RX63T グループ
USB ホストフラッシュブートローダ

R01AN1418JJ0100
Rev.1.00
2014.05.14

要旨

本アプリケーションノートでは、RX63Tグループの USB2.0 ホスト／ファンクションモジュールを使用し、シングルチップモードでホスト動作させ、USB 経由で内蔵フラッシュメモリの書き換えを行う「USB ホストフラッシュブートローダ」について説明します。

なお、本アプリケーションノートでは以下のアプリケーションノートのサンプルコードおよびドライバを使用しています。「ルネサス USB デバイス USB Host Mass Storage Class Driver」に「ルネサス USB デバイス USB Basic Firmware」、「M3S-TFAT-Tiny：FAT ファイルシステムソフトウェア」が同梱されています。

内蔵フラッシュメモリの消去／書き込み：
「RX600＆RX200 シリーズ RX 用のシンプルフラッシュ API」Rev.2.40 (R01AN0544JU0240)
USB 通信：
「ルネサス USB デバイス USB Basic Firmware」Rev.2.00 (R01AN0512JJ0200)
「ルネサス USB デバイス USB Host Mass Storage Class Driver」Rev.2.00 (R01AN0513JJ0200)
FAT ファイルシステム：
「M3S-TFAT-Tiny：FAT ファイルシステムソフトウェア」Rev.1.00 (R20AN0038JJ0100)

本アプリケーションノートの特長を以下に示します。

USB メモリに格納したモトローラ S フォーマットのプログラムを書き込み可能。
モトローラ S フォーマットのプログラムを格納した、USB マスストレージデバイス（USB メモリ）の接続を認識すると、マイコン内蔵フラッシュメモリを消去し、プログラムを書き込みます。
書き込んだプログラムの実行が可能。
マイコン内蔵フラッシュメモリに書き込んだモトローラ S フォーマットのプログラムを実行することが出来ます。

USB 仕様
USB2.0 規格のフルスピード転送に対応しています。
USB マスストレージクラスの Bulk-Only Transport(BOT)に対応しています。
USB マスストレージサブクラスの SFF-8070i（ATAPI）および SCSI に対応しています。

対象デバイス

RX63Tグループ　144,120pin

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

目次

1. 仕様

本アプリケーションノートのサンプルコードは、RX63T-H-RSK 上で動作します。

RX63T-H-RSK に接続されたスイッチ（SW3）を押していない状態でリセットを解除すると、接続された USB メモリ内のモトローラ S フォーマットのプログラム（ファイル名：download.mot）を内蔵フラッシュメモリに書き込みます。書き込み終了後、SW3 を押した状態でリセットを解除すると、内蔵フラッシュメモリに書き込んだプログラム（以降：ダウンロードコード）を実行します。

なお、サンプルコードが書き換える領域はユーザマットの一部のみとなります。サンプルコードが使用している領域の書き換えは行いません。詳細は「5.1.動作概要」を参照ください。

内蔵フラッシュメモリへの書き込み状況および結果は、RX63T-H-RSK に接続された、LED、LCD に表示します。表示の内容は「5.5.サンプルコードの LED,LCD 表示」を参照ください。

表 1.1に使用する周辺機能と用途を、図 1.1 に使用例を示します。

表1.1　使用する周辺機能と用途

周辺機能	用途
ROM（コード格納用フラッシュメモリ）	ROM P/E モードによる 内蔵フラッシュメモリの書き換え
USB2.0 ホスト／ファンクションモジュール	USB メモリとの通信

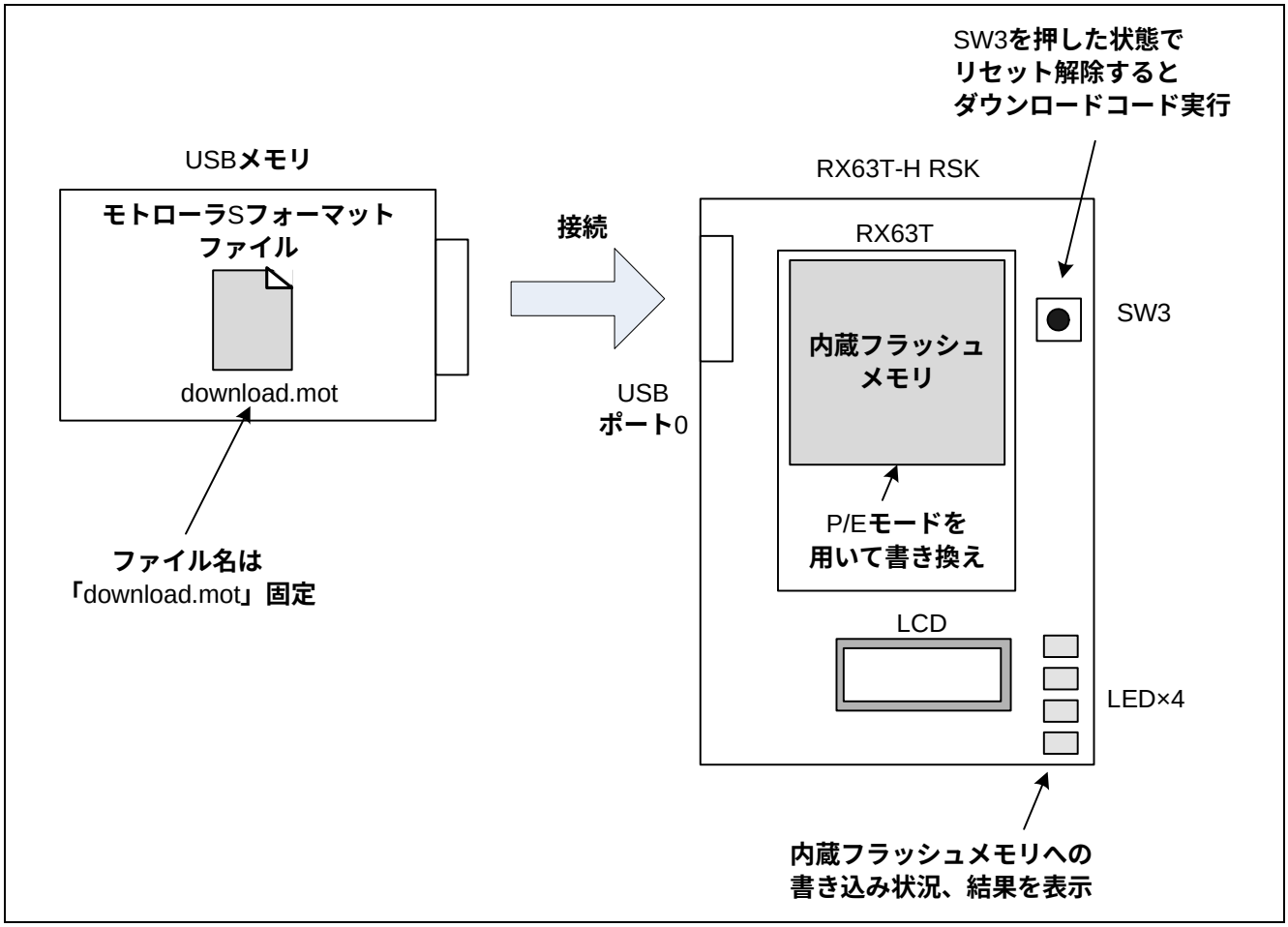

図1.1　使用例

2. 動作確認条件

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

表2.1 動作確認条件

項目	内容
使用マイコン	R5F563TEADFB（RX63T グループ）
動作周波数	• メインクロック: 12MHz • PLL: 192MHz (メインクロック 1 分周 16 逓倍) • システムクロック(ICLK): 96MHz (PLL 2 分周) • 周辺モジュールクロック B (PCLKB): 48MHz (PLL 4 分周) • USB に供給される USB クロック (UCLK): 48MHz (PLL4 分周) • FlashIF クロック (FCLK): 48MHz (PLL4 分周)
動作電圧	5.0V
統合開発環境	ルネサスエレクトロニクス製 High-performance Embedded Workshop Version 4.09.01.007
C コンパイラ	ルネサスエレクトロニクス製 RX Standard Toolchain Version 1.2.1.0
	-cpu=rx600 -include="$(WORKSPDIR)¥WorkSpace¥ANSI" -include="$(WORKSPDIR)¥WorkSpace¥SmplMain¥APL" -include="$(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥inc" -include="$(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥USBHW" -include="$(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥USBHW¥DEF" -include="$(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥USBHW¥REG" -include="$(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥USRCFG" -include="$(WORKSPDIR)¥WorkSpace¥MSCFW¥include" -include="$(WORKSPDIR)¥WorkSpace¥MSCFW¥TFAT¥lib_src" -include="$(WORKSPDIR)¥WorkSpace¥USBSTDFW¥include" -include="$(WORKSPDIR)¥WorkSpace¥FLASH" -include="$(WORKSPDIR)¥WorkSpace¥FLASH¥src" -include="$(WORKSPDIR)¥WorkSpace¥FLASH¥r_bsp" -include="$(WORKSPDIR)¥WorkSpace¥FLASH¥r_bsp¥mcu¥rx63t" -include="$(WORKSPDIR)¥WorkSpace¥FLASH¥r_bsp¥board¥rskrx63t_144pin" -define=USB_FW_PP=USB_FW_NONOS_PP,USB_TFAT_USE_PP=1,R_FLASH_USB -output=obj="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -nostuff -listfile="$(CONFIGDIR)¥$(FILELEAF).lst" -optimize=0 -nologo
iodefine.h のバージョン	0.50
エンディアン	リトルエンディアン
動作モード	シングルチップモード
サンプルコードのバージョン	Version 1.00
使用ボード	Renesas Starter Kit for RX63T-H

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- ルネサス USB デバイス USB Basic FirmwareRev.2.00 (R01AN0512JJ)
- ルネサス USB デバイス USB Host Mass Storage Class DriverRev.2.00 (R01AN0513JJ)
- M3S-TFAT-Tiny：FAT ファイルシステムソフトウェア Rev.1.00 (R20AN0038JJ)
- RX600&RX200 シリーズ RX 用のシンプルフラッシュ APIRev.2.40 (R01AN0544JU)

4. ハードウェア説明

4.1 使用端子一覧

表 4.1に使用する周辺機能と用途を示します。

表4.1 使用端子と機能

端子名	入出力	内容
USB0_DP	入出力	ポート 0 USB 内蔵トランシーバ D+入出力端子 USB バスの D+端子に接続
USB0_DM	入出力	ポート 0 USB 内蔵トランシーバ D-入出力端子 USB バスの D-端子に接続
P13/USB0_VBUSEN	出力	ポート 0 外部電源チップへの VBUS（5V）の供給許可信号
PE1/USB0_OVRCURA	入力	ポート 0 外部オーバカレント検出信号を接続します。また OTG 電源チップとの接続時には VBUS コンパレータ信号を接続します
PE3	入力	サンプルコードのモード選択用端子
P71	出力	LED0 接続端子
P72	出力	LED1 接続端子
P73	出力	LED2 接続端子
P33	出力	LED3 接続端子
PG4	出力	LCD モジュール制御用端子
PG5	出力	LCD モジュール制御用端子
PG0	出力	LCD モジュール制御用端子
PG1	出力	LCD モジュール制御用端子
PG2	出力	LCD モジュール制御用端子
PG3	出力	LCD モジュール制御用端子

5. ソフトウェア説明

5.1 動作概要

5.1.1 リセット解除後の動作

サンプルコードは、マイコンのリセット解除後に SW3(PE3)の状態をチェックします。このとき SW3 が押していない状態(PE3＝H)であれば、USB ホストフラッシュブートローダを実行し、USB 経由で内蔵フラッシュメモリを書き換えます。また、SW3 を押した状態(PE3＝L)であればダウンロードコードを実行します。

図 5.1にリセット解除後の動作を示します。

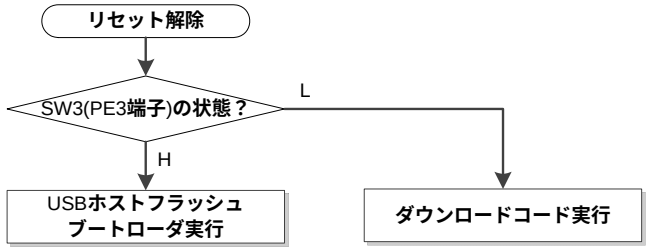

図5.1　リセット解除後の動作

5.1.2 書き換え対象

USB ホストフラッシュブートローダが書き換える対象はユーザマットの一部（以降：ダウンロードエリア）のみとなります。サンプルコードが使用している領域（FFFE0000h～FFFFFFFFh）の書き換えは行いません。図 5.2にメモリ配置を示します。

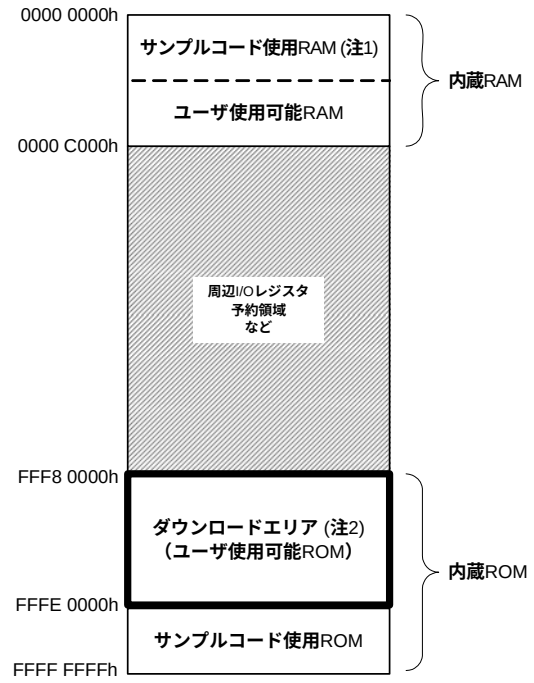

図5.2　メモリ配置

5.1.3　ダウンロードエリアの書き換え

USB ホストフラッシュブートローダは、以下の手順でダウンロードエリアの書き換えを行います。図 5.3 にダウンロードエリアの書き換え動作を示します。

① USB デバイスの接続を監視します。
② USB の接続を検出すると、接続されたデバイスの情報を取得し、アクセス可能か判定します。
③ 接続された USB デバイス（USB メモリ）へのアクセスが可能ならば、ファイル名「download.mot」のモトローラ S フォーマットファイルを検索します。
④ ファイル名「download.mot」のモトローラ S フォーマットファイルを確認すると、ダウンロードエリアを消去します。
⑤ ダウンロードエリアの消去後、USB メモリ内のモトローラ S フォーマットファイルから 2048byte 分データを読み出し、内蔵 RAM に格納します。
⑥ 内蔵 RAM へのデータ格納後、データの解析処理を行い、128byte 単位でダウンロードエリアに書き込みます。
⑦ 全データの書き込みを完了するまで、⑤、⑥の処理を繰り返します。
なお、モトローラ S フォーマットファイルの終端は、エンドレコード（S7,S8,S9 レコード）で判定しています。
⑧ 正常にダウンロードエリアの消去/書き込み処理を完了すると、I/O ポートに接続された LED、LCD で正常終了を知らせます。

