RZ/T1 グループ

R01AN4432JJ0200
Rev.2.00
2019.9.30

USB 経由シリアル FlashROM 書き込みサンプルプログラム

R-IN Engine 搭載製品

要旨

本アプリケーションノートは、USB Peripheral Communications Device Class(以下、PCDC)を使用した USB 経由でシリアル FlashROM にユーザアプリケーションプログラム(以下、ユーザプログラム)を書き込むサンプルプログラムについて説明します。

USB 経由シリアル FlashROM 書き込みサンプルプログラム(以下、USB シリアル書き込みサンプル)の特長を以下に示します。

- ✓ 本サンプルプログラムは、ローダプログラム部、ユーザプログラム部、および USB シリアル書き込みサンプル部で構成されています。
- ✓ ローダプログラムは、RZ/T1 のブート処理後にクロック発生回路、バスステートコントローラなどの初期化後、シリアル FlashROM に書き込まれている下記のいずれかのプログラムを ATCM 領域にコピーし、コピーしたプログラムを起動します。
  - ・ユーザプログラム
  - ・USB シリアル書き込みサンプル

  なお、ローダプログラムは RZ/T1 の P44 端子の状態を見て ATCM 領域にコピーするプログラムを判断します。
- ✓ ユーザプログラムは評価ボードの LED 制御を行います。評価ボードの SW2 を押下すると、Arm® Cortex®-R4 コアは外部端子割り込み処理により、評価ボードの LED1 の点灯／消灯の状態を LED データとして共有メモリ領域に書き込みます。一方 Cortex-M3 コアは常に共有メモリの LED データを読み込み、点灯／消灯状態を評価ボードの LED1 に反映させます。

  なお、ユーザプログラムは、「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を使用しています。ユーザプログラムの詳細な仕様に関しては「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を参照してください。
- ✓ USB シリアル書き込みサンプルは、ホスト PC からターミナルソフト(Tera Term など)により送信されるコマンドを USB 経由で受信し、ユーザプログラムのダウンロード、シリアル FlashROM への書き込み、セクターイレースなどを行います。

動作確認デバイス

RZ/T1 グループ R-IN Engine 搭載製品

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.　システム仕様

1.1　　システム構成

図 1.1 に本サンプルプログラムのシステム構成を示します。

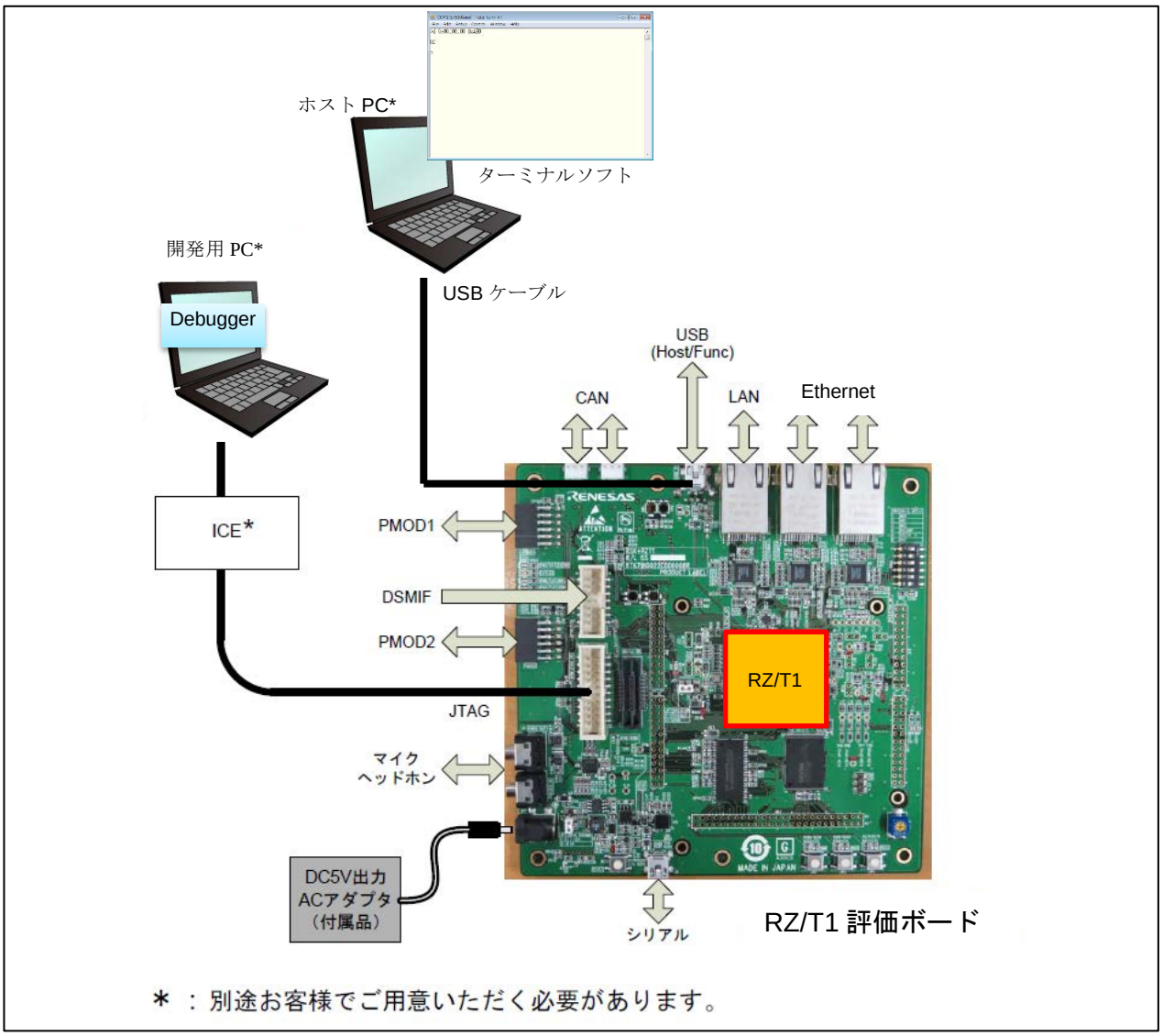


図 1.1　システム構成

1.2　動作環境

本マニュアルのサンプルプログラムは、下記の環境を想定しています。

表 1.1 動作環境

項目	内容
使用ボード	RZ/T1 評価ボード RTK7910018C00000BE
MCU	RZ/T1　(R-IN Engine 内蔵版) R7S910018
動作周波数	CPU クロック（CPUCLK）：450MHz（Arm Cortex-R4） システムクロック（ICLK）：150MHz（Arm Cortex-M3）
動作電圧	3.3V
動作モード	SPI ブートモード
使用デバイス	シリアル FlashROM(64Mbyte) Macronix 製 MX25L51245GMI-10G(セクターサイズ：64Kbyte)
統合開発環境	IAR システムズ社製 Embedded Workbench® for Arm Version 8.30.1 RENESAS 製 e² studio 6.1.0
エミュレータ	IAR システムズ社製 I-jet SEGGER 製 J-Link
ターミナルソフト	Tera Term v4.97
ホスト PC	Windows 10 Enterprise Intel(R) Core(TM) i5-6300U CPU@2.4GHz 2.5GHz

1.3　端子設定

1.3.1　動作モードの選択

RZ/T1 は、外部端子(MD0、MD1、MD2)で動作モードを選択します。

以下の表に、RZ/T1 の各モード設定端子のレベルと動作モードとの関係を示します。

表 1.2 動作モード選択

モード設定端子			動作モード
MD2	MD1	MD0	
Low	Low	Low	SPI ブートモード(シリアル FlashROM) SPI マルチ I/O バス空間に接続されたシリアル FlashROM からブートします。
Low	High	Low	16 ビットバスブートモード(NOR フラッシュ) CS0 空間に接続された NOR フラッシュ(バス幅 16 ビット)からブートします。
Low	High	High	32 ビットバスブートモード(NOR フラッシュ) CS0 空間に接続された NOR フラッシュ(バス幅 32 ビット)からブートします。 (RZ/T1 評価ボードでは設定禁止)
上記以外			予約（設定禁止）

これらの動作モードを選択する仕組みとして、RZ/T1 評価ボードでは、上記 MD0、MD1、MD2 は DIP-SW（SW4-1、4-2、4-3）に接続されています。評価ボードでは SW4 の設定により動作モードを選択可能です。本サンプルプログラムでは SPI ブートモードを対象とします。以下の表で塗りつぶしてある設定にします。

表 1.3 SW4 の設定

サンプルプログラム	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16 ビットバスブートモード	ON	OFF	ON	ON	ON	OFF
SPI ブートモード	ON	ON	ON	ON	ON	OFF

動作モードの選択は、電源を投入する前に行ってください。

1.3.2　USB シリアル書き込みサンプルへの切り替え

本サンプルプログラムでは、RZ/T1 のリセット解除後に P44 端子(評価ボードの SW3)のレベルをソフトウェアで判定し、USB シリアル書き込みサンプルとユーザプログラムを切り替えます。

表 1.4 USB シリアル書き込みサンプル切り替え

P44 端子	プログラム
Low(SW3 押下)	USB シリアル書き込みサンプル
High	ユーザプログラム

USB シリアル書き込みサンプルが起動された場合、評価ボードの LED10 が点灯します。

1.4　電源の選択

RZ/T1 評価ボードでは、電源選択用ジャンパ(JP2、JP7)を実装しています。

以下の表で塗りつぶしてある設定にします。

表 1.5 JP2、JP7 の設定

ジャンパ	設定	機能
JP2 システム電源選択	1-2	7～12V 電源を使用
	2-3	5V 電源を使用
JP7 VCCQ33B 供給元選択	1-2	RZ/T1 デジタル 3.3V 電源から供給
	2-3	RZ/T1 デジタル 1.2V 電源から供給

電源選択用ジャンパの設定は、電源を投入する前に行ってください。

2. 関連ドキュメント

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

- RZ/T1 グループ R-IN Engine 搭載製品　初期設定
  ドキュメント番号：R01AN2989JJxxxx(日本語版)/R01AN2989EJxxxx (英語版)
- RZ/T1 グループ USB Peripheral Communications Device Class Driver
  ドキュメント番号：R01AN2631JJxxxx (日本語版)/R01AN2631EJxxxx (英語版)
- RZ/T1 グループ・ユーザーズマニュアル ハードウェア編
  ドキュメント番号：R01UH0483JJxxxx (日本語版)/R01UH0483EJxxxx (英語版)
- RZ/T1 グループ シリアルフラッシュサンプルプログラム
  ドキュメント番号：R01AN3010JJxxxx (日本語版)/R01AN3010EJxxxx (英語版)
- RZ/T1 評価ボード RTK7910022C00000BR ユーザーズマニュアル
  ドキュメント番号：R20UT3124JJxxxx (日本語版)
- Renesas Starter Kit+ for RZ/T1 User's Manual
  ドキュメント番号：R20UT3242EGxxxx (英語版)

xxxx：リビジョン

3. 周辺機能説明

クロック発生回路（CPG）、割り込みコントローラ（ICUA）、エラーコントロールモジュール（ECM）、拡張内蔵 RAM、汎用入出力ポート、USB2.0HS ファンクションモジュール(USBf)についての基本的な内容は、RZ/T1 グループ・ユーザーズマニュアル ハードウェア編を参照してください。

4. ハードウェア説明

図 4.1 にハードウェア構成例を示します。

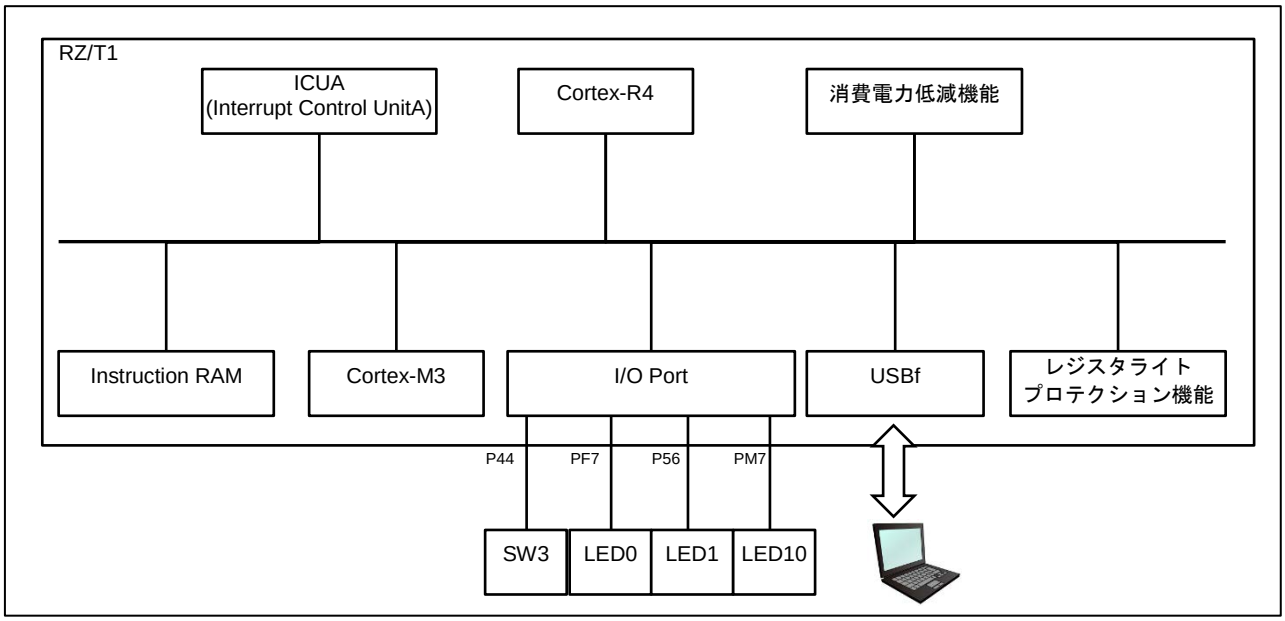


図 4.1 ハードウェア構成例

5. ソフトウェア説明

以降、特に明記しない場合は EWARM（IAR システムズ社製）を使用した場合について説明を行います。

5.1 ローダプログラム動作概要

RZ/T1 はリセット解除後のブート処理で、外付けのシリアル FlashROM に格納されたローダプログラムを内蔵 RAM（BTCM）へコピーします。

ローダプログラムでは、ブート処理後にリセット判定、クロック設定、バス設定などを行います。その後、RZ/T1 の P44 端子を確認して、シリアル FlashROM に格納されたユーザプログラム領域または、USB シリアル書き込みサンプル領域を内蔵 RAM(ATCM)にコピーします。

ユーザプログラムを内蔵 RAM（ATCM）にコピーする場合、ローダプログラムはシリアル FlashROM に書き込まれているユーザプログラム情報テーブルを参照して、内蔵 RAM（ATCM）にコピーします。本サンプルプログラムでは、ユーザプログラム情報テーブルはアドレス 0x300A0000 に格納します。表 5.1 にユーザプログラム情報テーブルを示します。

次に MPU 設定、キャッシュ設定を行い、例外ベクタをロウベクタ状態に切り替え、内蔵 RAM(ATCM)にコピーしたプログラムの先頭番地へ分岐します。

表 5.1 ユーザプログラム情報テーブル

No	Address(Offset)	Description
1	0x00	シリアル FlashROM のユーザプログラム領域の先頭アドレス
2	0x04	ATCM に格納するユーザプログラム領域の先頭アドレス
3	0x08	ATCM に格納するユーザプログラム領域の最終アドレス
4	0x0C	シリアル FlashROM の Cortex-M3 用ユーザプログラム領域の先頭アドレス
5	0x10	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域の先頭アドレス
6	0x14	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域のサイズ(e² studio 版は最終アドレス)
7	0x18	シリアル FlashROM のユーザプログラム用変数領域の先頭アドレス
8	0x1C	ATCM に格納するユーザプログラム用変数領域の先頭アドレス
9	0x20	ATCM に格納するユーザプログラム用変数領域の最終アドレス
10	0x24	Dummy
11	0x28	bss 領域の先頭アドレス
12	0x2C	bss 領域の最終アドレス

