RL78/G23

UART 通信によるブート・スワップを使用したセルフ・プログラミング

要旨

本アプリケーションノートは、UART 通信を使用したセルフ・プログラミングの概要を説明します。

フラッシュ・セルフ・プログラミング・コード (Renesas Flash Driver RL78 Type01) を使用し、フラッシュ・メモリのブート領域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

最初に、現在のプログラム・バージョンを LCD モジュールに表示します。その後、UART 通信で START コマンドを受信すると、「フラッシュ・アクセス中」を示す LED1 を点灯し、コード・フラッシュ・プログラミング・モードに移行します。コード・フラッシュ・メモリのブート・クラスタ 1 (04000H ～ 07FFFH) に書き込まれているデータを消去して WRITE コマンドを待ちます。

WRITE コマンドを受信および書き換え用データを受信すると、ブート・クラスタ 1 を書き換えます。書き換えが完了し、END コマンドを受信すると LED1 を消灯します。これまでの処理が正常終了していると、内部リセットを発生させ、ブート・スワップを実行します。再起動後、書き換え後のプログラム・バージョンを LCD モジュールに表示します。

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 UART0	書き換えデータの取得
シリアル・インタフェース IICA0	LCDモジュールとの通信

1.1.1　フラッシュ・セルフ・プログラミング・コード (Renesas Flash Driver RL78 Type01) 概要

フラッシュ・セルフ・プログラミング・コードは、RL78 マイクロコントローラに搭載されたコード・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。

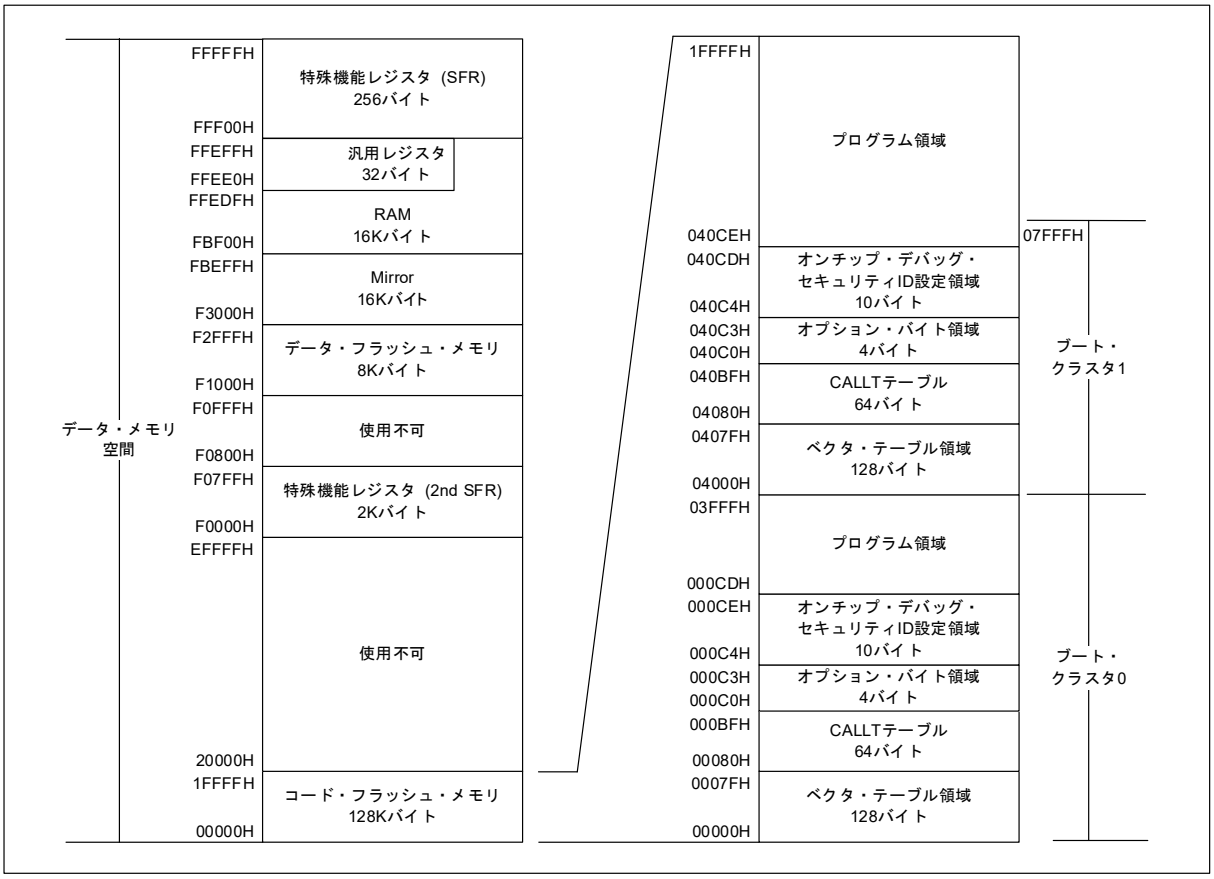
フラッシュ・セルフ・プログラミング・コードをユーザ・プログラムから呼び出すことにより、コード・フラッシュ・メモリの内容を書き換えることができます。

セルフ・プログラミングを行うためにはセルフ・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 言語またはアセンブリ言語でユーザ・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・メモリについて

RL78/G23 (R7F100GLG) 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　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 (040C0H - 040C3H) 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、040C0H - 040C3H に、000C0H - 000C3H と同じ値を設定してください。

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。

図 1-2　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・クラスタ0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・コード (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・クラスタ 0 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、セルフ・プログラミング時は無効となります。

1.1.3　セルフ・プログラミング

RL78/G23 には、セルフ・プログラミングを行うためのフラッシュ・セルフ・プログラミング・コード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・コードの各関数を呼び出すことでセルフ・プログラミングを行います。

RL78/G23 のセルフ・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。ただし、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・コード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・コード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。

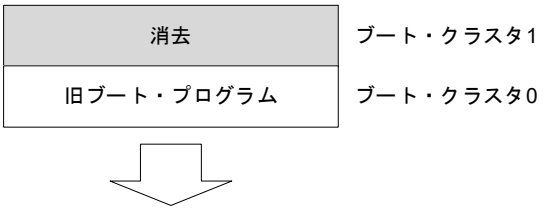
1.1.4　ブート・スワップ機能

ベクタ・テーブル・データ、プログラムの基本機能およびフラッシュ・セルフ・プログラミング・コードを配置している領域の書き換え中に、電源瞬断または外部要因によるリセット発生によって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。

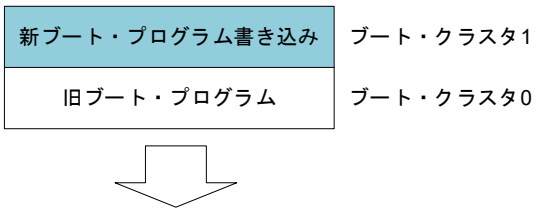
ブート・スワップ機能では、ブート・プログラム領域であるブート・クラスタ 0 とブート・スワップ対象領域であるブート・クラスタ 1 を切り替えます。書き換え処理を行う前に、新しいブート・プログラムをブート・クラスタ 1 に書き込んでおきます。このブート・クラスタ 1 とブート・クラスタ 0 をスワップし、ブート・クラスタ 1 をブート・プログラム領域にします。これによって、ブート・プログラム領域の書き換え中に電源の供給が瞬断しても、次のリセット・スタートはブート・クラスタ 1 からブートを行うため、正常にプログラムが動作します。

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
r_CF_EraseBlock関数を利用し、ブート・クラスタ1(ブロック8~15)を消去します。

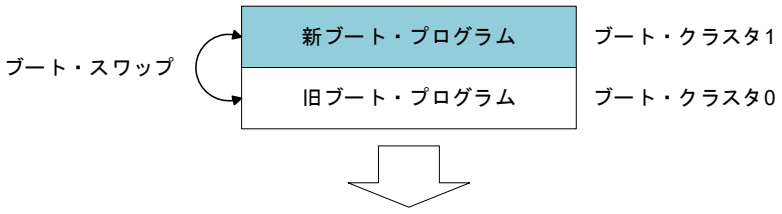


②ブート・クラスタ1へ新ブート・プログラム書き込み
r_CF_WriteData関数を利用し、ブート・クラスタ1に新ブート・プログラムを書き込みます。
r_CF_VerifyData関数を利用し、ブート・クラスタ1のベリファイを行います。
ここまでの処理で電源瞬断またはリセット発生等によって新ブート・プログラムが正常に書き込まれなかった場合でも、旧ブート・プログラムから起動が行われるため、プログラムが正常に動作します。



③ブート・スワップ・ビットの設定
r_RequestBootSwap関数を利用し、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瞬断またはリセット発生した場合は、書き換えが完了した新ブート・プログラムから起動が行われるため、プログラムが正常に動作します。

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。

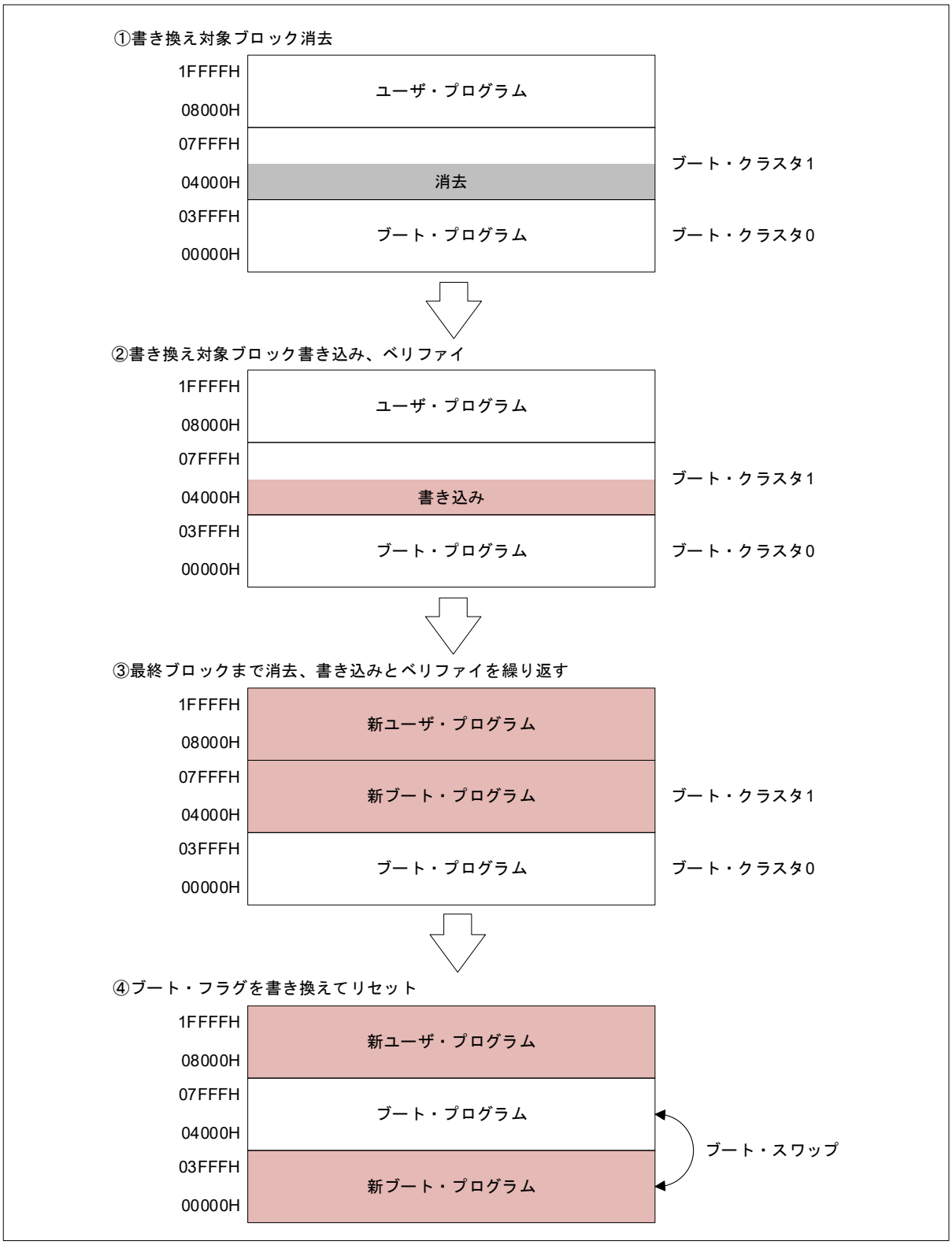


⑤ブート・スワップ完了

1.1.5　フラッシュ書き換え

セルフ・プログラミングのプログラム書き換え手順を以下に記載します。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。

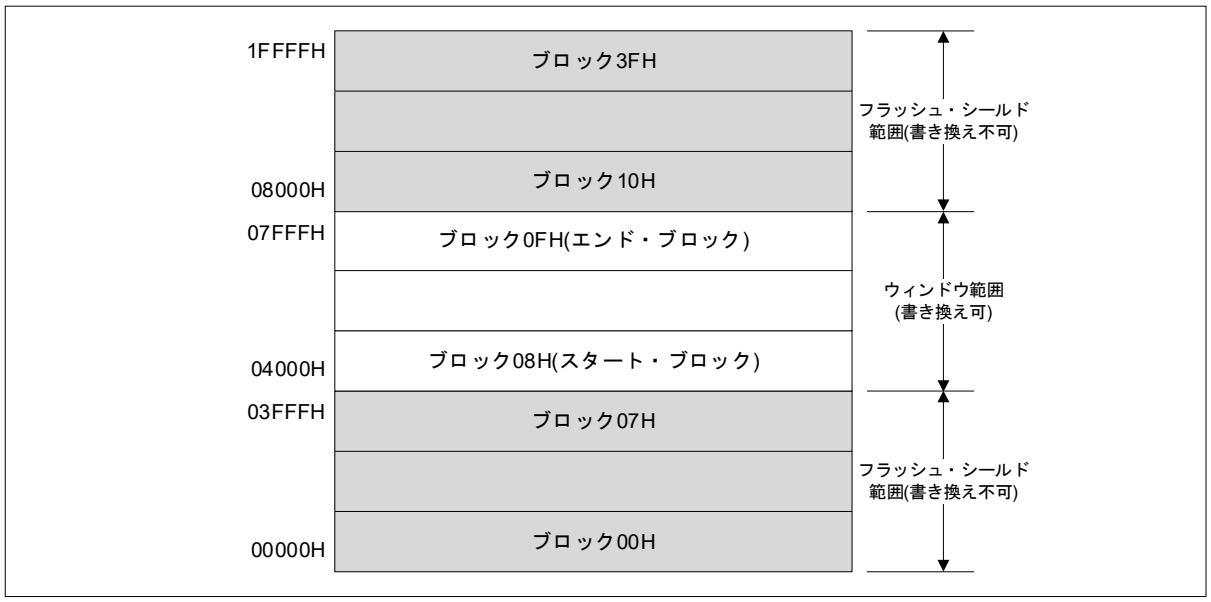
本アプリケーションノートでは、書き換え対象をブート領域に限定しています。



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と消去をセルフ・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 0FH の場合のイメージ図を記載します。



1.1.7　通信仕様

本アプリケーションノートは、UART 通信を利用してセルフ・プログラミングを行います。受信した START コマンド、WRITE コマンド、END コマンドに対応した処理を行います。正常終了の場合は、正常応答 (01H) を送信し、異常終了の場合は、データを送信せず、LCD モジュールに“ERROR!”と表示して処理を終了します。以下に UART 通信設定と各コマンドの仕様を記載します。

表 1-2　UART 通信設定

データ長	8ビット
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	115,200 bps

- START コマンド

START コマンドを受信すると、セルフ・プログラミングの初期設定を行います。正常終了の場合は、正常応答 (01H) を送信します。異常終了の場合は、データを送信しません。

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)

- 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、256 バイトの書き込み毎にベリファイを行います。正常終了の場合、正常応答 (01H) を送信します。異常終了の場合は、データを送信しません。

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)

- END コマンド

END コマンドを受信すると、応答通知として 01H を送信します。次にブート・フラグの反転を行います。正常終了の場合、リセットを発生させ、ブート・スワップを行います。異常終了の場合は、ブート・スワップを行いません。

START コード (01H)	データ長 (0002H)	コマンド (04H)	データ (なし)	チェックサム (1 バイト)

- 異常終了

LCD モジュールに“ERROR!”と表示して処理を終了します。

- 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。

1.1.8　フラッシュ・セルフ・プログラミング・コードの取得方法

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、RFD_RL78_TYPE1 フォルダ内にファイルをコピーしてください。

フラッシュ・セルフ・プログラミング・コードは、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

本アプリケーションノートでは、UART 通信を使用したセルフ・プログラミングの方法を説明します。

(1)　初期設定

ポートの初期設定

- P53 を出力ポートに設定します。 (初期値：ハイ･レベル、LED1 消灯状態。)

シリアル・アレイ・ユニットの初期設定

- チャネル 0、1 を UART として使用します。
- データ出力は P12/TxD0 端子、データ入力は P11/RxD0 端子を使用します。
- 動作クロックを CK00、クロックソースを fCLK/2 に設定します。
- 割り込み要因を転送完了割り込みに設定
- パリティなし、LSB ファースト、ストップ・ビットを 1 ビット、データ長を 8 ビットに設定します。
- 非反転 (標準) 送信に設定します。
- ボーレートを 115,200bps に設定します。

