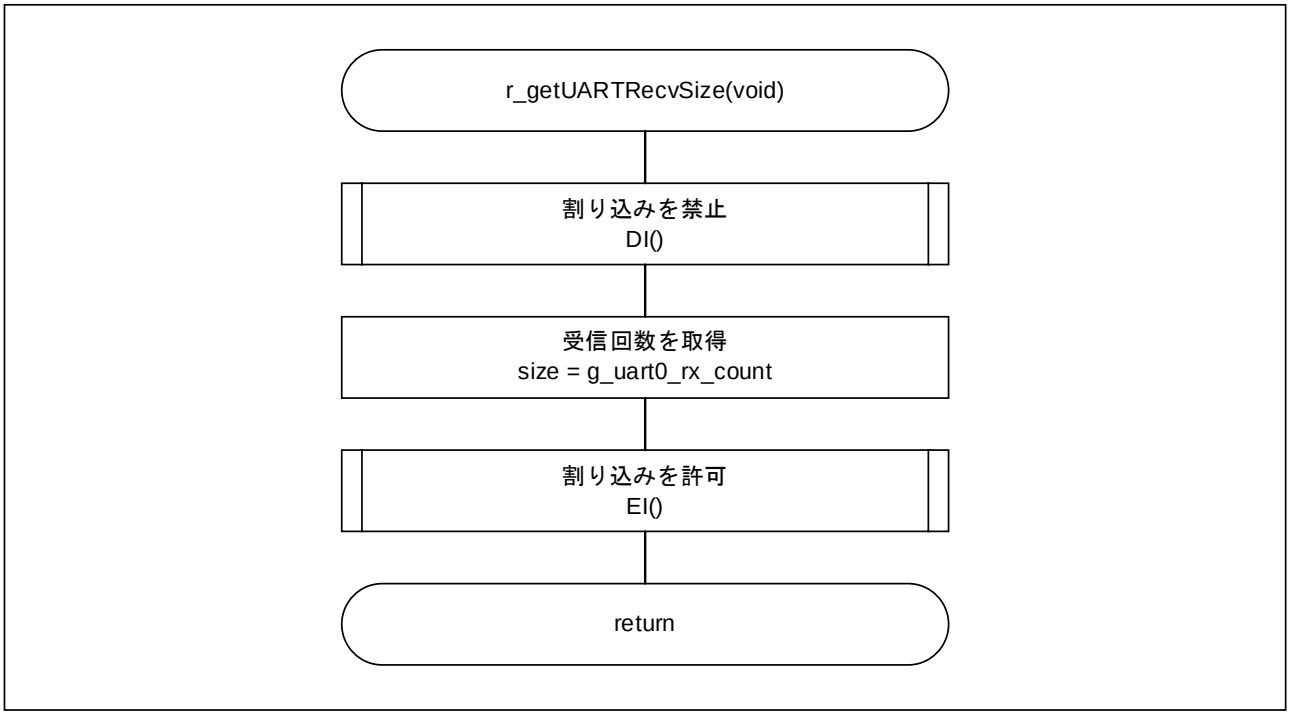
図 4-24　非同期コマンドパケット受信処理



4.11.20 受信データサイズ取得処理

図 4-25 に受信データサイズ取得処理のフローチャートを示します。

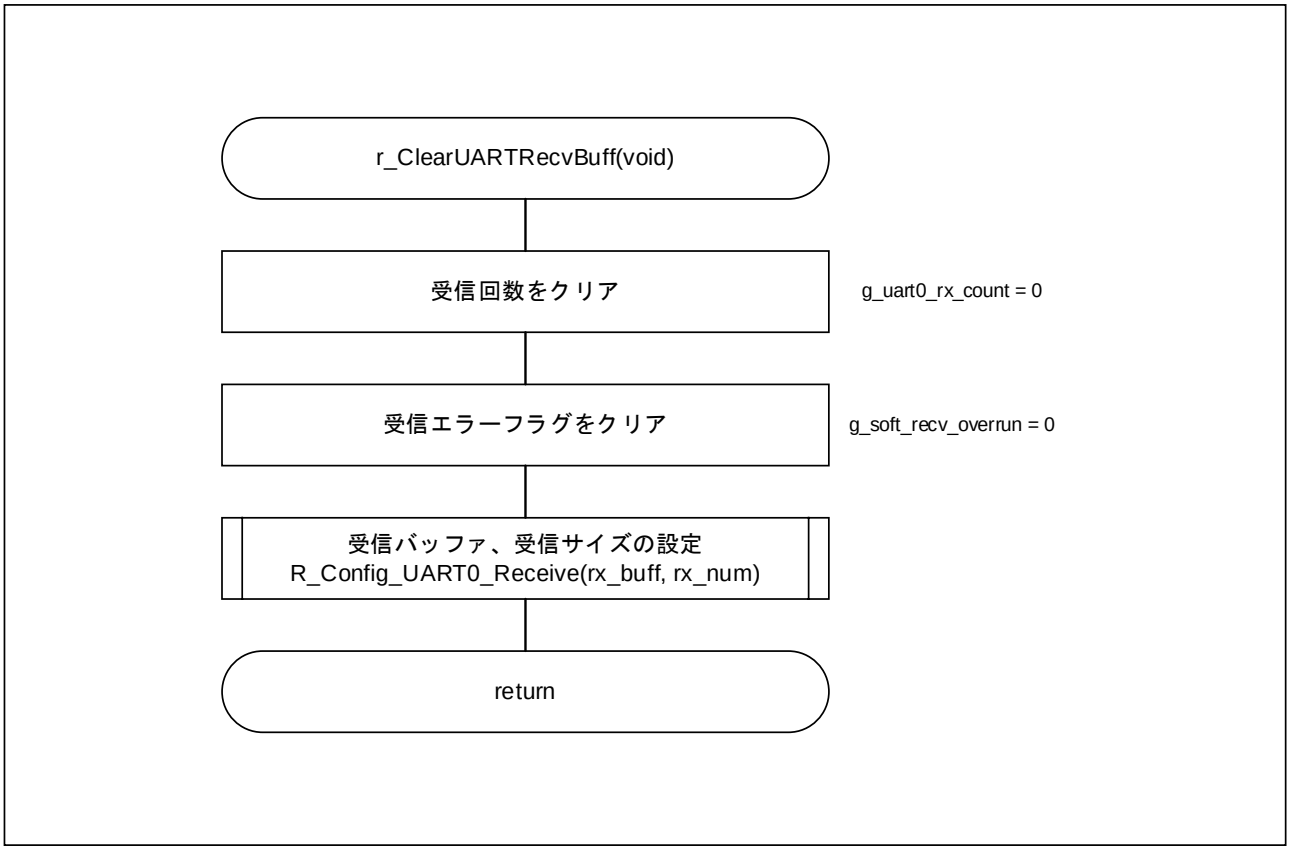
図 4-25　受信データサイズ取得処理



4.11.21 受信バッファクリア処理

図 4-26 に受信バッファクリア処理のフローチャートを示します。

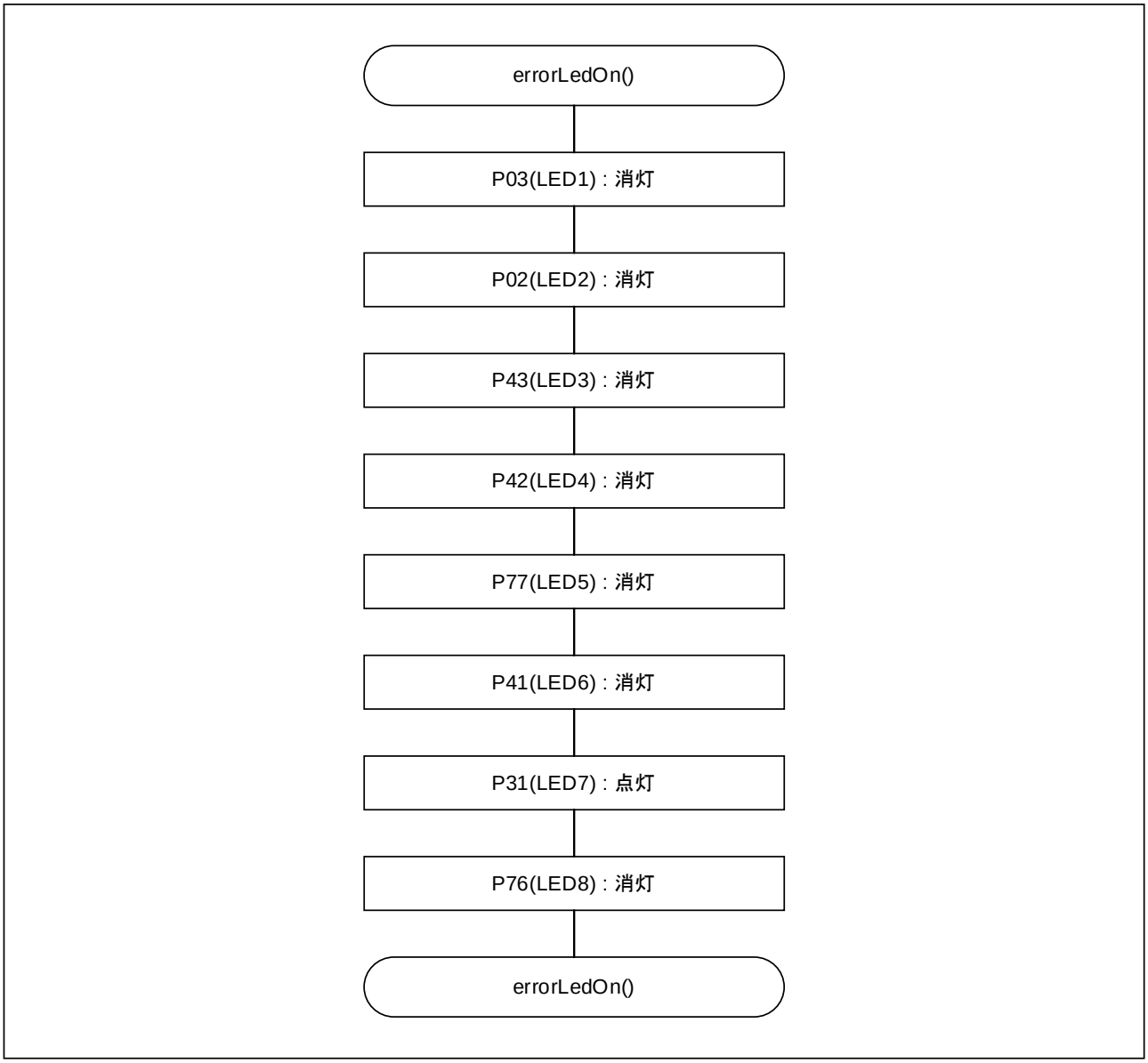
図 4-26　受信バッファクリア処理



4.11.22 エラーLED 点灯処理

図 4-27 にエラーLED 点灯処理のフローチャートを示します。