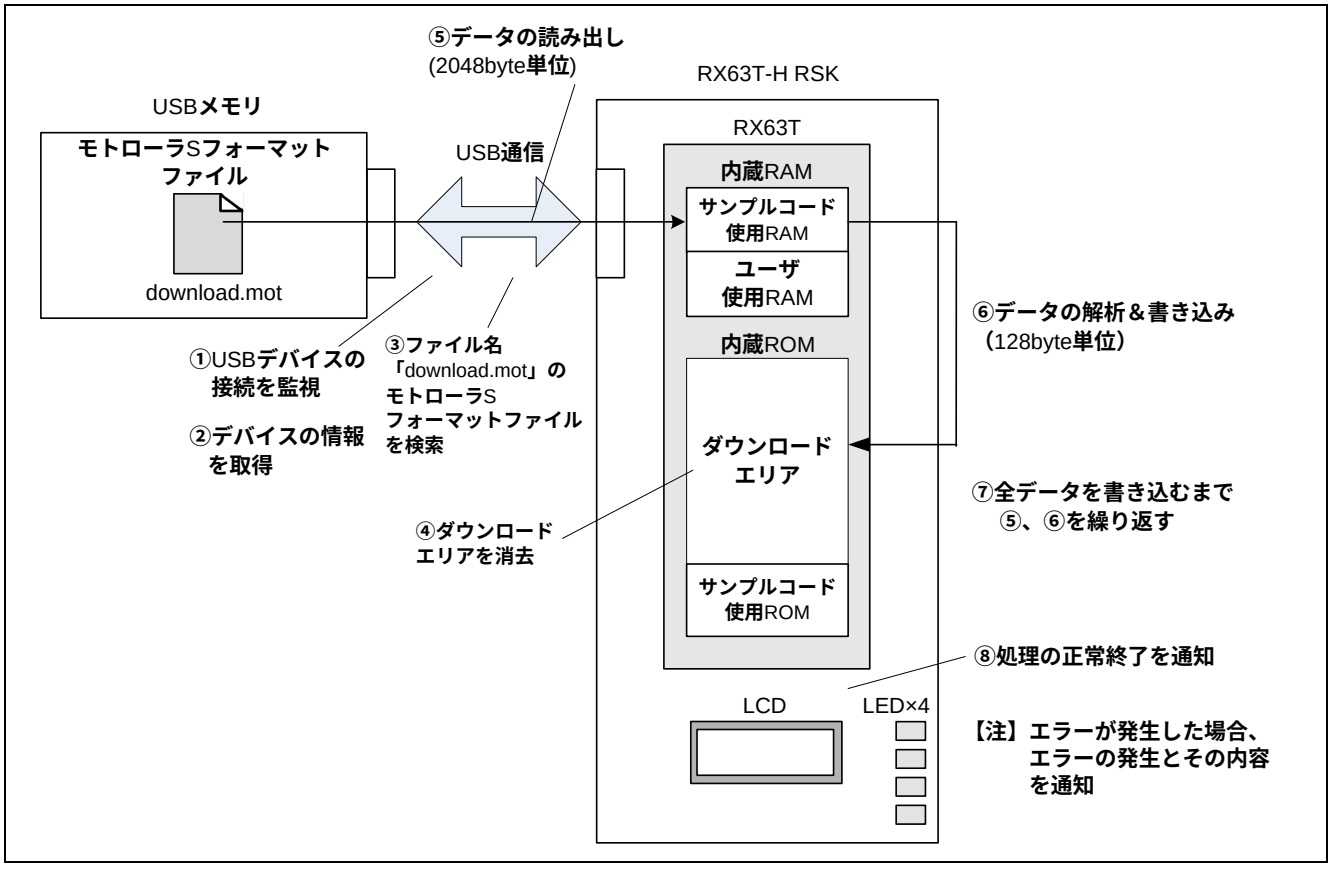

図5.3　ダウンロードエリアの書き換え動作

【注】　サンプルコード実行中にエラーが発生した場合は、LED、LCD でそのエラー内容を通知し、処理を終了します。エラーの発生条件および LED、LCD 表示については「5.5.サンプルコードの LED,LCD 表示」を参照ください。

5.2 ダウンロードコードの実行開始位置

サンプルコードは、マイコンのリセット解除後 SW3 の状態が L である場合、ダウンロードコードを実行します。この時サンプルコードは、アドレス”FFFD FFFCh”に書かれているアドレス番地から実行します。つまり、ダウンロードコードにとってのリセットベクタが” FFFD FFFCh”になります。ダウンロードコードでは、予め” FFFD FFFCh”に開始アドレスが格納されるようにしておいて下さい。

図 5.1にダウンロードコードのリセットベクタを示します。

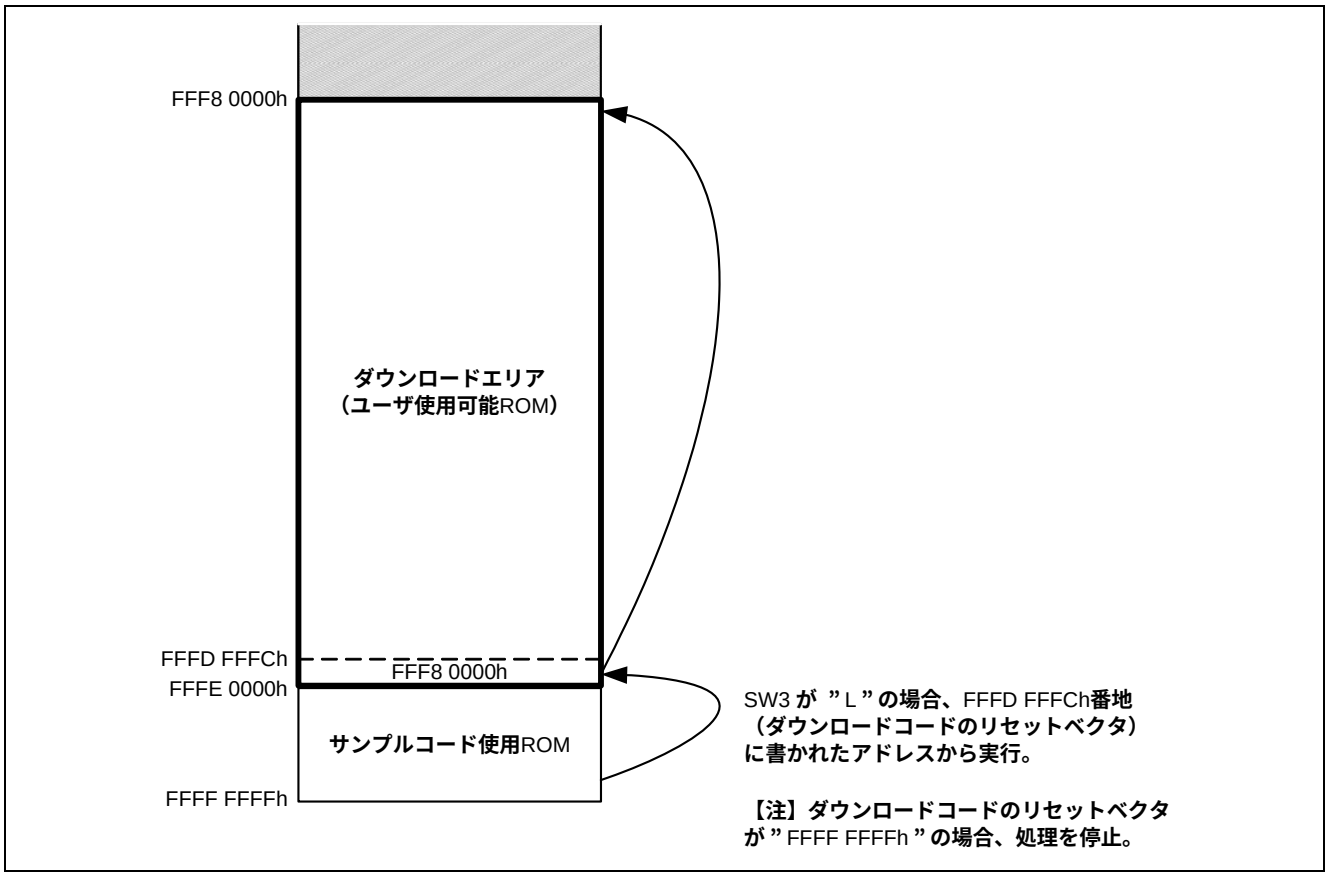

図5.4 ダウンロードコードのリセットベクタ

【注】 ダウンロードコードのリセットベクタに何も書かれていない場合（ダウンロードコードのリセットベクタが”FFFF FFFFh”の場合）は、while(1)で自番地ループを行い処理を停止します。

5.3 サンプルコードのソフトウェア構成

サンプルコードでは、USB との通信に、

「ルネサス USB デバイス USB Basic Firmware」と

「ルネサス USB デバイス USB Host Mass Storage Class Driver」を、

FAT ファイルシステムとして

「M3S-TFAT-Tiny：FAT ファイルシステムソフトウェア」

を使用しています。

また、内蔵フラッシュメモリの消去処理および書き込み処理には、

「RX600&RX200 シリーズ RX 用のシンプルフラッシュ API」を、

を使用しています。

図 5.5にサンプルコードのソフトウェア構成を、表 5.1にソフトウェアの概要を示します。

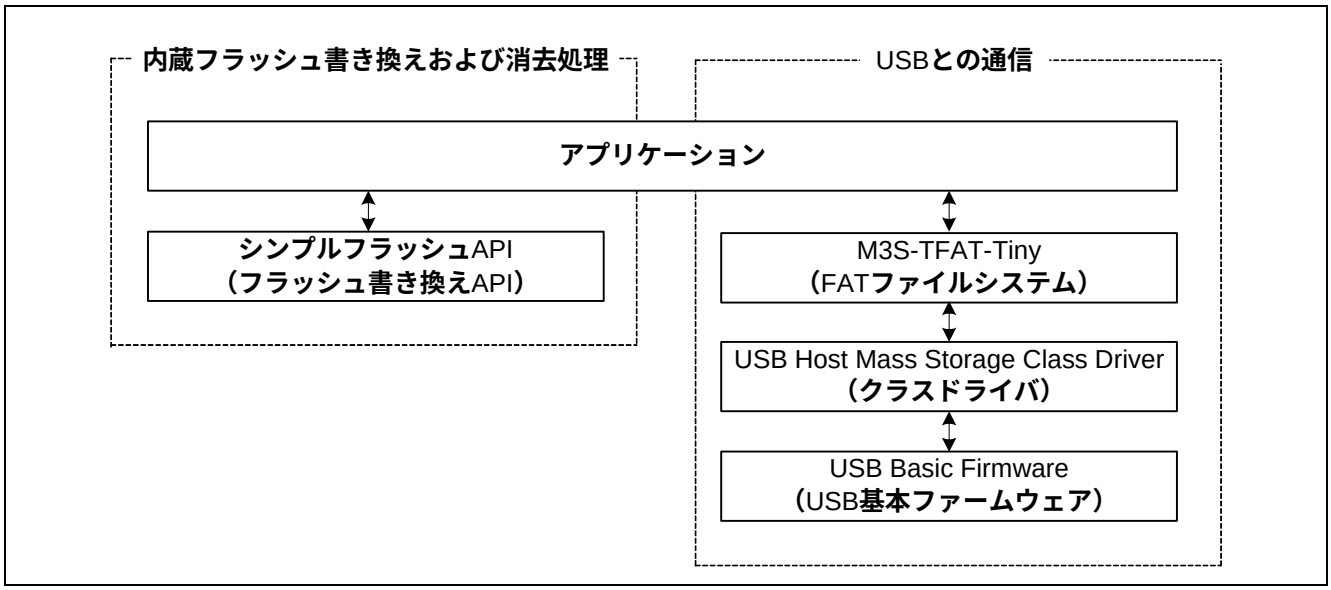

図5.5 サンプルコードのソフトウェア構成

表5.1 ソフトウェアの概要

モジュール名	概要
アプリケーション	FAT ライブラリ関数を使用し、USB メモリ内のモトローラ S フォーマットのプログラムを読み出します。また、シンプルフラッシュ API 関数を使用して、内蔵フラッシュメモリの消去、書き換えを行います。
RX 用のシンプルフラッシュ API	内蔵フラッシュメモリの消去/書き換えを行う API です。
M3S-TFAT-Tiny	FAT12、FAT16 に対応した FAT ファイルシステムです。
USB Host Mass Storage Class Driver	USB マスストレージクラスの Bulk-Only Transport (BOT)プロトコルに対応したクラスドライバです。
USB Basic Firmware	USB インタフェース制御用のサンプルプログラムです。

5.4　書き込み時のデータの流れ

ダウンロードコード書き込み時における、マイコン内部のデータの流れを図 5.6に示します。

① USB ドライバにより取得したデータを受信用リングバッファに転送します。
② モトローラ S フォーマットの 1 レコードをモトローラ S フォーマット用バッファ（ASCII）にコピーします。
③ モトローラ S フォーマットのヘッダ部分を解析すると同時に、ASCII コードのデータを Binary データに変換し、モトローラ S フォーマット用バッファ（Binary）に格納します。
なお、本アプリケーションノートで対応するモトローラ S フォーマットの仕様は、「7. 」を参照ください。
④ 書き込み用バッファにデータを格納します。

RX63Tのユーザマットへの書き込み単位は 128byte となっています。そのため、サンプルコードでは、ユーザマットへの一回の書き込みサイズ 128byte とし、書き込みバッファに格納される書き込みデータの合計が 128byte になるまで②～④の処理を繰り返しています。書き込みデータの合計が 128byte を超えてしまった場合は、超えた分のデータを一時保存しておき、次回の 128byte 書き込み時に使用しています。

⑤ 準備された書き込みデータ(128byte)を、シンプルフラッシュ API を使用してフラッシュメモリへ書き込みます。書き込み後、一時保存バッファに格納したデータサイズが書き込み単位より大きい場合は、④に戻り、続けて書き込みます。

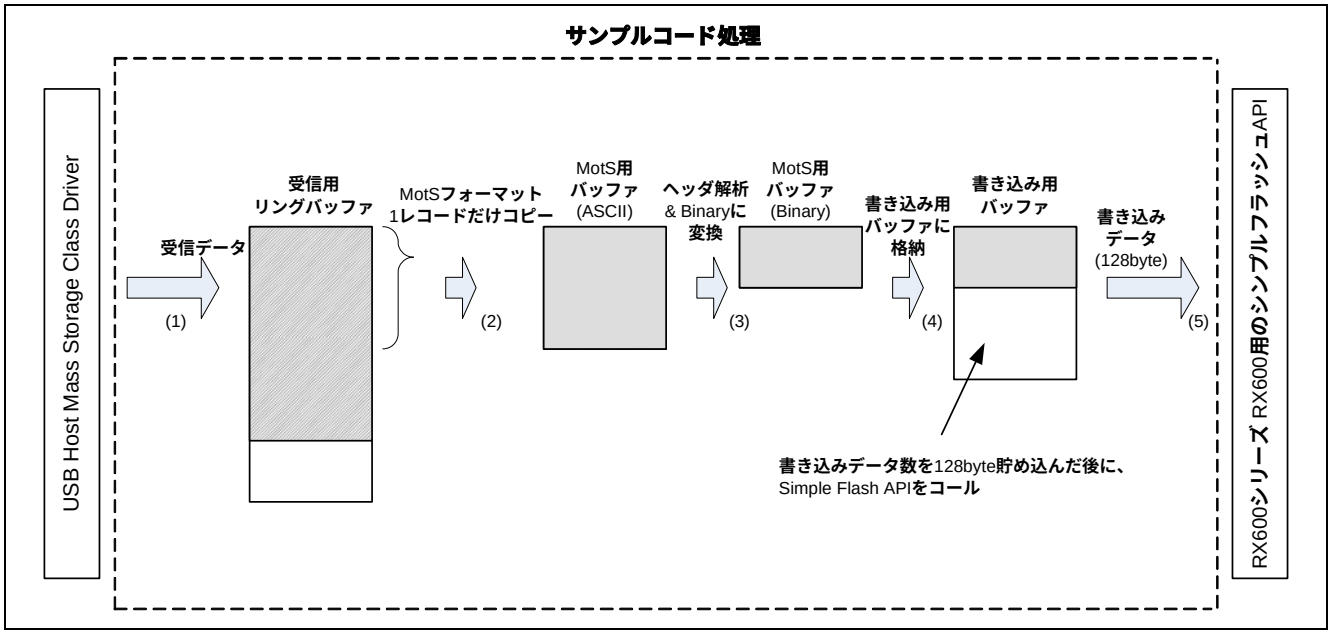

図5.6　書き込み時のデータの流れ

図 5.7に書き込み時のデータ構造を示します。

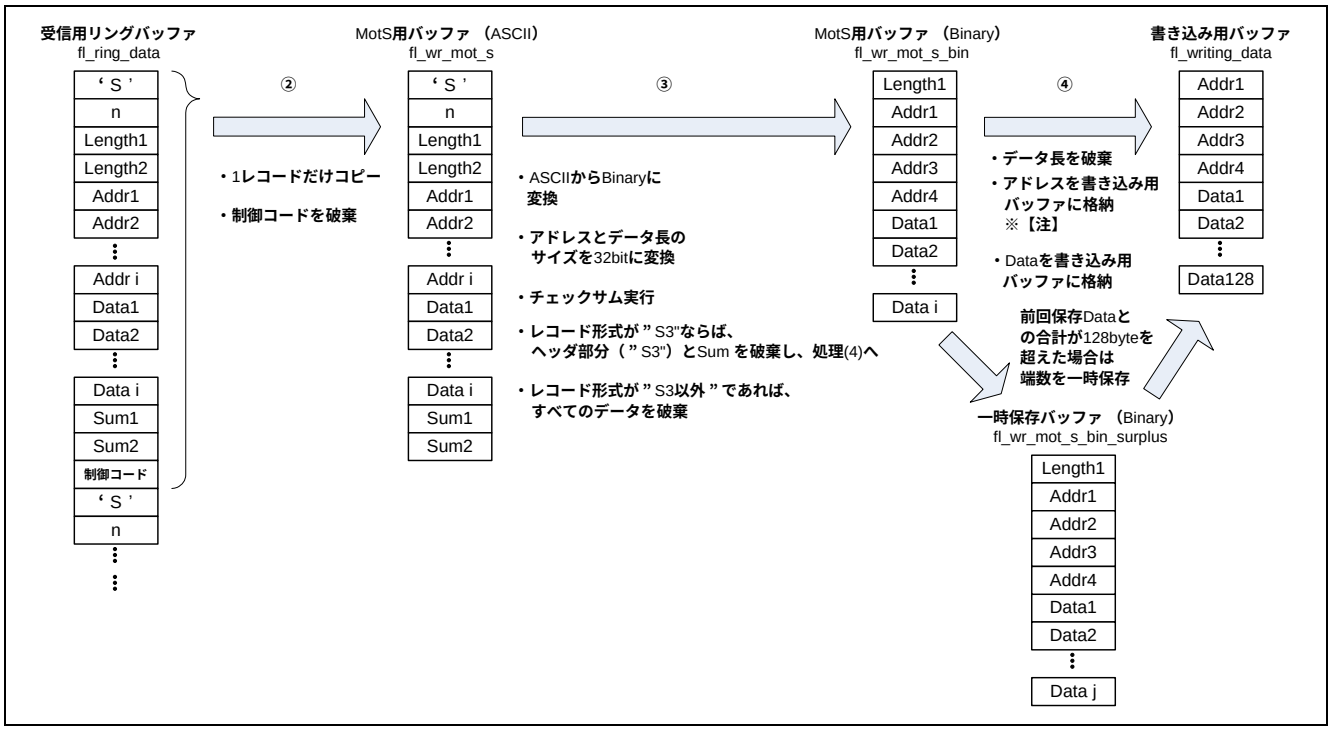

図5.7　書き込み時のデータ構造

【注】 RX63Tグループの内蔵フラッシュメモリは、書き込み対象の先頭アドレスを 128byte 境界にそろえる必要があるため、サンプルコードでは書き込み用バッファにアドレスを格納する際、書き込みアドレスの先頭が 128byte 境界となるよう処理を行っています。処理の詳細は、「5.14.11.ダウンロードエリアへの書き込みデータ作成」のフローチャートを参照ください。