図 5.1 にブート処理後の動作概要を示します。

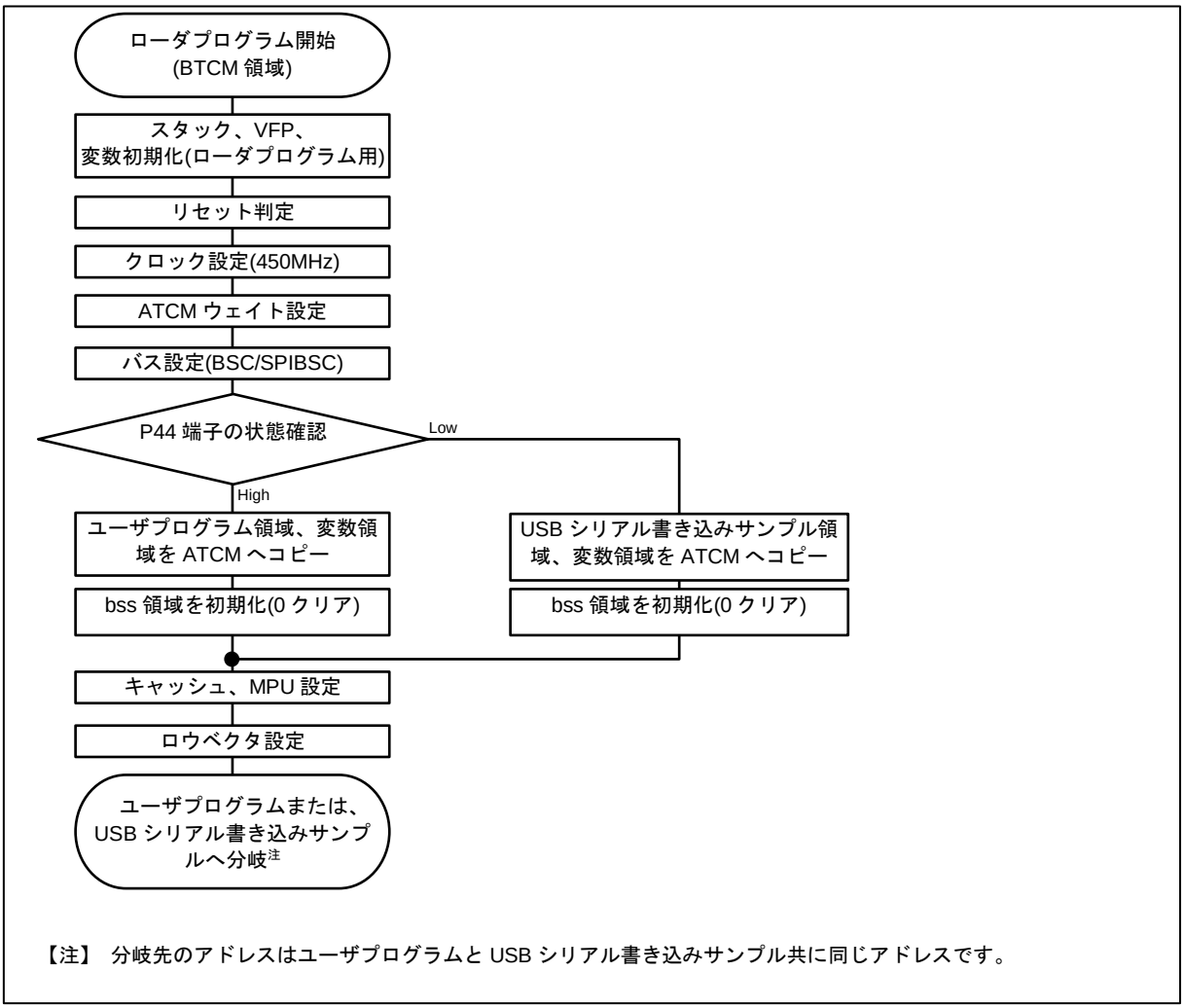


図 5.1 ブート処理後の動作概要

5.1.1　ユーザプログラム情報テーブル

以下に EWARM 版および e² studio 版のユーザプログラム情報テーブルを示します。

■EWARM 版

No	Address(Offset)	Description	Member	Value
1	0x00	シリアル FlashROM のユーザプログラム領域の先頭アドレス	user_prg_s_r_saddr	__section_begin ("USER_PRG_RBLOCK")
2	0x04	ATCM に格納するユーザプログラム領域の先頭アドレス	user_prg_d_r_saddr	__section_begin ("USER_PRG_WBLOCK")
3	0x08	ATCM に格納するユーザプログラム領域の最終アドレス	user_prg_d_r_eaddr	__section_end ("USER_PRG_WBLOCK")
4	0x0C	シリアル FlashROM の Cortex-M3 用ユーザプログラム領域の先頭アドレス	cm3_s_saddr	__section_begin ("CM3_SECTION ")
5	0x10	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域の先頭アドレス	cm3_d_saddr	__section_begin ("CM3_SECTION_WBLOCK ")
6	0x14	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域のサイズ	cm3_d_size	__section_size ("CM3_SECTION")
7	0x18	シリアル FlashROM のユーザプログラム用変数領域の先頭アドレス	user_prg_s_w_saddr	__section_begin ("USER_DATA_RBLOCK")
8	0x1C	ATCM に格納するユーザプログラム用変数領域の先頭アドレス	user_prg_d_w_saddr	__section_begin ("USER_DATA_WBLOCK")
9	0x20	ATCM に格納するユーザプログラム用変数領域の最終アドレス	user_prg_d_w_eaddr	__section_end ("USER_DATA_WBLOCK")
10	0x24	Dummy	dummy_addr	0x00000000
11	0x28	bss 領域の先頭アドレス	user_prg_d_saddr	__section_begin ("USER_DATA_ZBLOCK")
12	0x2C	bss 領域の最終アドレス	user_prg_d_eaddr	__section_end ("USER_DATA_ZBLOCK")

user_prg_info_tbl.h でユーザプログラム情報テーブル構造体を宣言し、loader_init2.c でユーザプログラム情報テーブルを宣言しています。ユーザプログラム情報テーブルは、リンカ設定ファイル(RZ_T1_R-IN_init¥Cortex-R4¥RZ_T1_init_boot¥src¥common¥serial_boot¥RZ_T1_init_serial_boot.icf)のセクションを使用することで、ビルド時に各領域のアドレス情報が格納されます。

【ユーザプログラム情報テーブル構造体(user_prg_info_tbl.h)】

```
typedef struct
{
    uint32_t user_prg_s_r_saddr;
    uint32_t user_prg_d_r_saddr;
    uint32_t user_prg_d_r_eaddr;
    uint32_t cm3_s_saddr;
    uint32_t cm3_d_saddr;
    uint32_t cm3_d_size;
    uint32_t user_prg_s_w_saddr;
    uint32_t user_prg_d_w_saddr;
    uint32_t user_prg_d_w_eaddr;
    uint32_t dummy_addr;
    uint32_t user_prg_d_saddr;
    uint32_t user_prg_d_eaddr;
}st_user_prg_info_tbl_t;
```

【ユーザプログラム情報テーブル(loader_init2.c)】

```
const st_user_prg_info_tbl_t Usr_Prog_Info_Table=
{
    (uint32_t) __section_begin("USER_PRG_RBLOCK"),
    (uint32_t) __section_begin("USER_PRG_WBLOCK"),
    (uint32_t) __section_end   ("USER_PRG_WBLOCK"),
    (uint32_t) __section_begin("CM3_SECTION"),
    (uint32_t) __section_begin("CM3_SECTION_WBLOCK"),
    (uint32_t) __section_size("CM3_SECTION"),
    (uint32_t) __section_begin("USER_DATA_RBLOCK"),
    (uint32_t) __section_begin("USER_DATA_WBLOCK"),
    (uint32_t) __section_end   ("USER_DATA_WBLOCK"),
    0x00000000U,
    (uint32_t) __section_begin("USER_DATA_ZBLOCK"),
    (uint32_t) __section_end   ("USER_DATA_ZBLOCK")
};
```

■e² studio 版(user_prog_info_tbl.asm)

No	Address(Offset)	Description	Value
1	0x00	シリアル FlashROM のユーザプログラム領域の先頭アドレス	_main_text
2	0x04	ATCM に格納するユーザプログラム領域の先頭アドレス	_main_text_start
3	0x08	ATCM に格納するユーザプログラム領域の最終アドレス	_text_end
4	0x0C	シリアル FlashROM の Cortex-M3 用ユーザプログラム領域の先頭アドレス	_cm3
5	0x10	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域の先頭アドレス	_cm3_start
6	0x14	Instruction RAM に格納する Cortex-M3 用ユーザプログラム領域の最終アドレス	_cm3_end
7	0x18	シリアル FlashROM のユーザプログラム用変数領域の先頭アドレス	_mdata
8	0x1C	ATCM に格納するユーザプログラム用変数領域の先頭アドレス	_data_start
9	0x20	ATCM に格納するユーザプログラム用変数領域の最終アドレス	_data_end
10	0x24	Dummy	0x00000000
11	0x28	bss 領域の先頭アドレス	__bss_start__
12	0x2C	bss 領域の最終アドレス	__bss_end__

user_prog_info_tbl.asm でユーザプログラム情報テーブルを宣言しています。ユーザプログラム情報テーブルは、リンカスクリプトファイル(¥RZ_T1_R-IN_init_sflash¥sample_cr4¥src¥linker_scriptHardwareDebug.ld)のセクションを使用することで、ビルド時に各領域のアドレス情報が格納されます。

【ユーザプログラム情報テーブル(user_prog_info_tbl.asm)】

```
_Usr_Prog_Info_Table:
    .word _main_text
    .word _main_text_start
    .word _text_end
    .word _cm3
    .word _cm3_start
    .word _cm3_end
    .word _mdata
    .word _data_start
    .word _data_end
    .word 0x00000000
    .word __bss_start__
    .word __bss_end__
```

5.2　メモリマップ

RZ/T1 グループのアドレス空間については、ユーザーズマニュアル ハードウェア編に記載しています。

5.3　サンプルプログラムのセクション配置

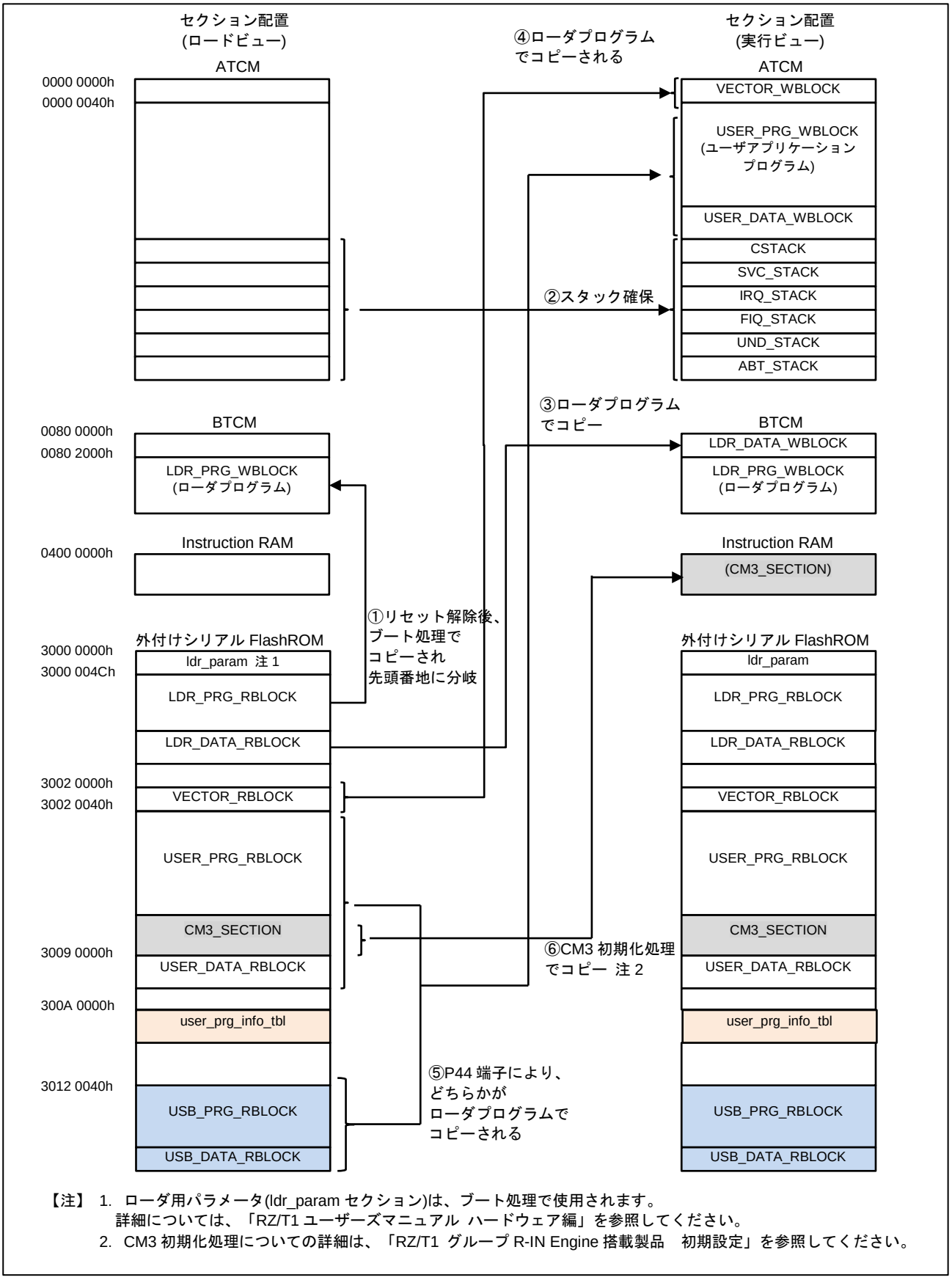
表 5.2 に Cortex-R4、表 5.3 に Cortex-M3 の使用するセクションを示し、図 5.2 に Cortex-R4、図 5.3 に Cortex-M3 のセクション配置を示します。

表 5.2 使用するセクション（Cortex-R4）

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_WBLOCK	リセット、例外ベクタテーブル	Code	―	ATCM
USER_PRG_WBLOCK	ユーザプログラム領域（実行用）	Code	―	ATCM
USER_DATA_WBLOCK	ユーザプログラム用変数領域（実行用）	Data	―	ATCM
CSTACK	スタック領域	Data	―	ATCM
SVC_STACK	スーパバイザ（SVC）モードのスタック領域	Data	―	ATCM
IRQ_STACK	IRQ モードのスタック領域	Data	―	ATCM
FIQ_STACK	FIQ モードのスタック領域	Data	―	ATCM
UND_STACK	未定義（UND）モードのスタック領域	Data	―	ATCM
ABT_STACK	アボート（ABT）モードのスタック領域	Data	―	ATCM
LDR_DATA_WBLOCK	ローダプログラム用変数領域（実行用）	Data	―	BTCM
LDR_PRG_WBLOCK	ローダプログラム領域（実行用）	Code	―	BTCM
ldr_param	ローダ用パラメータ	Data	FLASH	―
LDR_PRG_RBLOCK	ローダプログラム領域（格納用）	Code	FLASH	―
LDR_DATA_RBLOCK	ローダプログラム用変数領域（格納用）	Data	FLASH	―
VECTOR_RBLOCK	リセット、例外ベクタテーブル領域（格納用）	Code	FLASH	―
USER_PRG_RBLOCK	ユーザプログラム領域（格納用）、Cortex-M3 用ユーザプログラム領域（格納用）	Code	FLASH	―
USER_DATA_RBLOCK	ユーザプログラム用変数領域（格納用）	Data	FLASH	―
user_prg_info_tbl	ユーザプログラム情報テーブル領域	Data	FLASH	―
USB_PRG_RBLOCK	USB シリアル書き込みサンプル領域(格納用)	Code	FLASH	―
USB_DATA_RBLOCK	USB シリアル書き込みサンプル用変数領域(格納用)	Data	FLASH	―

表 5.3 使用するセクション（Cortex-M3）

領域の名前	内容	タイプ	ロード領域	実行領域
vectors	ベクタ領域	Code	―	Instruction RAM
readonly	ユーザプログラム領域	Code	―	Instruction RAM
_SHARED_MEM	共用メモリ領域	Data	―	Data RAM
readwrite	ユーザプログラム用変数領域	Data	―	Data RAM
HEAP	ヒープ領域	Data	―	Data RAM
CSTACK	スタック領域	Data	―	Data RAM



【注】 1. ローダ用パラメータ(ldr_param セクション)は、ブート処理で使用されます。
詳細については、「RZ/T1 ユーザーズマニュアル ハードウェア編」を参照してください。
2. CM3 初期化処理についての詳細は、「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を参照してください。