図 4-27　エラーLED 点灯処理



5. 書き込み用 GUI

この GUI は exe ファイルを実行するだけでデバイスに書き込みができます。書き込むファイルはバイナリ・ファイル(.bin)を選択してください。再度書き込む場合は、一度 GUI を再起動してから書き込みを行ってください。

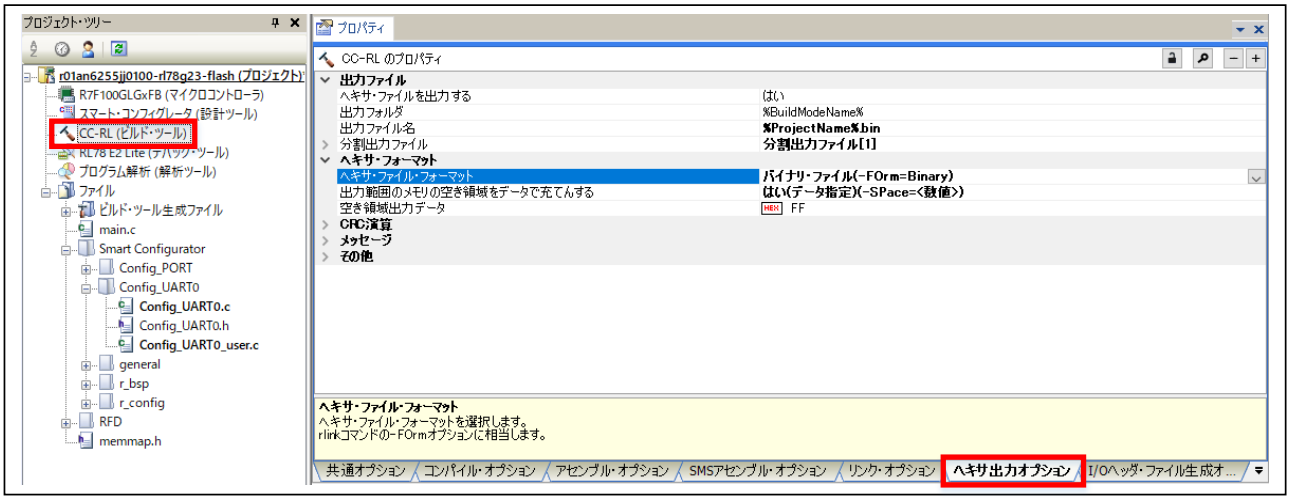
5.1 書き込みに必要なファイルの生成方法

この GUI で書き込むバイナリ・ファイル(.bin)は以下の手順で生成してから書き込みを行ってください。

5.1.1 CS+での生成方法

プロジェクトツリーにある「CC-RL (ビルドツール) 」を選択し、「ヘキサ出力オプション」のタブを開きます。

図 5-1 CS+での生成方法(1/6)