5.5 サンプルコードの LED,LCD 表示

サンプルコードではプログラムの進行状況および結果を RX63T-H-RSK に接続された LED、LCD に表示します。

表 5.2 にサンプルコードの LED 表示一覧を示します。

表5.2　サンプルコードの LED 表示一覧

○：点灯、●：消灯

LED 表示					内容
LED3	LED2	LED1	LED0	順序	
●	●	●	●	↓	**ダウンロード処理の実行中、LED は 2 進カウントアップ表示を行います。** **LED 表示は、RX63T グループのユーザマットへの書き込み単位である** 128byte×128（16K byte **書き込み）ごとに更新されます。**
●	●	●	○		
●	●	○	●		
●	●	○	○		
●	○	●	●		
●	○	●	○		
●	○	○	●		
●	○	○	○		
○	●	●	●		
○	●	●	○		
○	●	○	●		
○	●	○	○		
○	○	●	●		
○	○	●	○		
○	○	○	●		
○	○	○	○		
●	●	●	○	↓	**サンプルコードが正常終了した場合、LED はシフト表示を行います。LED 表示は、500ms 経過ごとに更新されます。**
●	●	○	●		
●	○	●	●		
○	●	●	●		
●	●	●	●	↓	**サンプルコードが異常終了した場合、LED はブリンク表示を行います。LED 表示は、500ms 経過ごとに更新されます。**
○	○	○	○		

表 5.3にサンプルコードの LCD 表示一覧を示します。

表5.3　サンプルコードの LCD 表示一覧

LCD 表示	内容
FLASH BOOT	リセット解除後、「USB ホストフラッシュブートローダ」が実行された場合に表示されます。
DETACH	USB を接続後、取り外した場合に表示されます。 ※【注】
ATTACH	リセット解除後、USB が接続された場合に表示されます。 ※【注】
FLASH UPDATE. . ..	ダウンロード処理の実行中に表示されます。
ERROR!! D OPEN	アクセス可能なドライブが検出されなかった場合に表示されます。（ドライブオープンエラー）
ERROR!! D MOUNT	ドライブのマウント処理に失敗した場合に表示されます。 （ドライブマウントエラー）
ERROR!! F OPEN	ファイルオープン処理に失敗した場合に表示されます。 （ファイルオープンエラー）
ERROR!! F READ	ファイルに読み出し処理に失敗した場合に表示されます。 （ファイルリードエラー）
ERROR!! F CLOSE	ファイルのクローズ処理に失敗した場合に表示されます。 （ファイルクローズエラー）
ERROR!! SUM	「7.モトローラ S フォーマット」を参照ください。 （チェックサムエラー）
ERROR!! MOTS	「7.モトローラ S フォーマット」を参照ください。 （フォーマットエラー）
ERROR!! ERASE	ダウンロードエリアの消去に失敗した場合に表示されます。 （イレーズエラー）
ERROR!! WRITE	ダウンロードエリアへの書き込みに失敗した場合に表示されます。（書き込みエラー）
ERROR!! ADDRESS	「7.モトローラ S フォーマット」を参照ください。 （アドレスエラー）
ERROR!! VERIFY	ダウンロードエリアへ書き込んだデータのベリファイ結果に異常があった場合に表示されます。（ベリファイエラー）
ERROR!! F END	FAT ライブラリでファイルの終端を検出したにもかかわらず、モトローラ S フォーマットのエンドコードを受信しなかった場合に表示されます。（ファイルエンドエラー）
ERROR!! ILL DET	ドライブオープン処理もしくは、ダウンロード処理実行中に USB のデタッチを検出した場合に表示されます。 （イリーガルデタッチエラー）
ERROR!! ENDIAN	サンプルコードとダウンロードコードのエンディアン（MDES の値）が不一致の場合に表示されます。 （エンディアンエラー）

【注】 USB 取り外し時の「DETACH」表示、USB 接続時の「ATTACH」表示は、「USB Host Mass Storage Class Driver」の仕様です。

5.6 必要メモリサイズ

表 5.4に必要メモリサイズを示します。

表5.4 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	121022 バイト	サンプルコードは、FFFE0000h ～ FFFFFFFFh に配置されているため書き換え可能な ROM 容量（ダウンロードエリアの容量）は「全体の ROM 容量 － 131072 バイト」となります。
RAM	38002 バイト	ダウンロードコード実行時、ユーザはこの領域を使用できます。
最大使用ユーザスタック	588 バイト	
最大使用割り込みスタック	152 バイト	

【注】 必要メモリサイズは C コンパイラのバージョンやコンパイルオプションにより異なります。

5.7 ファイル構成

表 5.5にサンプルコードで使用するファイルを示します。なお、統合開発環境で自動生成されるファイルは除きます。

表5.5 サンプルコードで使用するファイル

ファイル名	概要	備考
r_flash_api_rx.c	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム	詳細は RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のアプリケーションノートを参照してください。
locking.c	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム	
mcu_locks.c	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム	
r_flash_api_rx_if.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム(r_flash_api_rx.c)の外部参照用インクルードヘッダ	
r_flash_api_rx_private.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のパラメータ設定用インクルードヘッダ	
r_flash_api_rx_config.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のパラメータ設定用インクルードヘッダ	
r_bsp.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラムの外部参照用インクルードヘッダ	
r_bsp_config.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラムのパラメータ設定用インクルードヘッダ	
r_flash_api_rx63t.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラムのパラメータ設定用インクルードヘッダ	
mcu_info.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のパラメータ設定用インクルードヘッダ	
locking.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム(locking.c)の外部参照用インクルードヘッダ	
mcu_locks.h	RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のプログラム(mcu_locks.c)の外部参照用インクルードヘッダ	
r_Flash_main.c	フラッシュ書き換えデータ処理	
r_Flash_main.h	フラッシュ書き換えデータ処理の外部参照用インクルードヘッダ	
r_Flash_buff.c	USB との受信用バッファ関連処理	
r_Flash_buff.h	USB との受信用バッファ関連処理の外部参照用インクルードヘッダ	
TrgtPrgDmmy.c	ダウンロードコード用の領域を確保するためのダミープログラム	
その他ファイル	USB Host Mass Storage Class Driver のプログラム	詳細はルネサス USB デバイス USB Host Mass Storage Class Driver 及び USB Basic Firmware のアプリケーションノートを参照してください。

5.8 オプション設定メモリ

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

表5.6 サンプルコードで使用するオプション設定メモリ

シンボル	アドレス	設定値	内容
OFS0	FFFF FF8Fh - FFFF FF8Ch	FFFF FFFFh	リセット後、IWDT は停止 リセット後、WDT は停止
OFS1	FFFF FF8Bh – FFFF FF88h	FFFF FFFFh	リセット後、電圧監視リセット 0 無効
MDES(*1)	FFFF FF83h – FFFF FF80h	FFFF FFFFh FFFF FFF8h	（シングルチップモード時） リトルエンディアン ビッグエンディアン

【注】 *1 本サンプルコードの設定はリトルエンディアンです。エンディアンの切り替えは 8.7 エンディアンを参照ください。

5.9 定数一覧

表 5.7にサンプルコードで使用する定数 1、表 5.8にサンプルコードで使用する定数 2を示します。

表5.7 サンプルコードで使用する定数 1

定数名	設定値	内容
FL_MODE_ENTRY_WAIT_LCD_PERIOD	100000	モードエントリ時の待ち時間
FL_UPDATE_WAIT_LED_PERIOD	128	ダウンロード処理時の LED 表示間隔の時間
FL_ERROR_WAIT_LED_PERIOD	500	エラー処理時の LED 表示間隔の時間
FL_DONE_WAIT_LED_PERIOD	500	正常終了処理時の LED 表示間隔の時間
FL_RINGBUFF_SIZE	4096	USB からのデータ受信用リングバッファサイズ
FL_TARGET_REST_VECT_ADDR	FFFDFFFCh	ダウンロードコードのリセットベクタアドレス
FL_START_BLOCK_NUM	14	ダウンロードエリアの最初のブロック
FL_END_BLOCK_NUM	37	ダウンロードエリアの最後のブロック
FL_START_WRITE_ADDRESS	FFF80000h	ダウンロードエリアの先頭アドレス
FL_END_WRITE_ADDRESS	FFFDFFFFh	ダウンロードエリアの最後尾アドレス
MDES_START_ADDRESS	FFFFFF80h	MDES 先頭アドレス
MDES_END_ADDRESS	FFFFFF83h	MDES 終了アドレス
FL_UPDATE_FILE_NAME	"download.mot"	ダウンロードコードのファイル名
FL_MOTS_LNG_MAX_DATA	0xFF	モトローラ S フォーマットのデータ長最大値
FL_MOTS_LNG_SIZE	1	モトローラ S フォーマットのデータ長バッファサイズ
FL_MOTS_ADDR_MIN_SIZE	2	モトローラ S フォーマットのアドレスバッファ最小サイズ
FL_MOTS0_ADDR_SIZE	2	S0 モトローラ S フォーマットのアドレスバッファサイズ
FL_MOTS3_ADDR_SIZE	4	S3 モトローラ S フォーマットのアドレスバッファサイズ
FL_MOTS7_ADDR_SIZE	4	S7 モトローラ S フォーマットのアドレスバッファサイズ
FL_MOTS8_ADDR_SIZE	3	S8 モトローラ S フォーマットのアドレスバッファサイズ
FL_MOTS9_ADDR_SIZE	2	S9 モトローラ S フォーマットのアドレスバッファサイズ
FL_MOTS_SUM_SIZE	1	モトローラ S フォーマットデータのチェックサムバッファサイズ

表5.8 サンプルコードで使用する定数 2

定数名	設定値	内容
FL_USB_RCV_BLANK_SIZE	FL_RINGBUFF_SIZE / 2	リングバッファの補充可能サイズ
FL_PORT_MDE	PORTE.PIDR.BIT.B3	SW3 接続ポート（PE3）の PORT レジスタ
FL_ PDR_MDE	PORTE.PDR.BIT.B3	SW3 接続ポート（PE3）の PDR レジスタ
ROM_PROGRAM_SIZE	128(*2)	対象デバイスに応じた、ユーザマットへの書き込み単位が設定されます。r_flash_api_rx63t.h に記述されており、インクルードすることで参照するようにしています。

【注】 *2 RX63Tグループを対象デバイスにした場合の値になります。

5.10 変数一覧

表 5.9にstatic 型変数(1)を、表 5.10 にstatic 型変数(2)を示します。

表5.9 static 型変数(1)

型	変数名	内容	使用関数
static FIL	fl_file	作成するファイルオブジェクト構造体を指す変数※【注】	R_Fl_Flash_Update
static FATFS	fl_fatfs	登録する作業領域（ファイルシステムオブジェクト構造体）を指す変数 ※【注】	R_Fl_Flash_Update
static uint8_t	fl_usb_read_data[FL_RINGBUFF_SIZE / 2]	USB メモリから読み出したデータを格納する領域	R_Fl_Flash_Update
static Fl_prg_mot_s_t	fl_wr_mot_s	モトローラ S フォーマット格納用バッファ（ASCII）	R_Fl_Prg_ProcessForMotS_data、R_Fl_Prg_StoreMotS
static Fl_prg_mot_s_binary_t	fl_wr_mot_s_bin	モトローラ S フォーマット格納用バッファ（Binary）	R_Fl_Prg_ProcessForMotS_data、R_Fl_Prg_MakeWriteData
static Fl_prg_mot_s_binary_t	fl_wr_mot_s_bin_surplus	余剰（書き込み単位を超えた）データを一時的に保存する領域（Binary）	R_Fl_Prg_ProcessForMotS_data、R_Fl_Prg_MakeWriteData、R_Fl_Prg_ClearMotSVariables
static Fl_prg_writing_data_t	fl_writing_data	書き込み領域（アドレス）を示す変数	R_Fl_Prg_ProcessForMotS_data、R_Fl_Prg_MakeWriteData、R_Fl_Prg_WriteData、R_Fl_Prg_ClearMotSVariables
static uint8_t	fl_communication_start_flg	モトローラ S フォーマットのスタートコード（S0）検出フラグ 0：スタートコード未検出 1：スタートコード検出	R_Fl_Prg_ProcessForMotS_data
static uint8_t	fl_communication_end_flg	モトローラ S フォーマットのエンドコード（S7、S8、S9）検出フラグ 0：エンドコード未検出 1：エンドコード検出	R_Fl_PrgramTrgtArea、R_Fl_Prg_ProcessForMotS_data
static uint8_t	fl_tfat_eof_flg	ファイルの終端検出フラグ 0：ファイルの終端未検出 1：ファイルの終端検出	R_Fl_Flash_Update、R_Fl_PrgramTrgtArea

※【注】 詳細は、M3S-TFAT-Tiny：FAT ファイルシステムソフトウェアをご参照ください。

表5.10 static 型変数(2)

型	変数名	内容	使用関数
static uint8_t	fl_ring_data[FL_RINGBUFF_SIZE]	受信用リングバッファ	R_Fl_RingEnQueue、R_Fl_RingDeQueue
static uint32_t	fl_queue_head	受信用リングバッファの読み出し位置を示す変数	R_Fl_RingEnQueue、R_Fl_RingDeQueue
static uint32_t	fl_queue_num	受信用リングバッファに格納されているデータ数	R_Fl_RingEnQueue、R_Fl_RingDeQueue、R_Fl_RingCheck
static const uint8_t	fl_tbl_msg_drive_open_err[]	ドライブオープンエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_drive_mount_err[]	ドライブマウントエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_file_open_err[]	ファイルオープンエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_file_read_err[]	ファイルリードエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_file_close_err[]	ファイルクローズエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_program_erase_err[]	イレーズエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_program_write_err[]	書き込みエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_s_format_sum_err[]	チェックサムエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_s_format_mots_err[]	フォーマットエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_program_verify_err[]	ベリファイエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_program_address_err[]	アドレスエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_illegal_detach_err[]	イリーガルデタッチエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_tfat_end_err[]	ファイルエンドエラー表示用変数	R_Fl_Error
static const uint8_t	fl_tbl_msg_endian_err[]	エンディアンエラー表示用変数	R_Fl_Error
static const uint8_t	*p_fl_output_msg[]	エラーメッセージ表示用ポインタ変数	R_Fl_Error