図 5.2 セクション配置（Cortex-R4）

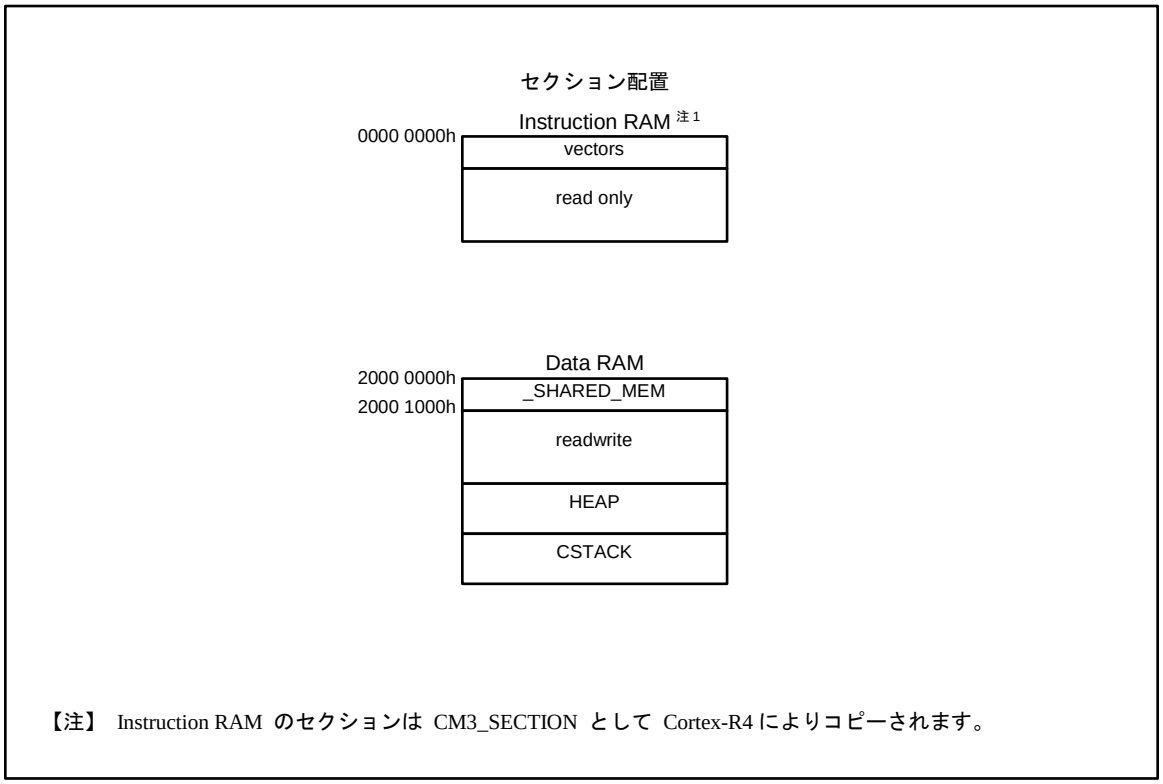


図 5.3 セクション配置（Cortex-M3）

5.4　USB シリアル書き込みサンプル

5.4.1　概要

USB シリアル書き込みサンプルは、USB 初期設定を行い、ホスト PC とハンドシェイクするためにターミナルソフトからの入力(任意データ)を待ちます。ターミナルソフトからの入力後、ホスト PC からのコマンドに応じて処理を行います。以下に USB シリアル書き込みサンプルの機能を示します。

1. シリアル FlashROM 領域にユーザプログラムを書き込み
2. シリアル FlashROM 領域の読み出し
3. セクターイレース
4. Protect Control

図 5.4 に USB シリアル書き込みサンプルの概要を示します。

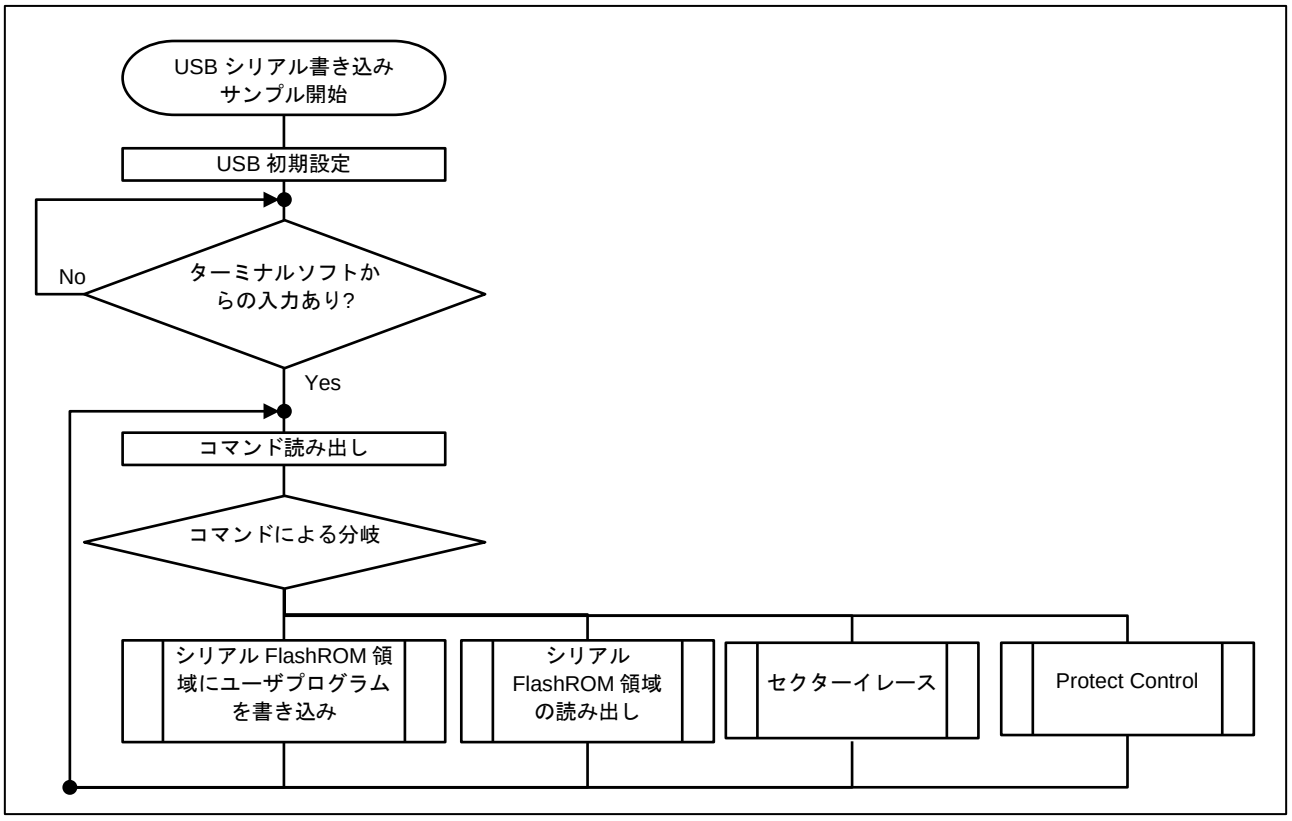


図 5.4 USB シリアル書き込みサンプル概要

5.4.2　ソフトウェア構成

図 5.5 に USB シリアル書き込みサンプルのソフトウェア構成を示します。

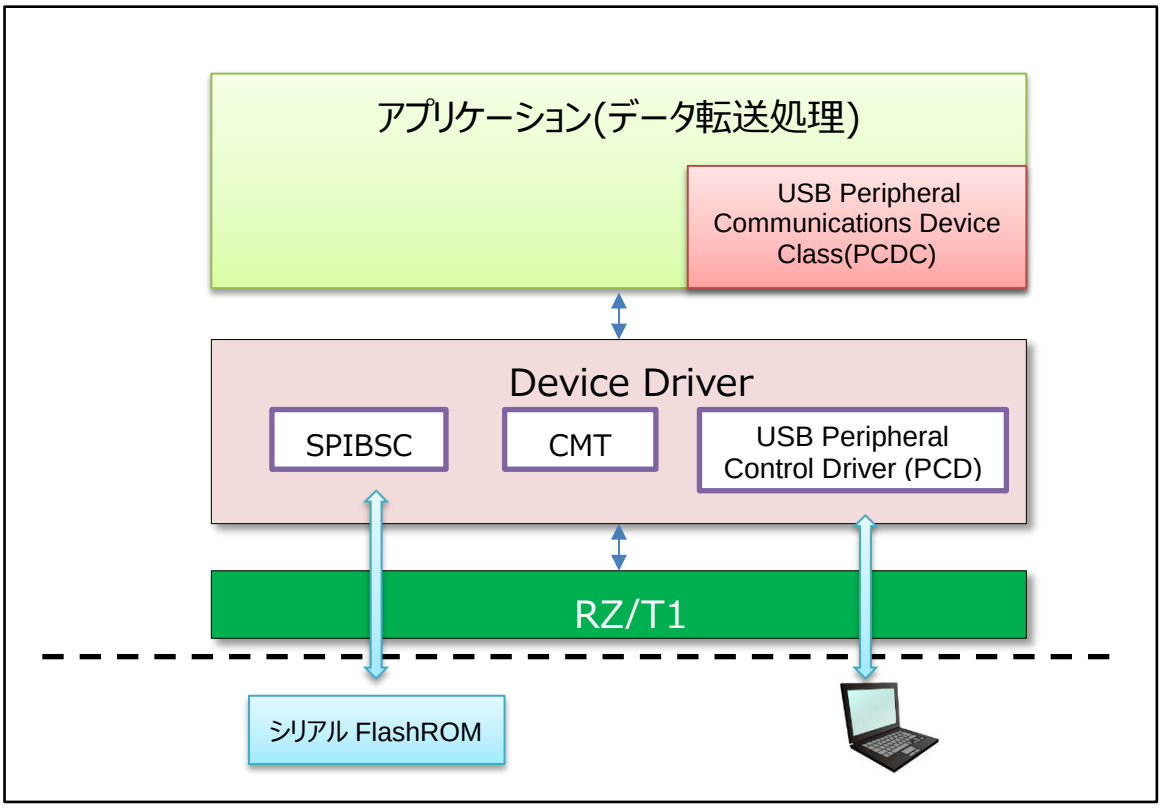


図 5.5 ソフトウェア構成

5.4.3　ソフトウェア詳細

(1)　定数一覧

表 5.4 に USB シリアル書き込みサンプルで使用する定数を示します。

表 5.4 定数一覧

定数	設定値	内容
CDC_DATA_LEN	1024	USB 転送データ長
EVENT_MAX	5	イベント最大数
TASK_LOOPS_BETWEEN_INSTRUCTIONS	0x5000000	初期接続メッセージ格納アドレス
USB_CDC_DATA_INVALID	0x00	USB 受信バッファにデータなし
USB_CDC_DATA_VALID	0x01	USB 受信バッファにデータあり
ALIGN_SIZE	0x20	アライメントサイズ
CDC_REV_DATA_MAX	CDC_DATA_LEN	USB 受信サイズ
CDC_REV_BUF_DATA_MAX	1024+512	USB 受信バッファサイズ
R_SFALSH_WRITE_SZ	0x400	シリアル FlashROM へ書き込むデータサイズ
R_USB_SEND_WAIT	59u	USB 送信後の Wait 時間
R_SFALSH_SEND_SZ	511u	最大 USB 送信サイズ
SPIBSC_MIN_ADDR	0x00000000	SPIBSC 開始アドレス(0x30000000 アドレスを 0x00000000 とします)
SPIBSC_MAX_ADDR	0x03FFFFFF	SPIBSC 終了アドレス(0x30000000 アドレスを 0x00000000 とします)
PRCR_ACCESS_UNLOCK	0x0000A503	MSTPCR レジスタを書き込み許可状態にする
PRCR_ACCESS_LOCK	0x0000A500	MSTPCR レジスタを書き込み禁止状態にする
LENGTH_PROTECT_CODE	4	SPIBSC プロテクトコードバッファサイズ
COMMAND_PROTECT_CONTROL	2	SPIBSC プロテクトコントロールコマンド
COMMAND_WRITE	3	SPIBSC ライトコマンド
COMMAND_READ	4	SPIBSC リードコマンド
COMMAND_SECTOR_ERASE	6	SPIBSC セクターイレースコマンド
COMMAND_ERROR	11	コマンドエラー
SCIF_CMD_PROTECT_LOWER	“p”	プロテクトコントロールコマンド(小文字)
SCIF_CMD_PROTECT_UPPER	“P”	プロテクトコントロールコマンド(大文字)
SCIF_CMD_WRITE_LOWER	“w”	ライトコマンド(小文字)
SCIF_CMD_WRITE_UPPER	“W”	ライトコマンド(大文字)
SCIF_CMD_READ_LOWER	“r”	リードコマンド(小文字)
SCIF_CMD_READ_UPPER	“R”	リードコマンド(大文字)
SCIF_CMD_SECTOR_ERASE_LOWER	“e”	セクターイレースコマンド(小文字)

SCIF_CMD_SECTOR_ERASE_LOWER	“E”	セクターイレースコマンド(大文字)
SCIF_COLUMN_LEN_ERR_CODE	2	エラーコード値の桁長
SCIF_LEN_ERR_CODE_STRING	6	エラーコード文字列の長さ
SCIF_LEN_CMD	1	コマンドの文字列長
SCIF_MAX_LEN_COMMAND	115	受け取るコマンドの最大文字列長
SCIF_LEN_CMD_SPACE	1	コマンドにおけるスペースの長さ
SCIF_LEN_RETURN_CODE	2	改行コード(CR+LF)の長さ
SCIF_LEN_RETURN_CODE_HALF	1	改行コードどちらか単体(CR or LF)の長さ
SCIF_LEN_PREFIX_HEX	2	16 進数文字列における”0x”部分の長さ
SCIF_MULTIPLIER_2_TO_16	4	16 を表すための 2 の累乗
SCIF_CLEAR_UPPER_COLUMN_10	10	0x0A 以上の値を文字列から変換する場合の補正値
SCIF_LEN_1BYTE_DATA	4	1byte データ(0xXX)の文字列長
SCIF_DIGIT_1BYTE_DATA	2	1byte データ桁の大きさ
SCIF_MAX_LEN_32BIT_STRING	10	32bit データの最大文字列長(“0x”を含む)
SCIF_CHANGE_VALUE_UPPERCASE	0x37	大文字を数値に変換するための数
SCIF_CHANGE_VALUE_LOWERCASE	0x57	小文字を数値に変換するための数
CMD_DEBUG_MODE_LOWER	'd'	コマンド入力開始文字ありにするコマンド(小文字)
CMD_DEBUG_MODE_UPPER	'D'	コマンド入力開始文字ありにするコマンド(大文字)
CMTW_TIMEOUT_REC_DATA	1464843	データ受信タイムアウト時間(10s)
CMTW_TIMEOUT_MESSAGE	1464843	データ送信タイムアウト時間(10s)
CMTW0_CMWI0_NUM	25	CMTW の割り込みベクタ番号
ERROR_CODE_SUCCESS	0x00	処理が成功した
ERROR_CODE_FILE_TRANSFER	0xFF	ファイル転送に失敗
ERROR_CODE_PARAM_ERROR	0xFE	パラメータが異常値
ERROR_CODE_VERIFY	0xFC	書き込んだデータのベリファイに失敗
ERROR_CODE_NO_CORRESPOND	0xFB	未対応のコマンドを受信した
ERROR_CODE_TIMEOUT	0xFA	タイムアウトエラーが発生した
ERROR_CODE_HW_ERROR	0xF8	HW エラーが発生
INTERNAL_ERROR_SUCCESS	0	処理が成功した(プログラム内部用戻り値)
INTERNAL_ERROR_SCIF_ERROR	-1	HW エラーが発生した(プログラム内部用戻り値)
INTERNAL_ERROR_SCIF_TIMEOUT	-2	タイムアウトエラーが発生した(プログラム内部用戻り値)

(2) 構造体/共用体/列挙型一覧

以下に USB シリアル書き込みサンプルで使用する構造体/共用体/列挙型を示します。

表 5.5 DEV_INFO_t 構造体

メンバ名	内容
uint16_t　state	State of the application
uint16_t　event_cnt	Event count
uint16_t　event[EVENT_MAX]	Event no.