「ヘキサ出力オプション」タブにある「出力ファイル」の「ヘキサ・ファイルを出力する」を「はい」に設定します。

「分割出力ファイル」を選択し、ダイアログに以下のテキストを記述します。

「XXXXXXXXXX.bin=0-YYYYYYYYY」

XXXXXXXXXX にはプロジェクト名、YYYYYYYYY には使用するデバイスのコード・フラッシュ・メモリの最終アドレスを記述してください。

図 5-2　CS+での生成方法(2/6)

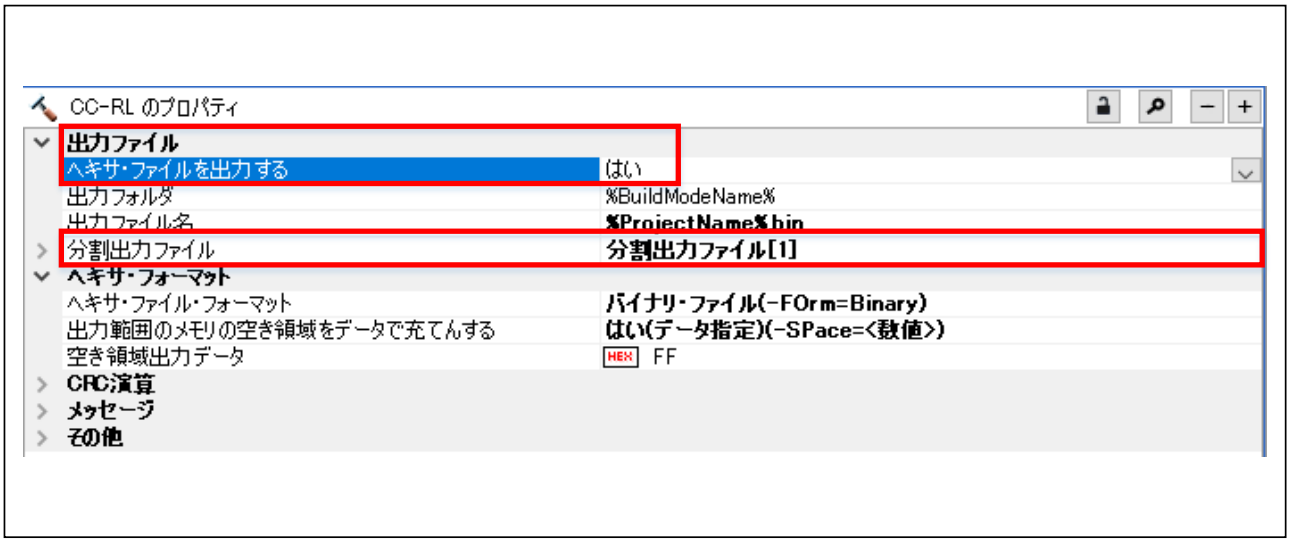
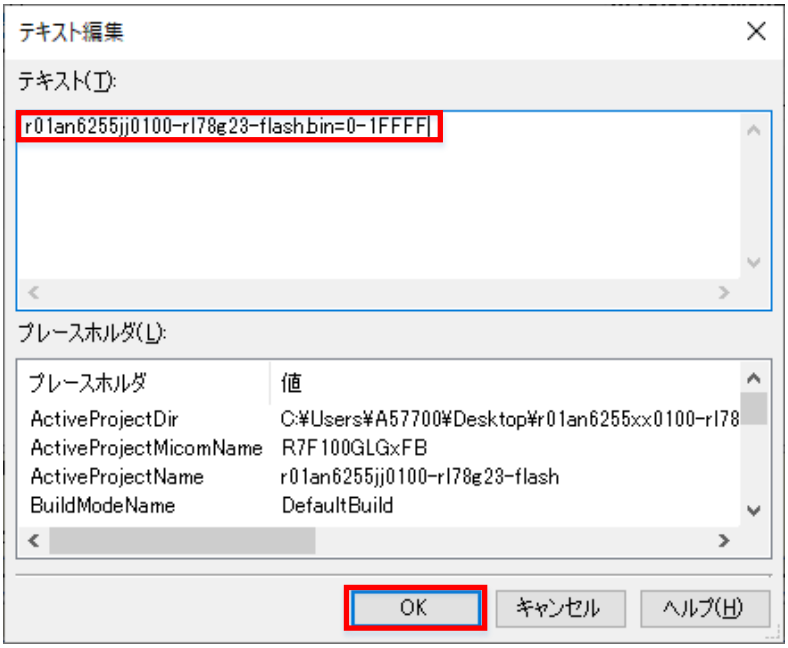
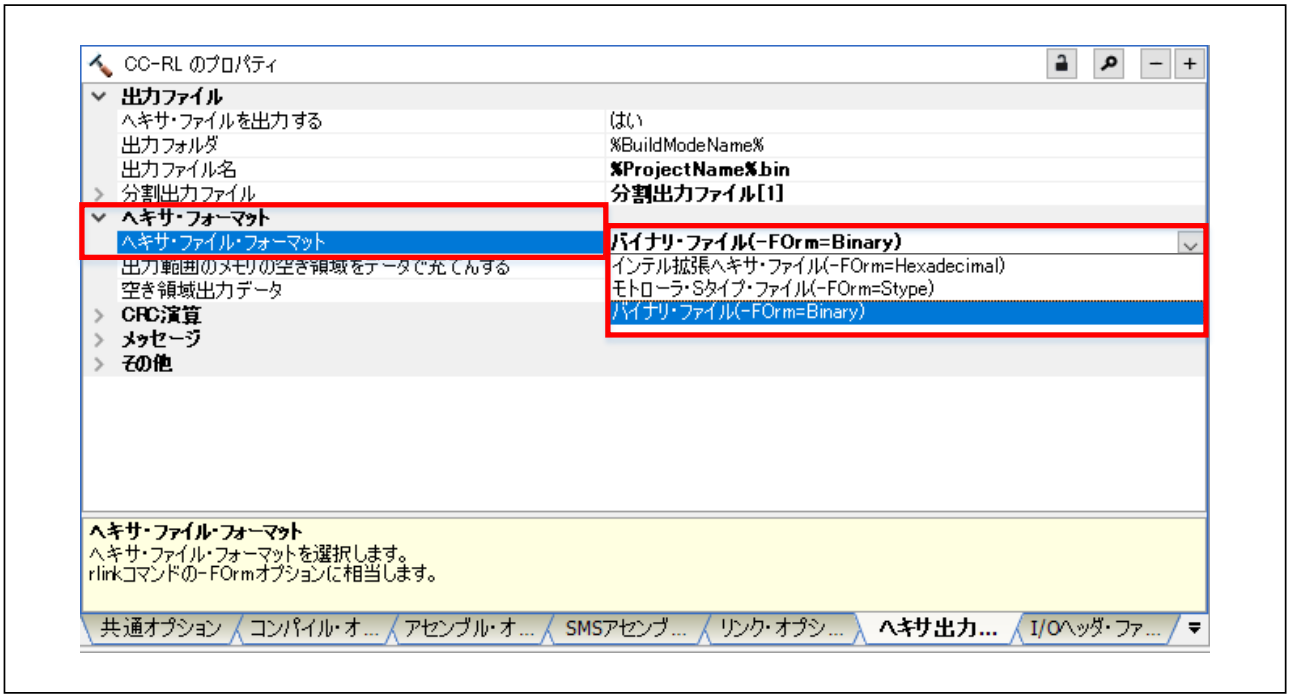