5.11 構造体/共用体一覧

図 5.8にサンプルコードで使用する構造体/共用体を示します。

```
/* buffer for mot S format data */
typedef struct {
    uint8_t type[2];                /* "S0", "S1" and so on */
    uint8_t len[2];                 /* "00"-“ FF" */
    uint8_t addr_data_sum[(FL_MOTS_LNG_MAX_DATA-FL_MOTS_LNG_SIZE)*2];
} Fl_prg_mot_s_t;

/* buffer for write data
     (this data is the converted data from mot S format data) */
typedef struct {
    uint8_t len;
    uint32_t addr;
    uint8_t data[(FL_MOTS_LNG_MAX_DATA-FL_MOTS_LNG_SIZE-FL_MOTS_ADDR_MIN_SIZE];
} Fl_prg_mot_s_binary_t;

/* buffer for writing flash */
typedef struct {
    uint32_t addr;
    uint8_t data[ROM_PROGRAM_SIZE];
} Fl_prg_writing_data_t;
```

図5.8 サンプルコードで使用する構造体/共用体

5.12 関数一覧

表 5.11に関数を示します。ただし、USB Host Mass Storage Class Driver、シンプルフラッシュ API、FAT ファイルシステムソフトウェアで使用しているものは除きます。

表5.11 関数

関数名	概要
R_Fl_Mode_Entry	モード選択
R_Fl_Flash_Update	書き換え処理のメインとなる関数
R_Fl_EraseTrgtArea	消去処理
R_Fl_Ers_EraseFlash	ダウンロードエリア消去
R_Fl_PrgramTrgtArea	書き込み処理
R_Fl_Prg_PrgramTrgtArea	ダウンロードエリアへの書き込み
R_Fl_Prg_StoreMotS	モトローラ S フォーマットデータ格納
R_Fl_Prg_ProcessForMotS_data	モトローラ S フォーマットデータのヘッダ解析、Binary 変換、書き込み
R_Fl_Prg_MotS_AsciiToBinary	モトローラ S フォーマットデータの ASCII -Binary 変換
R_Fl_Prg_MakeWriteData	ダウンロードエリアへの書き込みデータ作成
R_Fl_Prg_WriteData	ダウンロードエリアへの書き込み
R_Fl_Prg_ClearMotSVariables	モトローラ S フォーマットデータ関連の変数クリア
R_Fl_Run_StopUSB	USB 停止
R_Fl_RcvDataString	USB 受信データ格納
R_Fl_Error	エラー処理
R_Fl_RingCheckBlank	受信用リングバッファの空き容量確認
R_Fl_RingEnQueue	USB 受信データ格納バッファへのデータ格納
R_Fl_RingDeQueue	USB 受信データ格納バッファからのデータ読み出し
R_Fl_RingCheck	USB 受信データ格納バッファデータ数確認
R_Fl_AsciiToHexByte	ASCII コードを Binary データへ変換

5.13 関数仕様

サンプルコードの関数仕様を示します。

R_Fl_Mode_Entry	
概　要	モードエントリ
ヘッダ	r_Flash_main.h
宣　言	void R_Fl_Mode_Entry(void)
説　明	SW3 の状態をチェックします。 SW3 が押されていない状態であれば「USB ホストフラッシュブートローダ」を実行します。 SW3 が押されている状態であれば「ダウンロードコード」を実行します。
引　数	なし
リターン値	なし

R_Fl_Flash_Update	
概　要	書き換え処理メイン
ヘッダ	r_Flash_main.h
宣　言	void R_Fl_Flash_Update (void)
説　明	ダウンロードエリアの消去処理を行う関数をコールします。 FAT ライブラリ関数をコールし、USB メモリ内のモトローラ S フォーマットファイルからデータを読み出します。 モトローラ S フォーマットデータの解析/ダウンロードエリアの書き換え処理を行う関数をコールします。 FAT ライブラリ関数の実行に失敗した場合、エラー関数をコールします。
引　数	なし
リターン値	なし

R_Fl_EraseTrgtArea	
概　要	消去処理
ヘッダ	なし
宣　言	static void R_Fl_EraseTrgtArea(void)
説　明	ダウンロードエリアを消去する関数を呼び出します。 ダウンロードエリアの消去に失敗した場合、エラー関数をコールします。
引　数	なし
リターン値	なし

R_Fl_Ers_EraseFlash	
概　要	ダウンロードエリア消去
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Ers_EraseFlash(void)
説　明	ダウンロードエリアを消去します。
引　数	なし
リターン値	消去が正常に完了した場合：FLASH_API_SAMPLE_OK 消去が正常に完了しなかった場合：FLASH_API_SAMPLE_NG
備考	消去中の割り込みによる ROM アクセスを防ぐため、プロセッサステータスワード（PSW）のプロセッサ割り込み優先レベル（IPL）を変更します。

R_Fl_PrgramTrgtArea	
概　要	ダウンロードエリアへの書き込み
ヘッダ	なし
宣　言	static void R_Fl_PrgramTrgtArea (void)
説　明	書き込み処理を行う関数を呼び出します。 モトローラ S フォーマットのエンドレコードを受信せず、ファイル終端に達した場合、エラー関数をコールします。
引　数	なし
リターン値	なし

R_Fl_ Prg_PrgramTrgtArea	
概　要	書き込み処理
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Prg_PrgramFlash(void)
説　明	受信用リングバッファにデータがあった場合、モトローラ S フォーマットのレコード一つ分を格納する関数を呼び出します。 モトローラ S フォーマットのレコードを一つ分格納したら、ヘッダ解析、バイナリデータへの変換、ダウンロードエリアへの書き込みを行う関数をコールします。
引　数	なし
リターン値	ダウンロードエリアへの書き込みが完了した場合：FLASH_API_SAMPLE_OK ダウンロードエリアへの書き込みが未完了の場合：FLASH_API_SAMPLE_NG

R_Fl_Prg_StoreMotS	
概　要	モトローラ S フォーマットデータ格納
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Prg_StoreMotS(uint8_t)
説　明	引数で受け取ったデータをモトローラ S フォーマットデータとして 1byte ずつ格納します。 最初に'S' (ASCII コード)を受け取るまでは、すべてのデータを破棄します。
引　数	第一引数：mot_data　　：モトローラ S フォーマットデータ
リターン値	一つ分のモトローラ S フォーマットデータ（'S'からチェックサムまで）を格納した場合：FLASH_API_SAMPLE_OK 一つ分のモトローラ S フォーマットデータを格納していない場合：FLASH_API_SAMPLE_NG
備考	本関数は、モトローラ S フォーマットデータを 1byte ずつ繰り返し引数で渡して使用します。 チェックサムの確認は行いません。

R_Fl_Prg_ProcessForMotS_data	
概　要	モトローラ S フォーマットレコードのヘッダ解析、Binary 変換、書き込み
ヘッダ	なし
宣　言	static void R_Fl_Prg_ProcessForMotS_data(void)
説　明	モトローラ S フォーマットのヘッダ解析を行い、Binary 変換する関数をコールします。 書き込みバッファへデータを格納する関数をコールします。 ダウンロードエリアへデータを書き込む関数をコールします。 モトローラ S フォーマットと異なるデータがあった場合、エラー関数をコールします。
引　数	なし
リターン値	なし

R_Fl_Prg_MotS_AsciiToBinary	
概　要	モトローラ S フォーマットデータ ASCII - Binary 変換
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Prg_MotS_AsciiToBinary (Fl_prg_mot_s_t *, Fl_prg_mot_s_binary_t *)
説　明	ASCII コードのモトローラ S フォーマットのデータを Binary データに変換します。 変換した Binary データのチェックサムを確認します。 モトローラ S フォーマットと異なるデータがあった場合、エラー関数をコールします。 チェックサムエラーが発生した場合、エラー関数をコールします。
引　数	第一引数：*tmp_mot_s　　：ASCII コードのモトローラ S フォーマットのデータのポインタ 第二引数：　　：Binary 変換されたデータを格納する変数のポインタ *tmp_mot_s_binary
リターン値	正常に変換が完了した場合：FLASH_API_SAMPLE_OK 正常に変換が完了しなかった場合：FLASH_API_SAMPLE_NG

R_Fl_Prg_MakeWriteData	
概　要	ダウンロードエリアへの書き込みデータ作成
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Prg_MakeWriteData(void)
説　明	RX63T のユーザマットへの書き込み単位である 128byte ずつに区切られたデータを作成します。
引　数	なし
リターン値	RX63T のユーザマットへの書き込み単位である 128byte の書き込みデータ作成が完了した場合：FLASH_API_SAMPLE_OK 128byte の書き込みデータ作成が未完了の場合：FLASH_API_SAMPLE_NG

R_Fl_Prg_WriteData	
概　要	ダウンロードエリアへの書き込み
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_Prg_WriteData(void)
説　明	ダウンロードエリアに書き込みを行います。 ダウンロードコードのエンディアン（MDES）を受信した場合は、サンプルコードのエンディアンと比較し、不一致であればエラー関数をコールします。 書き込んだデータのベリファイを行います。 書き込みに失敗した場合、エラー関数をコールします。 ベリファイエラーが発生した場合、エラー関数をコールします。
引　数	なし
リターン値	書き込みが正常に完了した場合：FLASH_API_SAMPLE_OK 書き込みが正常に完了しなかった場合：FLASH_API_SAMPLE_NG
備考	書き込み中の割り込みによる ROM アクセスを防ぐため、プロセッサステータスワード（PSW）のプロセッサ割り込み優先レベル（IPL）を変更します。

R_Fl_Prg_ClearMotSVariables	
概　要	モトローラ S フォーマットデータ関連の変数クリア
ヘッダ	なし
宣　言	static void R_Fl_Prg_ClearMotSVariables(void)
説　明	モトローラ S フォーマット用変数をクリアします。
引　数	なし
リターン値	なし

R_Fl_Run_StopUSB	
概　要	USB 停止
ヘッダ	r_Flash_main.h
宣　言	void R_Fl_Run_StopUSB(void)
説　明	USB を停止します。
引　数	なし
リターン値	なし

R_Fl_RcvDataString

概　要	USB 受信データ格納
ヘッダ	なし
宣　言	static Fl_API_SMPL_rtn_t R_Fl_RcvDataString(void *, uint16_t)
説　明	USB が受信したデータを受信用リングバッファに格納します。
引　数	第一引数：*p_tranadr　：USB が受信したデータが格納されているバッファのポインタ 第二引数：length　：USB が受信したデータ数
リターン値	格納が完了した場合：FLASH_API_SAMPLE_OK 受信用リングバッファがいっぱいだった場合：FLASH_API_SAMPLE_NG

R_Fl_Error

概　要	エラー処理
ヘッダ	r_Flash_main.h
宣　言	void R_Fl_Error(Fl_err_tbl_num_t err_num)
説　明	USB 停止関数をコールします。 LED、LCD にエラー表示を行います。
引　数	第一引数：err_num　：エラーNo.
リターン値	なし

R_Fl_RingCheckBlank

概　要	受信用リングバッファの空き容量確認
ヘッダ	r_Flash_buff.h
宣　言	Fl_API_SMPL_rtn_t R_Fl_RingCheckBlank(void)
説　明	受信用リングバッファにファイルリード関数で読み出すデータ 1 回分（2048byte）の空きがあるかを確認します。
引　数	なし
リターン値	空きがあった場合：FLASH_API_SAMPLE_OK 空きがなかった場合：FLASH_API_SAMPLE_NG

R_Fl_RingEnQueue

概　要	受信用リングバッファへのデータ格納
ヘッダ	r_Flash_buff.h
宣　言	Fl_API_SMPL_rtn_t R_Fl_RingEnQueue(uint8_t)
説　明	受信用リングバッファへデータを格納します。
引　数	第一引数：enq_data　：格納データ
リターン値	格納完了の場合：FLASH_API_SAMPLE_OK バッファフルの場合：FLASH_API_SAMPLE_NG

R_Fl_RingDeQueue

概　要	**受信用リングバッファからのデータ読み出し**
ヘッダ	r_Flash_buff.h
宣　言	Fl_API_SMPL_rtn_t R_Fl_RingDeQueue(uint8_t *)
説　明	**受信用リングバッファからデータを読み出します。**
引　数	**第一引数：***p_deq_data　　**：読み出したデータを格納するバッファのポインタ**
リターン値	**正常にデータを読み出した場合：**FLASH_API_SAMPLE_OK **読み出すデータがなかった場合：**FLASH_API_SAMPLE_NG

R_Fl_RingCheck

概　要	**受信用リングバッファのデータ数確認**
ヘッダ	r_Flash_buff.h
宣　言	uint32_t R_Fl_RingCheck(void)
説　明	**受信用リングバッファのデータ数を確認します。**
引　数	**なし**
リターン値	**受信データ数を返します。**

R_Fl_AsciiToHexByte

概　要	ASCII **コードから** Binary **データへの変換**
ヘッダ	r_Flash_buff.h
宣　言	uint8_t R_Fl_AsciiToHexByte(uint8_t, uint8_t)
説　明	2byte **の** ASCII **コードデータを** 1byte **の** Binary **データへ変換します。**
引　数	**第一引数：**in_upper　　**：**ASCII **コードデータ（上位）** **第二引数：**in_lower　　**：**ASCII **コードデータ（下位）**
リターン値	Binary **に変換されたデータを返します。**