シリアル・インタフェース IICA の初期設定

- IICA0 (P60/SCLA0, P61/SDAA0) を使用します。
- IICA0 動作クロックを fCLK/2 に設定します。
- 自局アドレスを 10H に設定します。
- 動作モードを標準に設定します。
- 転送クロックを 80,000 bps に設定します。
- INTIICA0 割り込みを許可します。

LCD モジュールの初期設定および現在のプログラム・バージョン表示

- 定数 LCD_STRING の文字列を LCD モジュールに表示します。

Renesas Flash Driver RL78 Type01 の初期化

(2)　START コマンド処理

- P53 をロウ・レベル出力に設定し、「フラッシュ・アクセス中」を示す LED1 を点灯させます。
- r_CF_EraseBlock 関数を利用し、ブート・クラスタ 1 (04000H ～ 07FFFH) のデータを消去します。
  正常終了の場合は、正常応答 (01H) を送信します。
  異常終了の場合は、データを送信しません。

(3) WRITE コマンド処理

- 書き込みデータ (256 バイト) を受信します。
- r_CF_WriteData 関数を利用し、書き込み先アドレスに受信データを書き込みます。
  書き込み先アドレスを書き込みサイズ分加算します。
- 256 バイト毎に r_CF_VerifyData 関数を利用し、書き込まれたデータと受信データをベリファイします。
- 正常終了の場合、正常応答 (01H) を送信します。
  異常終了の場合は、データを送信しません。

(4) END コマンド処理

- P53 をハイ・レベル出力に設定し、「フラッシュ・アクセス中」を示す LED1 を消灯します。
- 正常応答 (01H) を送信します。
- r_RequestBootSwap 関数を利用し、ブート・フラグの値を反転します。
  ret_value が正常の場合、内部リセットを発生させます。
  内部リセット発生によって、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。
  正常終了の場合、ブート・フラグを反転してリセットを発生させ、ブート・スワップを行います。異常終了の場合は、ブート・スワップを行いません。

(5) 異常終了処理

- LCD モジュールに“ERROR!”と表示して処理を終了します。

注意 1.　ブート・クラスタ 1 の最終アドレス (07FFFH) まで書き込みが完了している場合、新たな WRITE コマンドを受信しても書き込み処理は実行されません。

注意 2.　END コマンド (04H) を受信した場合は、常に、正常応答 (01H) を送信し、P52 をハイ・レベル出力 (LED1 消灯) に変更します。r_RequestBootSwap 関数の実行により、ブート・スワップを行います。

注意 3.　セルフ・プログラミングが正常終了しなかった場合は、LCD モジュールに“ERROR!”と表示し、以降の処理は行いません。

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

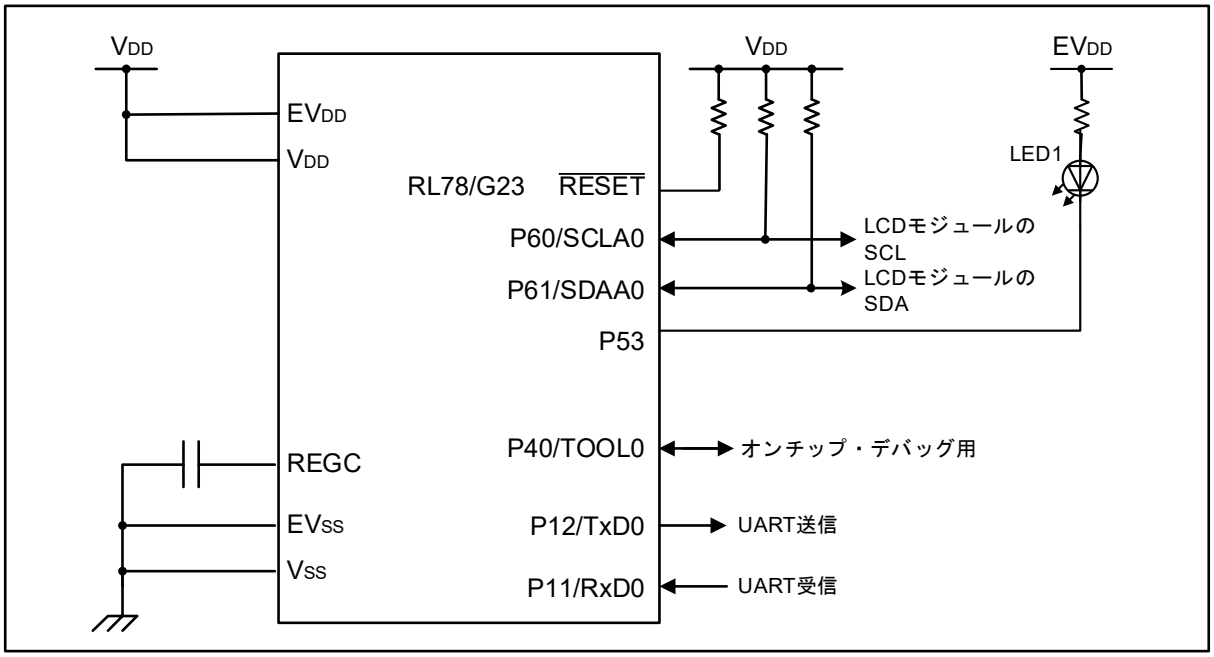
周辺機能	用途
使用マイコン	RL78/G23 (R7F100GLG)
使用ボード	RL78/G23-64p Fast Prototyping Board (RTK7RLG230CLG000BJ)
動作周波数	高速オンチップ・オシレータ・クロック (f_{IH}) : 32MHz
動作電圧	3.3V (3.1V～3.5V で動作可能) LVD 検出電圧: リセット・モード 立ち上がり時 TYP. 1.90 V (1.84 V ～ 1.95 V) 立ち下がり時 TYP. 1.86 V (1.80 V ～ 1.91 V)
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ for CC V8.05.00
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 V2021-04 (21.4.0)
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 V4.21.1
スマート・コンフィグレータ (SC)	ルネサス エレクトロニクス製 V1.0.1
ボードサポートパッケージ (BSP)	ルネサス エレクトロニクス製 V1.00

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
P53	出力	フラッシュ・アクセス状態を示す LED1 の点灯/消灯
P60/SCLA0、P61/SDAA0	入出力	LCD モジュールとの I2C 通信

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.　ソフトウェア設定

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 検出電圧 : 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	10000100B	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 040C0H-040C3H 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

サンプル・プログラムでは、ブート・クラスタ 1 に書き込むデータをローカル変数に保存します。ローカル変数はスタック領域に配置されることから、cstart.asm を修正して、任意のスタック領域の確保およびスタック領域の初期化を行います。

```
;$IF (__RENESAS_VERSION__ < 0x01010000)                先頭行に ';' を追加しコメントアウト
;--------------------------------------------------------------------------
;     stack area
;--------------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
        .DS         0x200
_stacktop:
;$ENDIF                                                 先頭行に ';' を追加しコメントアウト
    .
    .
    .
    .
        ;------------------------------------------------
        ; setting the stack pointer
        ;------------------------------------------------
;$IF (__RENESAS_VERSION__ >= 0x01010000)                先頭行に ';' を追加しコメントアウト
;       MOVW        SP,#LOWW(__STACK_ADDR_START)       先頭行に ';' を追加しコメントアウト
;$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                                                 先頭行に ';' を追加しコメントアウト

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

4.2.2　書き換え用プログラムの RAM 領域への配置

ブート・クラスタ 1 の書き換えに使用するプログラムを RAM 領域へ配置します。ブート・クラスタ 1 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      $.L12_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     $.L11_TEXT
```

※　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。

※　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。

※　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　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 040C4H - 040CDH 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。

4.4　サンプル・プログラム使用リソース

4.4.1　ROM 領域セクション一覧

表 4-3 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 4-3　ROM 領域セクション一覧

セクション名	配置内容
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

4.4.2　RAM 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 RAM 領域セクションの一覧を示します。

表 4-4　RAM 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_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・メモリ制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

4.5　定数一覧

表 4-5 にサンプル・プログラムで使用する定数を示します。

表 4-5　定数一覧

定数名	設定値	内容
LED_ON	00H	LED ON
LED_OFF	01H	LED OFF
START_WRITE_ADDRESS	00004000H	書き込み開始アドレス
END_WRITE_ADDRESS	00007FFFH	書き込み終了アドレス
WRITE_DATA_SIZE	0100H	コード・フラッシュ・メモリの書き込みサイズ (256 バイト)
CF_BLOCK_SIZE	0800H	コード・フラッシュ・メモリのブロック・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
CPU_FREQUENCY	32	CPU 動作周波数
COMMAND_START	02H	コマンドコード: START
COMMAND_WRITE	03H	コマンドコード: WRITE
COMMAND_END	04H	コマンドコード: END
COMMAND_ERROR	FFH	コマンドコード: ERROR
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 bit0: 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 bit1: 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 bit3: 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 bit4: 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 bit5: エクストラ領域シーケンサ・エラー
VALUE_U08_SHIFT_ADDR_TO_BLOCK_CF	11	CodeFlash のブロック番号算出時に行うビットシフト用定数
VALUE_U08_SHIFT_ADDR_TO_BLOCK_DF	8	DataFlash 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	1 ビット・マスク値
VALUE_U01_MASK1_1BIT	1	1 ビット・マスク値
VALUE_U08_MASK0_8BIT	00H	8 ビット・マスク値
VALUE_U08_MASK1_8BIT	FFH	8 ビット・マスク値

4.6　列挙型

表 4-6 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 4-6　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_ERR_CFDF_SEQUENCER	10H	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	11H	エクストラ領域シーケンサ・エラー
ENUM_RET_ERR_ERASE	12H	消去エラー
ENUM_RET_ERR_WRITE	13H	書き込みエラー
ENUM_RET_ERR_BLANKCHECK	14H	ブランク・エラー
ENUM_RET_ERR_CHECK_WRITE_DATA	15H	書き込みデータのリード値比較エラー
ENUM_RET_ERR_MODE_MISMATCHED	16H	モード不一致エラー
ENUM_RET_ERR_PARAMETER	17H	パラメータ・エラー
ENUM_RET_ERR_CONFIGURATION	18H	デバイス構成・エラー

4.7　変数一覧

表 4-7 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 4-7　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フラグ	r_Send_nByte r_Config_UART0_callback_sendend
uint8_t	f_UART0_receiveend	UART0 にてデータの受信が完了したことを示すフラグ	r_Receive_nByte r_Config_UART0_callback_receiveend

4.8　関数一覧

表 4-8 と表 4-9 にサンプル・プログラムで使用する関数を示します。

表 4-8　関数一覧 (1/2)

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_write	WRITE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_Config_UART0_callback_receiveend	UART0 受信完了割り込み時のコールバック処理
r_RecvPacket	UART0 コマンド受信処理
r_ReceivePacketAnalyze	UART0 コマンド解析処理
r_Receive_nByte	UART0 データ受信処理
r_Send_nByte	UART0 データ送信処理
r_SendACK	UART0 正常応答送信処理
r_Config_IICA0_callback_master_sendend	IICA0 送信完了割り込み時のコールバック処理
r_Config_IICA0_callback_master_error	IICA0 送信エラー割り込み発生時のコールバック処理
r_LCM_init	LCDモジュール 初期化処理
r_LCM_clear	LCDモジュール 表示消去処理クリアディスプレイ
r_LCM_send_string	LCDモジュール 文字列送信処理
r_LCM_send_command	LCDモジュール コマンド送信処理
r_LCM_send_data	LCDモジュール データ送信処理
r_LCM_turn_sendend_on	LCDモジュール 通信終了フラグ設定
r_LCM_wait_sendend	LCDモジュール 通信終了待ち処理

表 4-9　関数一覧 (2/2)

R_RFD_Init 注	RFD RL78 Type01の初期化処理
R_RFD_SetFlashMemoryMode 注	フラッシュ・メモリ制御モード変更処理
R_RFD_EraseCodeFlashReq 注	コード・フラッシュ消去処理
R_RFD_WriteCodeFlashReq 注	コード・フラッシュ書き込み処理
R_RFD_CheckCFDFSeqEndStep1 注	コード／データ・フラッシュ領域シーケンサの動作終了確認処理
R_RFD_CheckCFDFSeqEndStep2 注	フラッシュ・メモリ・シーケンサ制御レジスタのクリアにより、コマンド動作が終了したかの確認処理
R_RFD_GetSeqErrorStatus 注	コード／データ・フラッシュ領域シーケンサ・コマンド、または、エクストラ領域シーケンサ・コマンドにより、発生したエラー情報の取得処理
R_RFD_ClearSeqRegister 注	コード／データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサ制御を行うレジスタのクリア処理
R_RFD_CheckExtraSeqEndStep1 注	エクストラ領域シーケンサの動作終了を確認処理
R_RFD_CheckExtraSeqEndStep2 注	エクストラ制御レジスタのクリアにより、コマンド動作が終了したかの確認処理
R_RFD_GetSecurityAndBootFlags 注	セキュリティ・フラグ、ブート領域切替フラグ情報の取得処理
R_RFD_SetDataFlashAccessMode 注	データ・フラッシュへのアクセスの許可／禁止を設定
R_RFD_SetExtraBootAreaReq 注	ブート領域切替フラグの書き込み処理
R_RFD_ForceReset 注	CPUの内部リセット要求

注.　フラッシュ・セルフ・プログラミング・コードで定義されている API 関数です。関数の詳細は ”RL78 ファミリ Renesas Flash Driver RL78 Type01 ユーザーズマニュアル” を参照してください

4.9　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	無し
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	無し
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_write	
概　要	WRITE 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_write(uint32_t* write_start_addr, uint8_t __near * write_data);
説　明	WRITE コマンド受信時の処理を行います。
引　数	uint32_t i_u32_start_addr: 書き込み開始アドレス uint8_t __near * inp_u08_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_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。正常終了時は内部リセットが発生し、再起動を行います。
引　数	無し
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 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 i_u32_start_addr: 書き込み開始アドレス uint16_t i_u16_write_data_length: 書き込みサイズ uint8_t __near * inp_u08_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 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_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 i_u32_start_addr, uint16_t i_u16_write_data_length, uint8_t __near * inp_u08_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 i_u32_start_addr: 書き込み開始アドレス uint16_t i_u16_write_data_length: 書き込みサイズ uint8_t __near * inp_u08_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	無し
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、Config_IICA0.h、LCM_driver.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_Config_UART0_callback_receiveend	
概　要	UART0 受信完了割り込み時処理
ヘッダ	r_cg_macrodriver.h、Config_IICA0.h、LCM_driver.h
宣　言	static void r_Config_UART0_callback_receiveend(void);
説　明	UART0 の受信完了割り込み時に呼ばれるコールバック関数です。
引　数	MD_STATUS flag : エラータイプ
リターン値	なし

r_RecvPacket	
概　要	UART0 コマンド受信処理
ヘッダ	r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	MD_STATUS r_RecvPacket(uint8_t *data, uint16_t *length);
説　明	UART0 を使用してコマンド受信処理を行います。 1 パケット分のデータ受信が完了するまで受信待ちを行います。
引　数	uint8_t *data: 受信データ格納先バッファのポインタ uint16_t *length: 受信データサイズ (コマンド部＋データ部＋チェックサム部の合計)
リターン値	MD_OK: 正常終了 (受信完了) COMMAND_ERROR: パラメータ・エラー

r_ReceivePacketAnalyze	
概　要	UART0 コマンド解析処理
ヘッダ	r_cg_userdefine.h
宣　言	uint8_t r_ReceivePacketAnalyze(uint8_t *rxbuf, uint16_t rxlength);
説　明	受信データのチェックサムの確認を行います。 チェックサムが一致した場合は、受信データのコマンドコードを取得します。
引　数	uint8_t *rxbuf: 受信データ uint16_t rxlength: 受信データサイズ (コマンド部＋データ部＋チェックサム部の合計)
リターン値	COMMAND_START: START コマンドを受信 COMMAND_WRITE: WRITE コマンドを受信 COMMAND_END: END コマンドを受信 COMMAND_ERROR: チェックサムエラー、またはコマンドコードエラー