表 5.6 STATE_t 列挙型

メンバ名	内容
STATE_ATTACH	アタッチ処理
STATE_DATA_TRANSFER	データ転送処理
STATE_DETACH	デタッチ処理

表 5.7 EVENT_t 列挙型

メンバ名	内容
EVENT_NONE	イベントなし
EVENT_CONFIGERD	USB デバイス接続完了
EVENT_USB_READ_START	USB データ受信要求
EVENT_USB_READ_COMPLETE	USB データ受信完了
EVENT_USB_WRITE_START	USB データ送信要求
EVENT_USB_WRITE_COMPLETE	USB データ送信完了
EVENT_COM_NOTIFY_START	Class Notification“SerialState” 送信要求
EVENT_COM_NOTIFY_COMPLETE	Class Notification“SerialState” 送信完了

表 5.8 USB_PCDC_APL_STATE 列挙型

メンバ名	内容
APP_STATE_IDLE	アイドル状態
APP_STATE_ECHO_MODE	エコーモード

表 5.9 bootloader_ctrl_t 構造体

メンバ名	内容
uint8_t　cmd	実行するコマンド
uint32_t　timeout_err_flag	タイムアウトエラーの発生状況を示す。true 時にエラー発生
uint8_t　*target_buf	コマンドの実行対象になるバッファのポインタ
uint8_t　*src_buf	データコピー時にコピー元になるバッファのポインタ
uint32_t　target_size	コマンドで取り扱うデータのサイズ
uint8_t　*protect_code	プロテクトコード領域へのポインタ
uint32_t　protect_code_size	プロテクトコード領域のサイズ

(3)　大域変数一覧

表 5.10 に大域変数を示します。

表 5.10　大域変数一覧

型	変数名	内容
static uint8_t	gb_rec_buf [R_SFALSH_WRITE_SZ]	シリアル FlashROM 用データバッファ
static uint8_t	gb_spibsc_protect_code [LENGTH_PROTECT_CODE]	SPIBSC のプロテクトコードを保存するためのバッファ
static uint32_t	g_cmtw_start_flag	CMTW2 重スタートチェックフラグ
static bootloader_ctrl_t	bootloader_param	USB シリアル書き込みサンプル管理パラメータ
static volatile uint8_t	usbf_wfin_flag	USB 送信完了フラグ
static uint8_t	cmd_enter_mode_en	コマンド入力開始文字フラグ (true がコマンド入力開始文字ありモード、false がコマンド入力開始文字なしモード)
static uint8_t	receive_timeout_en	USB 受信タイムアウト設定フラグ
uint8_t	cdc_trans_data_base [CDC_DATA_LEN + ALIGN_SIZE]	USB 送受信バッファ
uint8_t*	cdc_trans_data	USB 送受信バッファのポインタ
static uint8_t	cdc_rev_data [CDC_REV_BUF_DATA_MAX]	USB 受信バッファ
static uint32_t	cdc_rev_data_pw	USB 受信バッファにデータが書き込まれた位置
static uint32_t	cdc_rev_data_pr	USB 受信バッファからデータが読み込まれた位置

(4)　関数一覧

表 5.11 に関数一覧を示します。

表 5.11　関数一覧

関数名	概要	スコープ	定義ファイル
main	ユーザプログラムメイン	global	main.c
port_init	ポート設定	global	main.c
ecm_init	ECM 初期設定	global	main.c
icu_init	割り込み設定	global	main.c
usbf_main	USB ファンクションメイン	global	r_usb_pcdc_apl.c
cdc_connect_wait 注 1	USB 接続待機	global	r_usb_pcdc_apl.c
cdc_detach_device 注 1	デタッチ	global	r_usb_pcdc_apl.c
cdc_configured 注 1	デバイスコンフィガード用コールバック関数	global	r_usb_pcdc_apl.c
cdc_detach 注 1	デタッチ処理コールバック関数	global	r_usb_pcdc_apl.c
cdc_default 注 1	デフォルト処理コールバック関数	global	r_usb_pcdc_apl.c
cdc_suspend 注 1	サスペンド処理コールバック関数	global	r_usb_pcdc_apl.c
cdc_resume 注 1	レジューム処理コールバック関数	global	r_usb_pcdc_apl.c
cdc_interface 注 1	インタフェース処理コールバック関数	global	r_usb_pcdc_apl.c
cdc_registration 注 1	デバイスドライバ登録	global	r_usb_pcdc_apl.c
apl_init 注 1	USB ファンクションメイン初期化	global	r_usb_pcdc_apl.c
cdc_event_set 注 1	イベント発行	global	r_usb_pcdc_apl.c
cdc_event_get 注 1	イベント取得	global	r_usb_pcdc_apl.c
r_data_trans_sFlash_writing	データ転送＆シリアル FlashROM 書き込みメイン	global	r_usb_data_trans_sFlash_writing.c
get_boot_param_pointer	USB シリアル書き込みサンプル管理パラメータのポインタを取得	global	r_usb_data_trans_sFlash_writing.c
scif_putc	文字を USB 送信	global	r_usb_data_trans_sFlash_writing.c
scif_getc	USB データ受信	static	r_usb_data_trans_sFlash_writing.c
scif_puts	文字列の USB 送信	static	r_usb_data_trans_sFlash_writing.c
spibsc_set_protect_code	SPIBSC のプロテクト制御コードを送信	static	r_usb_data_trans_sFlash_writing.c
spibsc_verify_data	SPIBSC へ書き込んだデータのベリファイを実行	static	r_usb_data_trans_sFlash_writing.c
scif_get_cmd	コマンドを取得	static	r_usb_data_trans_sFlash_writing.c
check_protect_control_command	Protect Control コマンドのフォーマットチェックを実行	static	r_usb_data_trans_sFlash_writing.c
check_write_command	write コマンドのフォーマットチェックを実行	static	r_usb_data_trans_sFlash_writing.c
check_read_command	read コマンドのフォーマットチェックを実行	static	r_usb_data_trans_sFlash_writing.c
check_sector_erase_command	セクターイレースコマンドのフォーマットチェックを実行	static	r_usb_data_trans_sFlash_writing.c
change_string_to_val	文字列を 1byte 単位の数値に変換する	static	r_usb_data_trans_

			sFlash_writing.c
change_errcode_to_string	数値を 16 進数表記の文字列に変換する	static	r_usb_data_trans_sFlash_writing.c
spibsc_init	SPIBSC の初期化	static	r_usb_data_trans_sFlash_writing.c
r_cdc_read_complete	USB 受信完了コールバック関数	static	r_usb_data_trans_sFlash_writing.c
r_cdc_write_complete	USB 送信完了コールバック関数	static	r_usb_data_trans_sFlash_writing.c
r_cdc_rev_data_is_valid	USB 受信バッファにデータの有無を確認	static	r_usb_data_trans_sFlash_writing.c
r_cdc_rev_data_clear	USB 受信バッファをクリア	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_write_data	USB 受信バッファへデータをライト	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_rev_data	USB 受信バッファのデータをリード	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_rev_1Bdata	USB 受信バッファのデータを 1 バイトリード	static	r_usb_data_trans_sFlash_writing.c
r_cdc_start	USB CDC スタート	static	r_usb_data_trans_sFlash_writing.c
r_usb_write_data_to_sFlash	データ受信とシリアル FlashROM への書き込みを実施	static	r_usb_data_trans_sFlash_writing.c
r_usb_read_data_from_sFlash	シリアル FlashROM の読み出しとデータ送信を実施	static	r_usb_data_trans_sFlash_writing.c
r_receive_data	データ受信を実施	static	r_usb_data_trans_sFlash_writing.c
r_write_sFlash	シリアル FlashROM への書き込みを実施	static	r_usb_data_trans_sFlash_writing.c
r_trans_data_payload	データ送信を実施	static	r_usb_data_trans_sFlash_writing.c

【注】 1．関数の詳細は、「RZ/T1 グループ USB Peripheral Communications Device Class Driver」を参照してください。

(5)　main

main	
概　要	ユーザプログラムメイン
宣　言	void main (void)
説　明	下記にフローチャートを示します。
引　数	None
リターン値	None
補足	本関数の開始アドレスを 0x00000040 に配置しています。

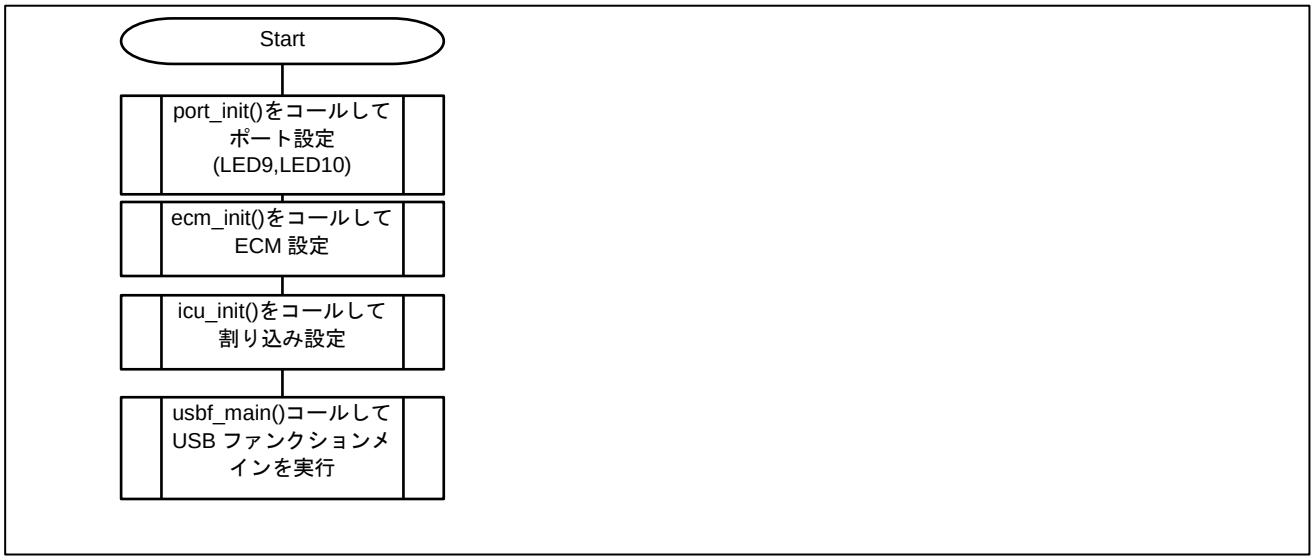


図 5.6 main 関数処理(Cortex-R4)

(6)　port_init

port_init	
概　要	ポート設定
宣　言	void port_init(void)
説　明	ポート PM7、ポート PM6 を出力設定し、low 出力します。
引　数	None
リターン値	None
補足	

(7)　ecm_init

ecm_init	
概　要	ECM 初期設定
宣　言	void ecm_init(void)
説　明	ECM 機能の初期化を行います
引　数	None
リターン値	None
補足	

(8)　icu_init

icu_init	
概　要	割り込み設定
宣　言	void icu_init(void)
説　明	割り込み許可に設定します。
引　数	None
リターン値	None
補足	

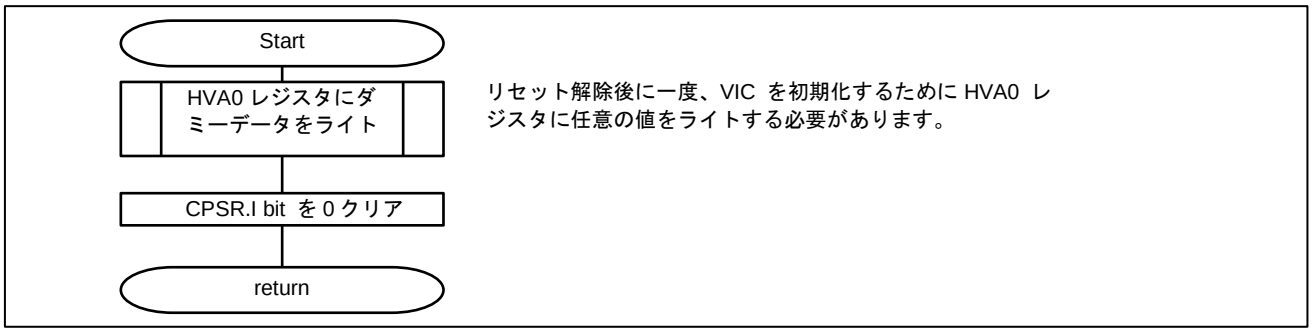


図 5.7 icu_init 関数処理

(9)　usbf_main

usbf_main	
概　要	USB ファンクションメイン
宣　言	void usbf_main(void)
説　明	サンプルプログラムのメイン処理です。下記にフローチャートを示します。 ステートとそのステートに関連するイベントにより管理を行っています。 ステートに関連するイベントの確認を行い、そのイベントに応じた処理を行います。 そのイベント処理後、アプリケーション は、必要に応じてステートを変化させます。 - STATE_ATTACH　アタッチ処理 - STATE_DATA_TRANSFER　データ転送＆シリアル FlashROM 書き込み処理 - STATE_DETACH　デタッチ処理
引　数	None
リターン値	None
補足	データ転送＆シリアル FlashROM 書き込みメイン処理は、無限ループとなるため、この関数に戻りません。

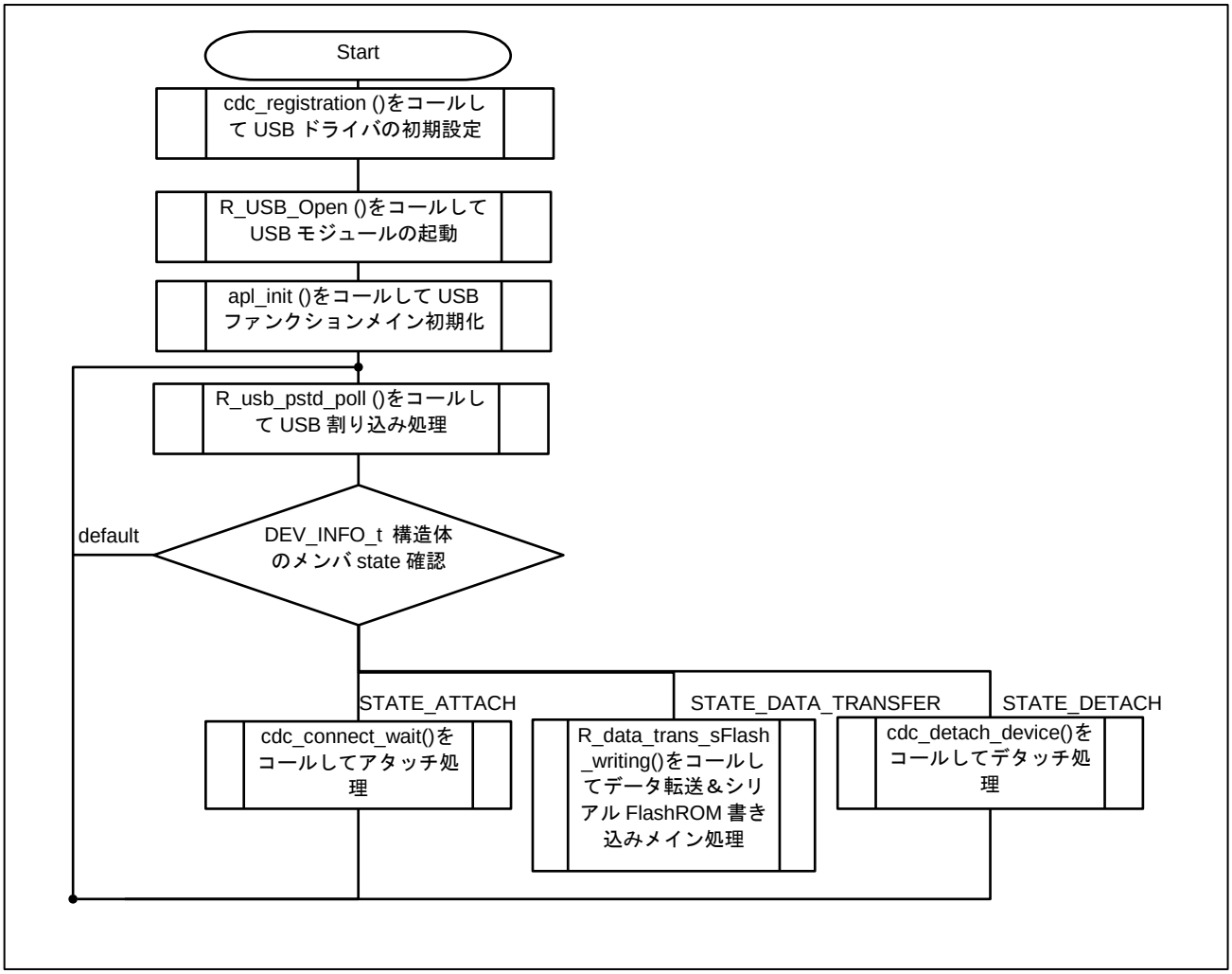
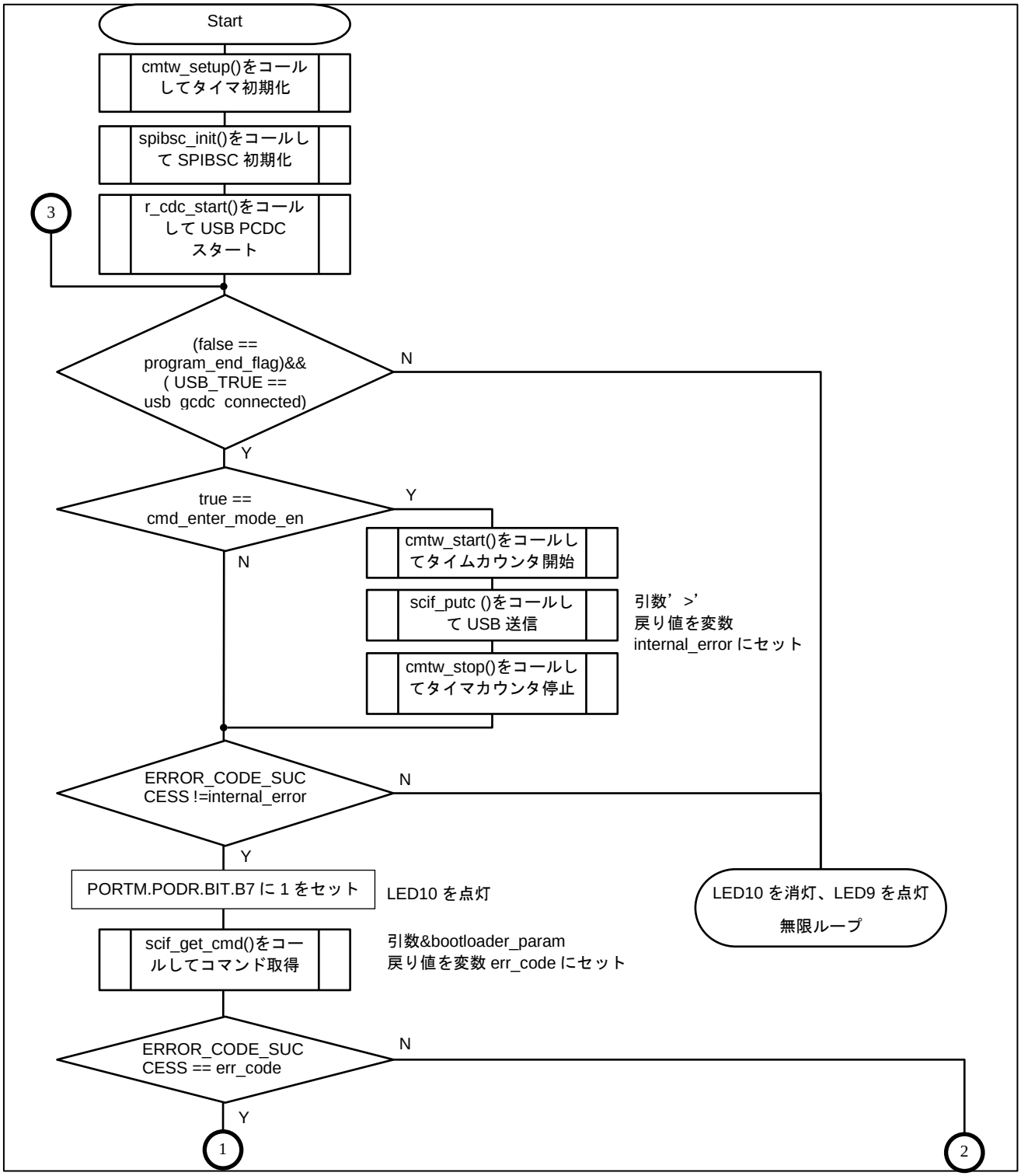