5.14 フローチャート

5.14.1 USB 処理メイン

図 5.9にUSB 処理メインのフローチャートを示します。

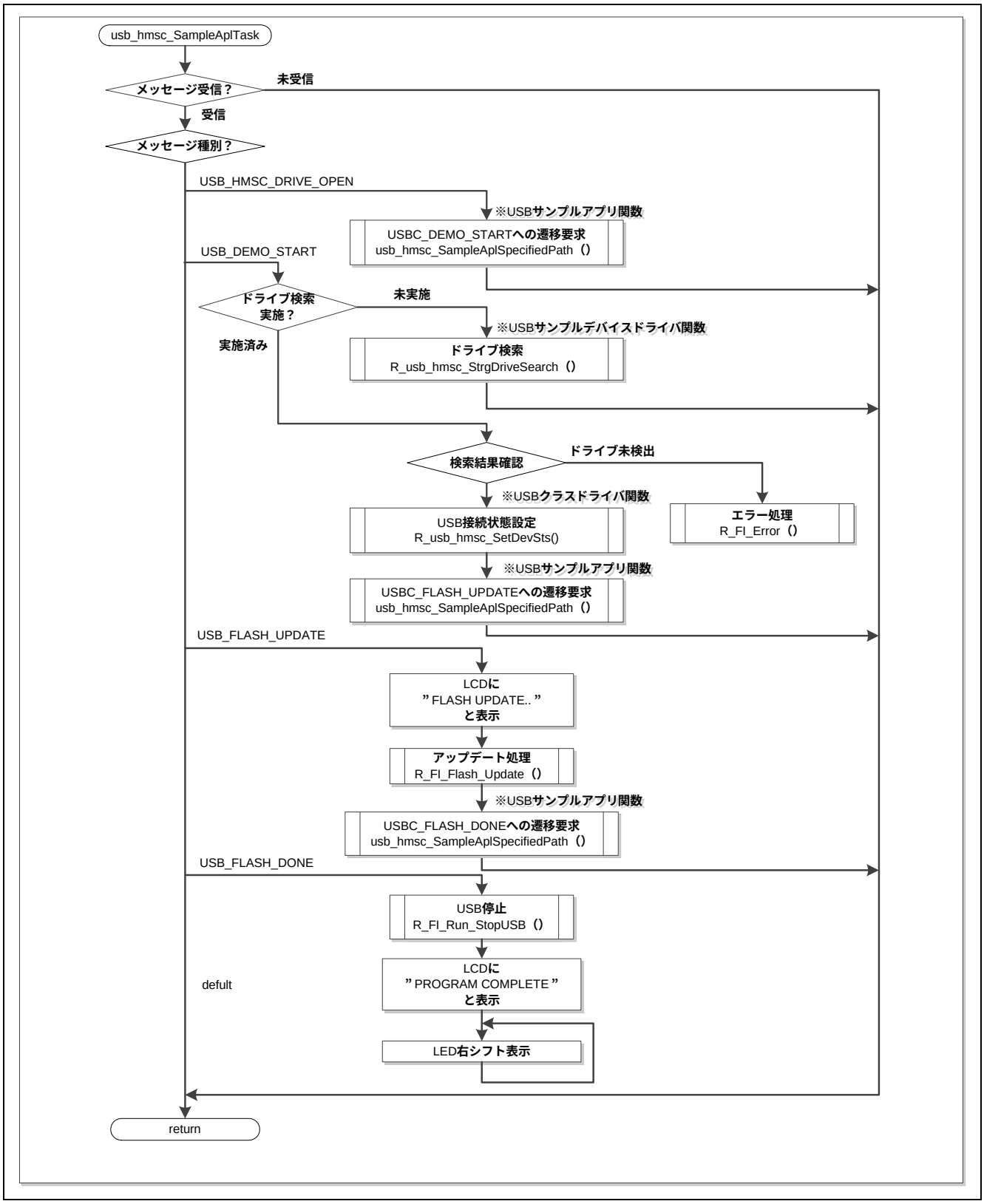

図5.9 USB 処理メイン

5.14.2　モードエントリ

図 5.10にモードエントリのフローチャートを示します。

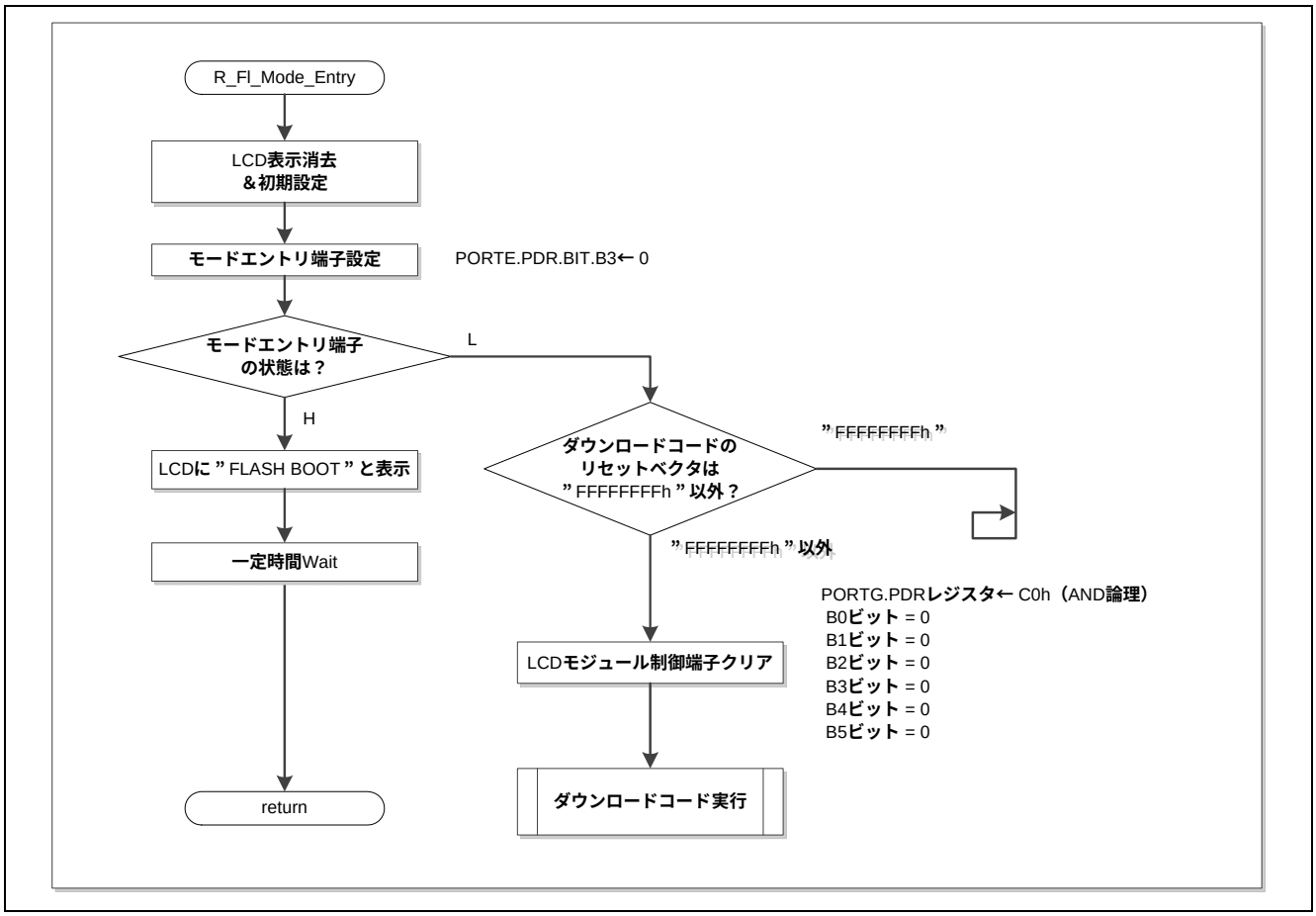

図5.10　**モードエントリ**

5.14.3　書き換え処理メイン

図 5.11 に書き換え処理メインのフローチャートを示します。

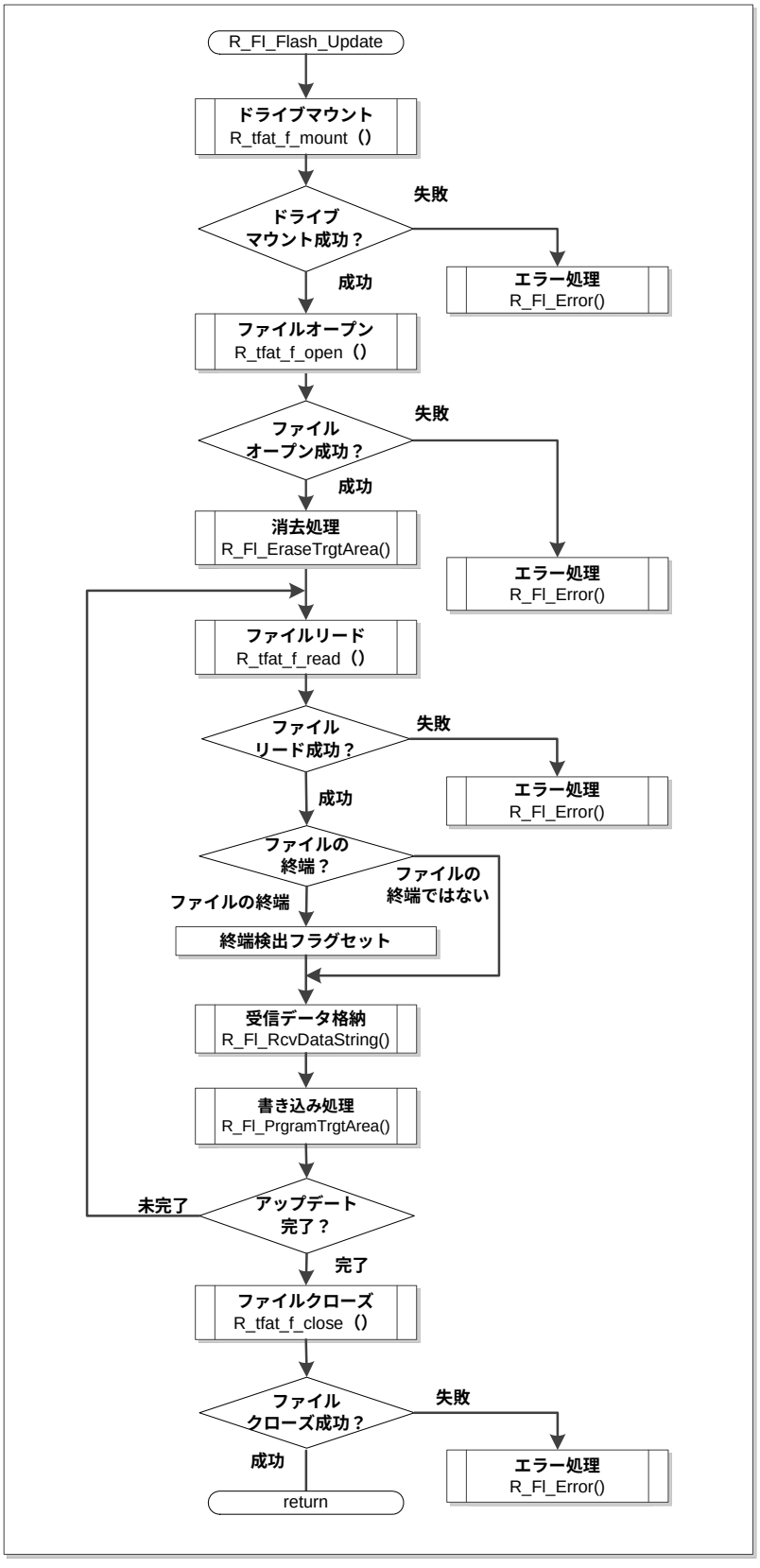

図5.11　書き換え処理メイン

5.14.4 消去処理

図 5.12 に消去処理のフローチャートを示します。

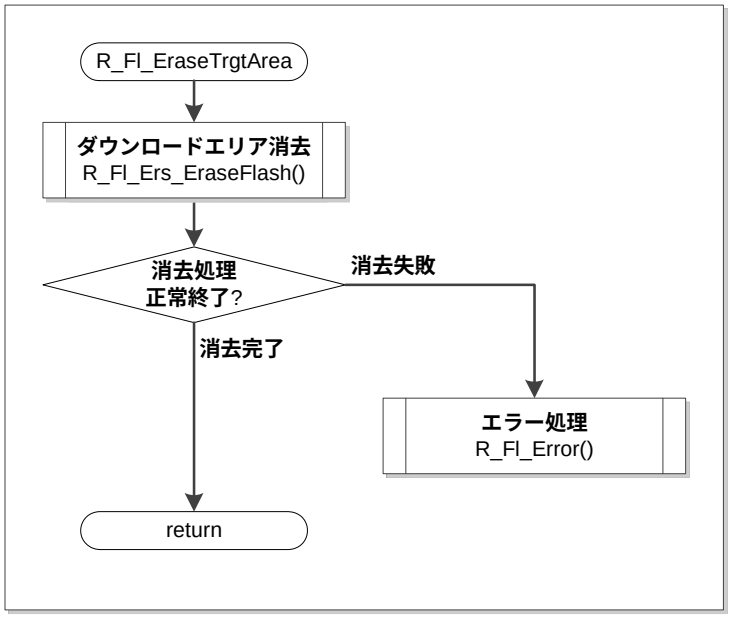

図5.12 消去処理

5.14.5 ダウンロードエリア消去

図 5.13 にダウンロードエリア消去のフローチャートを示します。

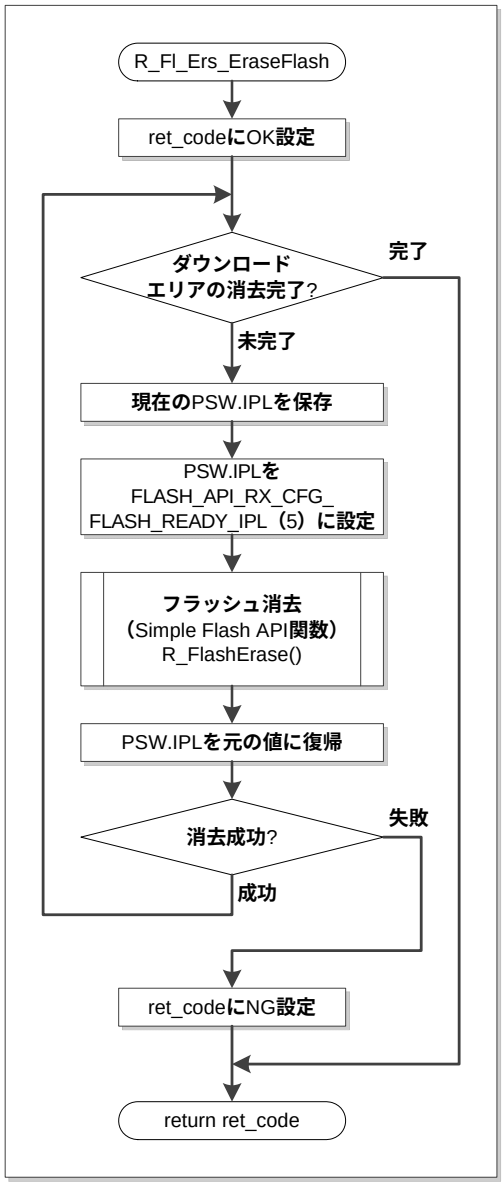

図5.13 ダウンロードエリア消去

5.14.6　書き込み処理

図 5.14に書き込み処理のフローチャートを示します。

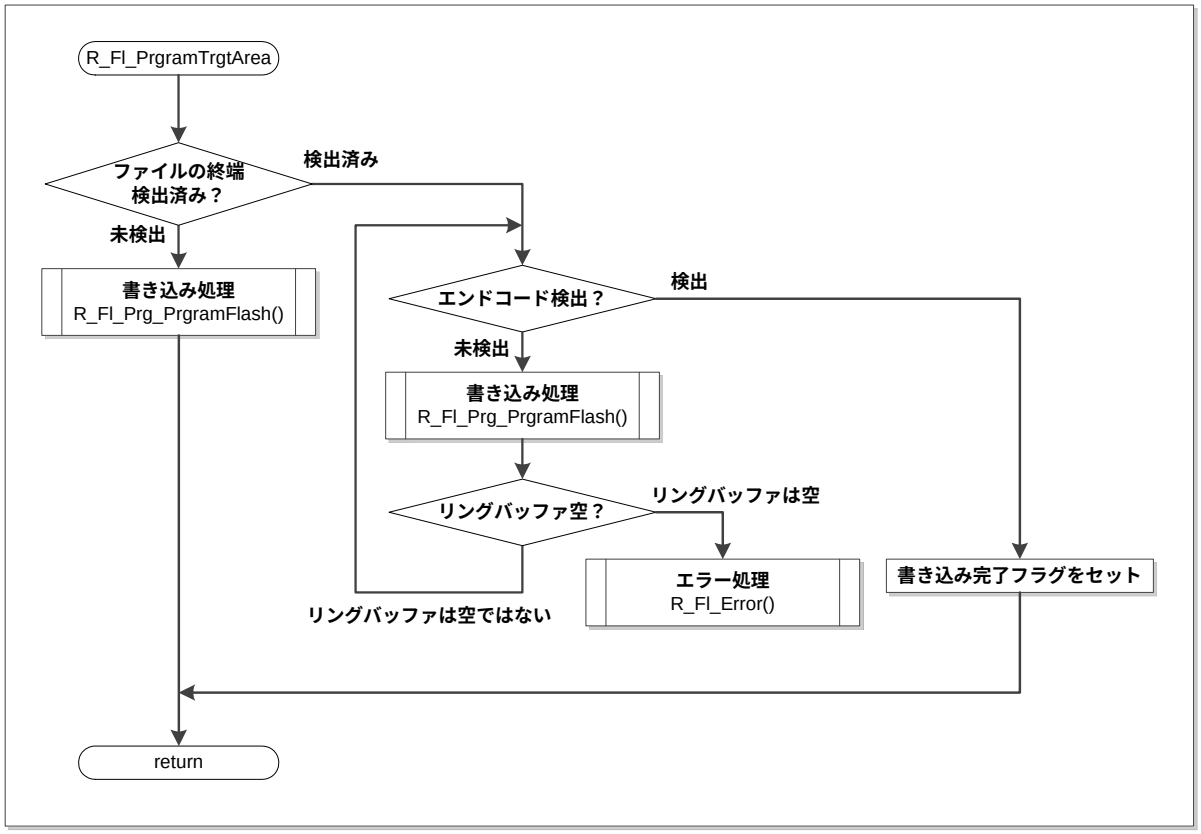

図5.14　**書き込み処理**

5.14.7　ダウンロードエリア書き込み

図 5.15 にダウンロードエリア書き込みのフローチャートを示します。

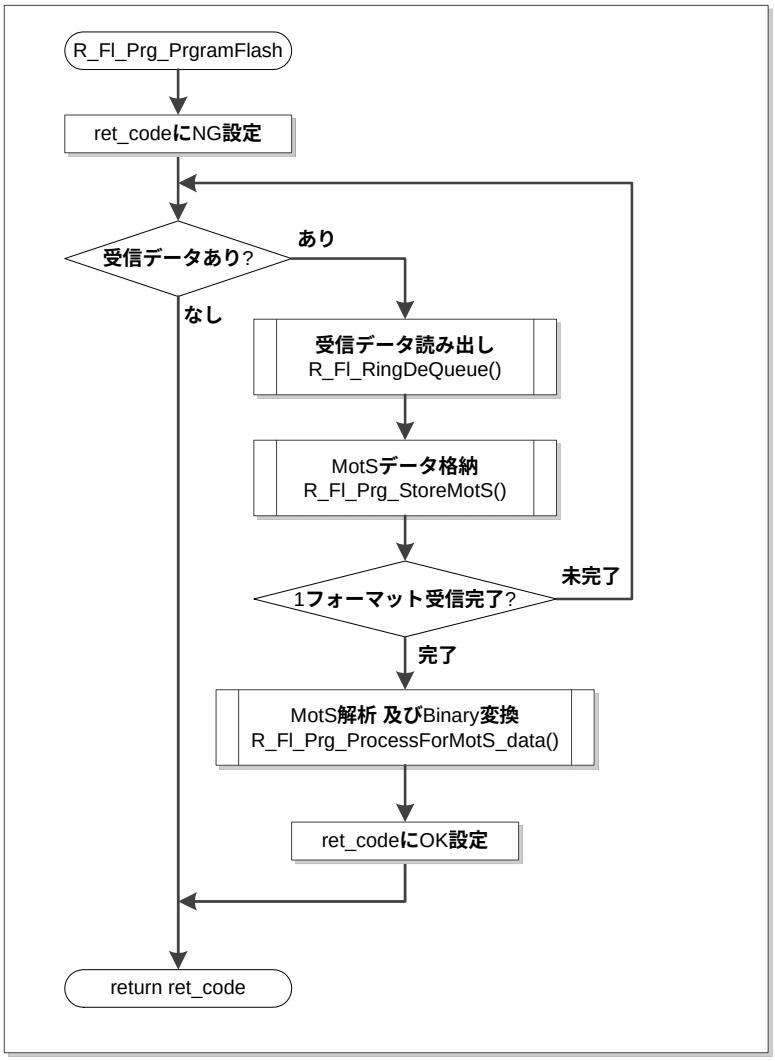

図5.15　ダウンロードエリア書き込み

5.14.8 モトローラ S フォーマットヘッダ解析、及び Binary 変換、書き込み

図 5.16にモトローラ S フォーマットヘッダ解析､及び Binary 変換､書き込みのフローチャートを示します。

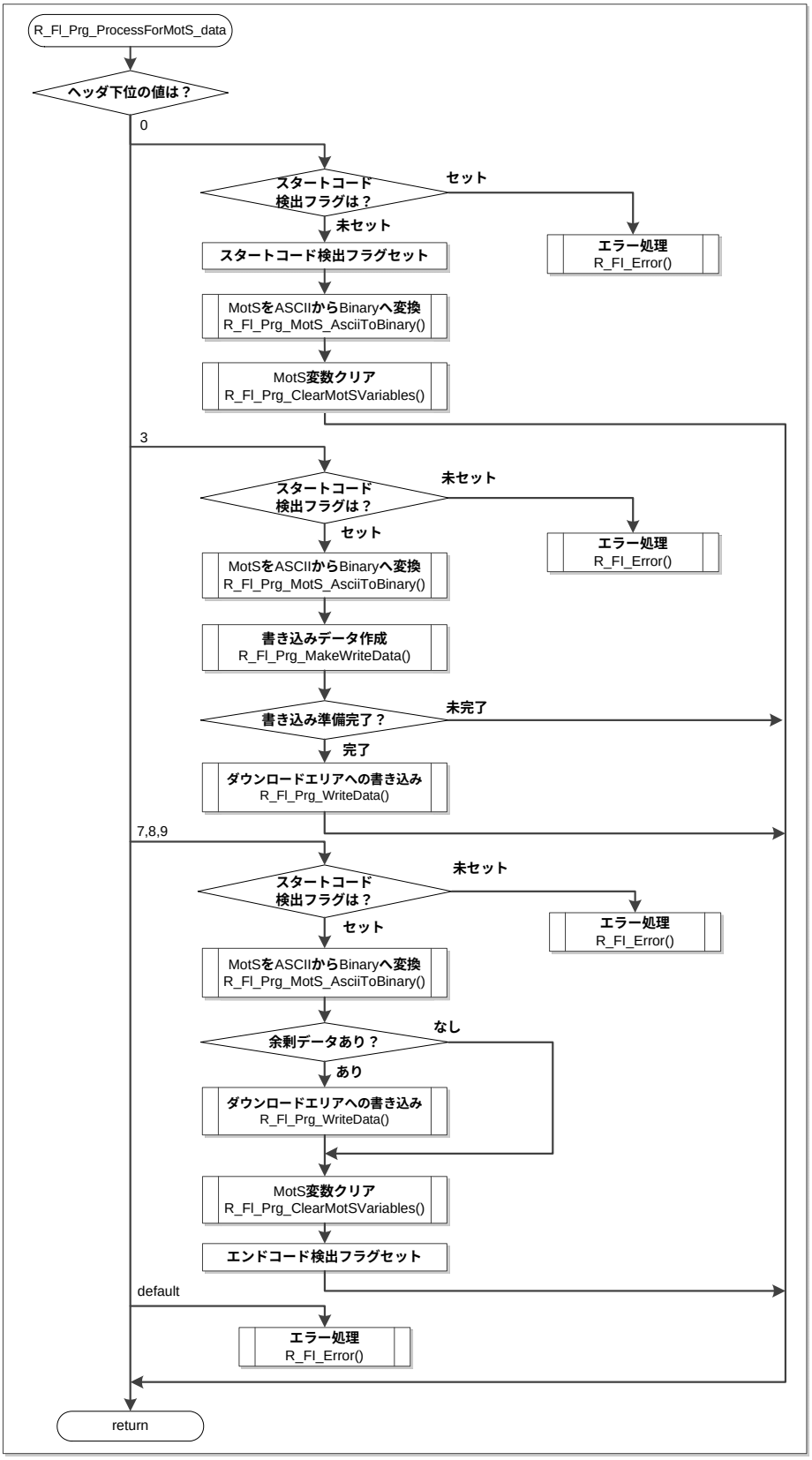

図5.16 モトローラ S フォーマットヘッダ解析、及び Binary 変換、書き込み

5.14.9 モトローラ S フォーマットデータ格納

図 5.17にモトローラ S フォーマットデータ格納のフローチャートを示します。

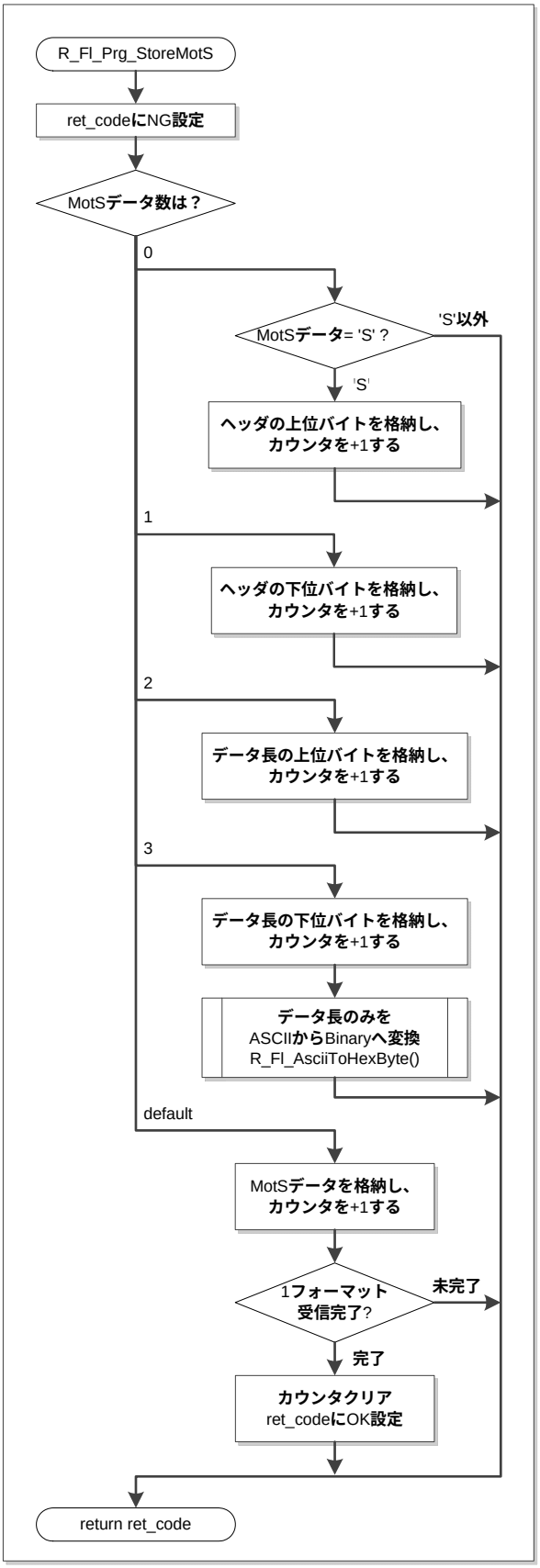

図5.17 モトローラ S フォーマットデータ格納