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

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 *tx_buff: 送信データ格納先バッファのポインタ const uint16_t tx_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_Config_IICA0_callback_master_sendend	
概　要	IICA0 送信完了割り込み時の処理
ヘッダ	r_cg_macrodriver.h、Config_IICA0.h、LCM_driver.h
宣　言	static void r_Config_IICA0_callback_master_receiveend(void);
説　明	IICA0 の送信完了割り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_Config_IICA0_callback_master_error	
概　要	IICA0 送信エラー発生時処理
ヘッダ	r_cg_macrodriver.h、Config_IICA0.h、LCM_driver.h
宣　言	static void r_Config_IICA0_callback_master_error(MD_STATUS flag);
説　明	IICA0 の送信エラー割り込み時に呼ばれるコールバック関数です。
引　数	MD_STATUS flag : エラータイプ
リターン値	なし

r_LCM_init	
概　要	LCDモジュール 初期化処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_init(void);
説　明	LCD モジュールを初期化します。
引　数	なし
リターン値	なし

r_LCM_clear	
概　要	LCDモジュール 表示消去処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_clear(void);
説　明	LCD モジュールに表示消去処理のコマンドを送信します。
引　数	なし
リターン値	なし

r_LCM_send_string	
概　要	LCDモジュール 文字列送信処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_send_string(uint8_t * const str, lcm_position_t pos);
説　明	LCD モジュールに str で渡された文字列を表示します。 表示させるラインは pos で指定します。
引　数	uint8_t * const str : 表示させる文字列 lcm_position_t pos : LCM_POSITION_TOP で上段に表示 LCM_POSITION_BOTTOM で下段に表示
リターン値	なし

r_LCM_send_command	
概　要	LCDモジュール コマンド送信処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_send_command(uint8_t command);
説　明	LCD モジュールに command で渡されたコマンドを送信します。
引　数	uint8_t command : LCD モジュールへ送信するコマンド
リターン値	なし

r_LCM_send_data	
概　要	LCDモジュール データ送信処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_send_data(uint8_t data);
説　明	LCD モジュールに data で渡されたデータを送信します。
引　数	uint8_t data : LCD モジュールへ送信するデータ
リターン値	なし

r_LCM_turn_sendend_on	
概　要	LCDモジュール 通信終了フラグ設定
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	void r_LCM_turn_sendend_on(void);
説　明	g_LCM_is_sendend に LCD モジュールとの IIC 通信終了フラグを設定します。
引　数	なし
リターン値	なし

r_LCM_wait_sendend	
概　要	LCDモジュール 通信終了待ち処理
ヘッダ	LCM_driver.h、Config_IICA0.h
宣　言	static void r_LCM_wait_sendend(void);
説　明	LCD モジュールとの IIC 通信が終了するまで待ち、コマンド実行ウエイト時間 (5ms) だけウエイトを実行します。
引　数	なし
リターン値	なし

4.10 フローチャート

4.10.1 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)

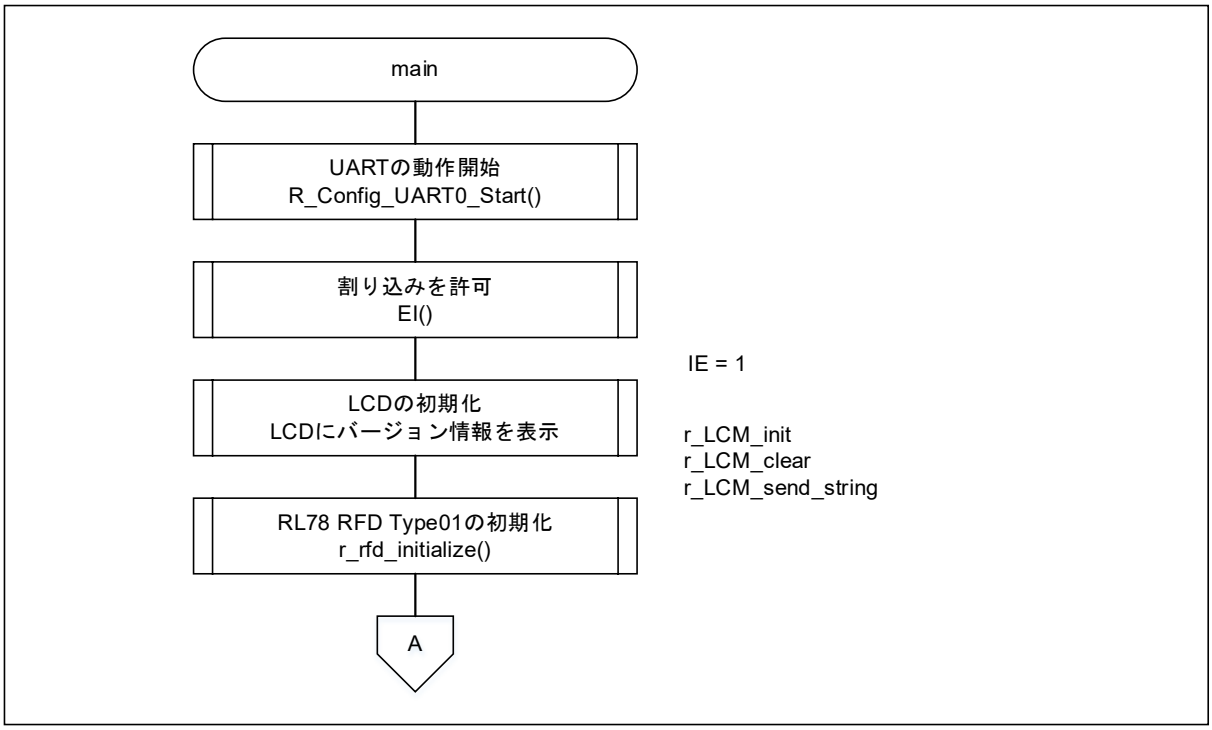
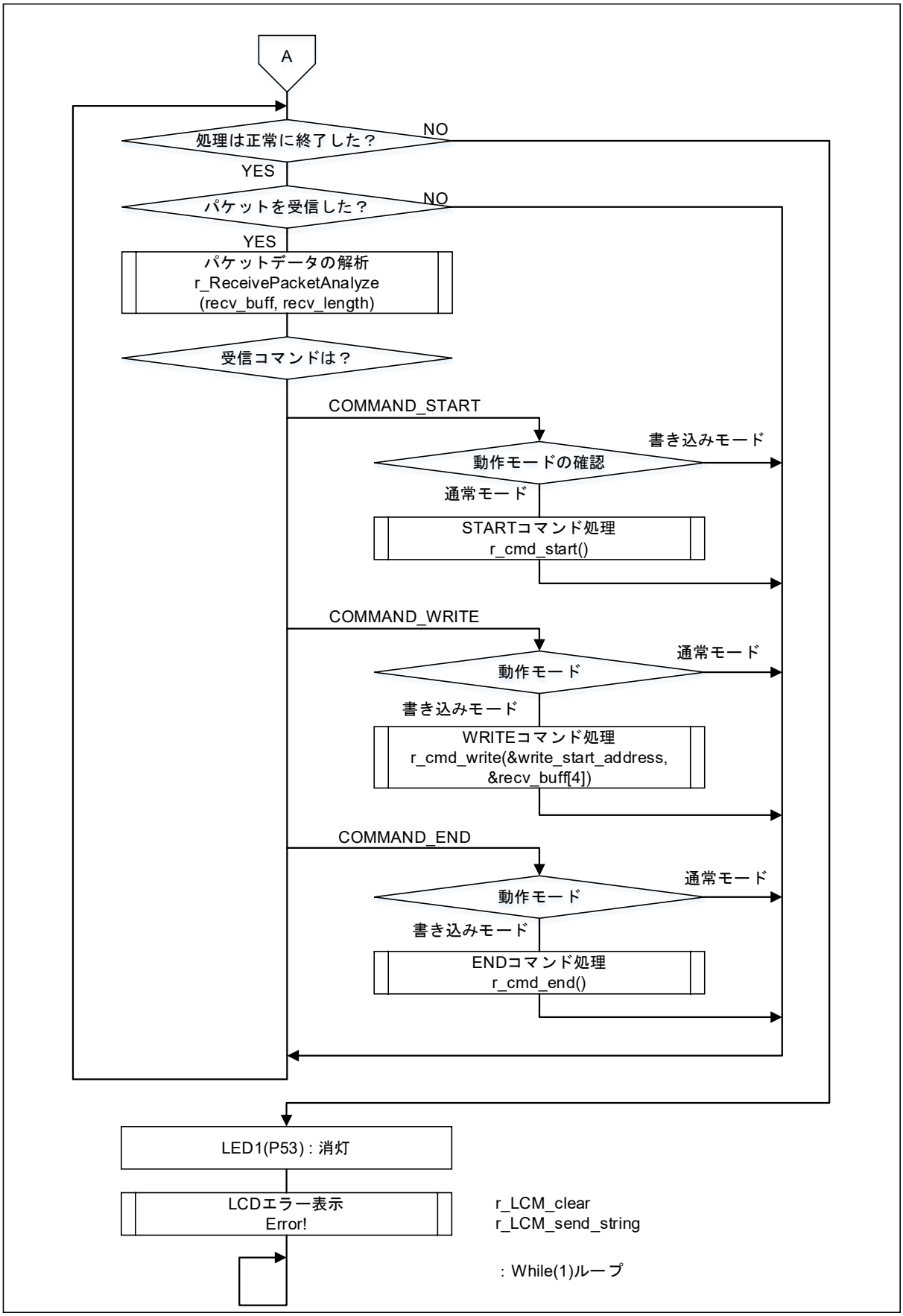


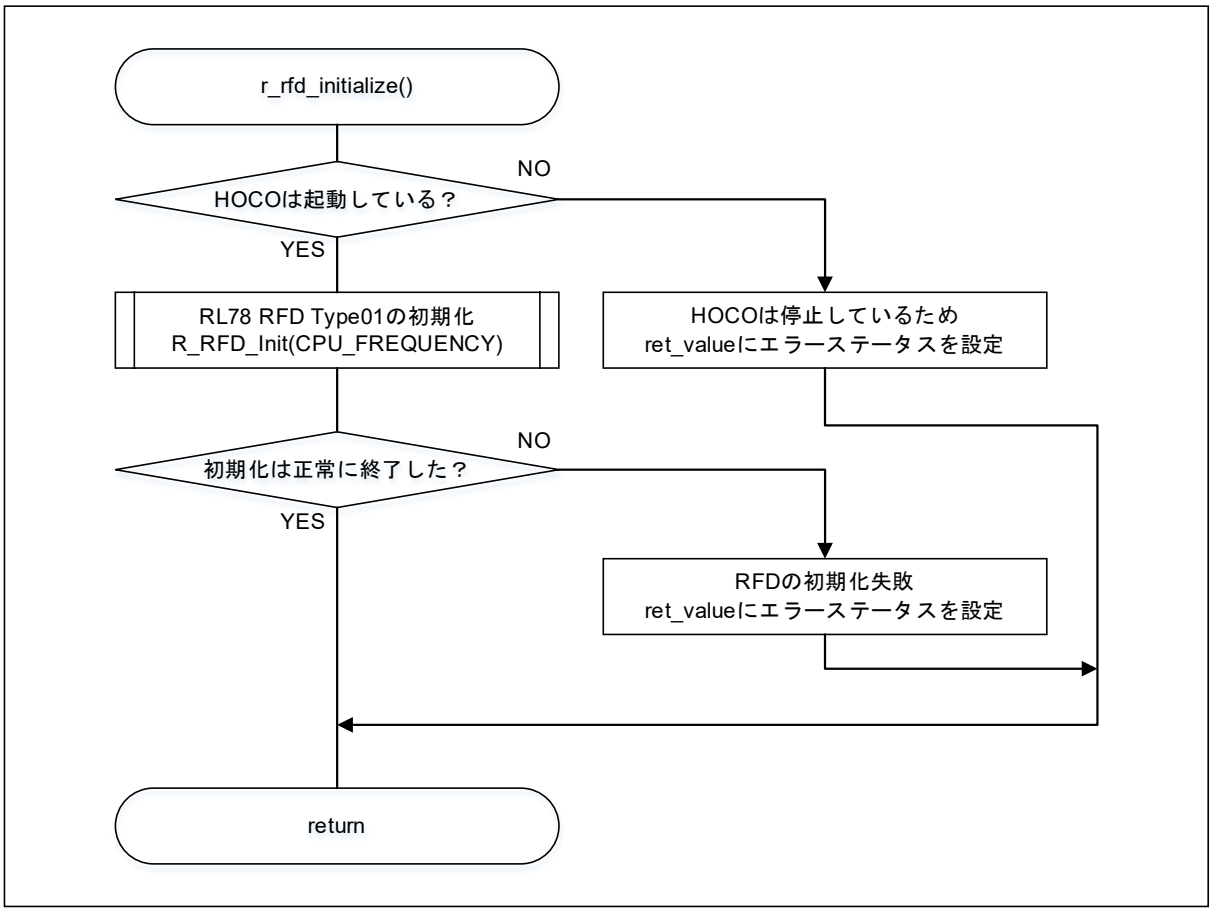
図 4-2　メイン処理 (2/2)


A
処理は正常に終了した？
NO
YES
パケットを受信した？
NO
YES
パケットデータの解析
r_ReceivePacketAnalyze
(recv_buff, recv_length)
受信コマンドは？
COMMAND_START
動作モードの確認
書き込みモード
通常モード
STARTコマンド処理
r_cmd_start()
COMMAND_WRITE
動作モード
通常モード
書き込みモード
WRITEコマンド処理
r_cmd_write(&write_start_address,
&recv_buff[4])
COMMAND_END
動作モード
通常モード
書き込みモード
ENDコマンド処理
r_cmd_end()
LED1(P53) : 消灯
LCDエラー表示
Error!
r_LCM_clear
r_LCM_send_string
：While(1)ループ