図 5.8 usbf_main 関数処理

(10)　r_data_trans_sFlash_writing

r_data_trans_sFlash_writing	
概　要	データ転送＆シリアル FlashROM 書き込みメイン
宣　言	void r_data_trans_sFlash_writing(void)
説　明	下記にフローチャートを示します。
引　数	None
リターン値	None
補足	



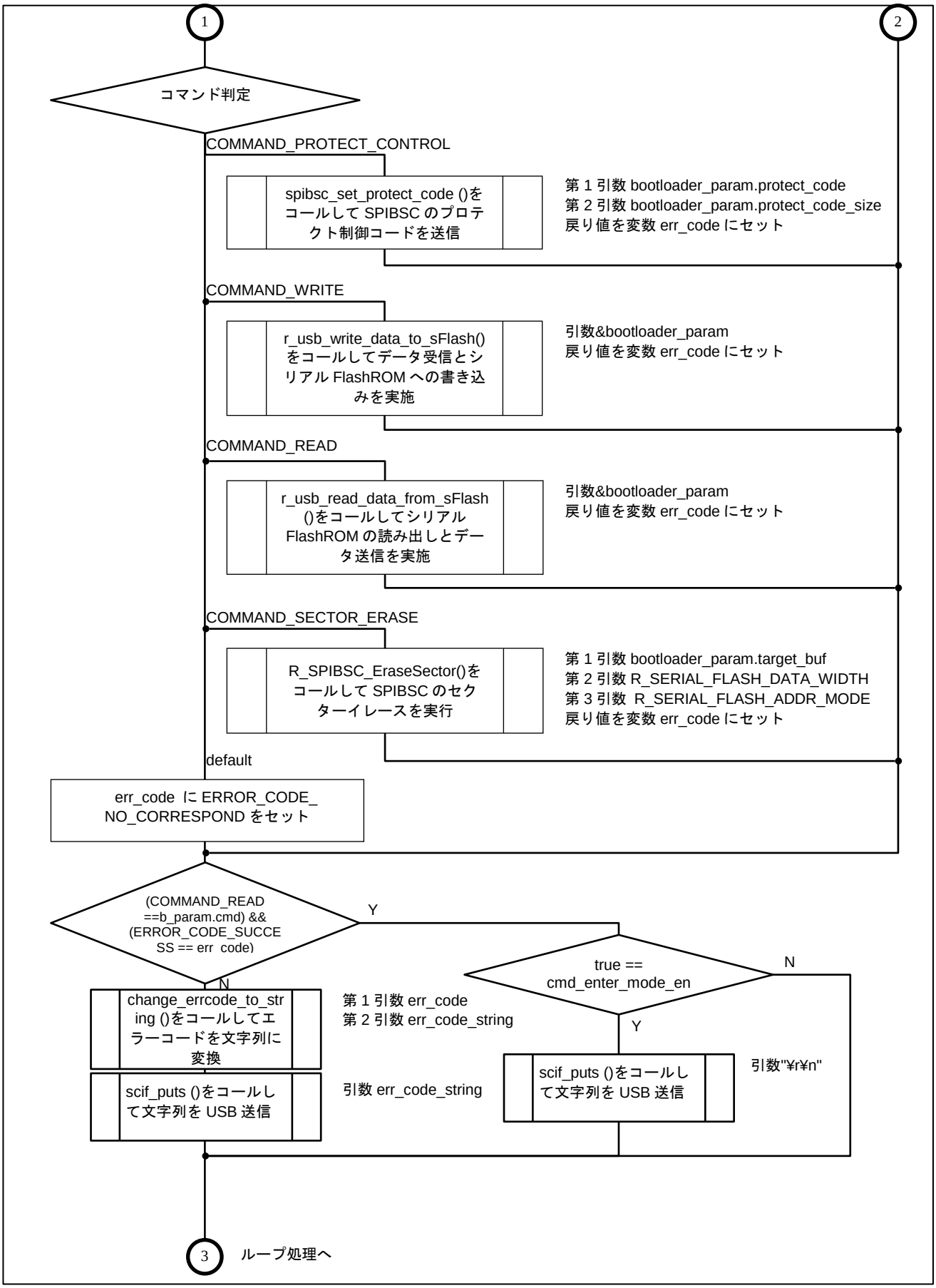


図 5.9 r_data_trans_sFlash_writing 関数処理

(11) get_boot_param_pointer

get_boot_param_pointer	
概　要	USB シリアル書き込みサンプル管理パラメータのポインタを取得
宣　言	bootloader_ctrl_t *get_boot_param_pointer(void)
説　明	変数 bootloader_param のポインタを渡す
引　数	None
リターン値	bootloader_param のポインタ
補足	

(12) scif_putc

scif_putc	
概　要	文字を USB 送信
宣　言	int32_t scif_putc(uint8_t cha)
説　明	文字を USB 送信します。
引　数	uint8_t cha　　文字
リターン値	INTERNAL_ERROR_SUCCESS：正常終了 INTERNAL_ERROR_SCIF_TIMEOUT：タイムアウト発生 INTERNAL_ERROR_SCIF_ERROR：USB がデタッチされた
補足	

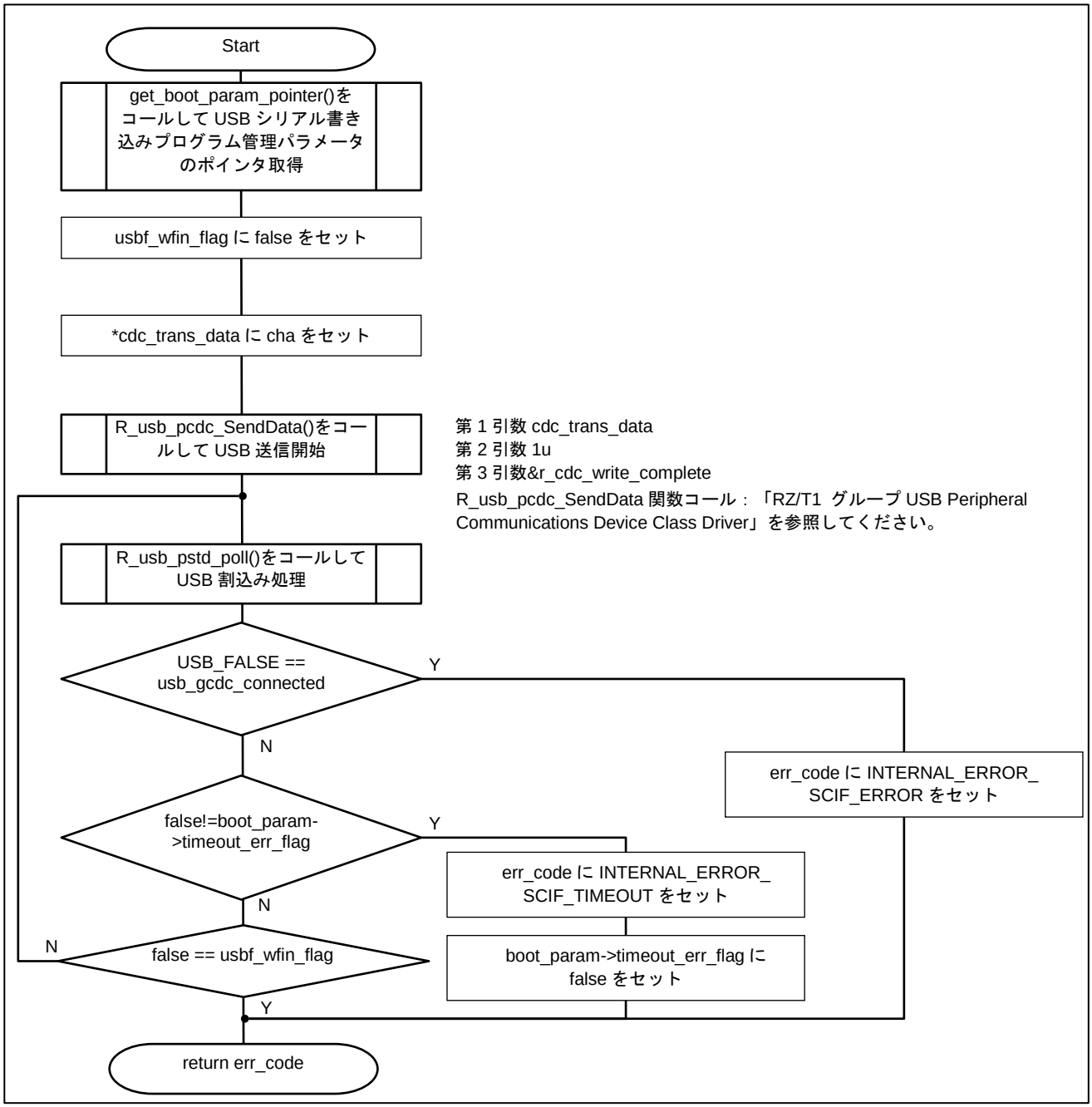
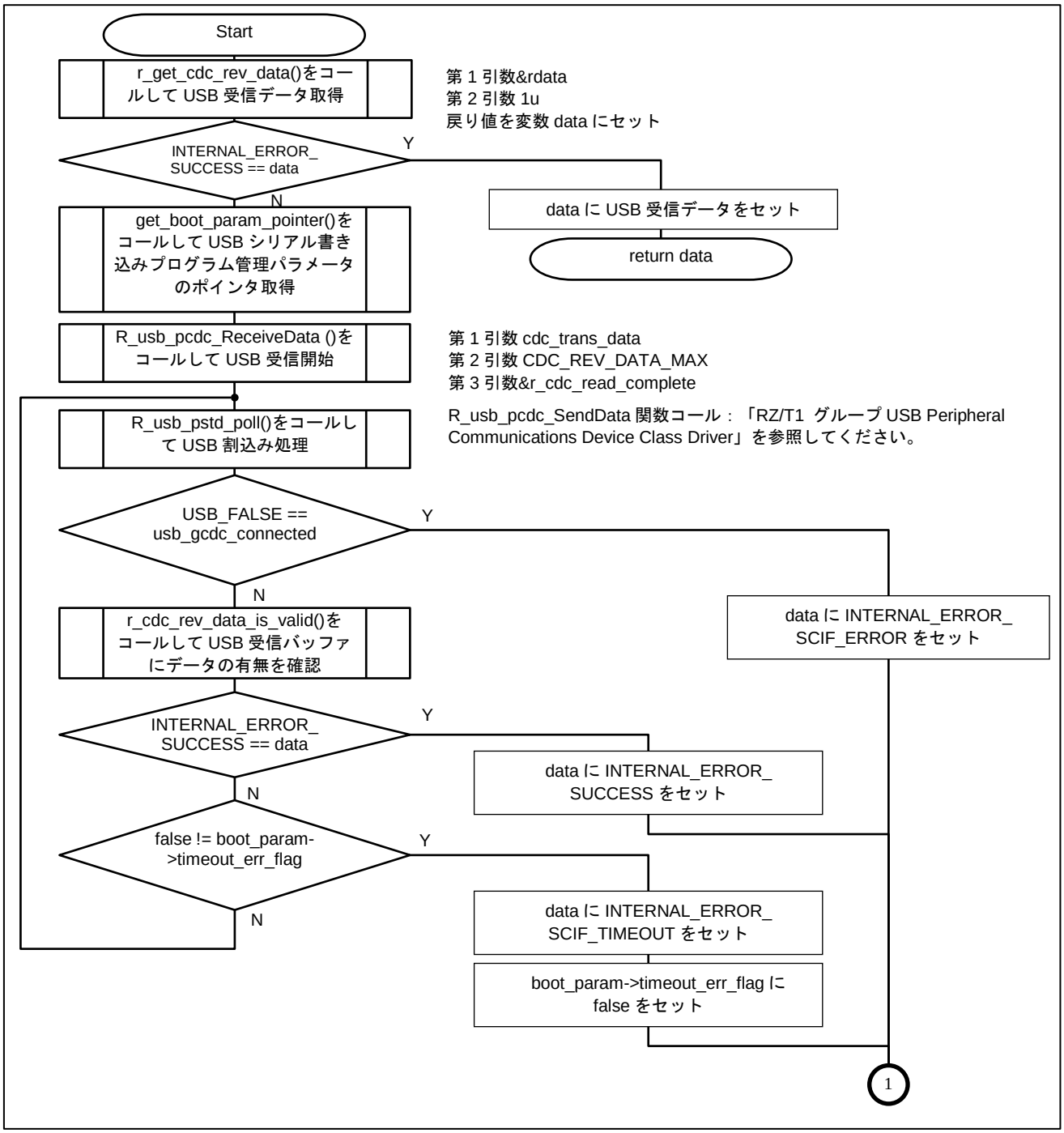


図 5.10 scif_putc 関数処理

(13) scif_getc

scif_getc	
概　要	USB データ受信
宣　言	static int32_t scif_getc(void)
説　明	USB データ受信を実行します
引　数	None
リターン値	正数値：受信データ INTERNAL_ERROR_SCIF_TIMEOUT：タイムアウト発生 INTERNAL_ERROR_SCIF_ERROR：受信エラーが発生、USB がデタッチされた
補足	