5.14.10 モトローラ S フォーマットデータ ASCII - Binary 変換

図 5.18にモトローラ S フォーマットデータ ASCII - Binary 変換のフローチャートを示します。

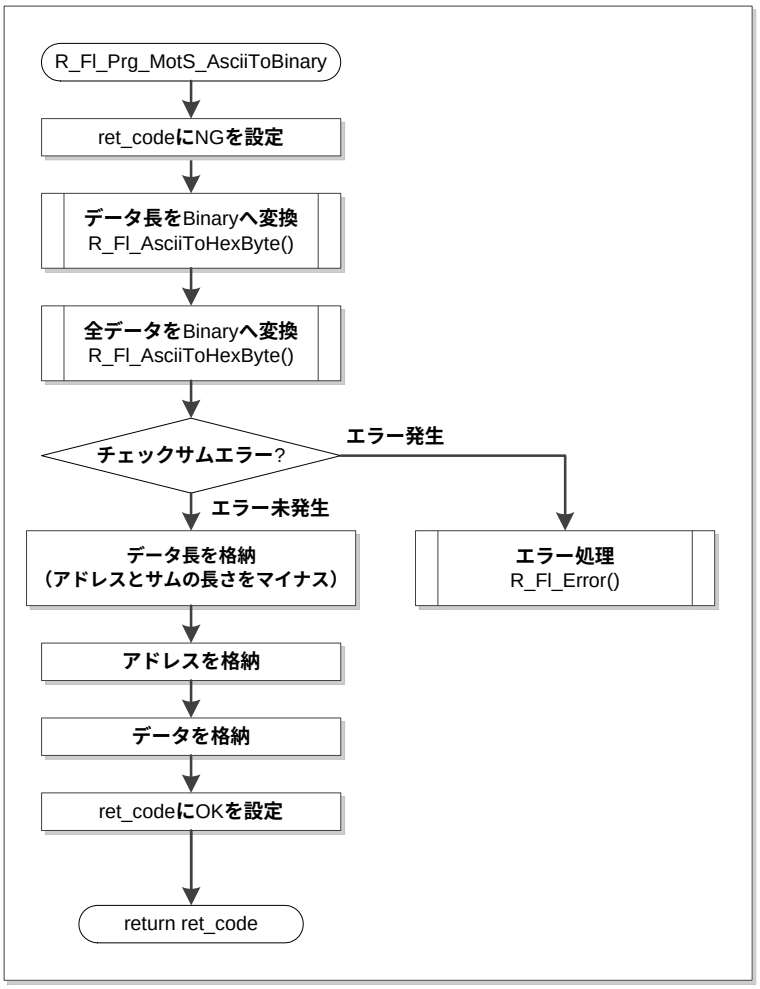

図5.18 モトローラ S フォーマットデータ ASCII - Binary 変換

5.14.11 ダウンロードエリアへの書き込みデータ作成

図 5.19にダウンロードエリアへの書き込みデータ作成のフローチャートを示します。

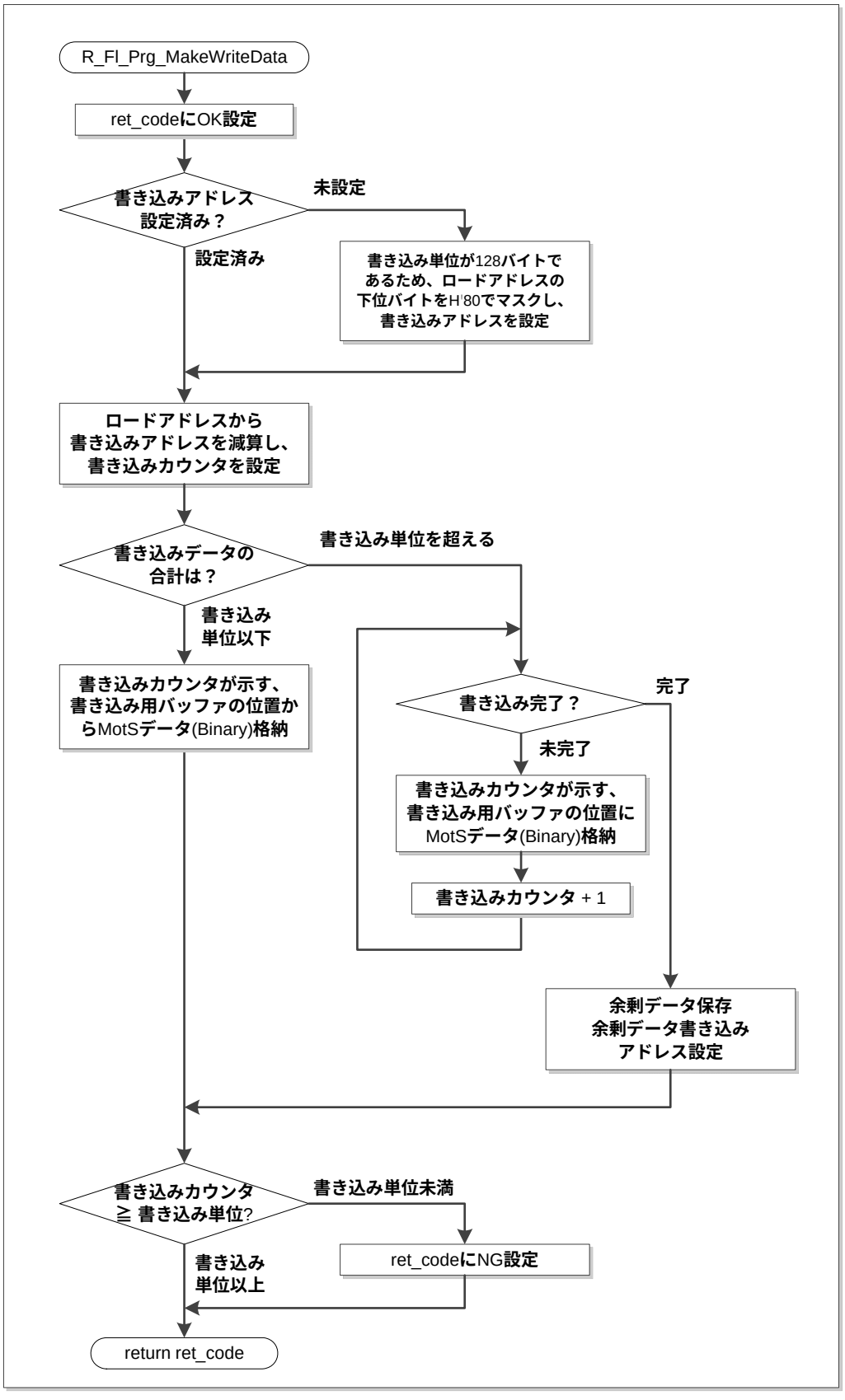

図5.19 ダウンロードエリアへの書き込みデータ作成

5.14.12　ダウンロードエリアへの書き込み

図 5.20にダウンロードエリアへの書き込みのフローチャートを示します。

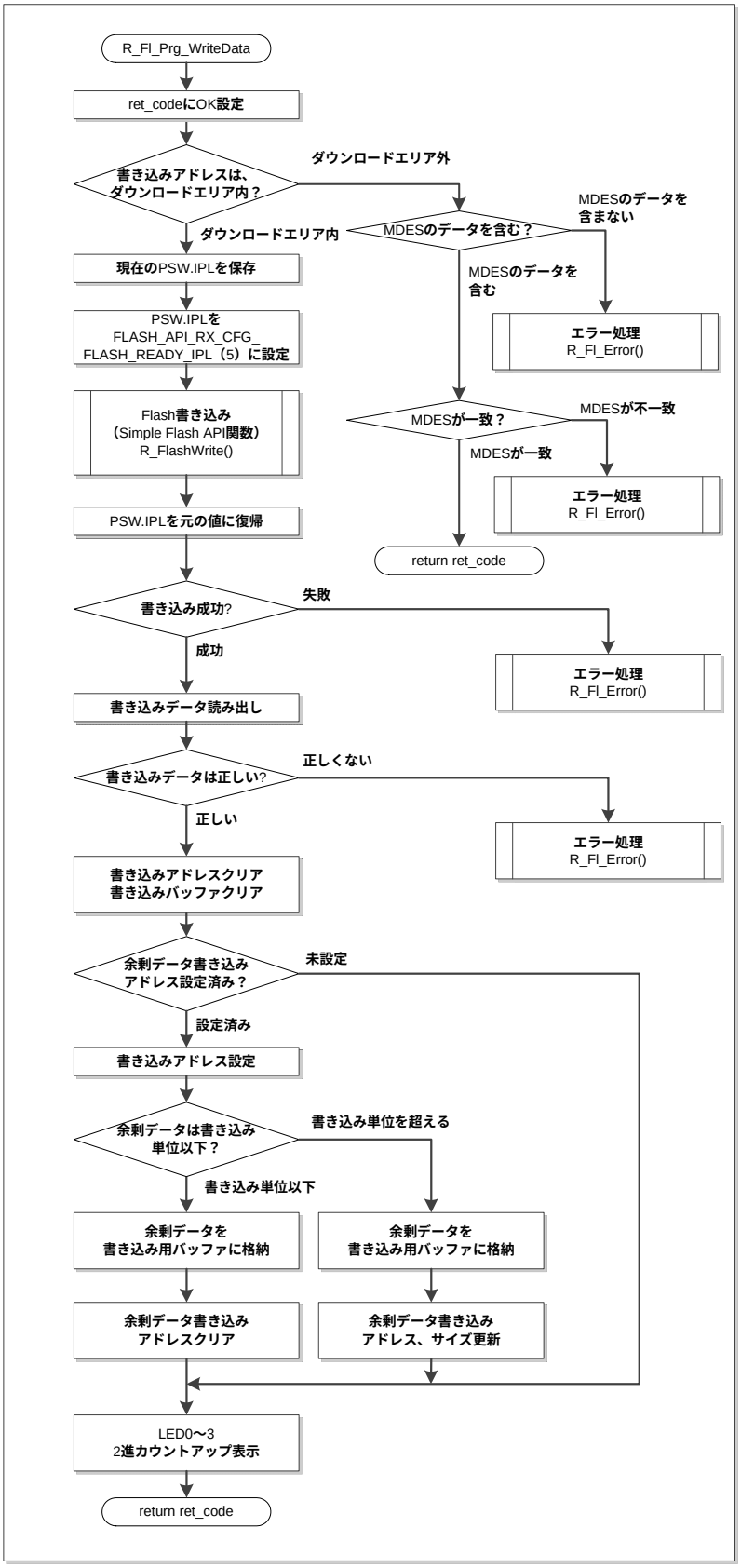

図5.20　ダウンロードエリアへの書き込み

5.14.13 モトローラ S フォーマットデータ関連の変数クリア

図 5.21にモトローラ S フォーマットデータ関連の変数クリアのフローチャートを示します。

図5.21 モトローラ S フォーマットデータ関連の変数クリア

5.14.14 USB 受信データ格納

図 5.22にUSB 受信データ格納のフローチャートを示します。

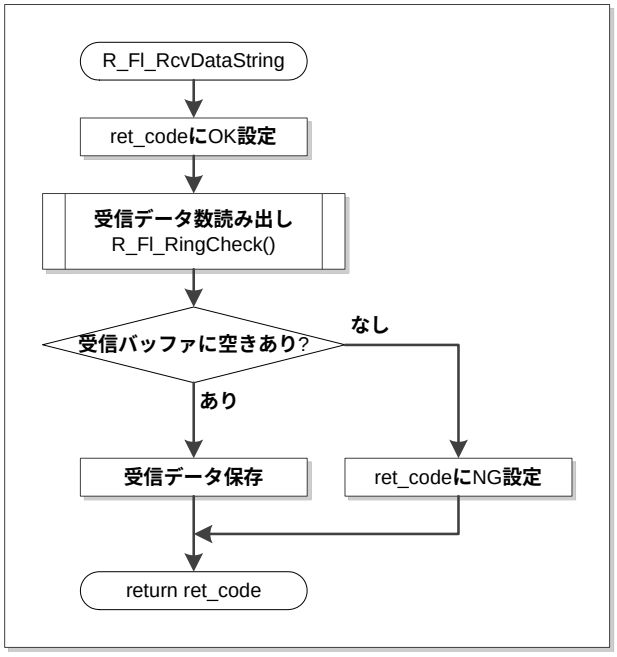

図5.22 USB 受信データ格納

5.14.15　受信用リングバッファの空き容量確認

図 5.23に受信用リングバッファの空き容量確認のフローチャートを示します。

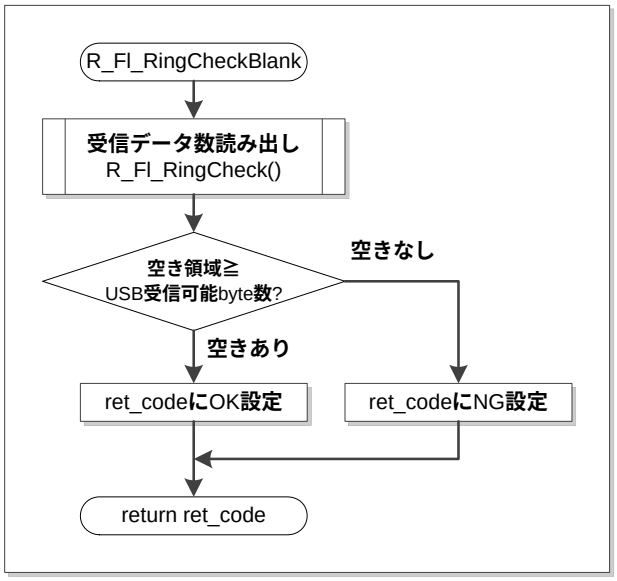

図5.23　受信用リングバッファの空き容量確認

5.14.16　USB 停止

図 5.24にUSB 停止のフローチャートを示します。

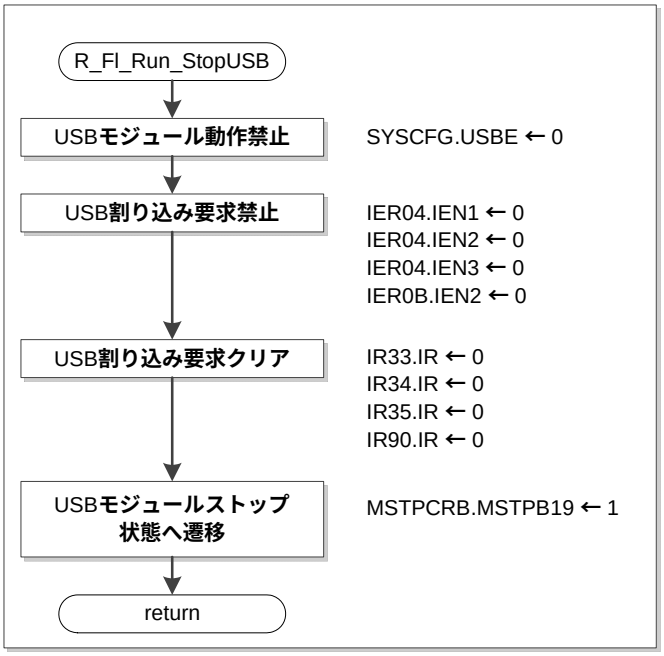

図5.24　USB 停止

5.14.17 エラー処理

図 5.25にエラー処理のフローチャートを示します。

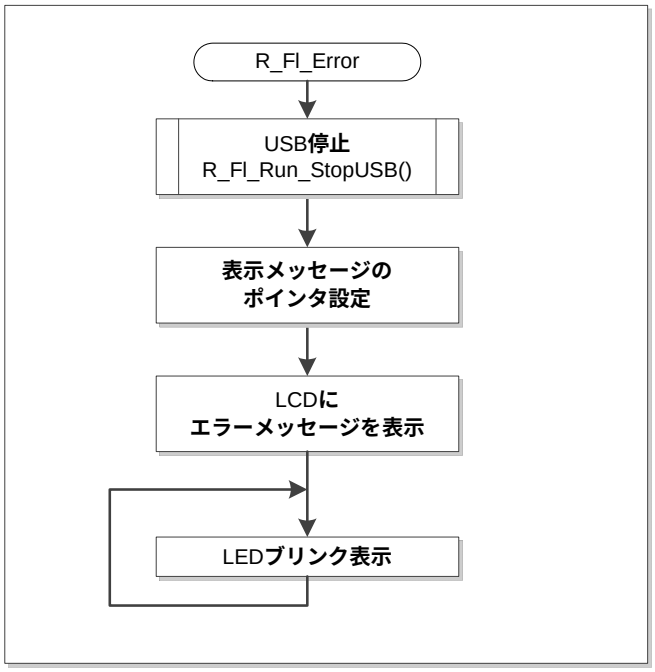

図5.25 エラー処理

5.14.18 受信用リングバッファへのデータ格納

図 5.26に受信用リングバッファへのデータ格納のフローチャートを示します。