図 5-3　CS+での生成方法(3/6)



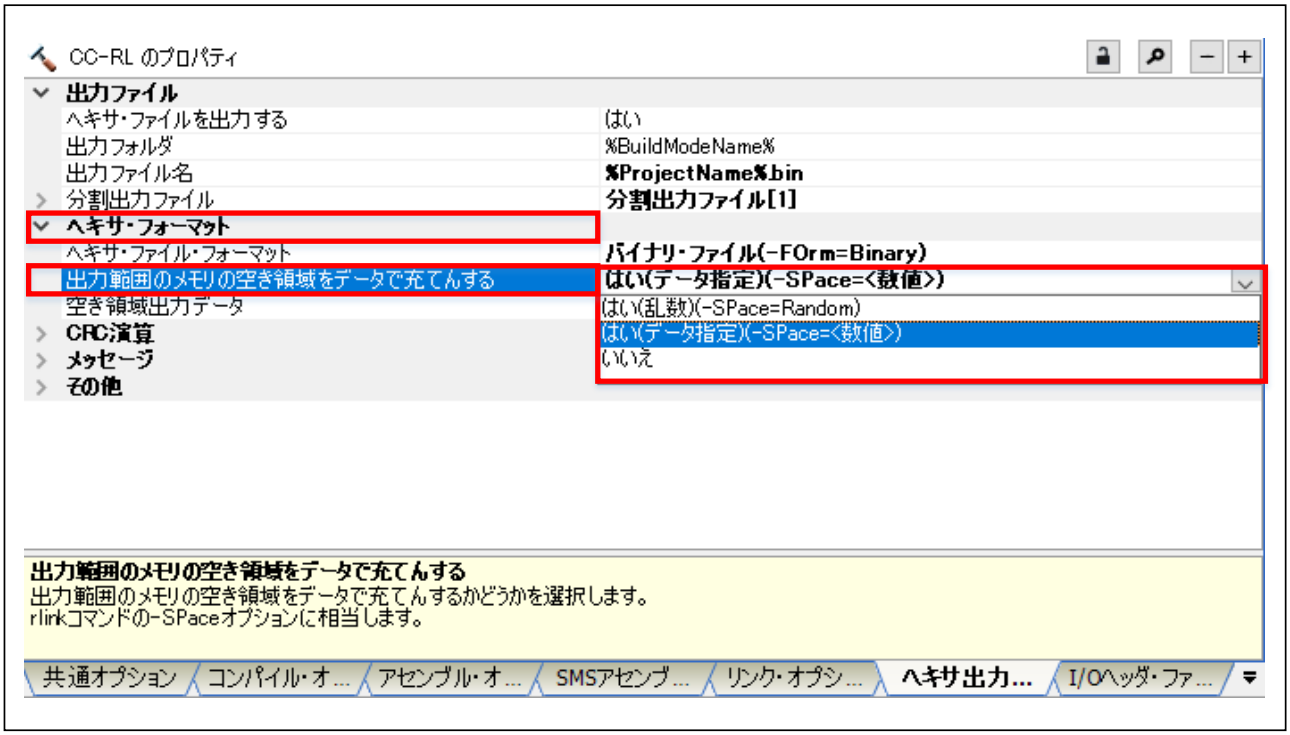
「ヘキサ出力オプション」タブにある「ヘキサ・フォーマット」の「ヘキサ・ファイル・フォーマット」を「バイナリ・ファイル (-FOrm=Binary) 」に設定します。

図 5-4　CS+での生成方法(4/6)



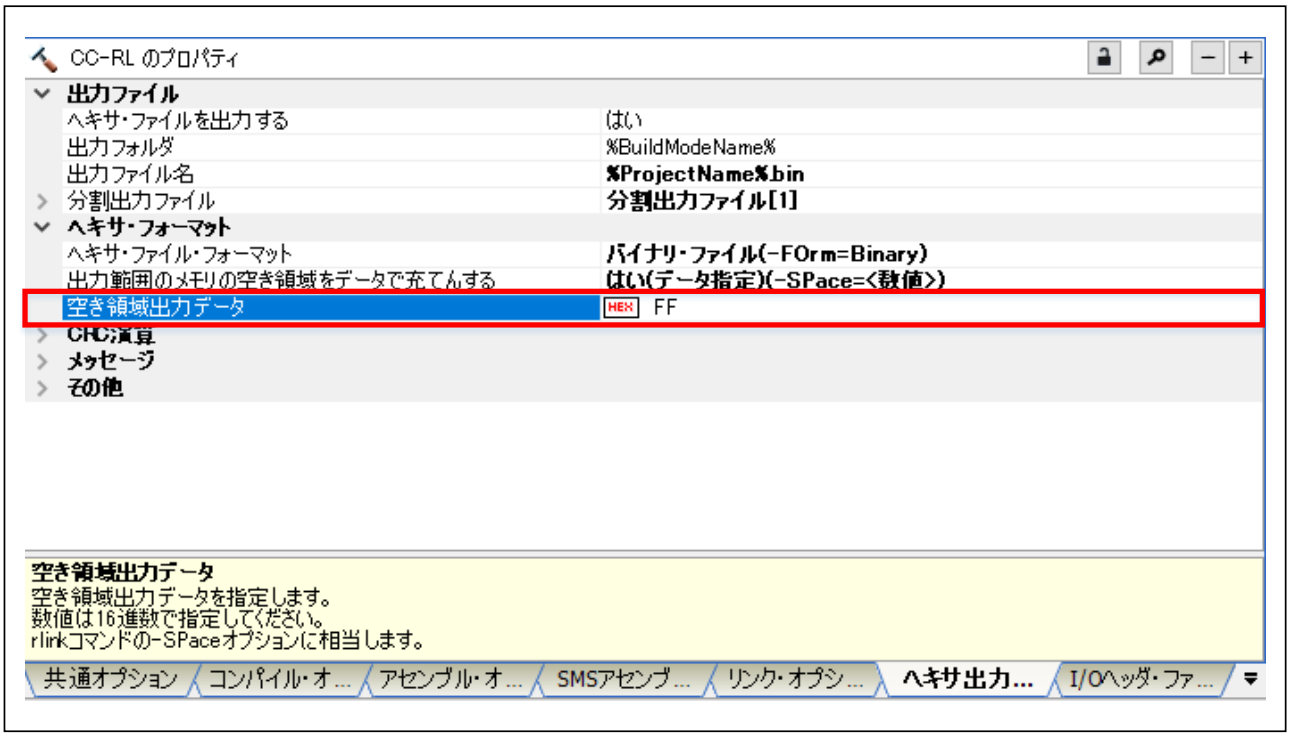
「ヘキサ出力オプション」タブにある「ヘキサ・フォーマット」の「出力範囲のメモリの空き領域をデータで充てんする」を「はい (データ指定) (-SPace=<数値>) 」に設定します。