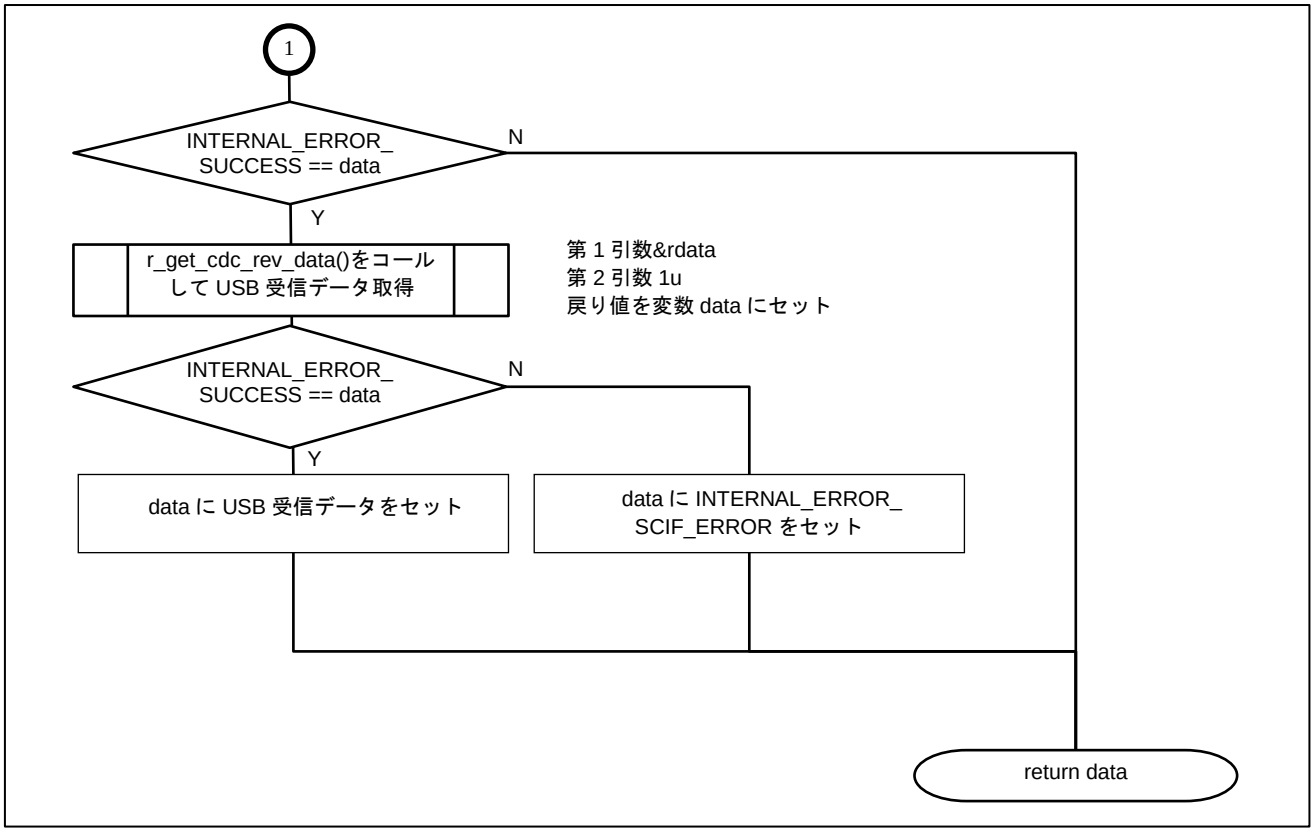


図 5.11 scif_getc 関数処理

(14) scif_puts

scif_puts	
概　要	文字列を USB 送信
宣　言	static int32_t scif_puts(uint8_t *p_str)
説　明	文字列の送信を実行します
引　数	uint8_t *p_str　　文字列データ
リターン値	INTERNAL_ERROR_SUCCESS：正常終了 INTERNAL_ERROR_SCIF_ERROR：引数エラー
補足	

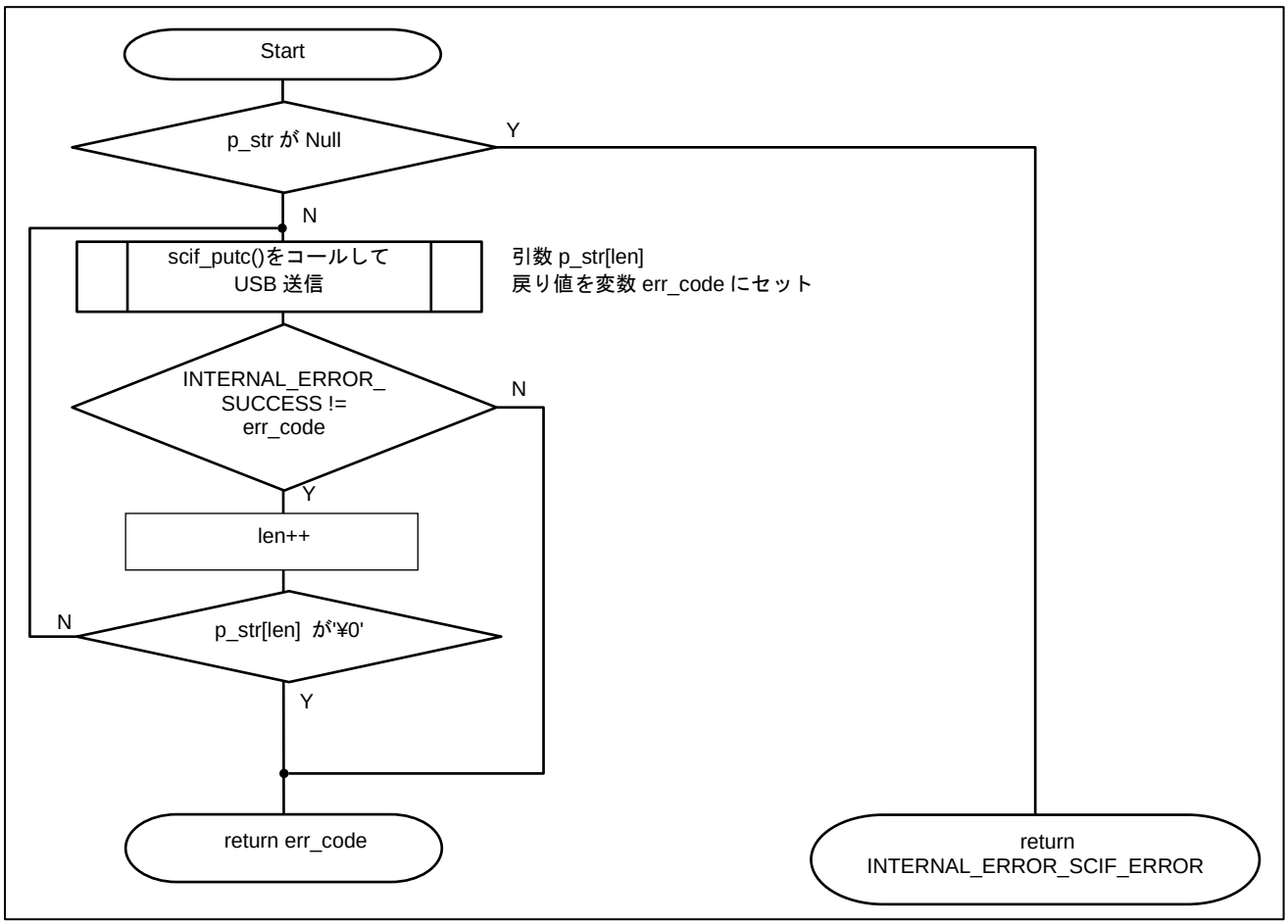
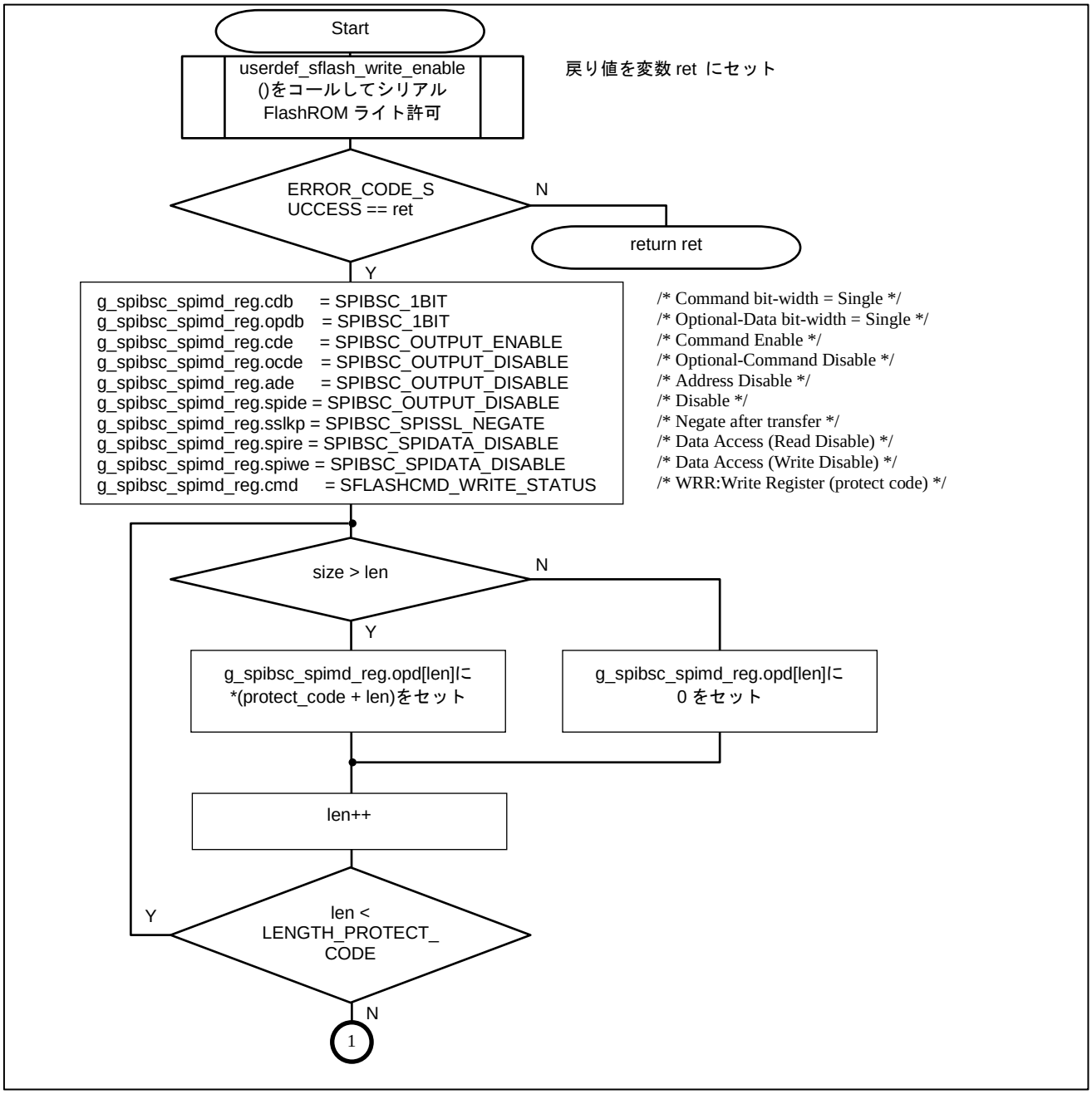


図 5.12 scif_puts 関数処理

(15) spibsc_set_protect_code

spibsc_set_protect_code		
概　要	SPIBSC のプロテクト制御コードを送信	
宣　言	static uint8_t spibsc_set_protect_code(uint8_t *protect_code, uint32_t size)	
説　明	下記にフローチャートを示します。	
引　数	uint8_t *protect_code	プロテクト制御コードのポインタ
	uint32_t size	プロテクト制御コード長
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_FILE_TRANSFER：シリアル FlashROM へのデータ転送失敗	
	ERROR_CODE_TIMEOUT：タイムアウト(10s)発生	

補足



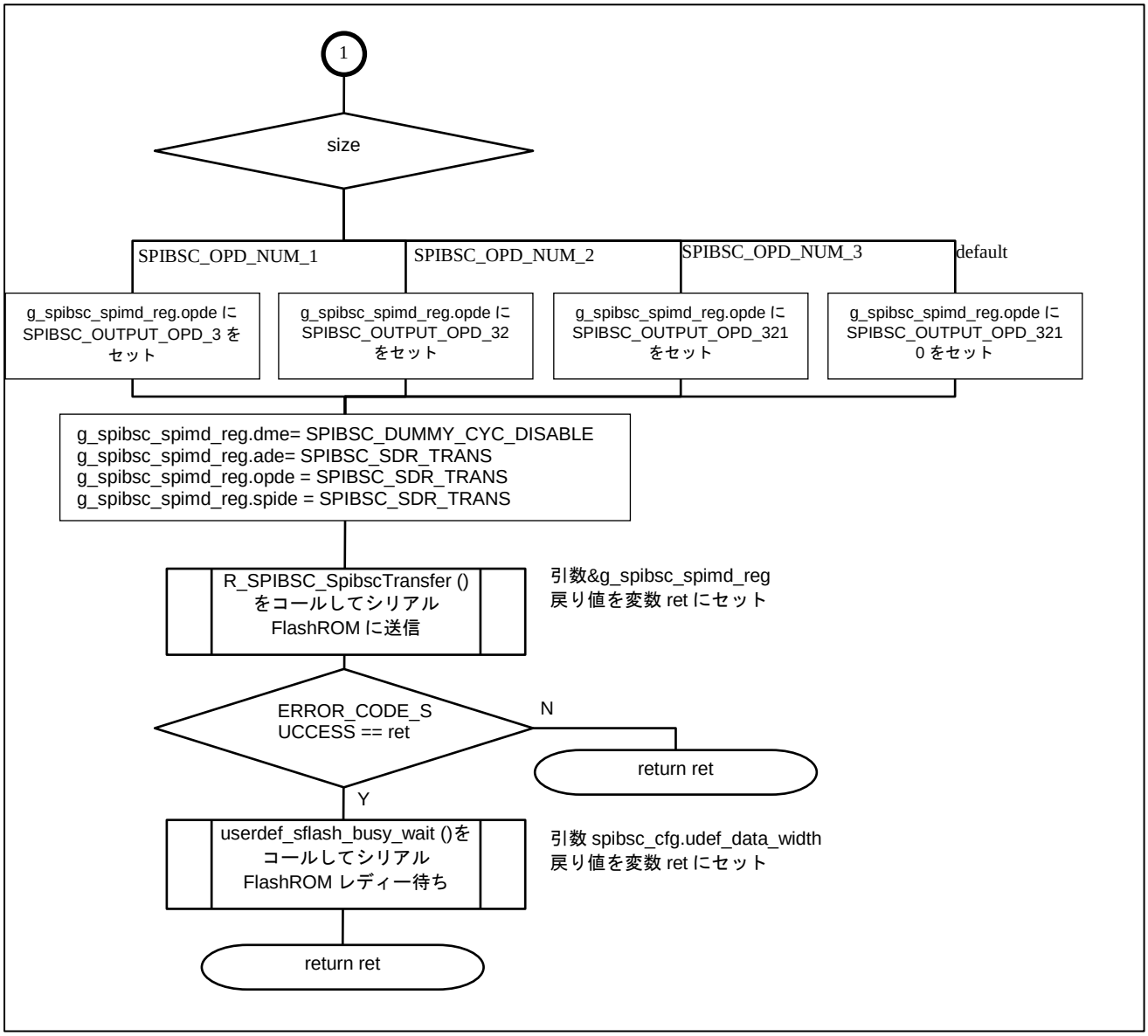


図 5.13 spibsc_set_protect_code 関数処理

(16) spibsc_verify_data

spibsc_verify_data	
概　要	SPIBSC へ書き込んだデータのベリファイを実行
宣　言	static uint8_t spibsc_verify_data(uint32_t target_addr, uint8_t *src_addr, int32_t size)
説　明	下記にフローチャートを示します。
引　数	uint32_t target_addr　　書き込み先データアドレス uint8_t *src_addr　　書き込み元データバッファのポインタ uint32_t size　　書き込みデータサイズ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_FILE_TRANSFER：シリアル FlashROM へのデータ転送失敗 ERROR_CODE_VERIFY：ベリファイエラー
補足	