4.10.2 RFD RL78 Type01 初期化処理

図 4-3 に RFD RL78 Type01 初期化処理のフローチャートを示します。

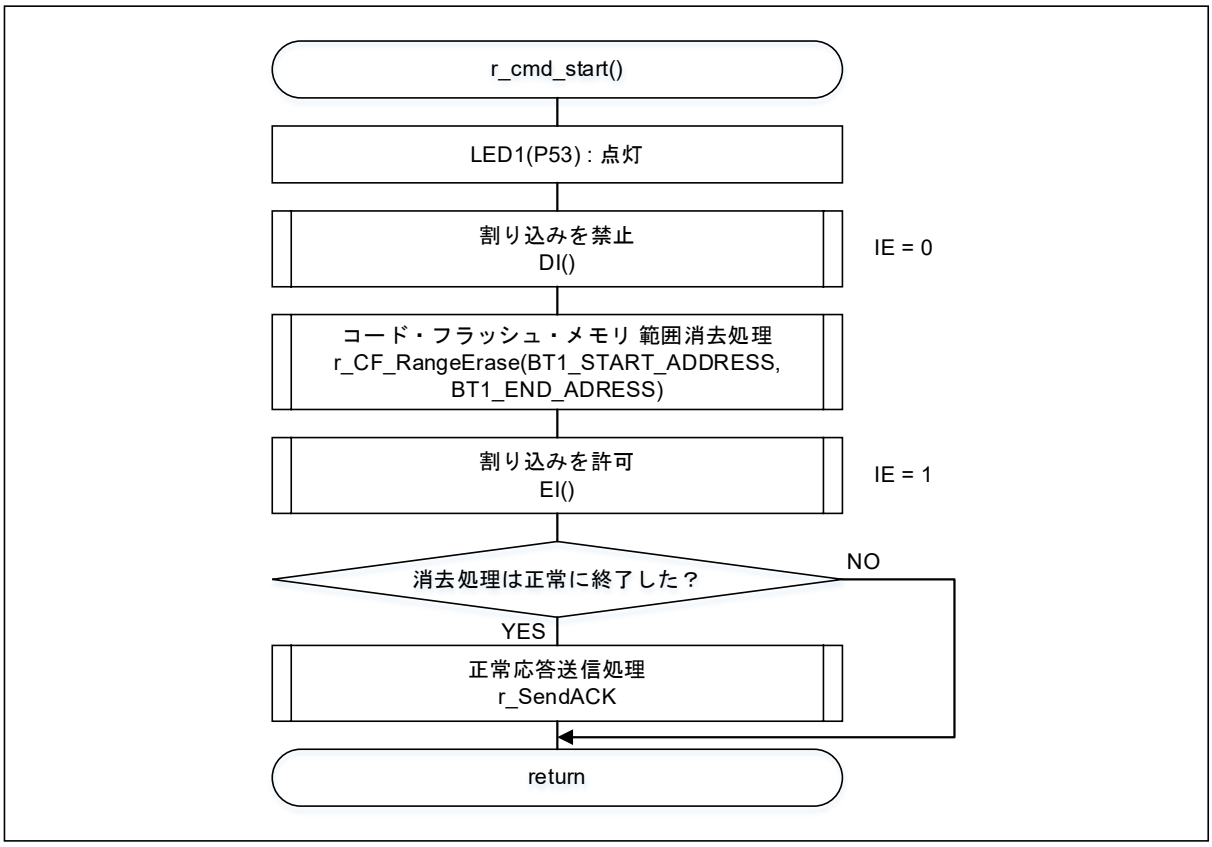
図 4-3　RFD RL78 Type01 初期化処理



4.10.3 START コマンド処理

図 4-4 にSTARTコマンド処理のフローチャートを示します。

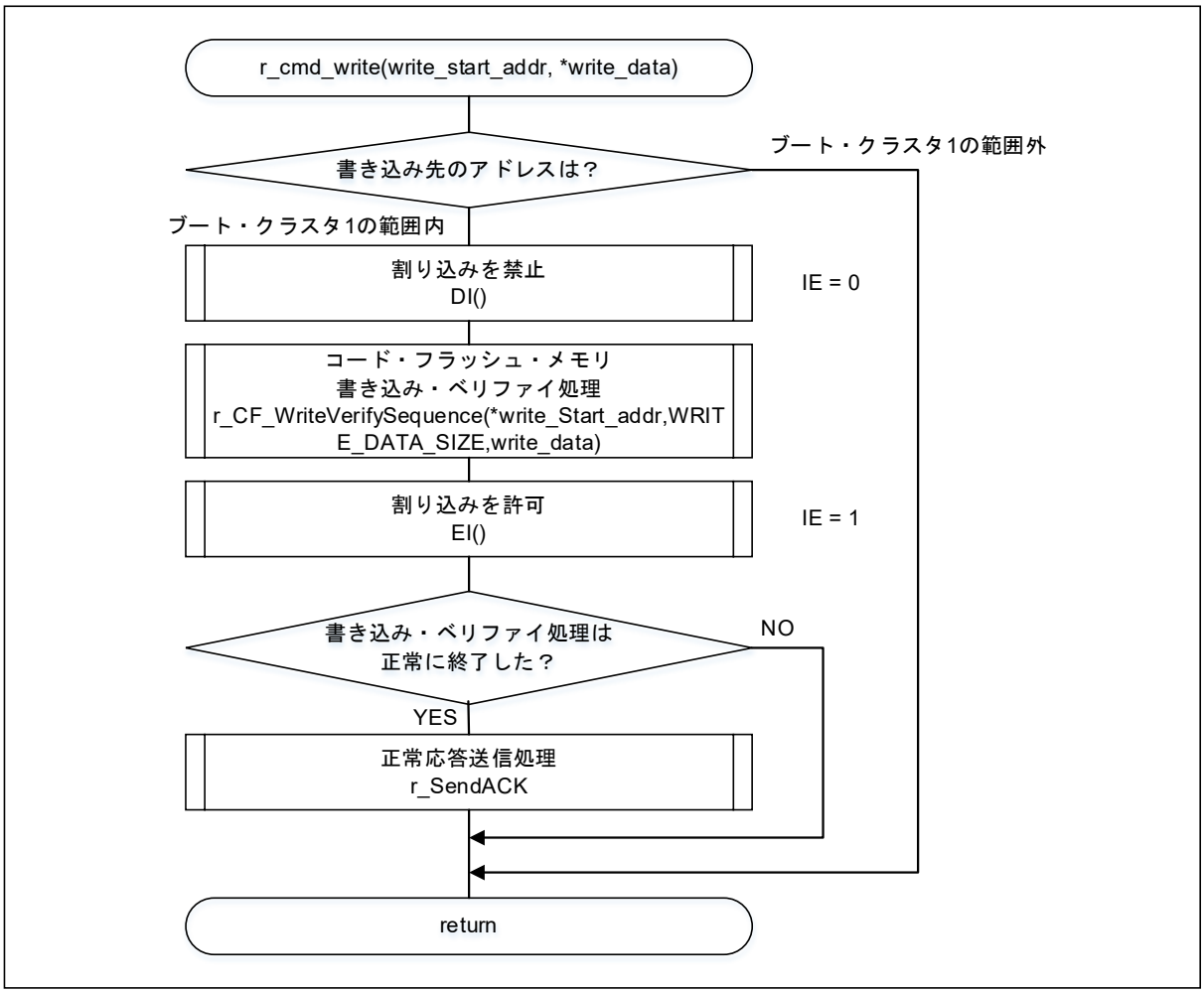
図 4-4　START コマンド処理



4.10.4 WRITE コマンド処理

図 4-5 にWRITEコマンド処理のフローチャートを示します。

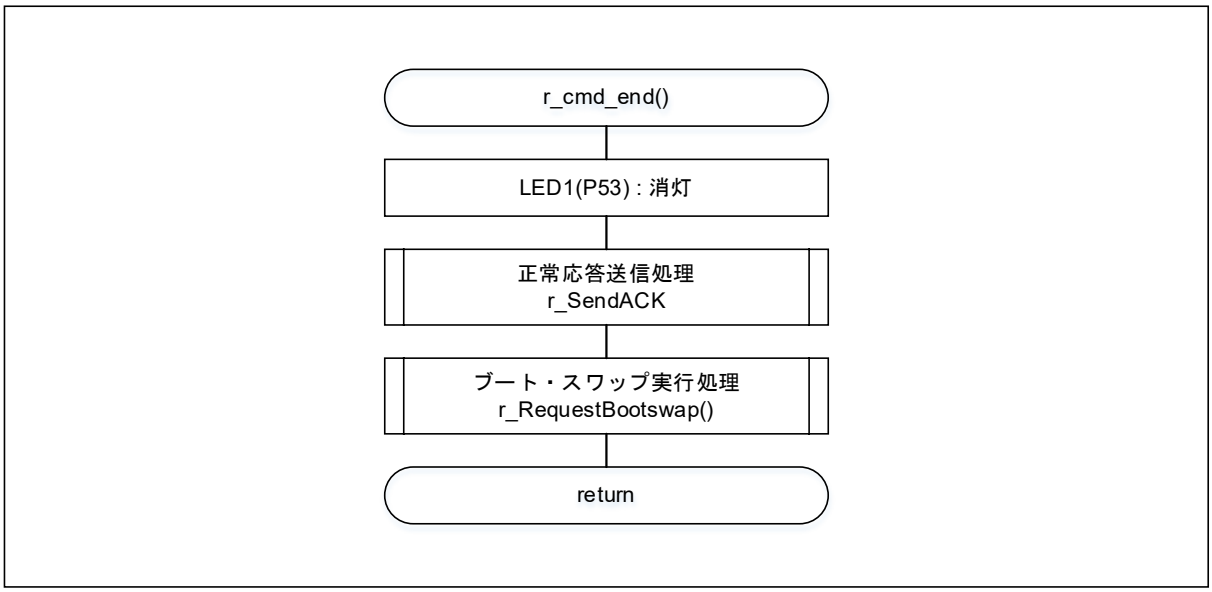
図 4-5　WRITE コマンド処理



4.10.5 END コマンド処理

図 4-6 にENDコマンド処理のフローチャートを示します。

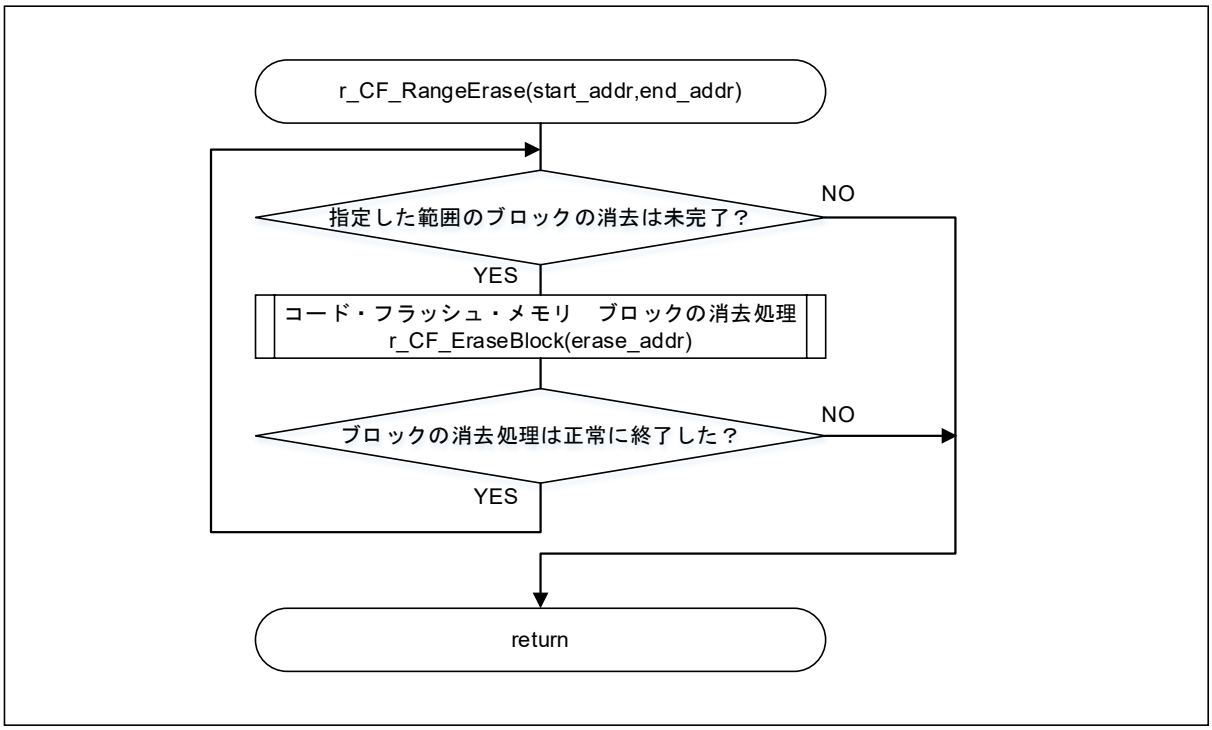
図 4-6　END コマンド処理



4.10.6 コード・フラッシュ・メモリ 範囲消去処理

図 4-7 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。

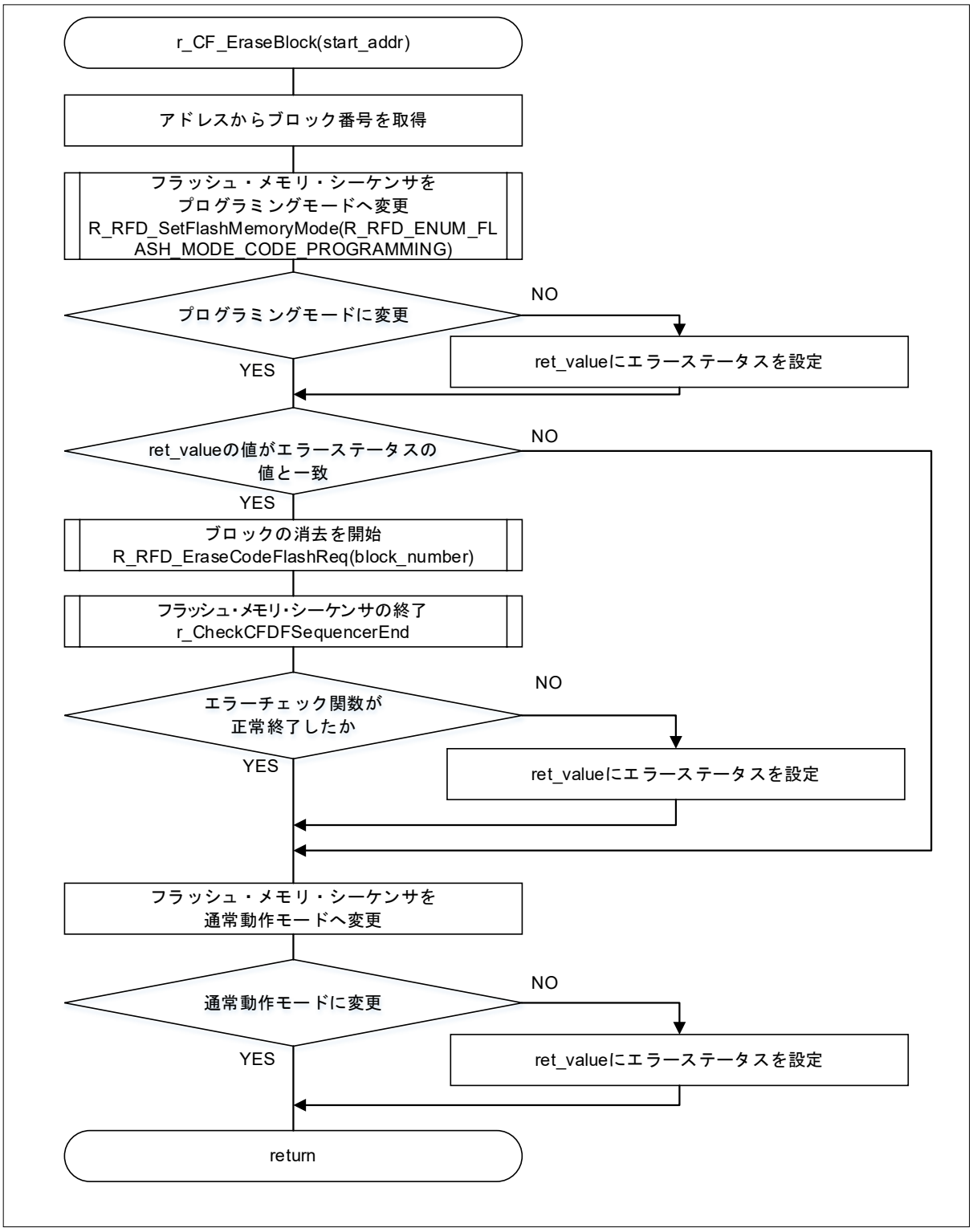
図 4-7　コード・フラッシュ・メモリ 範囲消去処理



4.10.7 コード・フラッシュ・メモリ ブロック消去処理

図 4-8 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。

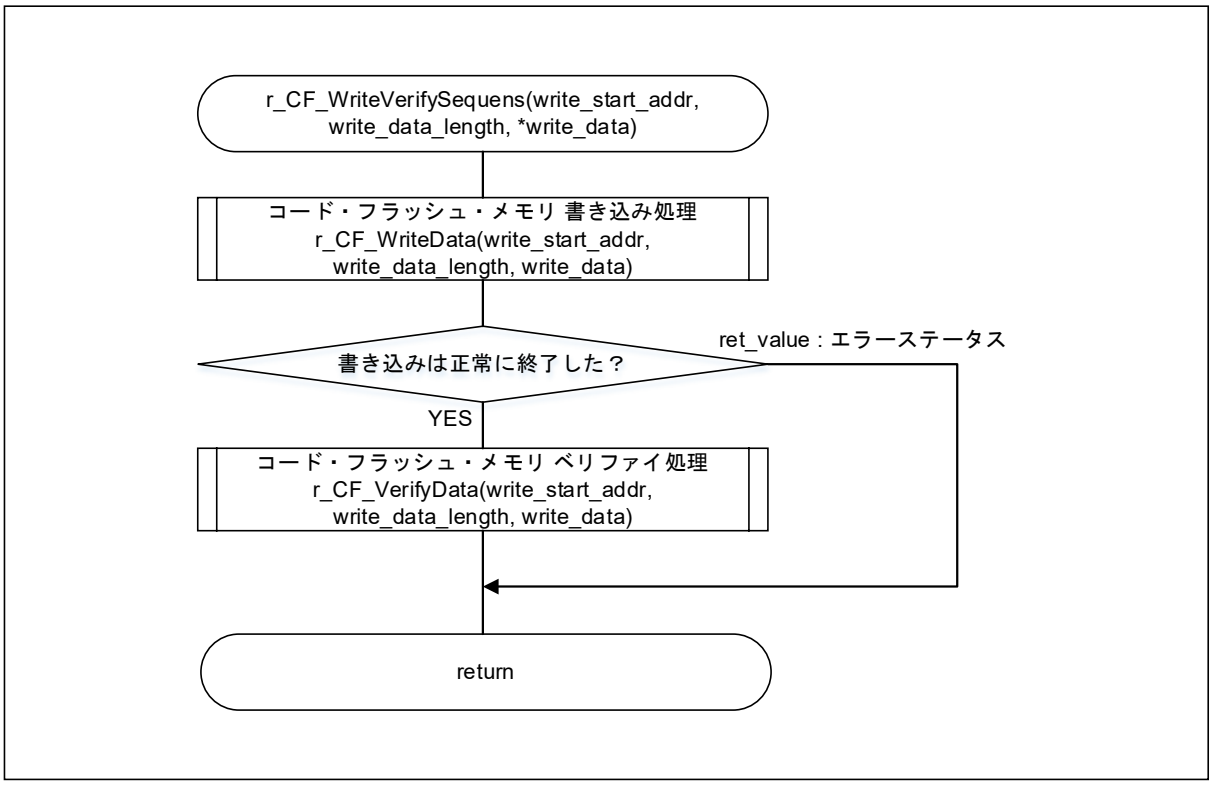
図 4-8　コード・フラッシュ・メモリ ブロック消去処理



4.10.8 コード・フラッシュ・メモリ 書き込み・ベリファイ処理