図 5-5　CS+での生成方法(5/6)



「ヘキサ出力オプション」タブにある「ヘキサ・フォーマット」の「空き領域データ」を「FF」に設定します。

図 5-6　CS+での生成方法(6/6)



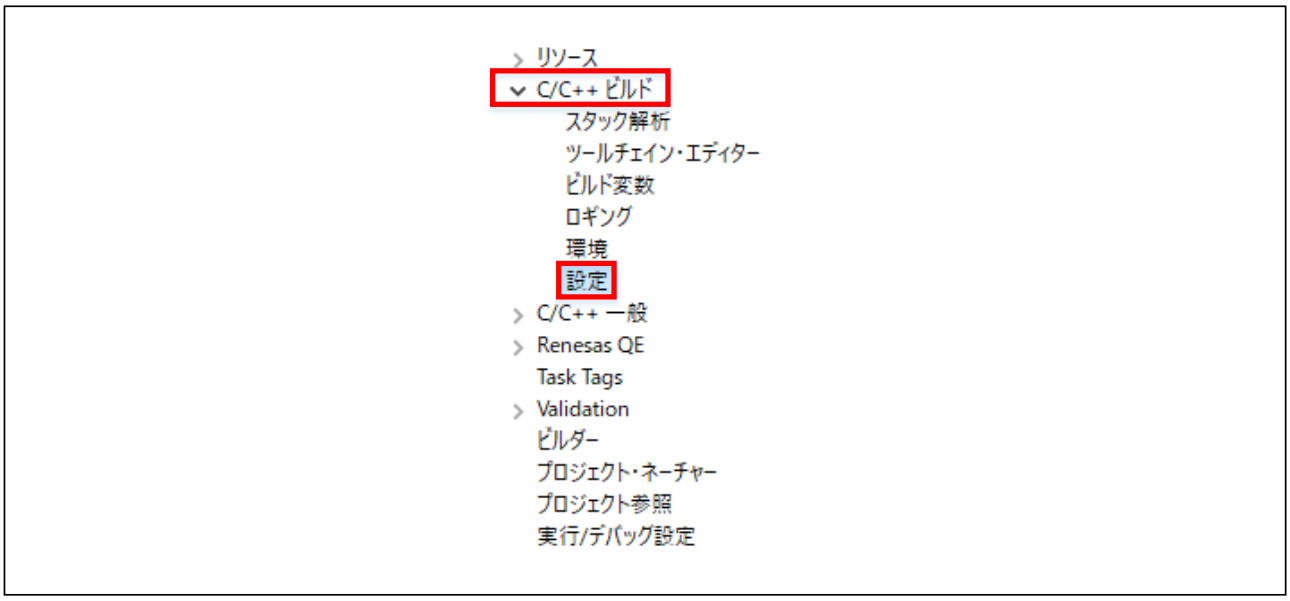
その後、ビルドを行うとバイナリ・ファイルが生成されます。

5.1.2　e2studio での生成方法

「プロジェクト」タブから「プロパティ」を選択します。

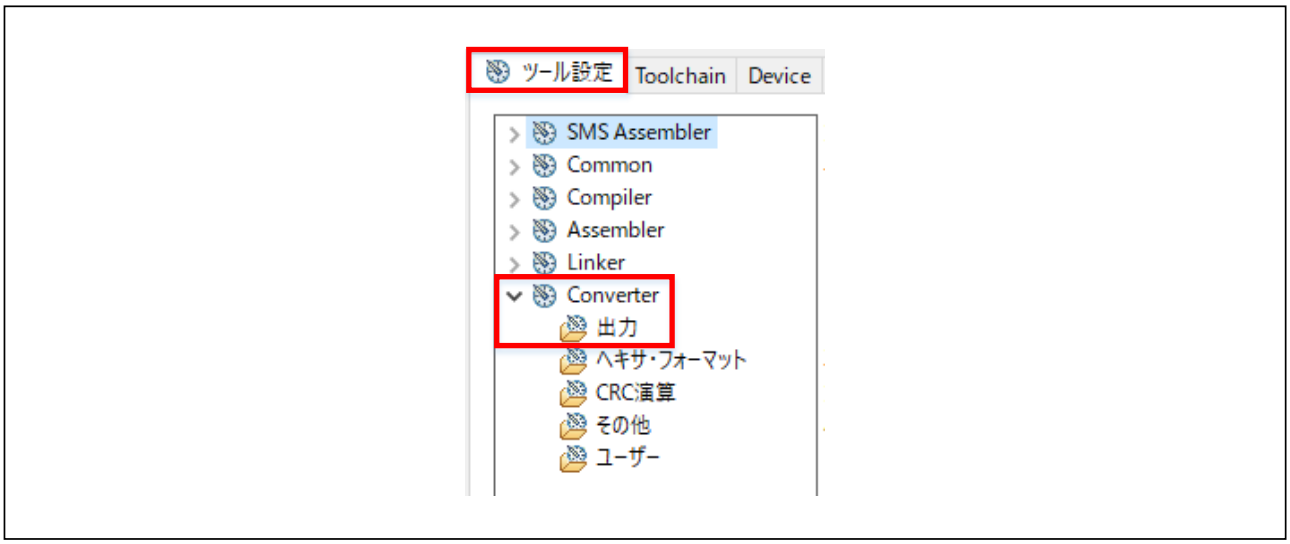
「C/C++ビルド」の「設定」を選択します。

図 5-7　e2studio での生成方法(1/3)



「ツール設定」のタブの「Converter」、「出力」を選択します。

図 5-8　e2studio での生成方法(2/3)