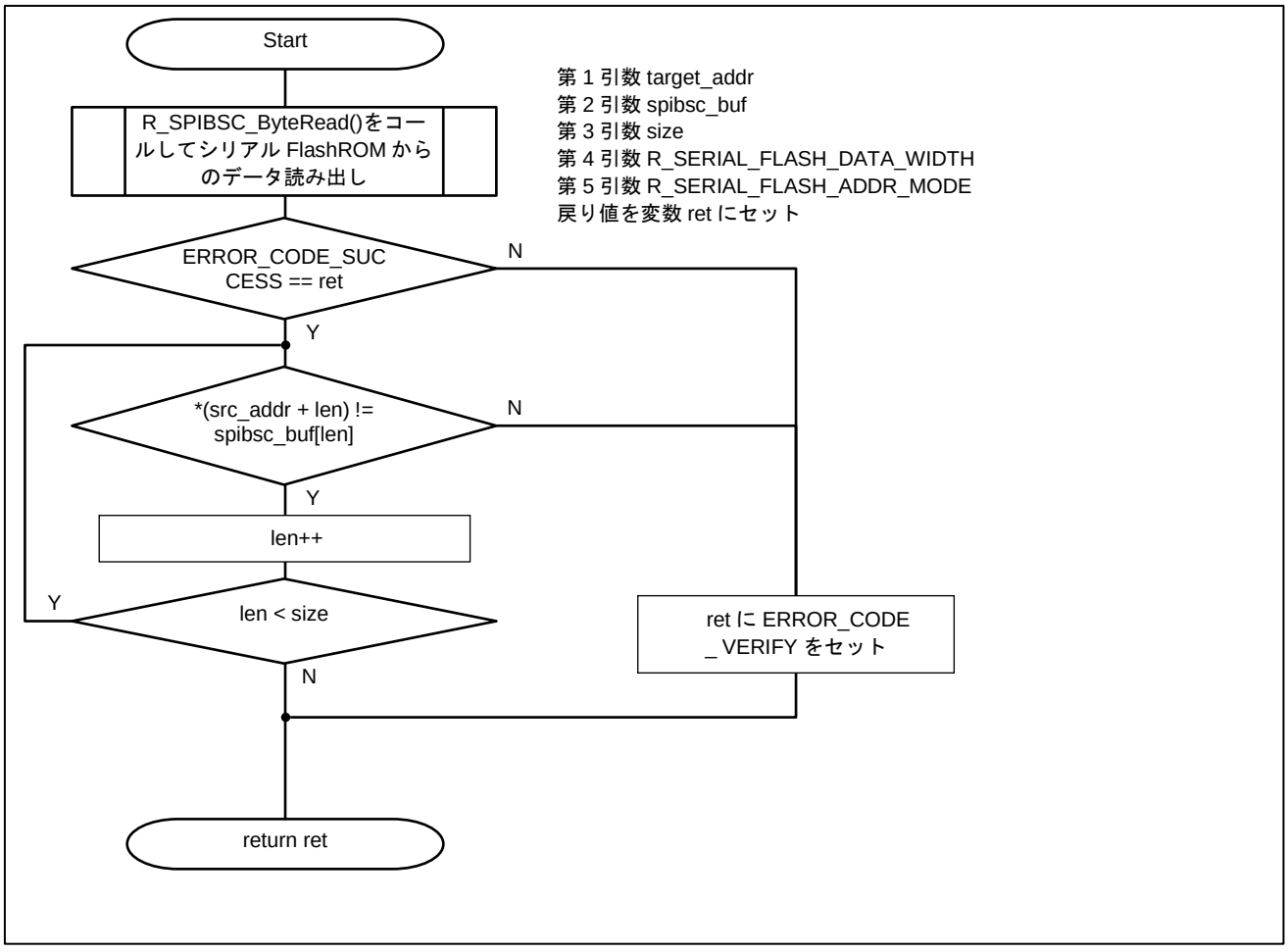
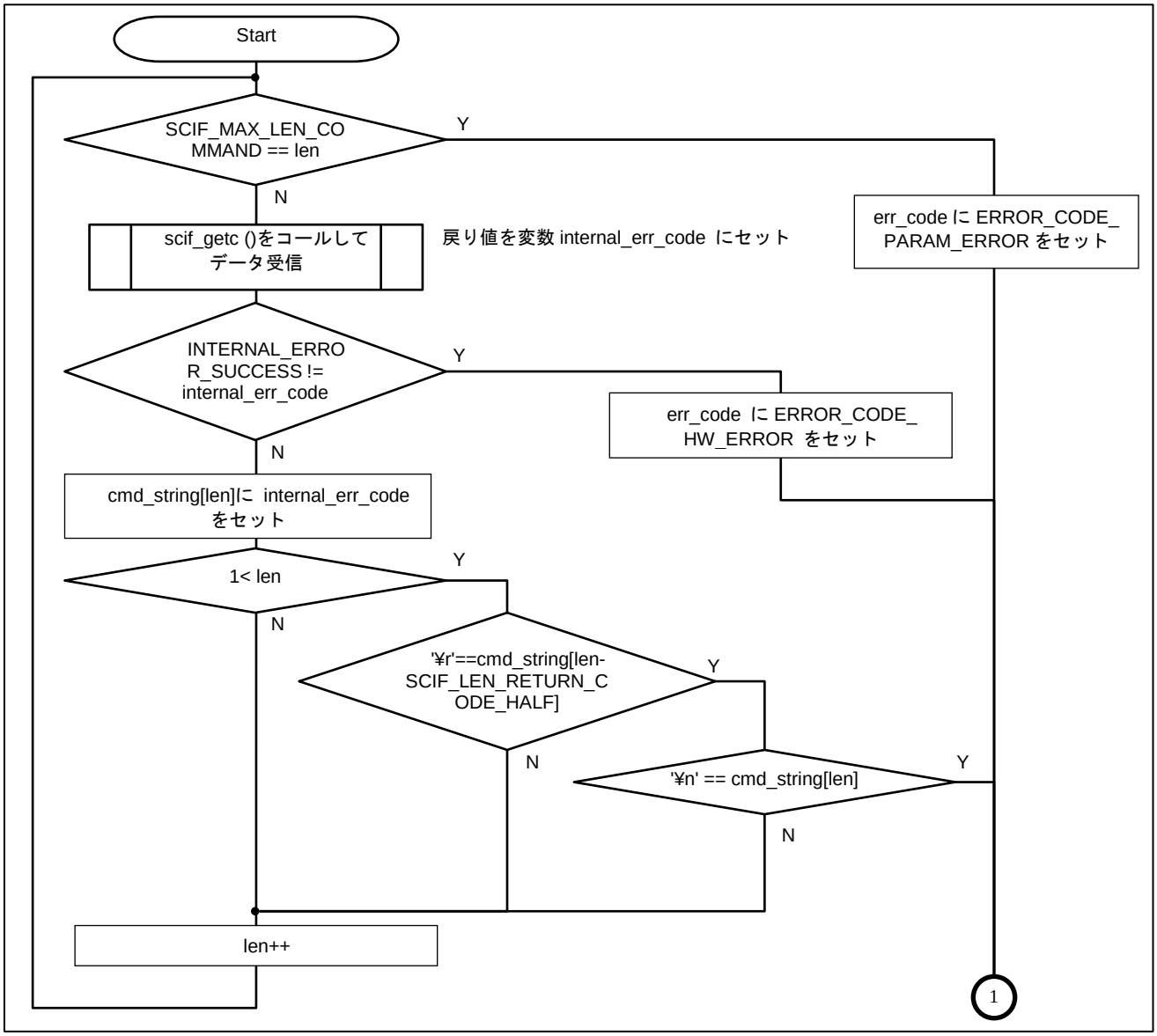


図 5.14　spibsc_verify_data 関数処理

(17) scif_get_cmd

scif_get_cmd	
概　要	コマンドを取得
宣　言	static uint8_t scif_get_cmd (bootloader_ctrl_t *bootloader_param)
説　明	下記にフローチャートを示します。
引　数	bootloader_ctrl_t *bootloader_param　USB シリアル書き込みサンプル管理パラメータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_PARAM_ERROR：受け取ったコマンドのフォーマットエラー ERROR_CODE_NO_CORRESPOND：対応していないコマンドを受け取った ERROR_CODE_HW_ERROR：HW エラー発生 ERROR_CODE_TIMEOUT：タイムアウト発生
補足	



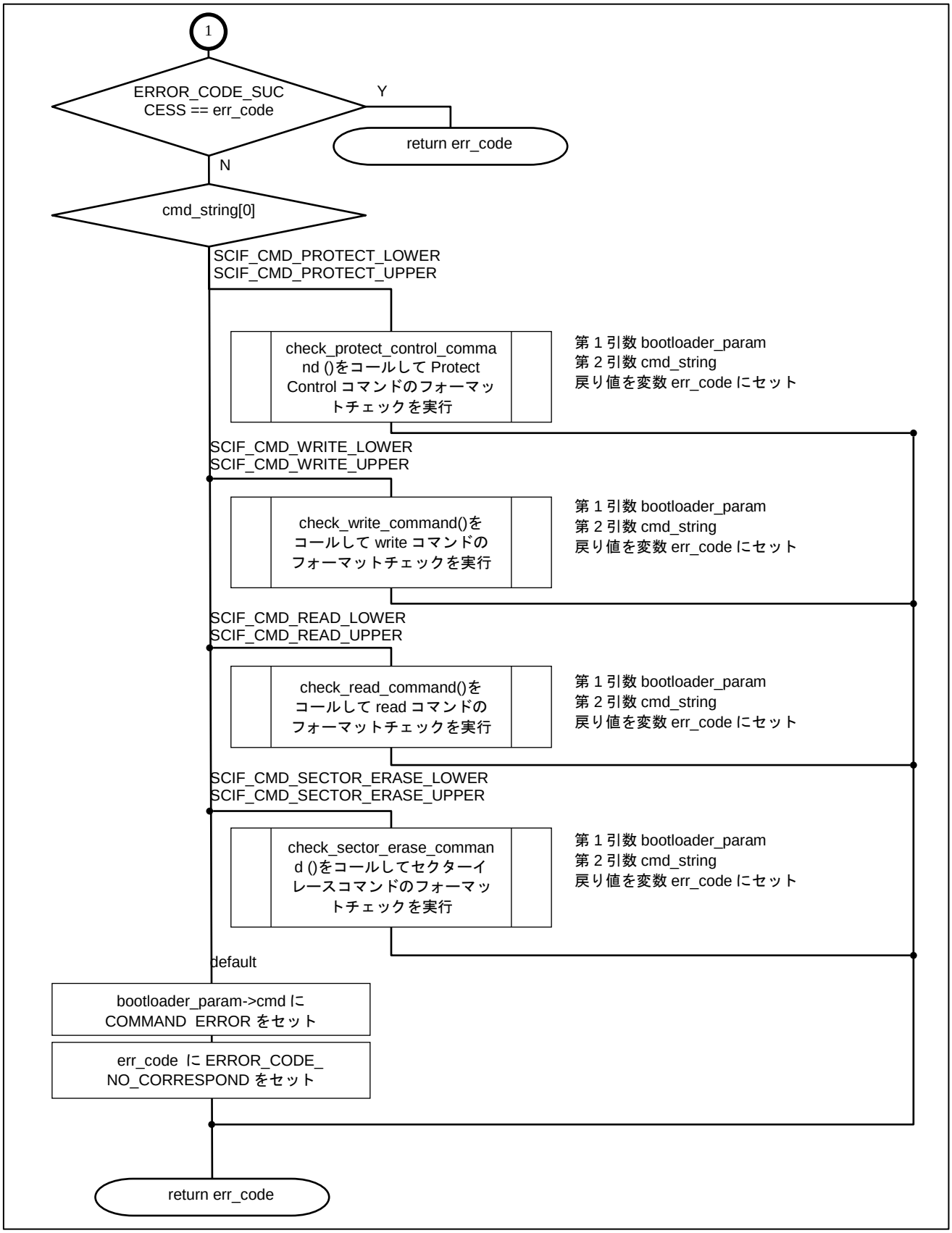
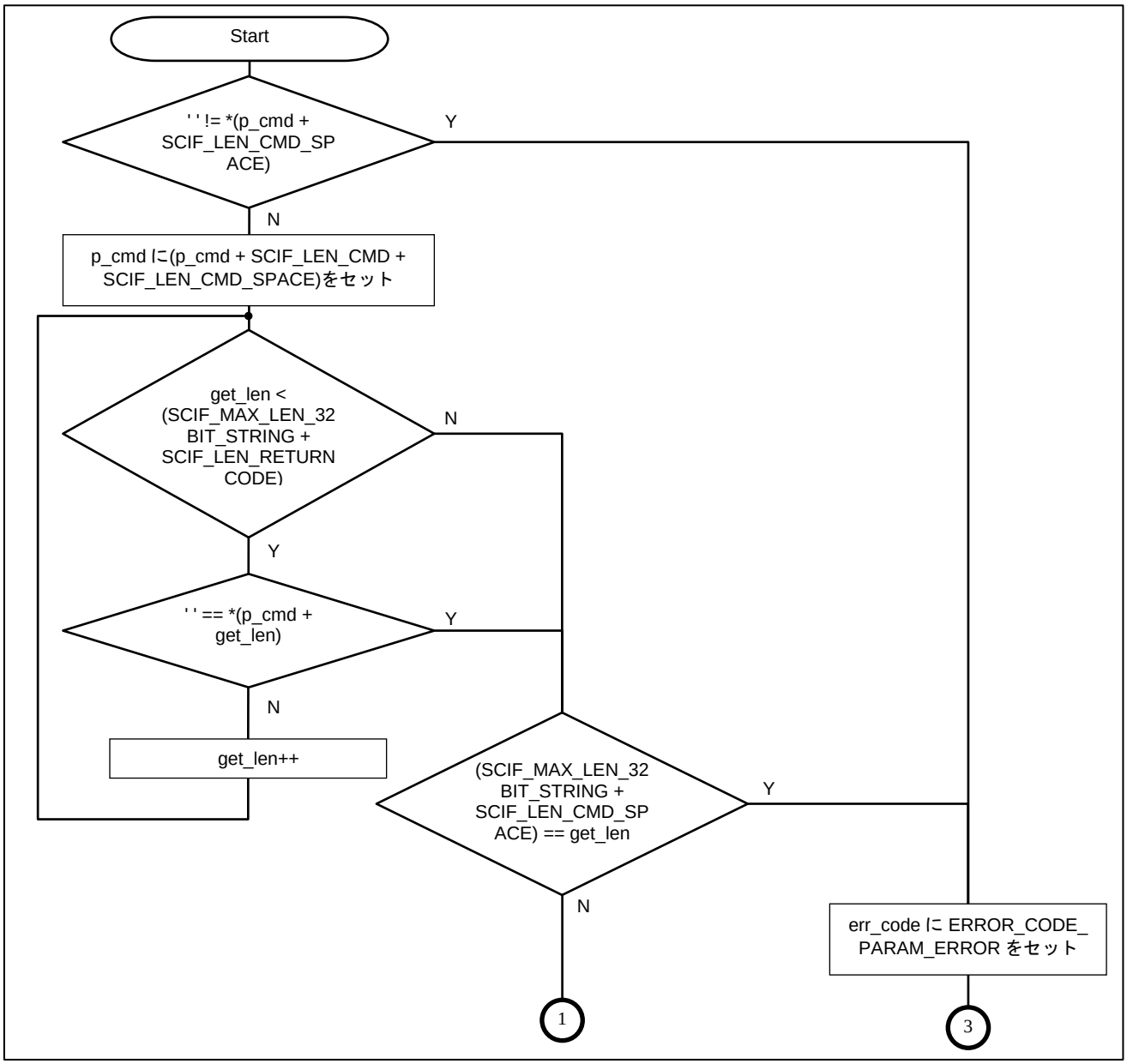
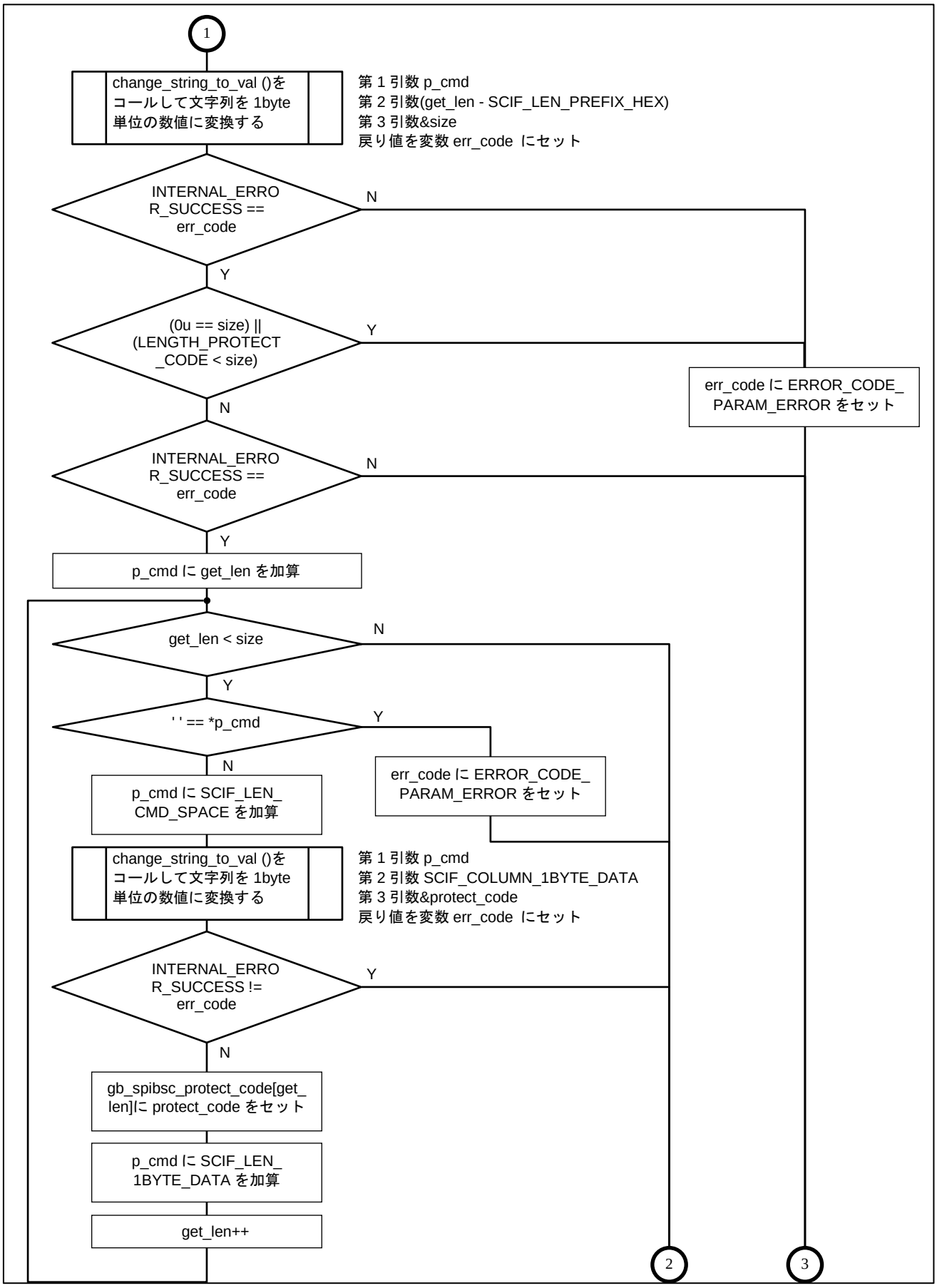