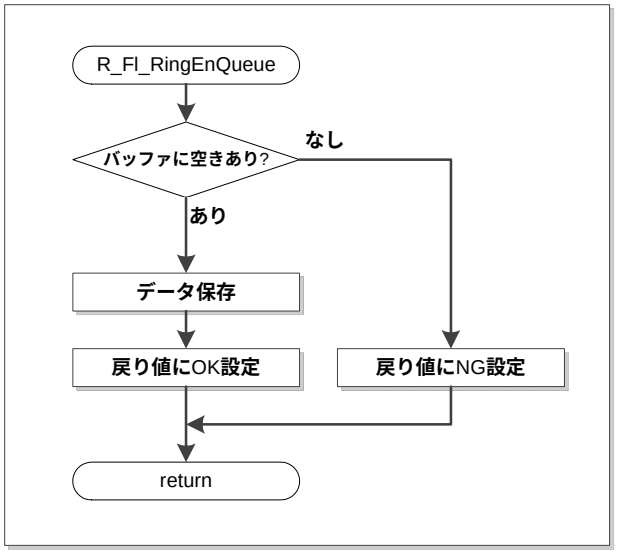

図5.26 受信用リングバッファへのデータ格納

5.14.19 受信用リングバッファからのデータ読み出し

図 5.27に受信用リングバッファからのデータ読み出しのフローチャートを示します。

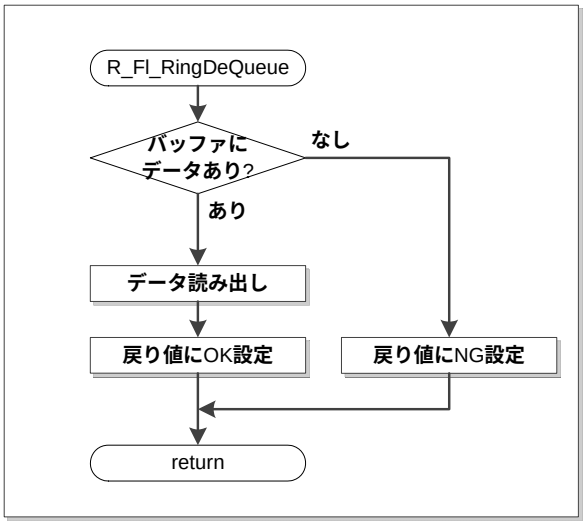

図5.27 受信用リングバッファからのデータ読み出し

5.14.20 受信用リングバッファのデータ数確認

図 5.28に受信用リングバッファのデータ数確認のフローチャートを示します。

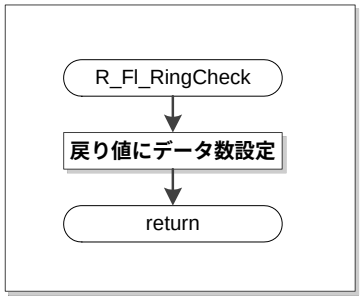

図5.28 受信用リングバッファのデータ数確認

5.14.21 ASCII コードから Binary データへの変換

図 5.29にASCII コードから Binary データへの変換のフローチャートを示します。

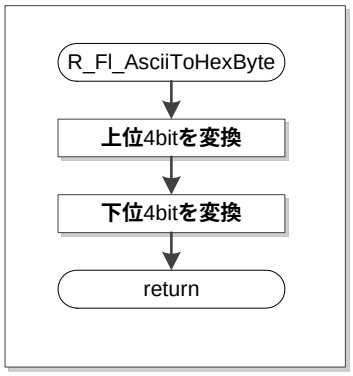

図5.29 ASCII コードから Binary データへの変換

6. ダウンロードコードの例

本アプリケーションノートのファイルには、ダウンロードコードの例（download.zip）が同梱されています。「2.2」に示されているボードの LED を順に点灯していくプログラムです。ダウンロードリセットベクタの設定やセクションの設定の参考にしてください。なお、ダウンロードコードは、512Kbyte の ROM 容量の使用を前提としています。

7. モトローラ S フォーマット

サンプルコードが対応するモトローラ S フォーマットについて説明します。

7.1 レコード形式

図 7.1にサンプルコードが対応するレコード形式を示します。

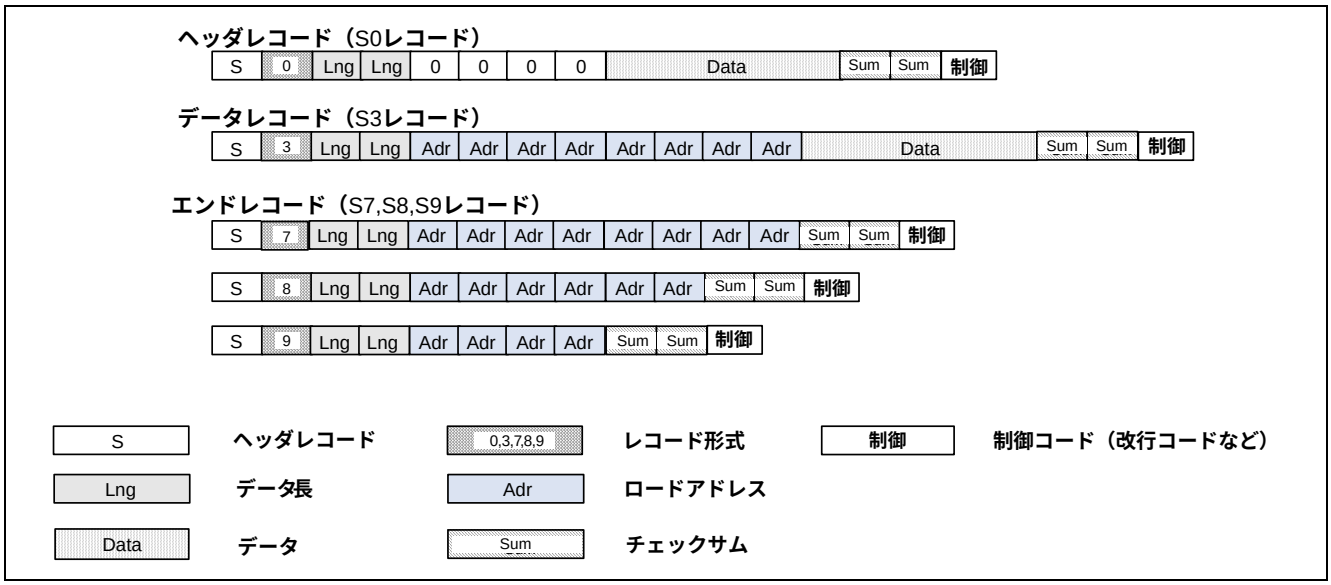

図7.1　サンプルコードが対応するレコード形式

7.2 レコード構成

図 7.2にサンプルコードが対応するレコード構成を示します。図 7.2の順序以外で構成されたモトローラ S フォーマットには対応していません。

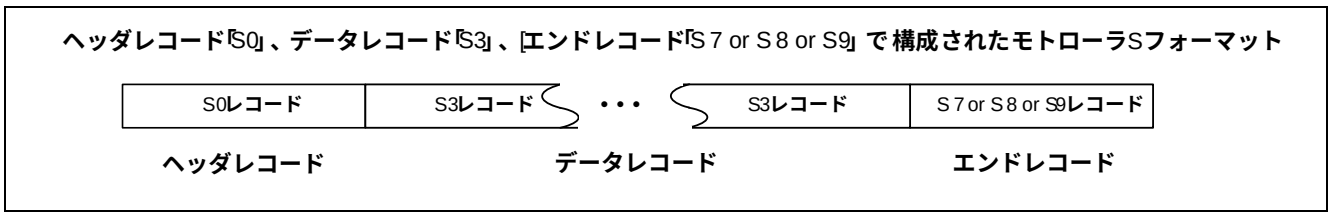

図7.2　サンプルコードが対応するレコード構成

7.3 ロードアドレス

サンプルコードでは、ロードアドレスが昇順のモトローラ S フォーマットのみ対応しています。降順およびアドレスが前後するモトローラ S フォーマットファイルは使用しないでください。