図 4-9 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。

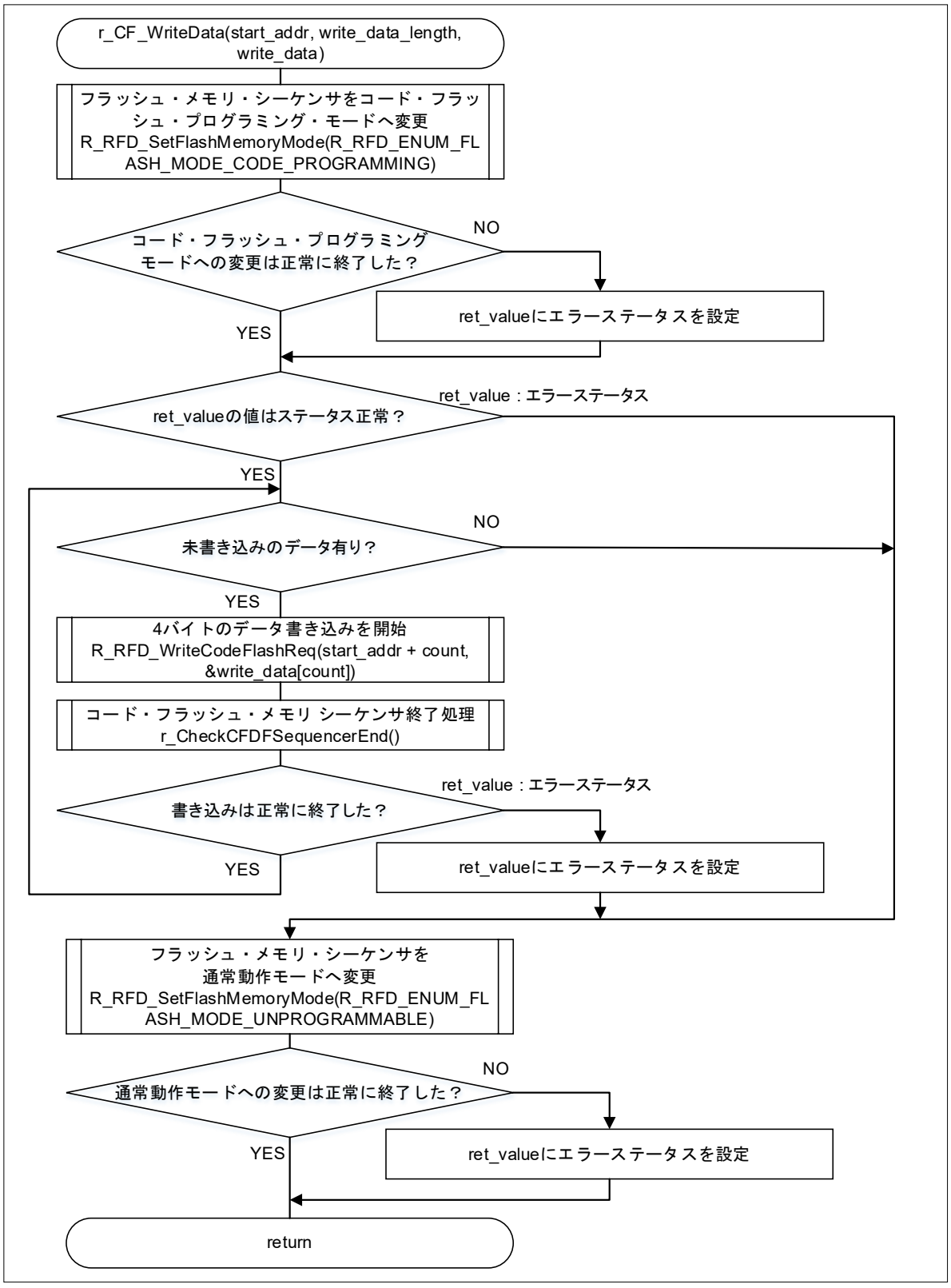
図 4-9　コード・フラッシュ・メモリ 書き込み・ベリファイ処理



4.10.9 コード・フラッシュ・メモリ 書き込み処理

図 4-10 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。

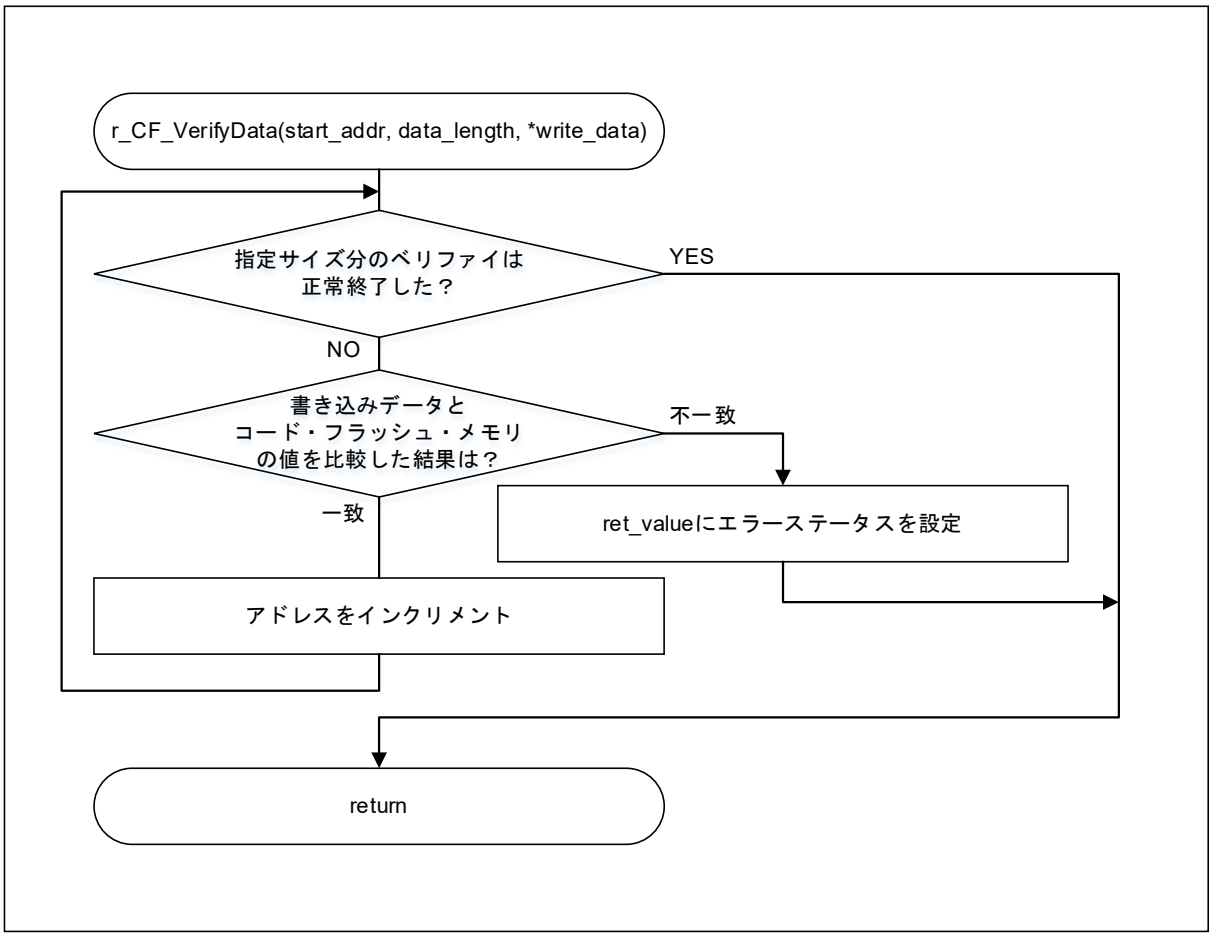
図 4-10　コード・フラッシュ・メモリ 書き込み処理



4.10.10　　コード・フラッシュ・メモリ ベリファイ処理

図 4-11 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 4-11　コード・フラッシュ・メモリ ベリファイ処理



4.10.11　　コード・フラッシュ・メモリ シーケンス終了処理

図 4-12、図 4-13 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 4-12　コード・フラッシュ・メモリ シーケンス終了処理 (1/2)

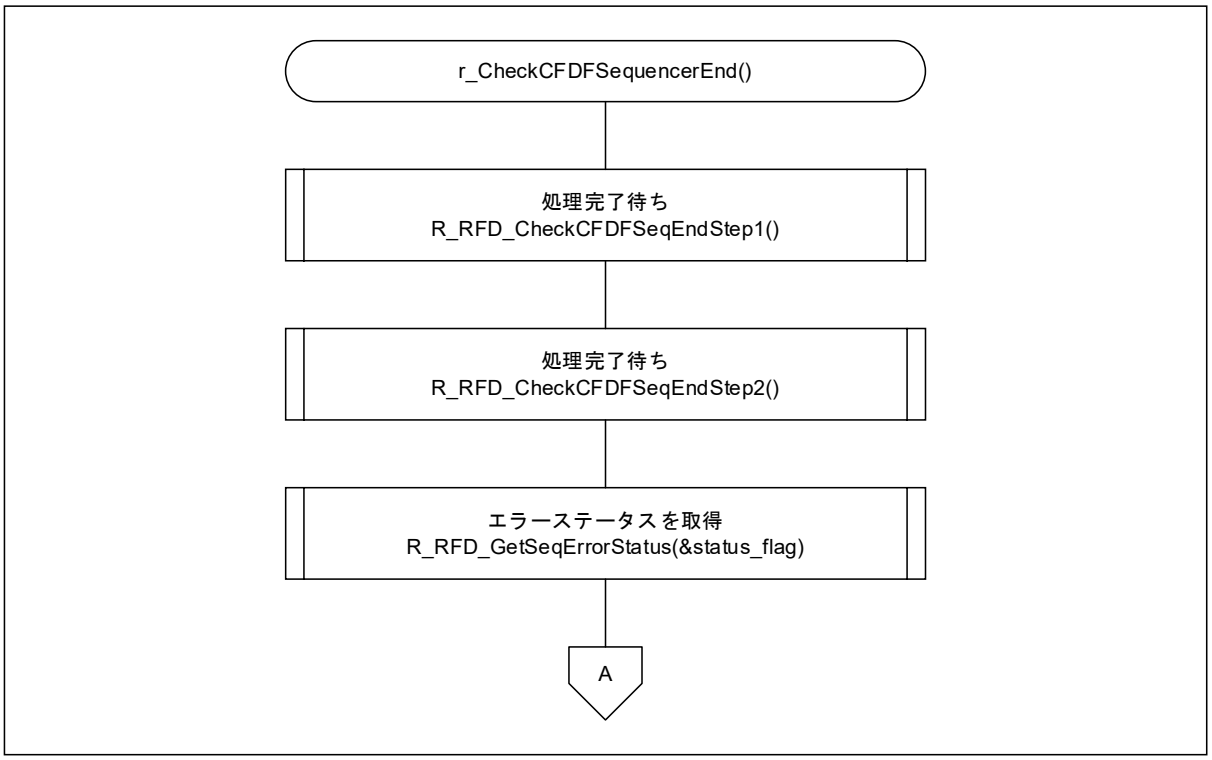
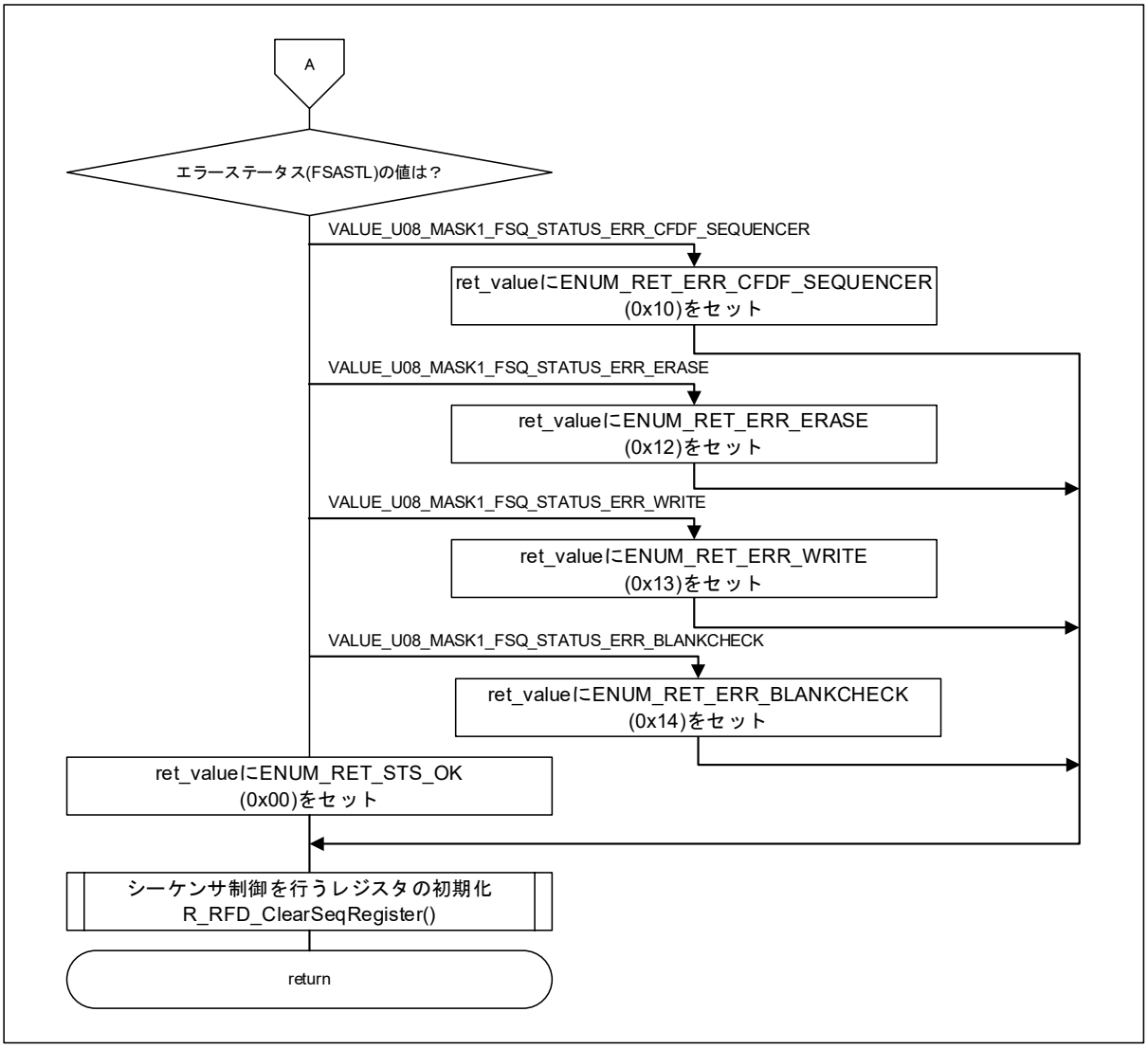


図 4-13　コード・フラッシュ・メモリ シーケンス終了処理 (2/2)



4.10.12　　エクストラ・メモリ シーケンス終了処理

図 4-14、図 4-15 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 4-14　エクストラ・メモリ シーケンス終了処理 (1/2)

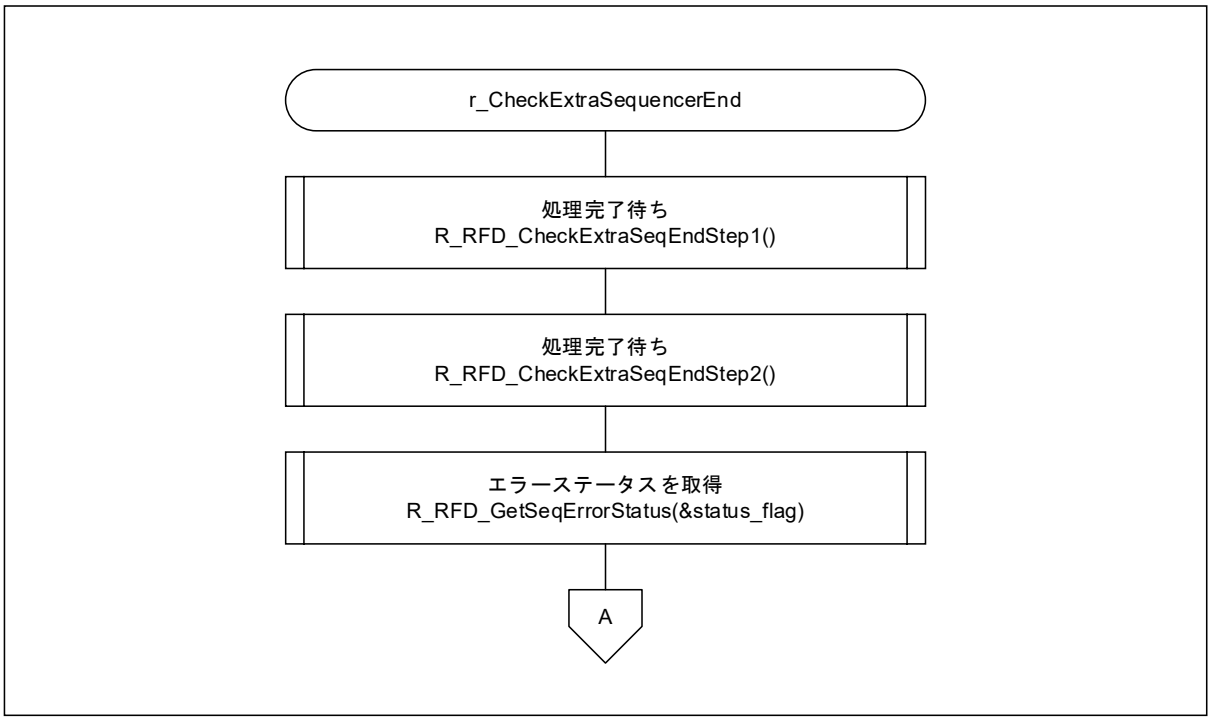
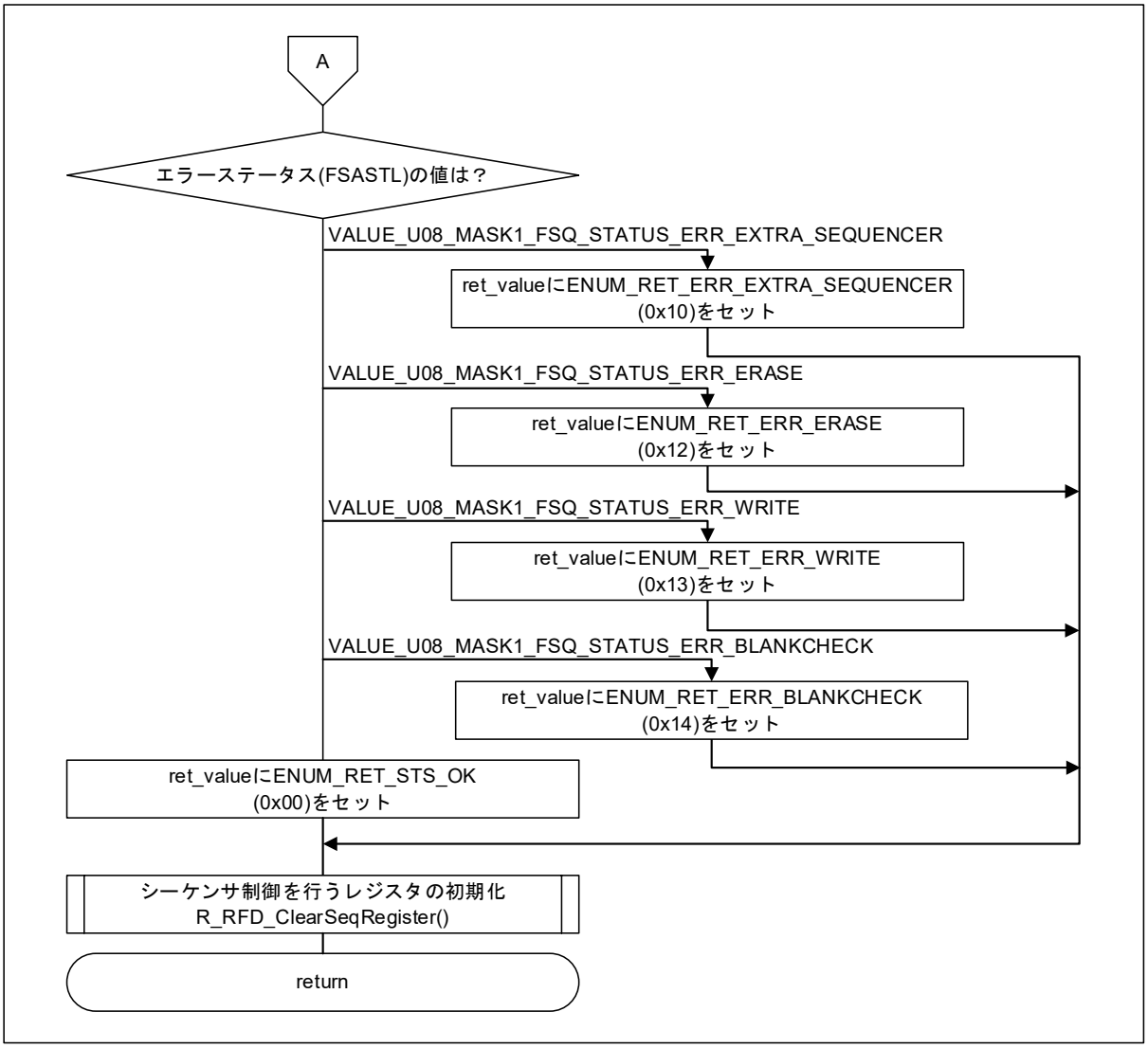