「ロード・モジュール・コンバータを実行する」を選択します。

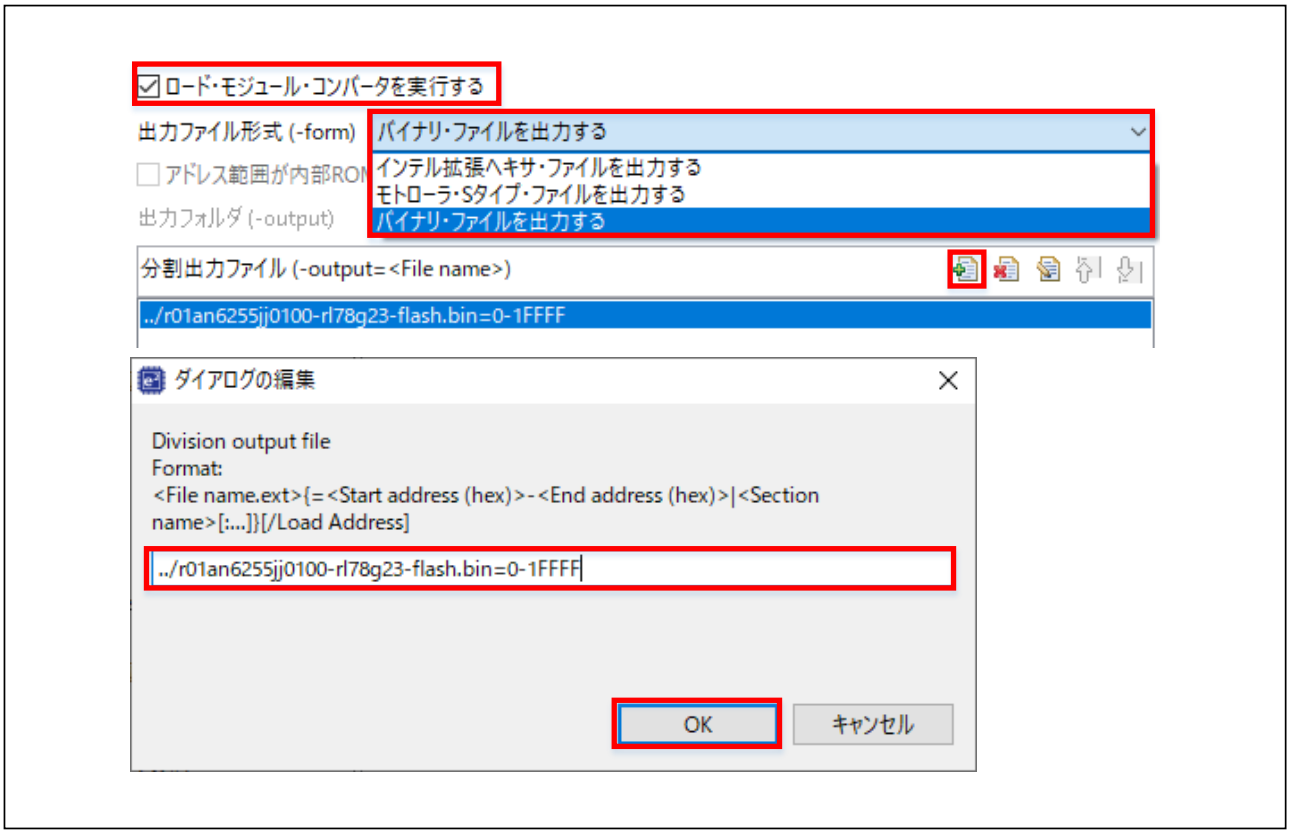
出力ファイル形式を「バイナリ・ファイルを出力する」に選択します。

「追加」ボタンを押して以下のテキストを記述します。

「../XXXXXX.bin=0-YYYYYY」

XXXXXX にはプロジェクト名、YYYYYY には使用するデバイスのコード・フラッシュ・メモリの最終アドレスを記述してください。

図 5-9　e2studio での生成方法(3/3)