7.4 エラー検出

サンプルコードでは、受信したモトローラ S フォーマットに異常があった場合、エラーを検出します。

(a) チェックサムエラー

受信したモトローラ S フォーマットのレコード毎にチェックサムの検査を行い、検査結果に異常があった場合、「チェックサムエラー」を検出します。

(b) フォーマットエラー

モトローラ S フォーマットが以下の条件に当てはまった場合、「フォーマットエラー」を検出します。

- 未対応レコード（S1、S2、S4、S5、S6）を検出した場合。
- ヘッダレコード（S0）を 2 回検出した場合
- ヘッダレコード（S0）検出前に、データレコード（S3）もしくは、エンドレコード（S7、S8、S9）を検出した場合。

図 7.3にフォーマットエラーの検出条件を示します。

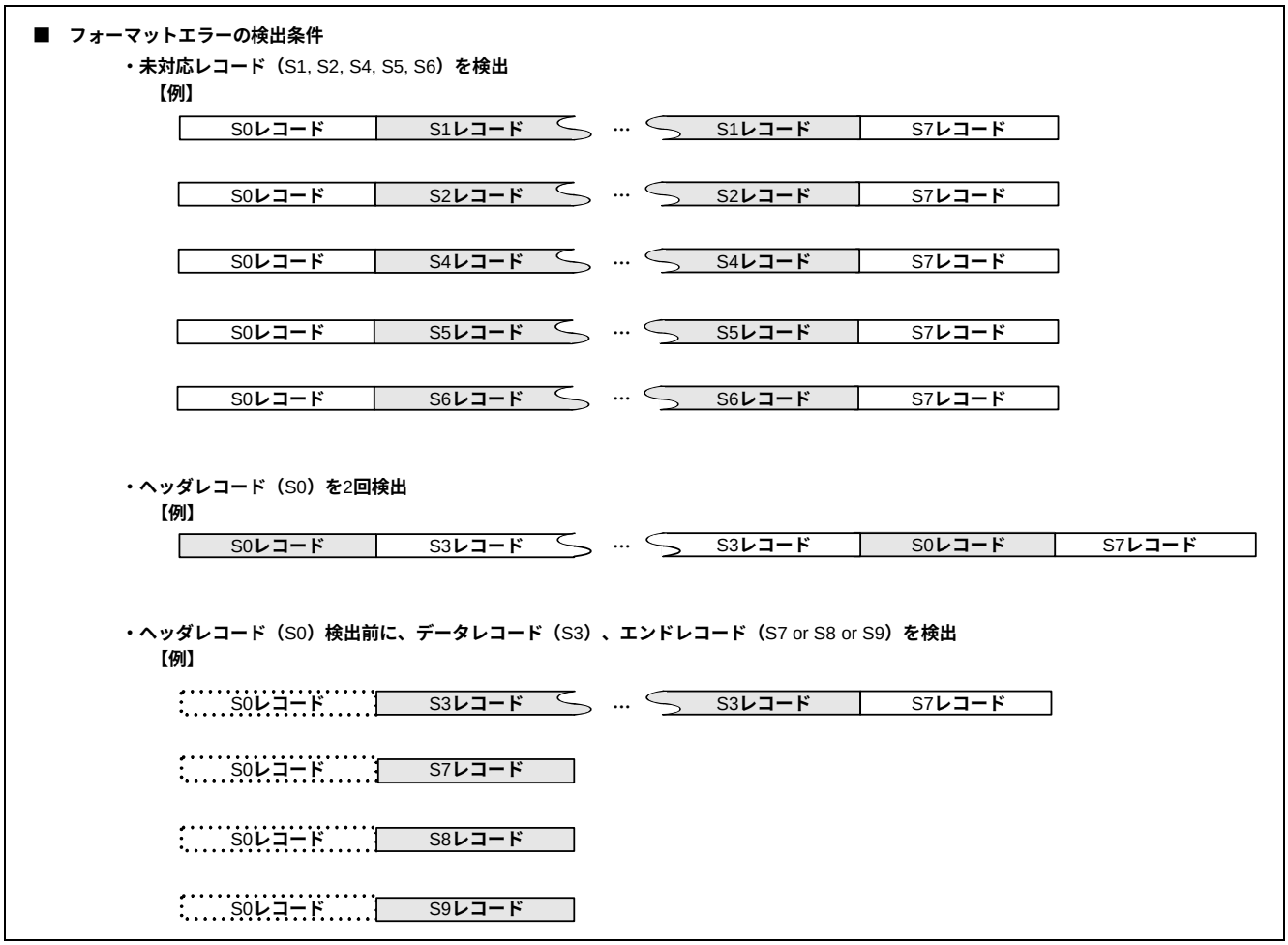

図7.3　フォーマットエラーの検出条件

(c) アドレスエラー

ダウンロードエリア外への書き込みデータを受信した場合、「アドレスエラー」を検出します。

8. 注意事項

8.1 書き込み、消去中の USB 切断

ダウンロードエリアへの書き込み、消去中は、USB を抜き差ししないでください。

8.2 HEW の設定

サンプルコードは、フラッシュ書き換え時に、ROM 上のコードを RAM に転送して実行します。これら設定の詳細は RX600&RX200 シリーズ RX 用のシンプルフラッシュ API のアプリケーションノートを参照してください。

8.3 固定ベクタテーブルの割り込み

サンプルコードは、固定ベクタテーブルに配置された割り込みの内、リセットのみに対応しています。それ以外の固定ベクタテーブルの割り込みを使用する場合は、サンプルコードを変更してご使用ください。

8.4 ダウンロードコードのリセットベクタ

サンプルコードを使用して書き込むダウンロードコードの実行開始位置は、ダウンロードリセットベクタ(FFFD FFFCh)の値で決定されます。従って、ダウンロードコードは、リセットベクタが FFFD FFFCh に配置されるように設定してください。詳細は「5.2.ダウンロードコードの実行開始位置」を参照してください。

また、ダウンロードコードの例については「6.ダウンロードコードの例」を参照してください。

8.5 ROM 容量の変更

サンプルコードが使用しているマイコンの ROM 容量は 512K byte です。ROM 容量が 384K byte または、256K byte のマイコンを使用する場合は、ファイル “r_Flash_main.h” 内の define “FL_END_BLOCK_NUM” を、使用する容量に合わせて変更してください。

表 8.1にROM 容量一覧を示します。

表8.1 ROM 容量一覧

製品型名	ROM 容量	ダウンロードエリア ROM 容量	ダウンロードエリア 開始アドレス	ダウンロードエリア ブロック番号
R5F563TE	512K	384K	FFF8 0000h	EB14～EB37
R5F563TC	384K	256K	FFFA 0000h	EB14～EB29
R5F563TB	256K	128K	FFFC 0000h	EB14～EB21

8.6 while(1)の処理

サンプルコードでは、USB 受信リングバッファがオーバフローした場合、while(1)でデッドロックします。ご注意ください。

8.7 エンディアン

本アプリケーションノートのサンプルコードは、リトルエンディアン/ビッグエンディアンの両方に対応しています。なお、エンディアンはフラッシュブートローダとダウンロードコードで同じエンディアン設定にしてください。

8.7.1 リトルエンディアン使用時

リトルエンディアンで動作する場合は以下の設定を行ってください。

1. コンパイラオプションの設定
   コンパイラオプションのエンディアンの設定で “Little-endian データ” を指定してください。5.8 オプション設定メモリの MDES はリトルエンディアンの値になります。

2. ユーザシステム定義ファイル（r_usb_usrconfig.h）の変更
   “r_usb_usrconfig.h” 内のマクロ定義 “USB_CPUBYTE_PP” の値を “USB_BYTE_LITTLE_PP” としてください。

3. TFAT ライブラリファイル（tfat_rx600_little.lib、tfat_rx600_big.lib）の変更
   リンカの入力ライブラリファイルオプションで
   “$(WORKSPDIR)¥WorkSpace¥MSCFW¥TFAT¥lib¥ tfat_rx600_little.lib” を指定してください。

8.7.2 ビッグエンディアン使用時

ビッグエンディアンで動作する場合は以下の設定を行ってください。

1. コンパイラオプションの設定
   コンパイラオプションのエンディアンの設定で “Big-endian データ” を指定してください。5.8 オプション設定メモリの MDES はビッグエンディアンの値になります。

2. ユーザシステム定義ファイル（r_usb_usrconfig.h）の変更
   “r_usb_usrconfig.h” 内のマクロ定義 “USB_CPUBYTE_PP” の値を “USB_BYTE_BIG_PP” としてください。

3. TFAT ライブラリファイル（tfat_rx600_little.lib、tfat_rx600_big.lib）の変更
   リンカの入力ライブラリファイルオプションで
   “$(WORKSPDIR)¥WorkSpace¥MSCFW¥TFAT¥lib¥tfat_rx600_big.lib” を指定してください。

8.8 RX600＆RX200 用のシンプルフラッシュ API からの変更点

サンプルコードは、RX600＆RX200 用のシンプルフラッシュ API のプログラムを流用しています。RX600＆RX200 用のシンプルフラッシュ API の仕様については RX600＆RX200 用のシンプルフラッシュ API のアプリケーションノートを参照してください。

8.8.1 変更箇所

RX600＆RX200 用のシンプルフラッシュ API から変更したファイルは “r_flash_api_rx_config.h”、“r_bsp_config.h”、“r_bsp.h” “r_flash_api_rx.c” です。

- ファイル “r_flash_api_rx_config.h” 内の変更箇所：
  ① フラッシュ書き込み/消去中の割り込みによる ROM アクセスを防ぐため、フラッシュ書き込み/消去時のプロセッサステータスワード（PSW）のプロセッサ割り込み優先レベル（IPL）を、以下のマクロ定義にて指定した値に変更します。本アプリケーションノートでは “5” に設定しています。
  マクロ定義：#define F FLASH_API_RX_CFG_FLASH_READY_IPL 5

  ② シンプルフラッシュ API の設定を変更しています。
  変更前：

```
//#define FLASH_API_RX_CFG_FLASH_TO_FLASH
#define FLASH_API_RX_CFG_IGNORE_LOCK_BITS
#define FLASH_API_RX_CFG_COPY_CODE_BY_API
```

  変更後：

```
#define FLASH_API_RX_CFG_FLASH_TO_FLASH
//#define FLASH_API_RX_CFG_IGNORE_LOCK_BITS
//#define FLASH_API_RX_CFG_COPY_CODE_BY_API
```

- ファイル “r_bsp_config.h” 内の変更箇所：
  ① シンプルフラッシュ API の設定を変更しています。※ICLK_HZ、PCLK_HZ、BCLK_HZ、FCLK_HZ は、USB Host Mass Storage Class Driver 内にある $(WORKSPDIR)¥WorkSpace¥HwResourceForUSB¥src¥rx_mcu.c の設定と合わせてください。
  変更前：#define BSP_CFG_BCK_DIV (8)
  変更後：#define BSP_CFG_BCK_DIV (4)

- ファイル “r_bsp.h” 内の変更箇所：
  ① USB Host Mass Storage Class Driver に付属しているプログラムと重複を防ぐため、“hwsetup.h”、“lowsrc.h”、“vecttbl.h”、“iodefine_rx63th.h”は以下のようにコメントアウトしています。
  “rskrx63t_144pin.h”に関しては未使用のため、コメントアウトしています。
  変更前：

```
#include "mcu/rx63t/register_access/iodefine_rx63th.h"
#include "board/rskrx63t_144pin/rskrx63t_144pin.h"
#include "board/rskrx63t_144pin/hwsetup.h"
#include "board/rskrx63t_144pin/lowsrc.h"
#include "board/rskrx63t_144pin/vecttbl.h"
```

  変更後：

```
//#include "mcu/rx63t/register_access/iodefine_rx63th.h"
//#include "board/rskrx63t_144pin/rskrx63t_144pin.h"
//#include "board/rskrx63t_144pin/hwsetup.h"
//#include "board/rskrx63t_144pin/lowsrc.h"
//#include "board/rskrx63t_144pin/vecttbl.h"
```

- ファイル “r_flash_api_rx.c” 内の変更箇所：
  ① 本 APN では、USB Host Mass Storage Class Driver に付属している“iodefine.h”を参照するため、以下のように変更しています。
  変更前：#if !defined(R_BSP_VERSION_MAJOR)
  #include "iodefine.h"
  #include "mcu_info.h"
  #endif
  変更後：#if !defined(R_BSP_VERSION_MAJOR)
  #include "iodefine.h"
  #include "mcu_info.h"
  #else
  #include "iodefine.h"
  #endif

8.9 USB Host Mass Storage Class Driver からの変更点

サンプルコードは USB Host Mass Storage Class Driver のプログラムを流用しています。USB Host Mass Storage Class Driver の仕様については USB Host Mass Storage Class Driver 及び USB Basic Firmware のアプリケーションノートを参照してください。

8.9.1 変更箇所

USB Host Mass Storage Class Driver から変更したファイルは

- r_usb_hmsc_apl.c
- dbsct_hmsc.c
- resetprg.c

の３つです。

- ファイル “r_usb_hmsc_apl.c” 内の変更箇所：
  #ifdef R_FLASH_USB で示しています。

- ファイル “dbsct_hmsc.c” 内の変更箇所：
  コメント “// Flash table” で示しています。

- ファイル “resetprg.c” 内の変更箇所：
  1. インクルードファイルを追加しています。
     追加：#include "r_Flash_main.h"
  2. 関数 “PowerON_Reset_PC” 内でモードエントリ関数を呼んでいます。
     追加：R_Fl_Mode_Entry();

8.9.2 追加ファイル

USB Host Mass Storage Class Driver に追加したファイルについては「5.7.ファイル構成」を参照してください。

8.9.3 追加セクション

表 8.2に USB Host Mass Storage Class Driver の追加セクションを示します。

表8.2 追加セクション

セクション名	概要
R_flash_api_sec	RAM 上で動作する Flash 書き換えコードの変数用セクション
D_flash_api_sec	RAM 上で動作する Flash 書き換えコードの初期化領域セクション
RPFRAM	RAM 上で動作する Flash 書き換えコード用セクション
TRGT_DMMY_FIXEDVECT	ダウンロードコード固定ベクタ用セクション

8.9.4 インクルードファイルディレクトリ

“WorkSpace¥FLASH”、“WorkSpace¥FLASH¥src”、“WorkSpace¥FLASH¥r_bsp”、“WorkSpace¥FLASH¥r_bsp¥mcu¥rx63t”、“WorkSpace¥FLASH¥r_bsp¥board¥rskrx63t_144pin” をインクルードファイルディレクトリに追加しています。

8.9.5 マクロ定義（コンパイラオプション）

- “R_FLASH_USB” をコンパイラオプションのマクロ定義に追加しています。

8.9.6 リンカの設定

ROM から RAM へ MAP するリンカの設定を追加しています。

- Rom “PFRAM” を RPFRAM へ MAP しています。
- Rom “D_flash_api_sec” を “R_flash_api_se————c” へ MAP しています。

9. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

10. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
RX63Tグループ ユーザーズマニュアル ハードウェア編 Rev.2.00
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境
RX ファミリ C/C++コンパイラパッケージ V.1.01 ユーザーズマニュアル Rev.1.00（V.1.02 添付資料含む）
（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。）

アプリケーションノート
ルネサス USB デバイス USB Host Mass Storage Class Driver Rev.2.00
（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。）

アプリケーションノート
ルネサス USB デバイス USB Basic Firmware Rev.2.00
（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。）

アプリケーションノート
M3S-TFAT-Tiny：FAT ファイルシステムソフトウェア Rev.1.00
（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。）

アプリケーションノート
RX600&RX200 シリーズ RX 用のシンプルフラッシュ API Rev.2.40
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.05.14	—	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. **未使用端子の処理**
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. **電源投入時の処置**
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. **リザーブアドレスのアクセス禁止**
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. **クロックについて**
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. **製品間の相違について**
   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。
   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

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

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

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

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

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

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/