図 4-15　エクストラ・メモリ シーケンス終了処理 (2/2)



4.10.13　　ブート・スワップ処理

図 4-16、図 4-17にブート・スワップ処理のフローチャートを示します。

図 4-16　ブート・スワップ処理 (1/2)

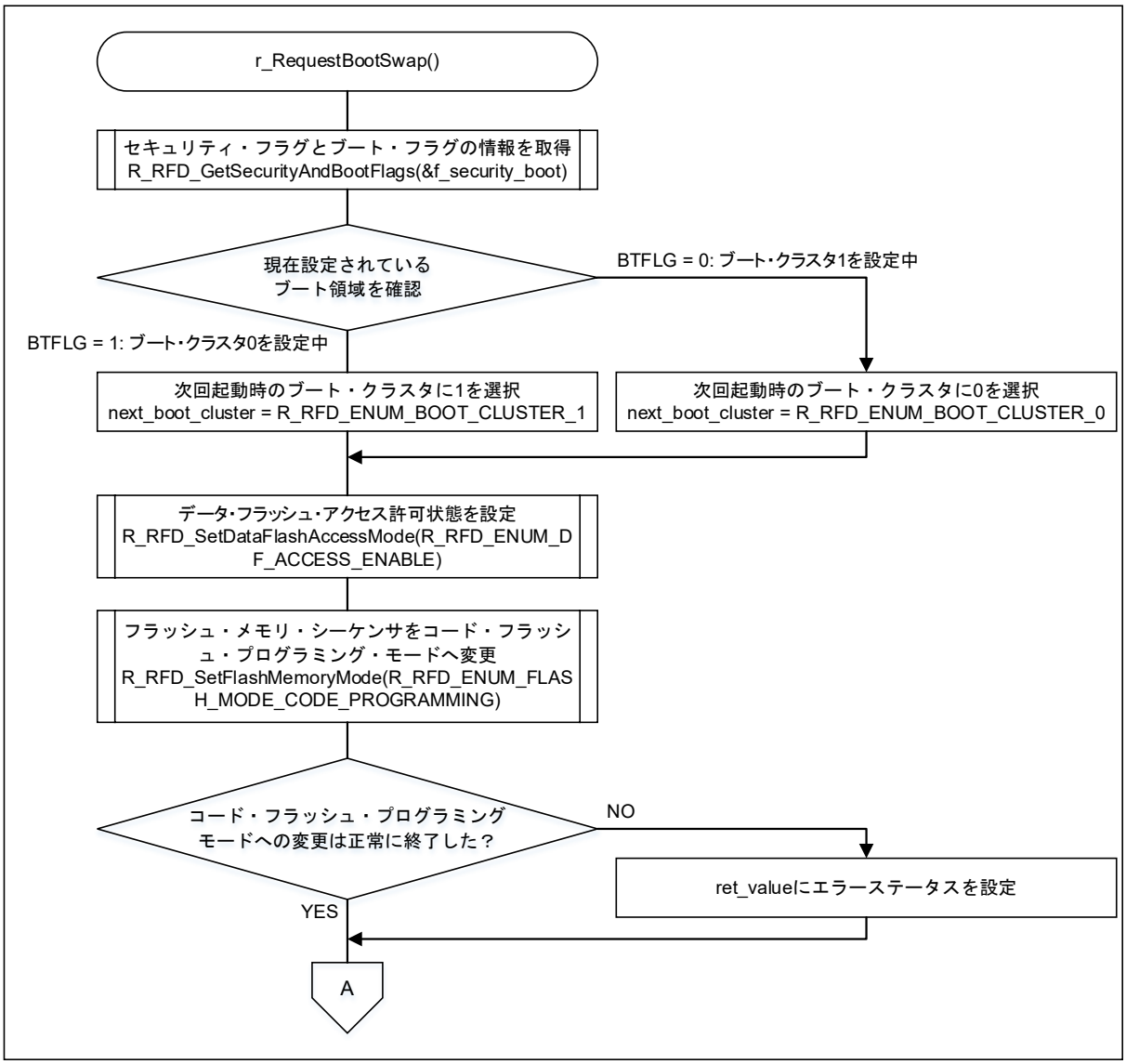
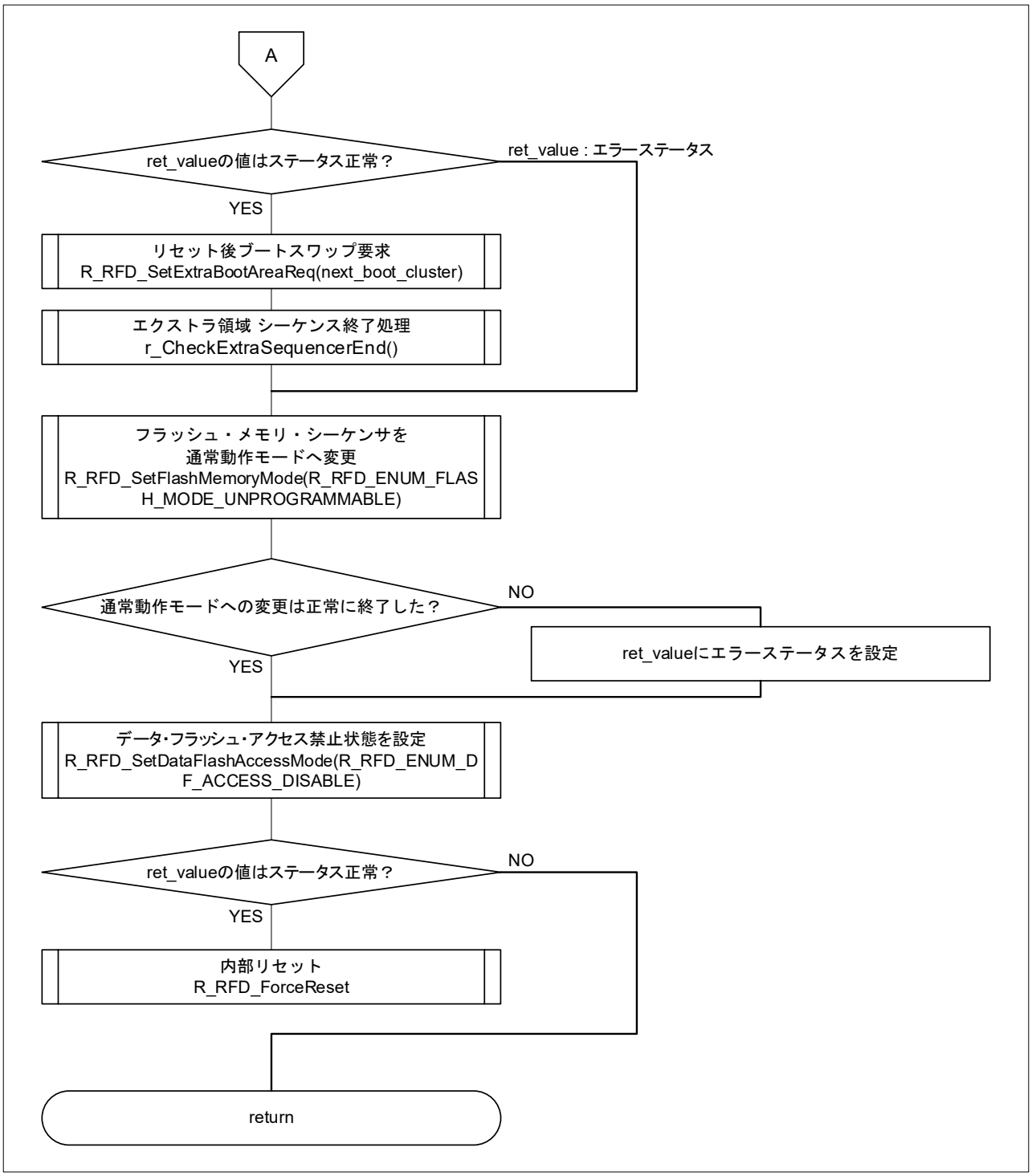


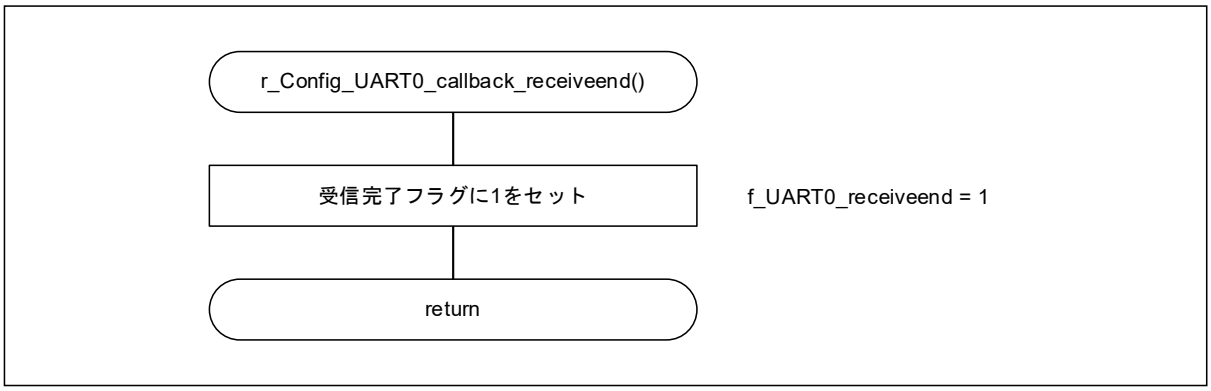
図 4-17　ブート・スワップ処理 (2/2)