その後、ビルドを行うとバイナリ・ファイルが生成されます。

5.1.3　IAR EW での生成方法

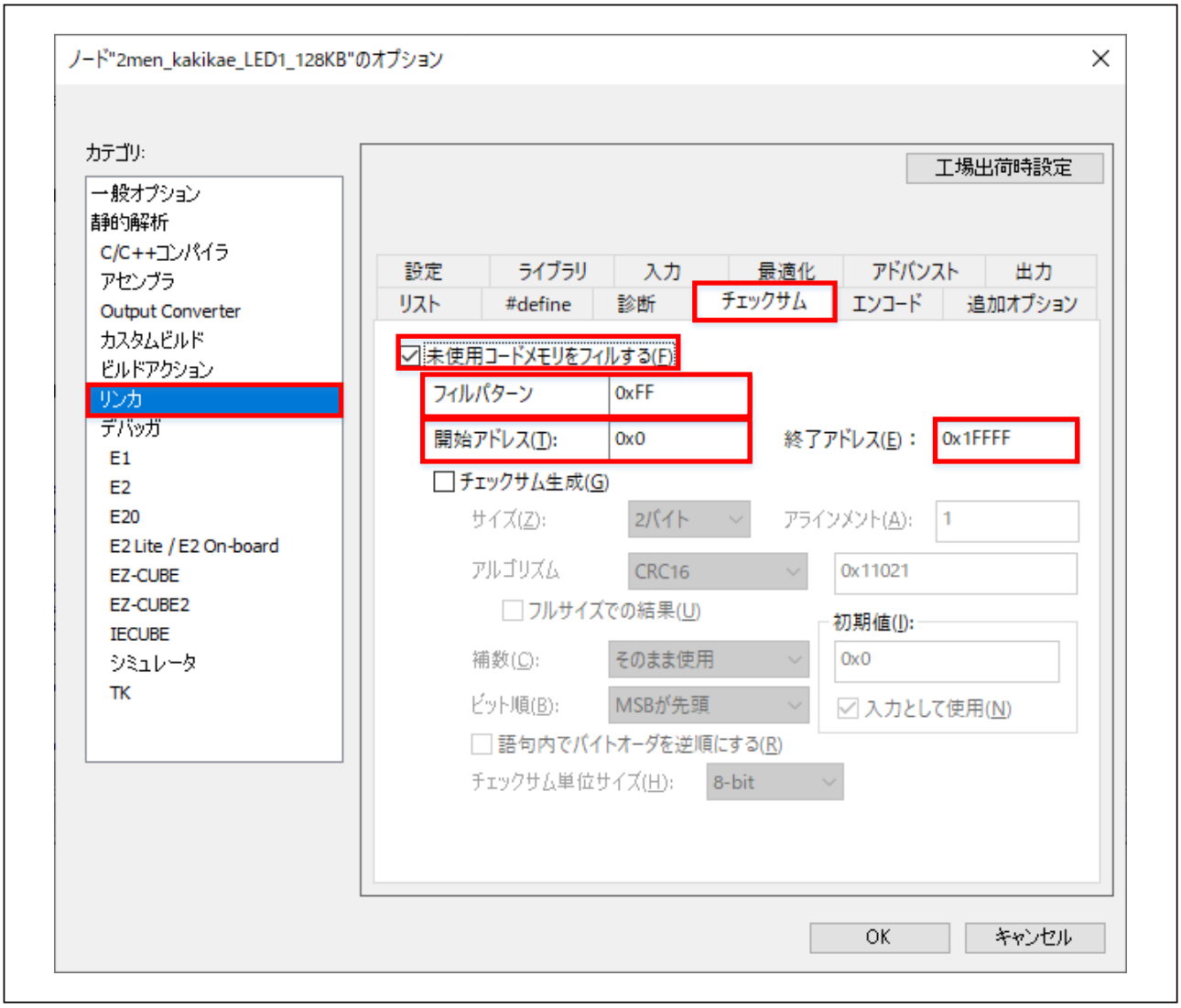
「プロジェクト」タブの「オプション」を選択します。

カテゴリの「リンカ」から「チェックサム」を選択します。

「未使用コードメモリをフィルする」にチェックを付けます。

フィルパターンに「0xFF」、開始アドレスに「0x0」、最終アドレスに「使用するデバイスのコード・フラッシュ・メモリの最終アドレス」を 0x を付けた 16 進数で記述してください。

図 5-10　IAR EW での生成方法(1/2)

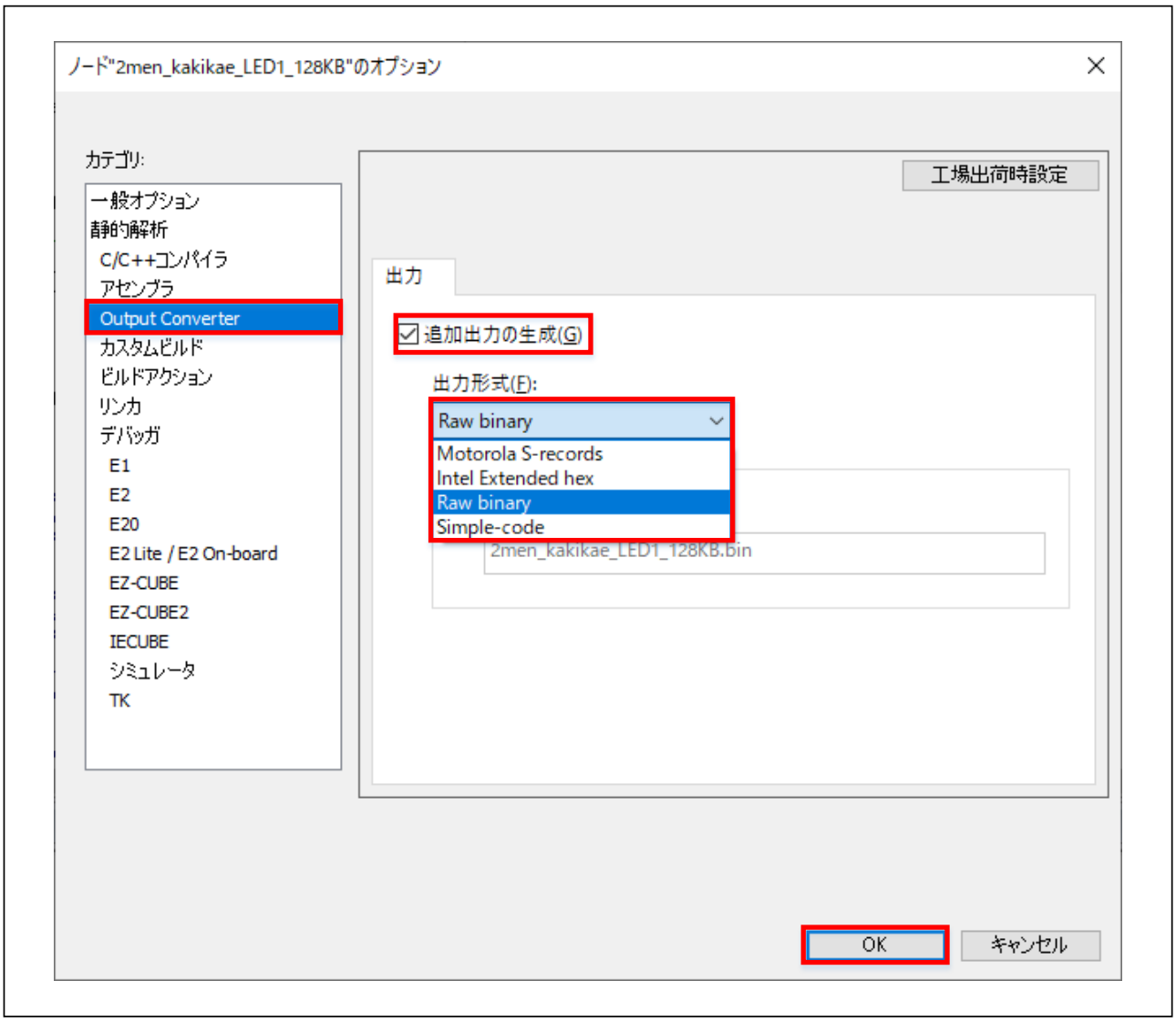


「Output Converter」タブの「追加出力の生成」にチェックを付けます。

出力形式に「Raw binary」を選択します。

その後「OK」を選択し、ビルドを行うとバイナリ・ファイルが生成されます。

図 5-11　IAR EW での生成方法(2/2)