4.10.14　　UART0 受信完了割込み時の処理

図 4-18 にUART0 受信完了割込み時の処理のフローチャートを示します。

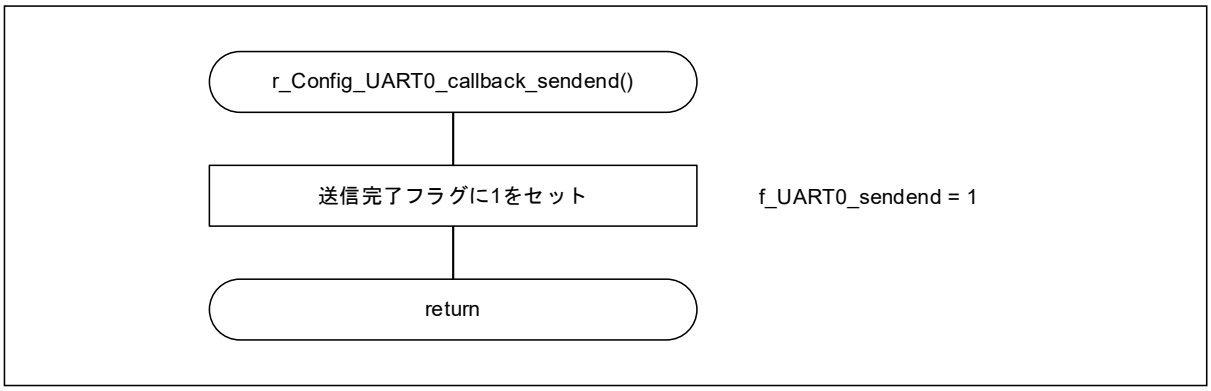
図 4-18　UART0 受信完了割込み時の処理



4.10.15　　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。

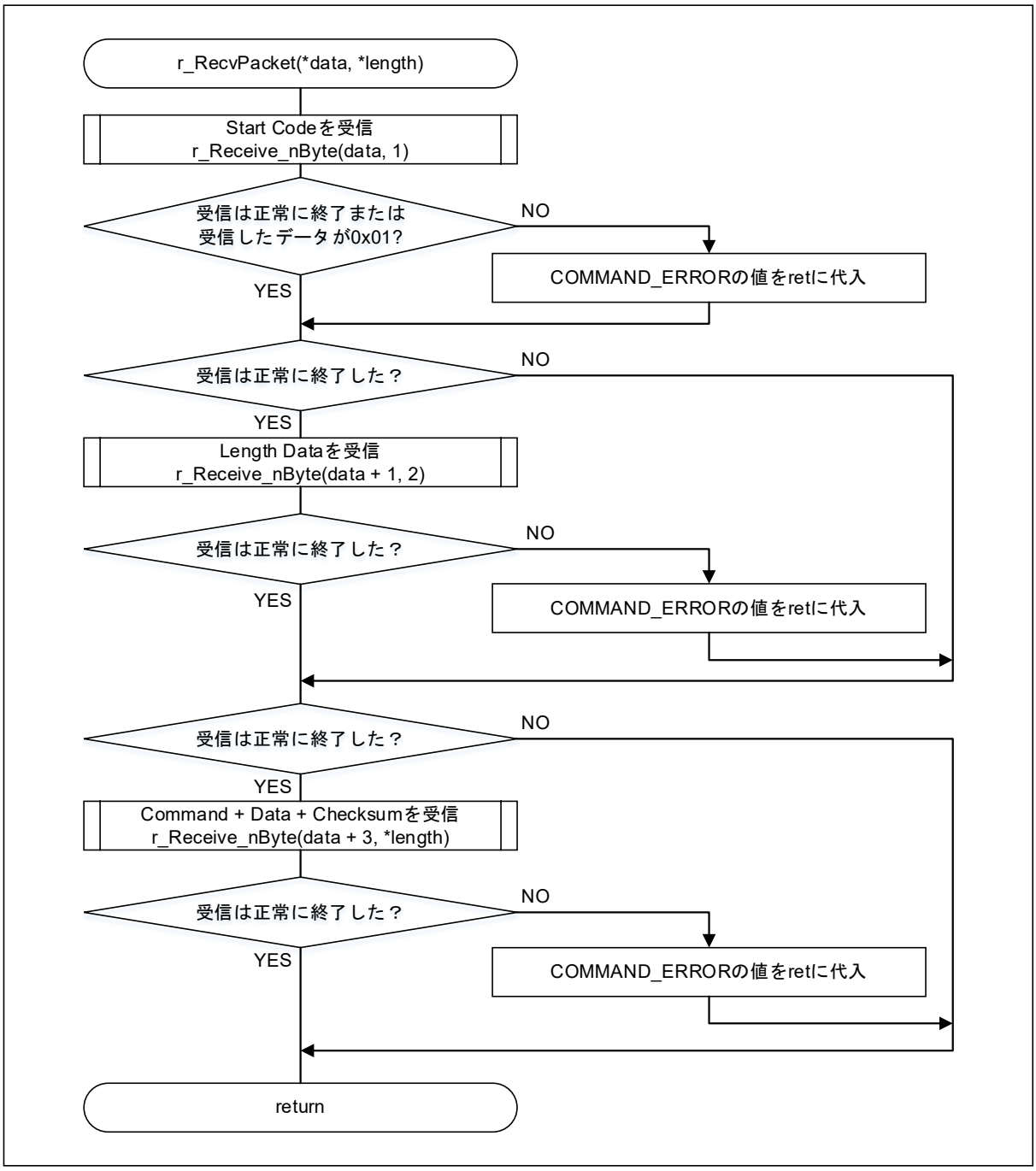
図 4-19　UART0 送信完了割込み時の処理



4.10.16　　UART0 コマンド受信処理

図 4-20 にUART0 コマンド受信処理のフローチャートを示します。

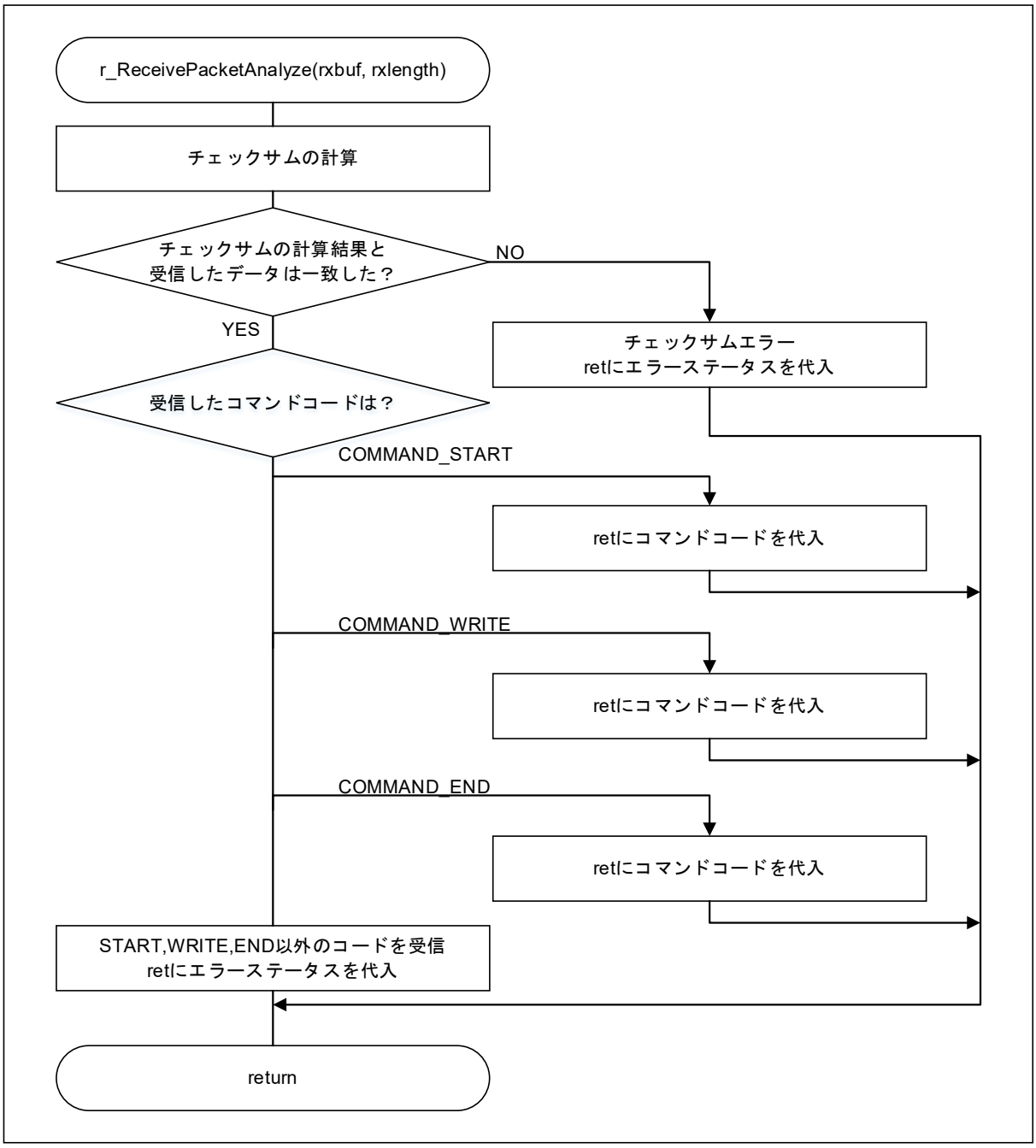
図 4-20　UART0 コマンド受信処理



4.10.17　　UART0 コマンド解析処理

図 4-21 にUART0 コマンド解析処理のフローチャートを示します。

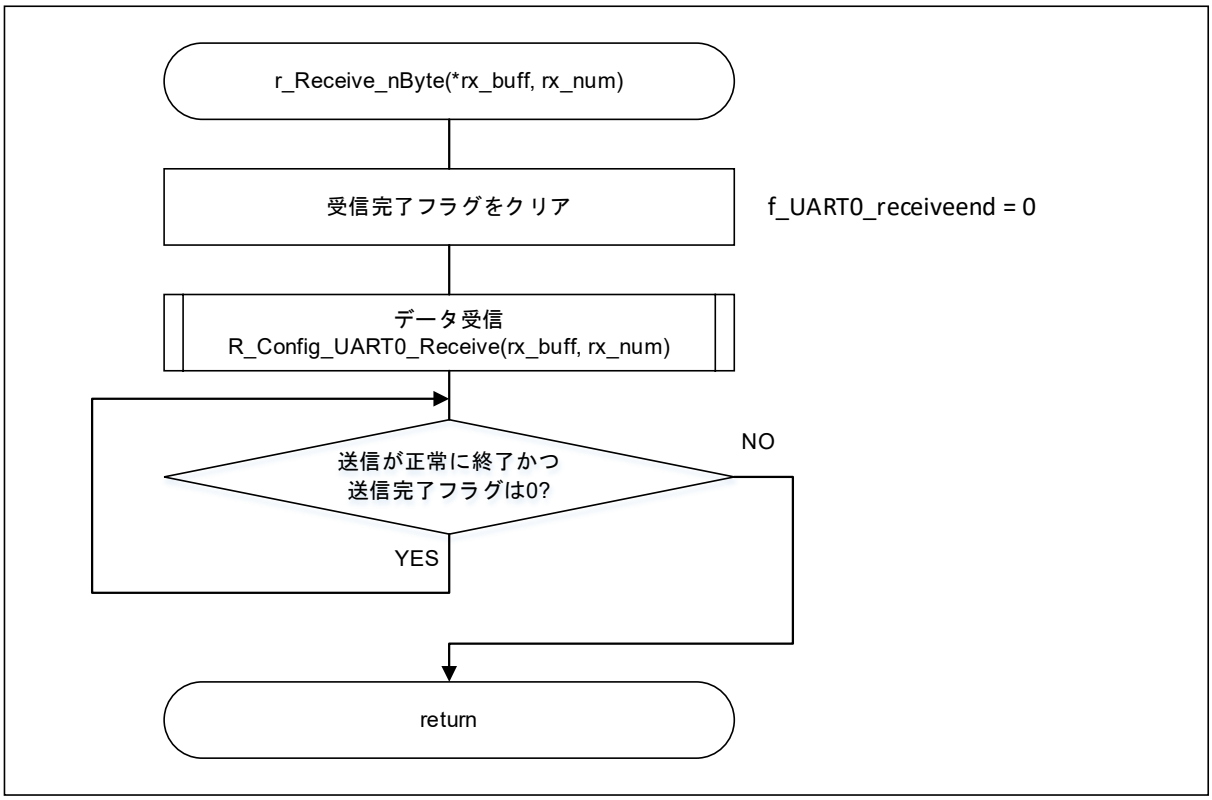
図 4-21　UART0 コマンド解析



4.10.18　　UART0 データ受信処理

図 4-22 にUART0 データ受信処理のフローチャートを示します。

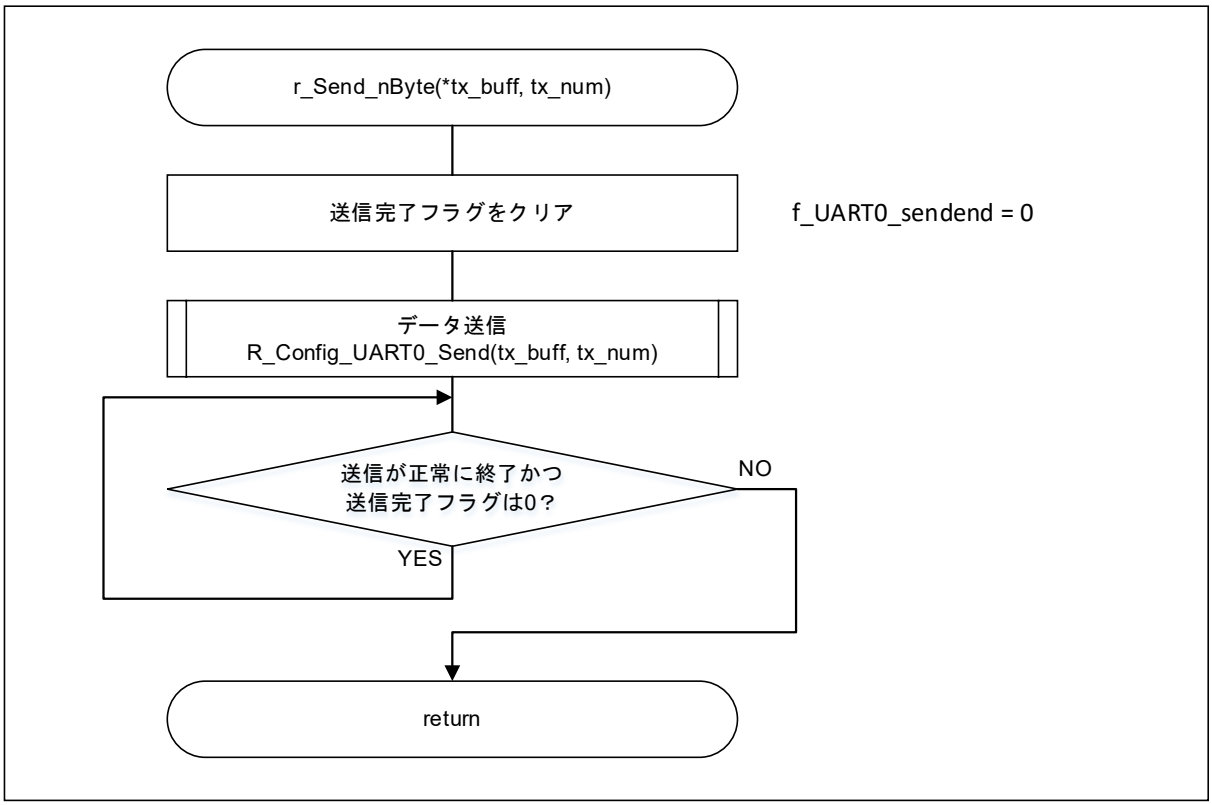
図 4-22　UART0 データ受信処理



4.10.19　　UART0 データ送信処理

図 4-23 にUART0 データ送信処理のフローチャートを示します。

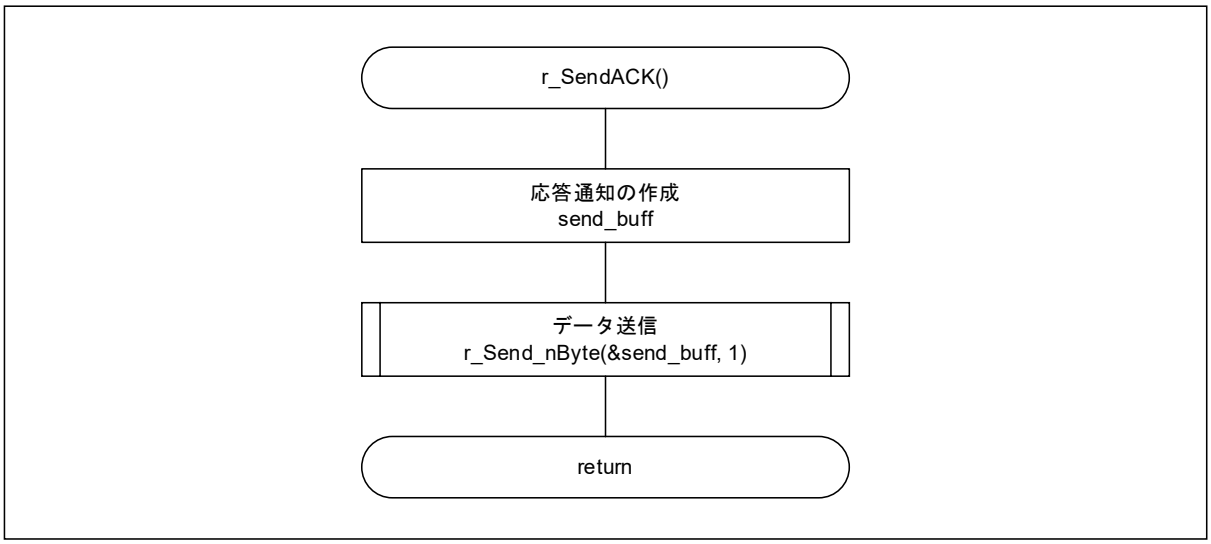
図 4-23　UART0 データ送信処理



4.10.20　　UART0 正常応答送信処理

図 4-24 にUART0 正常応答送信処理のフローチャートを示します。

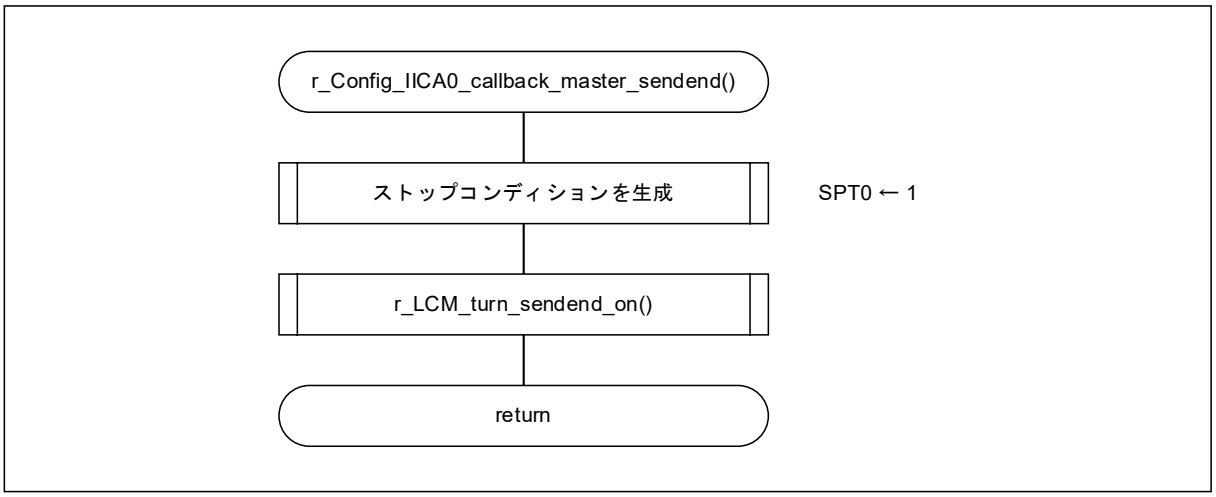
図 4-24　UART0 正常応答送信処理



4.10.21　　IICA0 送信完了割込み時の処理

図 4-25 に IICA0 送信完了割込み時の処理のフローチャートを示します。

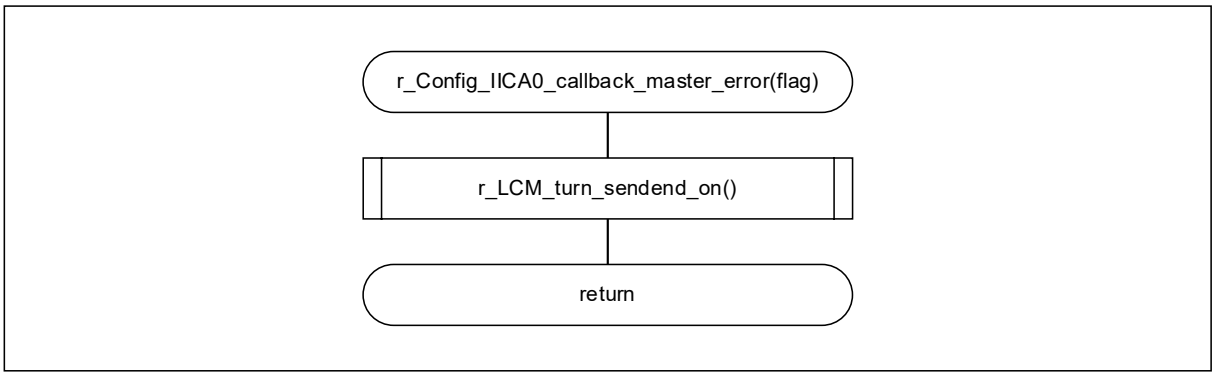
図 4-25　IICA0 送信完了割込み時の処理



4.10.22　　IICA0 送信エラー発生時の処理

図 4-26 に IICA0 送信エラー発生時の処理のフローチャートを示します。

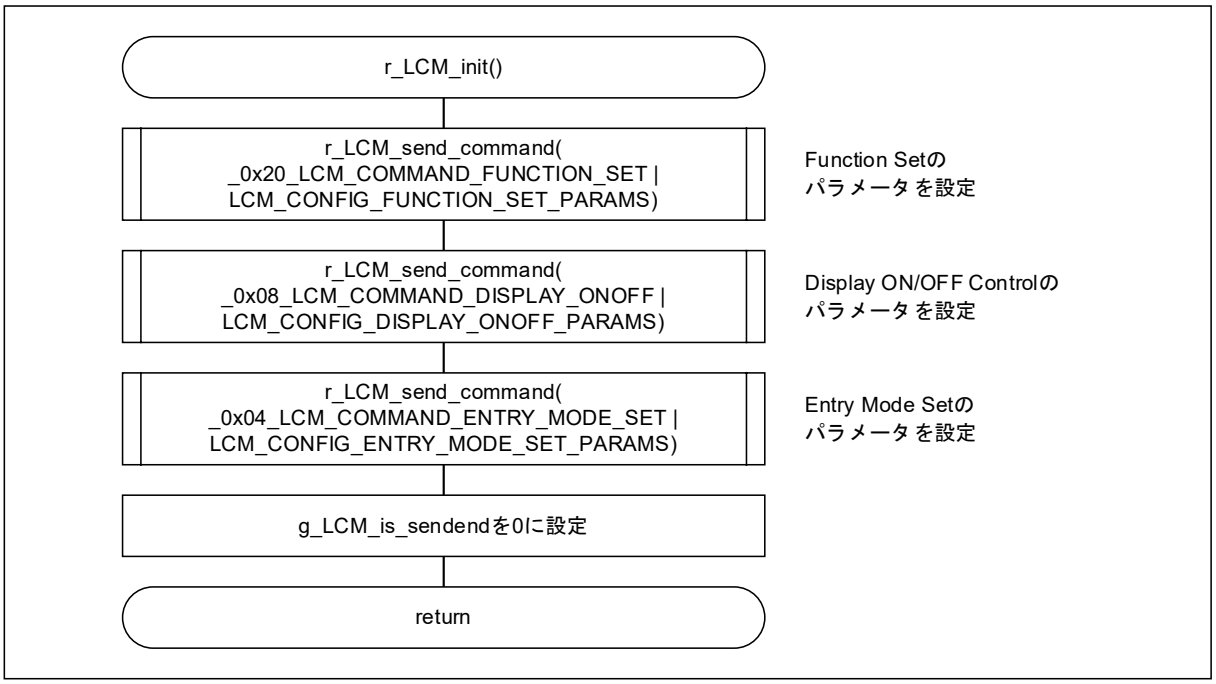
図 4-26　IICA0 送信エラー発生時の処理



4.10.23　　LCD モジュール 初期化

図 4-27 に LCD モジュール初期化のフローチャートを示します。

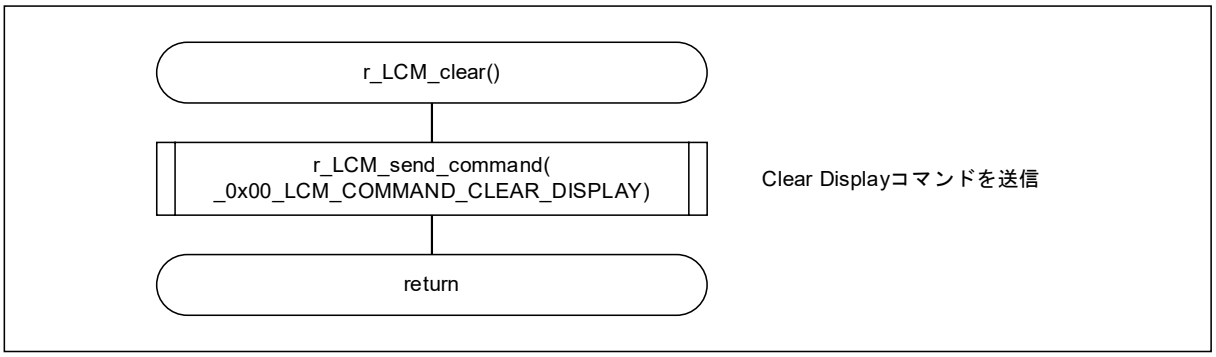
図 4-27　LCD モジュール初期化



4.10.24　　LCD モジュール 表示消去処理

図 4-28 に LCD モジュール 表示消去処理のフローチャートを示します。

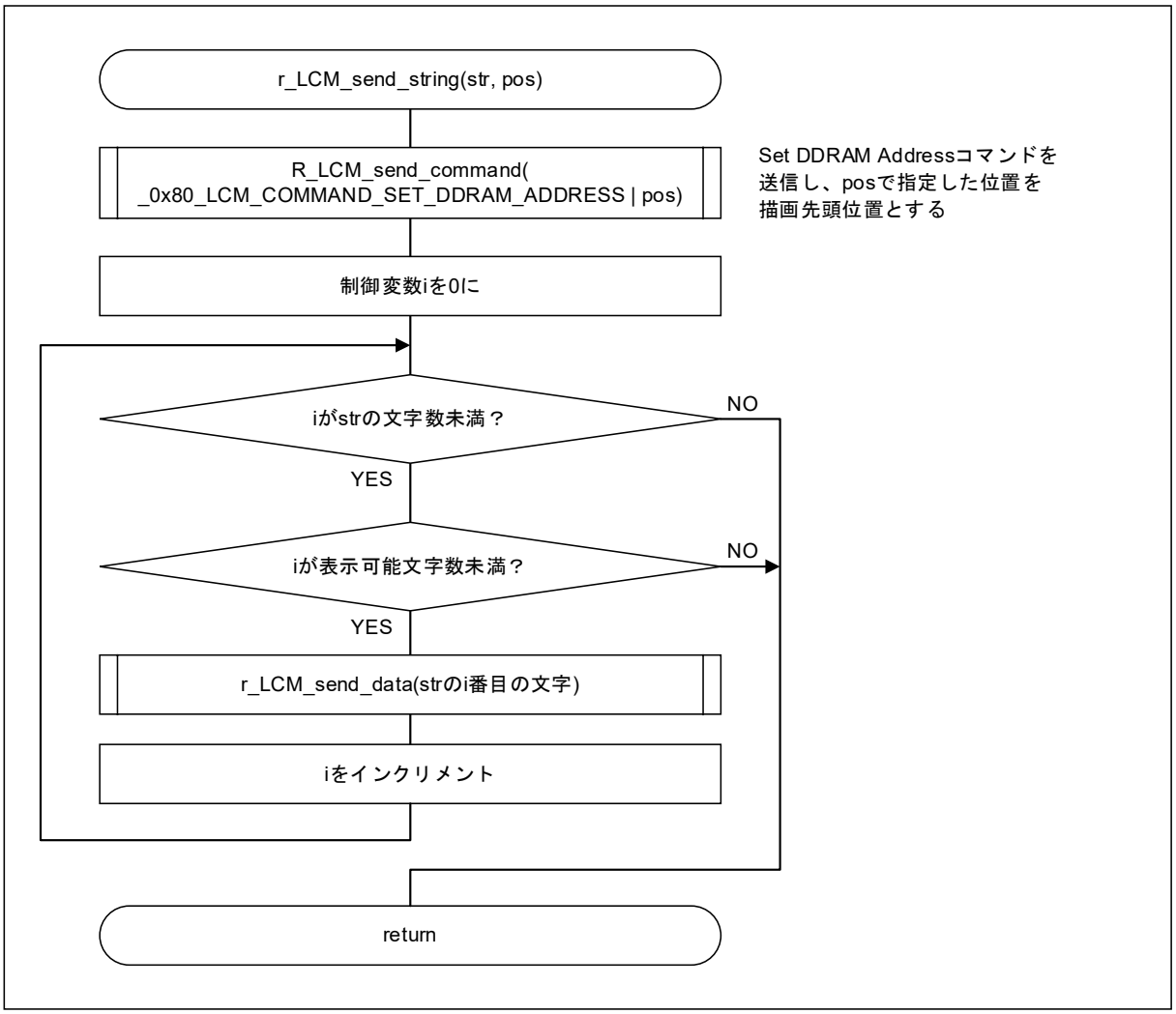
図 4-28　LCD モジュール 表示消去処理



4.10.25　　LCD モジュール 文字列送信処理

図 4-29 に LCD モジュール 文字列送信処理のフローチャートを示します。

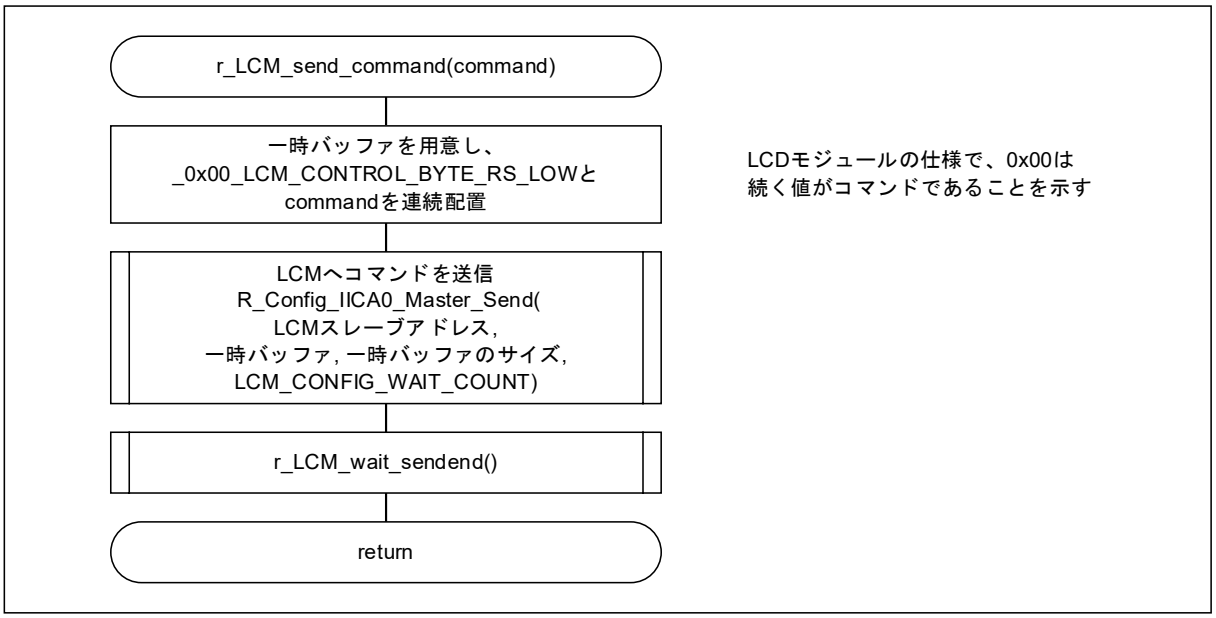
図 4-29　LCD モジュール 文字列送信処理



4.10.26　　LCD モジュール コマンド送信処理

図 4-30 に LCD モジュール コマンド送信処理のフローチャートを示します。

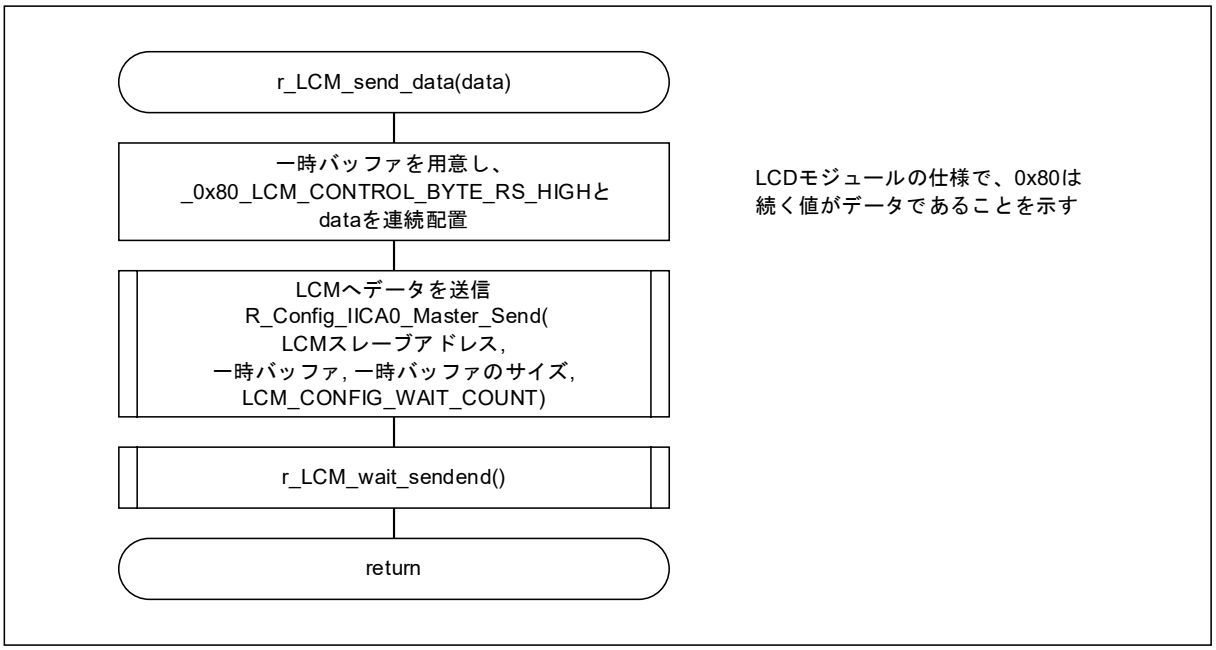
図 4-30　LCD モジュール コマンド送信処理



4.10.27　　LCD モジュール データ送信処理

図 4-31 に LCD モジュール データ送信処理のフローチャートを示します。

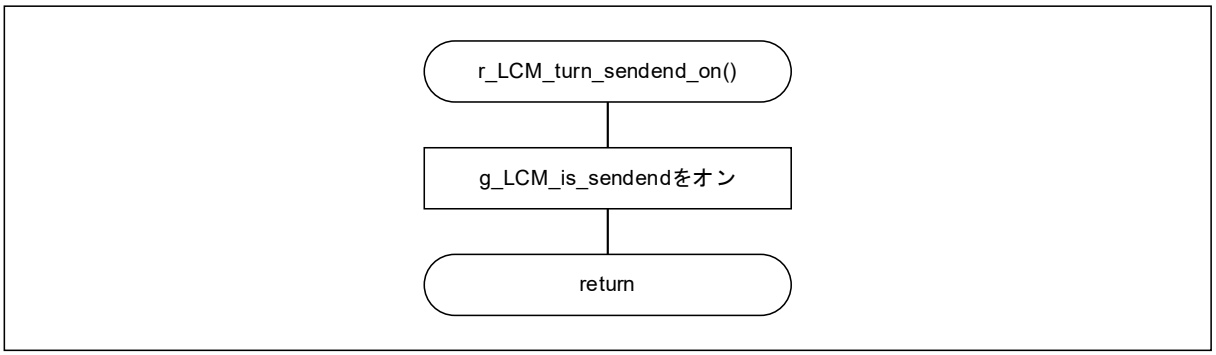
図 4-31　LCD モジュール データ送信処理



4.10.28　　LCD モジュール 通信終了フラグ設定

図 4-32 に LCD モジュール 通信終了フラグ設定のフローチャートを示します。

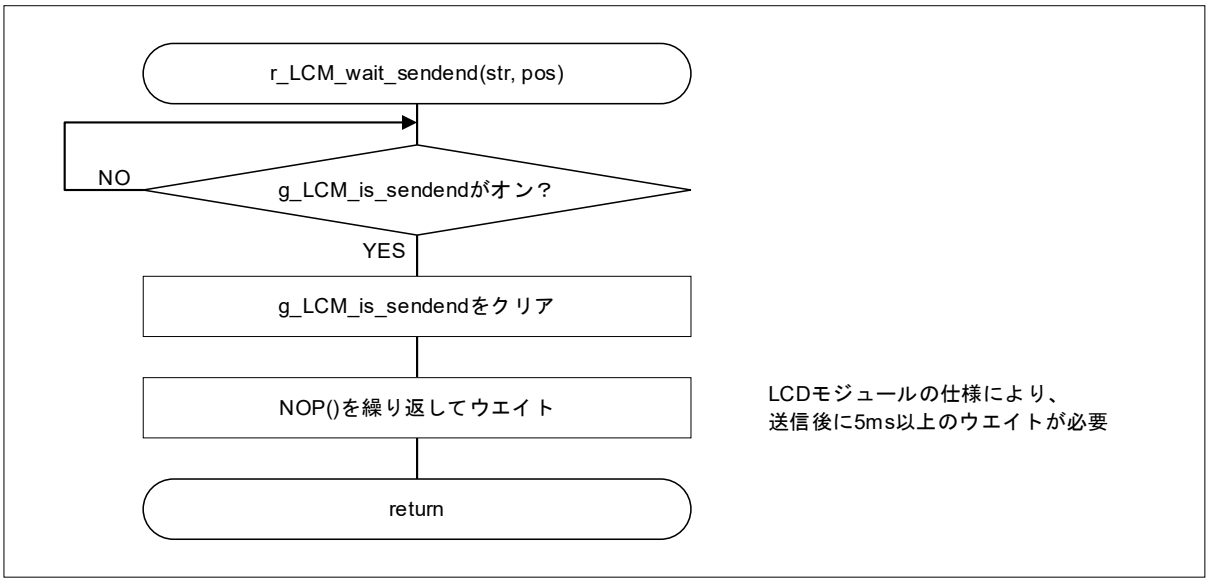
図 4-32　LCD モジュール 通信終了フラグ設定



4.10.29　　LCD モジュール 通信終了待ち処理

図 4-33 に LCD モジュール 通信終了待ち処理のフローチャートを示します。

図 4-33　LCD モジュール 通信終了待ち処理



5.　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。

6.　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	2021.03.22	—	初版発行
1.01	2021.06.04	14	動作確認条件を更新

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。