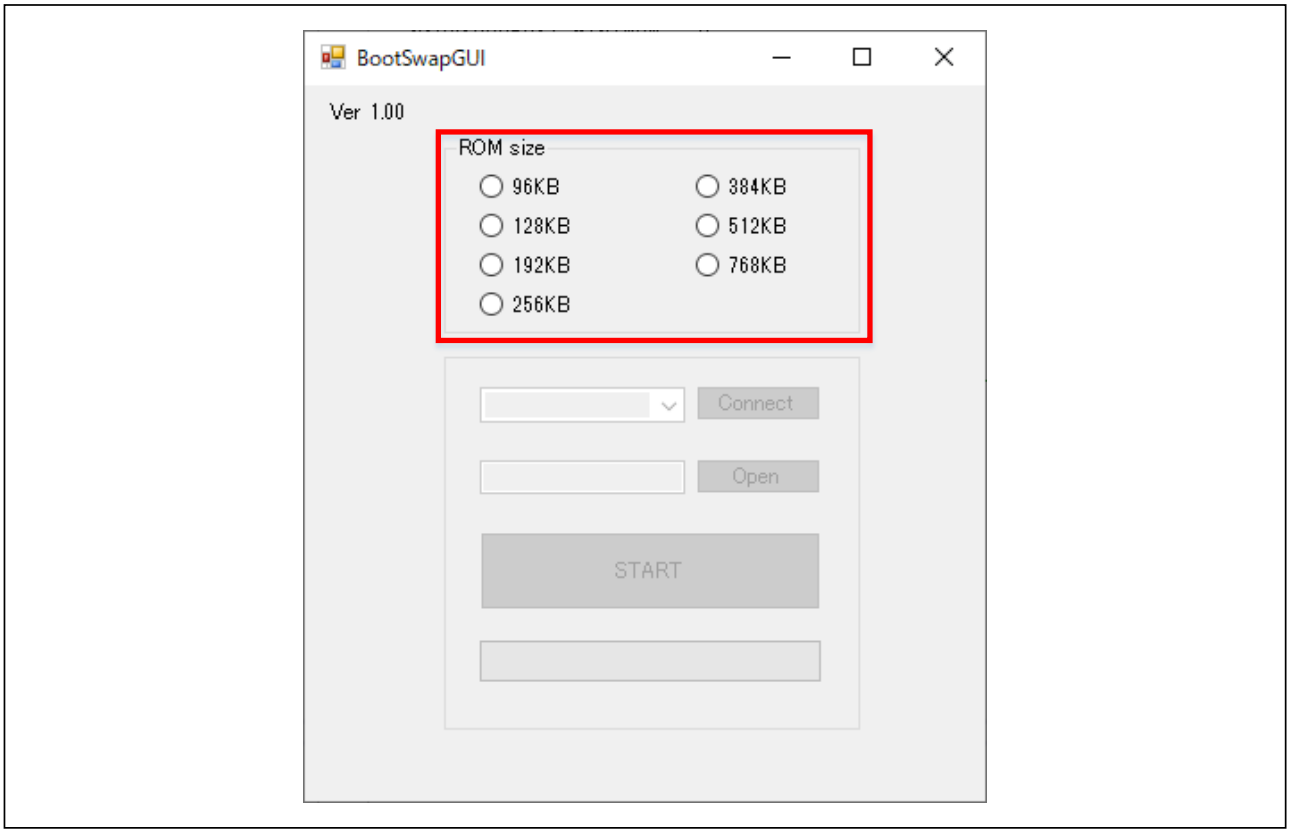


5.2　GUI 操作手順

BootSwapGUI.exe を起動します。

使用するデバイスの ROM サイズをラジオボタンから選択します。

図 5-12　GUI 使用方法(1/2)



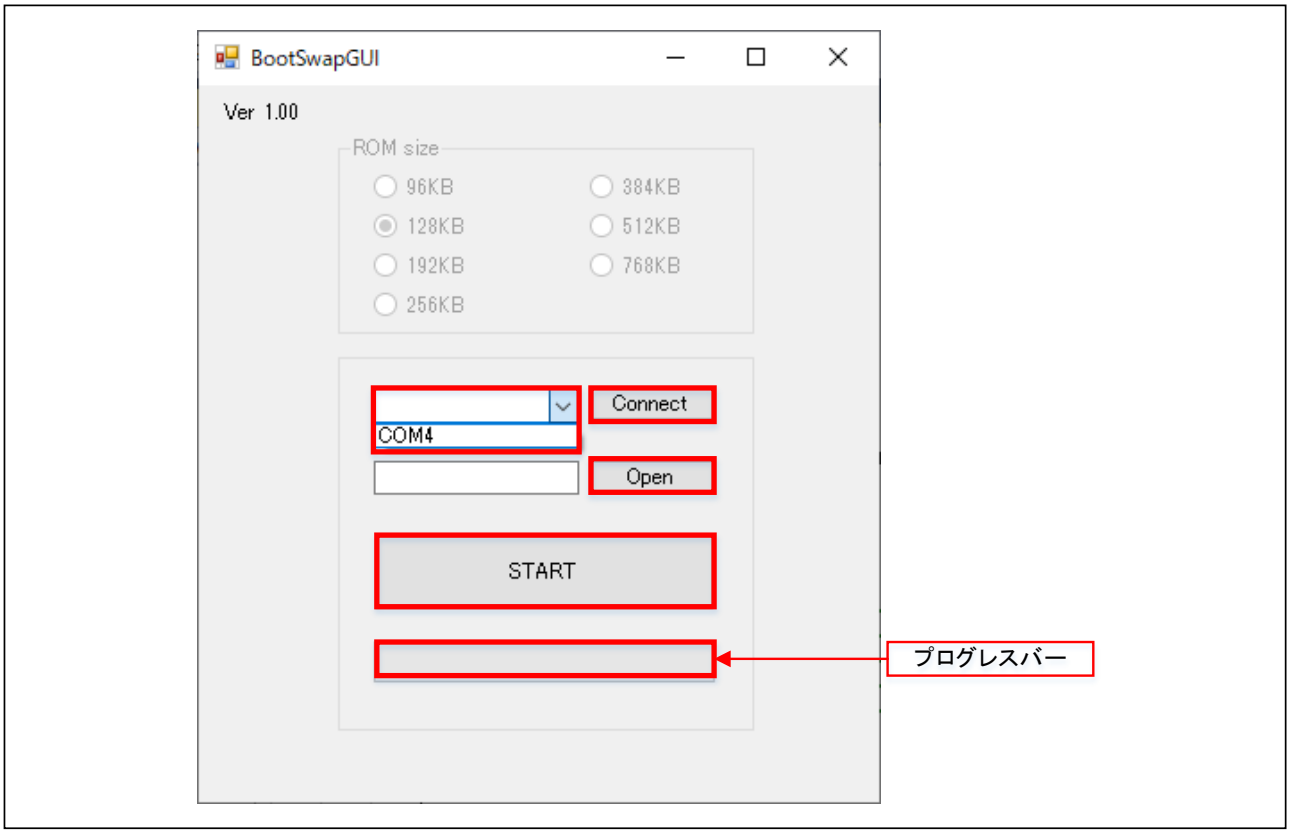
ドロップダウンリストから有効な COM ポートを選択し、「 Connect 」を押して接続します。

「 Open 」を押して書き込み用のプログラム(.bin)を選択します。

「 START 」を押して書き込みを開始します。

プログレスバーに書き込みの進捗が表示されます。

図 5-13　GUI 使用方法(2/2)



書き込み完了後、BootSwapGUI.exe を終了してください。

6. サンプル・プログラムの入手方法

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。

7. 参考ドキュメント

RL78/G23 ユーザーズマニュアルハードウェア編 (R01UH0896J)

RL78 ファミリユーザーズマニュアルソフトウェア編 (R01US0015J)

（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

（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。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.08.04	—	初版発行

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、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想定していません。たとえ、当社が想定してい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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、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。
10. 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。
11. 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。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。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。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。