図 5.15　scif_get_cmd 関数処理

(18) check_protect_control_command

check_protect_control_command		
概　要	Protect Control コマンドのフォーマットチェックを実行	
宣　言	static uint8_t check_protect_control_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
説　明	下記にフローチャートを示します。	
引　数	bootloader_ctrl_t *bootloader_param	USB シリアル書き込みサンプル管理パラメータのポインタ
	uint8_t *p_cmd	受信したコマンドデータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_PARAM_ERROR：フォーマットエラー	
補足		




1
change_string_to_val ()をコールして文字列を 1byte 単位の数値に変換する
第 1 引数 p_cmd
第 2 引数(get_len - SCIF_LEN_PREFIX_HEX)
第 3 引数&size
戻り値を変数 err_code にセット
INTERNAL_ERRO R_SUCCESS == err_code
N
Y
(0u == size) || (LENGTH_PROTECT _CODE < size)
Y
N
err_code に ERROR_CODE_ PARAM_ERROR をセット
INTERNAL_ERRO R_SUCCESS == err_code
N
Y
p_cmd に get_len を加算
get_len < size
N
Y
' ' == *p_cmd
Y
N
err_code に ERROR_CODE_ PARAM_ERROR をセット
p_cmd に SCIF_LEN_ CMD_SPACE を加算
change_string_to_val ()をコールして文字列を 1byte 単位の数値に変換する
第 1 引数 p_cmd
第 2 引数 SCIF_COLUMN_1BYTE_DATA
第 3 引数&protect_code
戻り値を変数 err_code にセット
INTERNAL_ERRO R_SUCCESS != err_code
Y
N
gb_spibsc_protect_code[get_ len]に protect_code をセット
p_cmd に SCIF_LEN_ 1BYTE_DATA を加算
get_len++
2
3

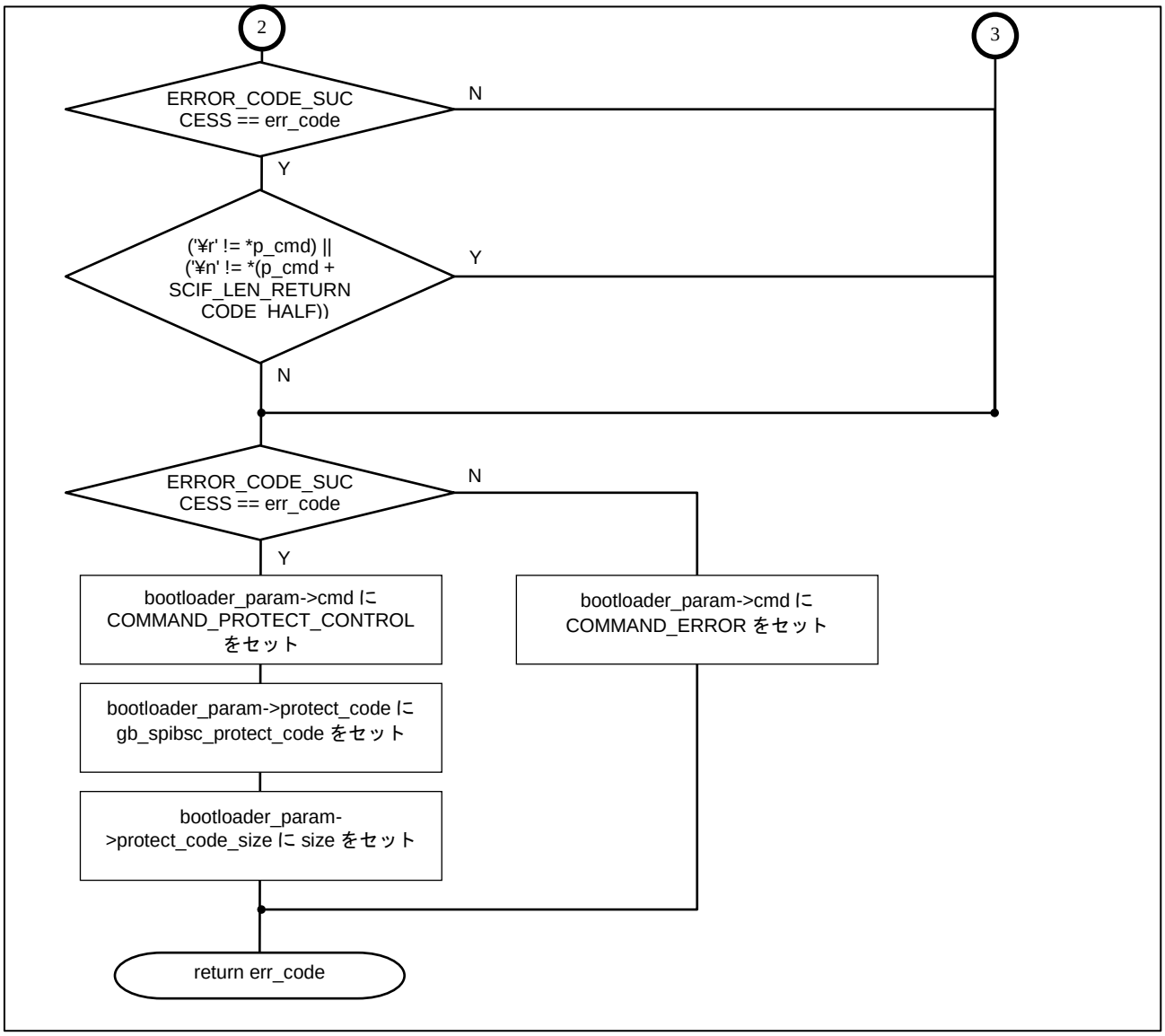
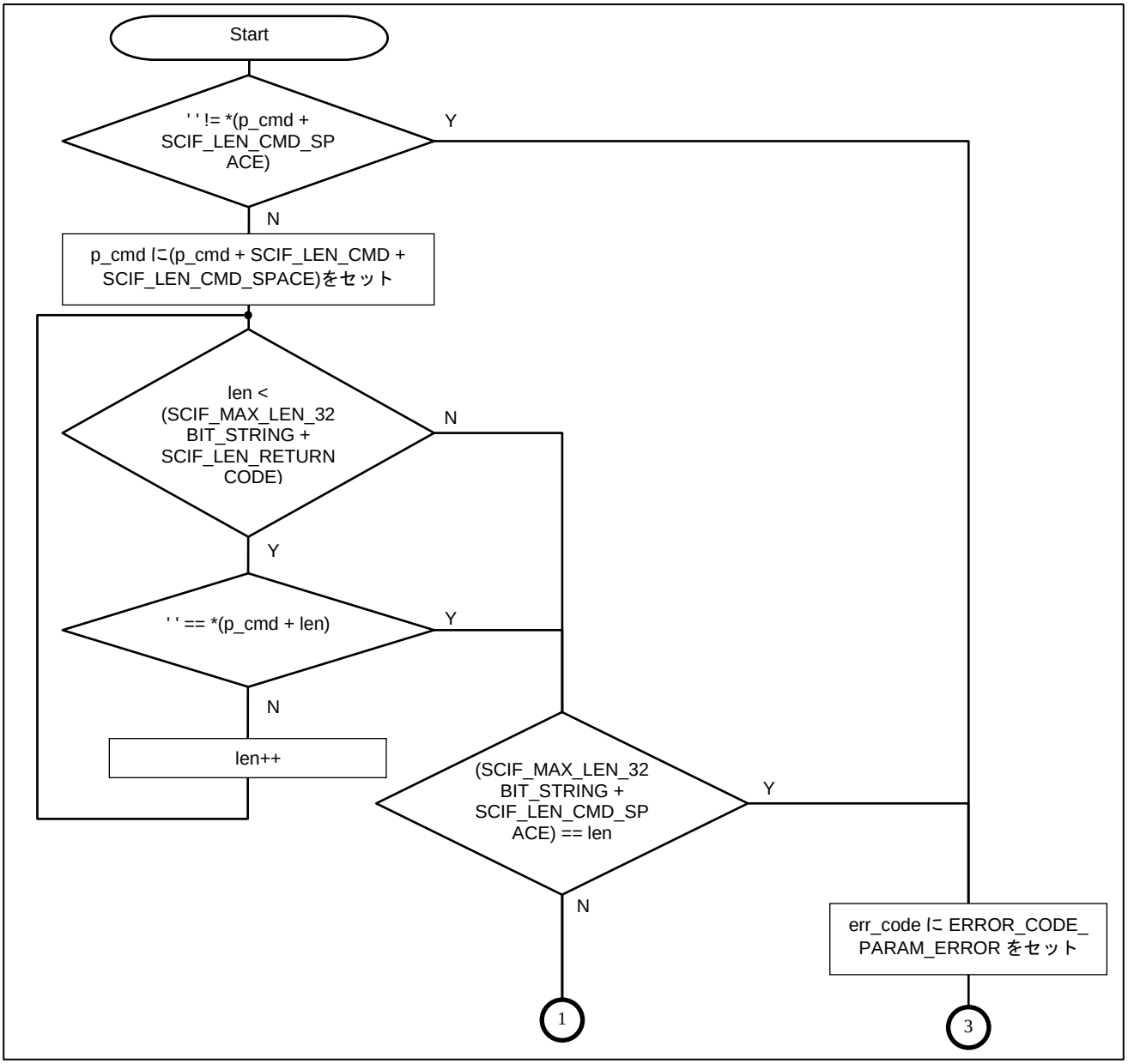
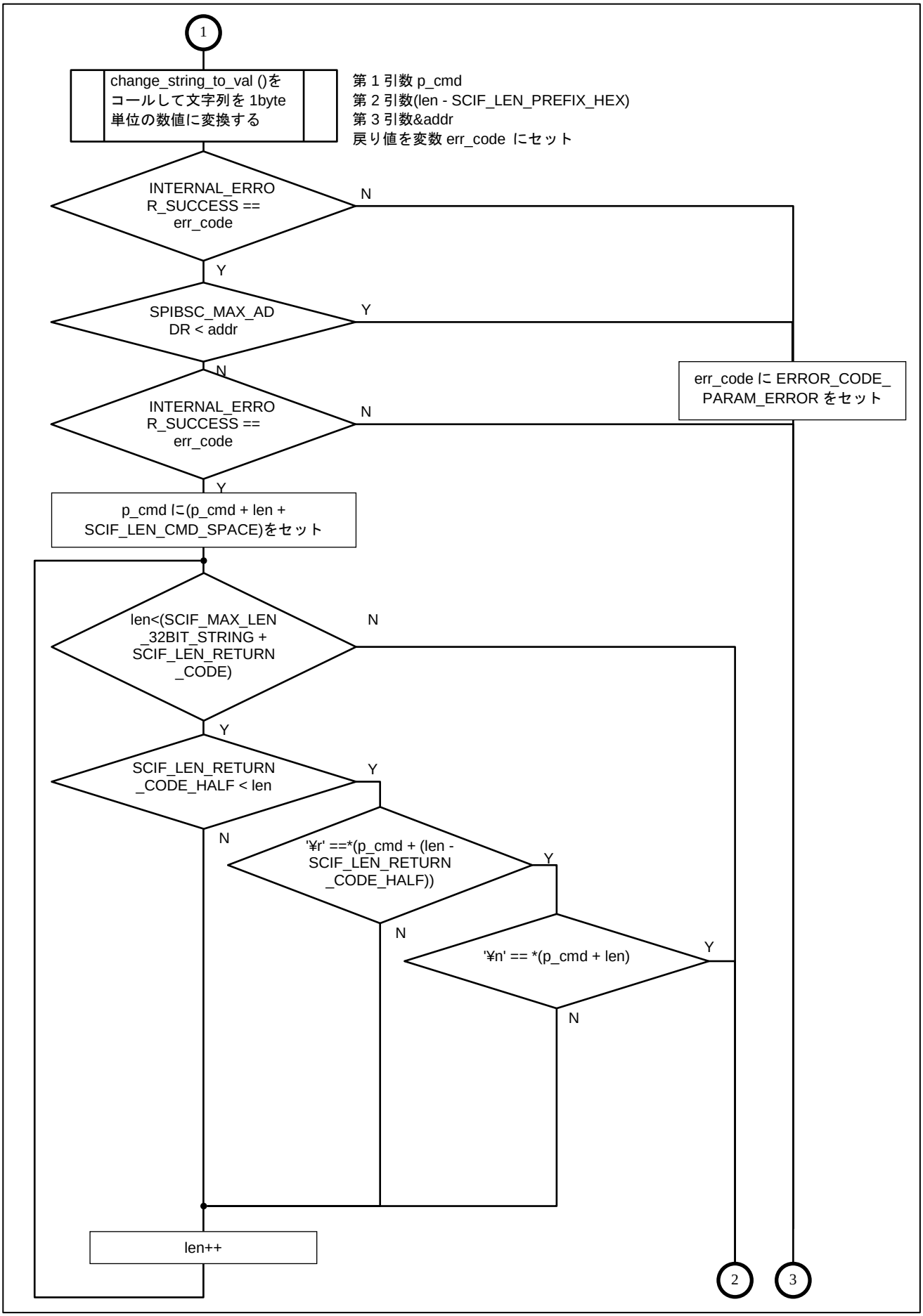


図 5.16 check_protect_control_command 関数処理

(19) check_write_command

check_write_command		
概　要	write コマンドのフォーマットチェックを実行	
宣　言	static uint8_t check_write_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
説　明	下記にフローチャートを示します。	
引　数	bootloader_ctrl_t *bootloader_param	USB シリアル書き込みサンプル管理パラメータのポインタ
	uint8_t *p_cmd	受信したコマンドデータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_PARAM_ERROR：フォーマットエラー	
補足		




1
change_string_to_val ()を
コールして文字列を 1byte
単位の数値に変換する
第 1 引数 p_cmd
第 2 引数(len - SCIF_LEN_PREFIX_HEX)
第 3 引数&addr
戻り値を変数 err_code にセット
INTERNAL_ERRO
R_SUCCESS ==
err_code
N
Y
SPIBSC_MAX_AD
DR < addr
Y
N
INTERNAL_ERRO
R_SUCCESS ==
err_code
N
err_code に ERROR_CODE_
PARAM_ERROR をセット
Y
p_cmd に(p_cmd + len +
SCIF_LEN_CMD_SPACE)をセット
len<(SCIF_MAX_LEN
_32BIT_STRING +
SCIF_LEN_RETURN
_CODE)
N
Y
SCIF_LEN_RETURN
_CODE_HALF < len
Y
N
'¥r' ==*(p_cmd + (len -
SCIF_LEN_RETURN
_CODE_HALF))
Y
N
'¥n' == *(p_cmd + len)
Y
N
len++
2
3

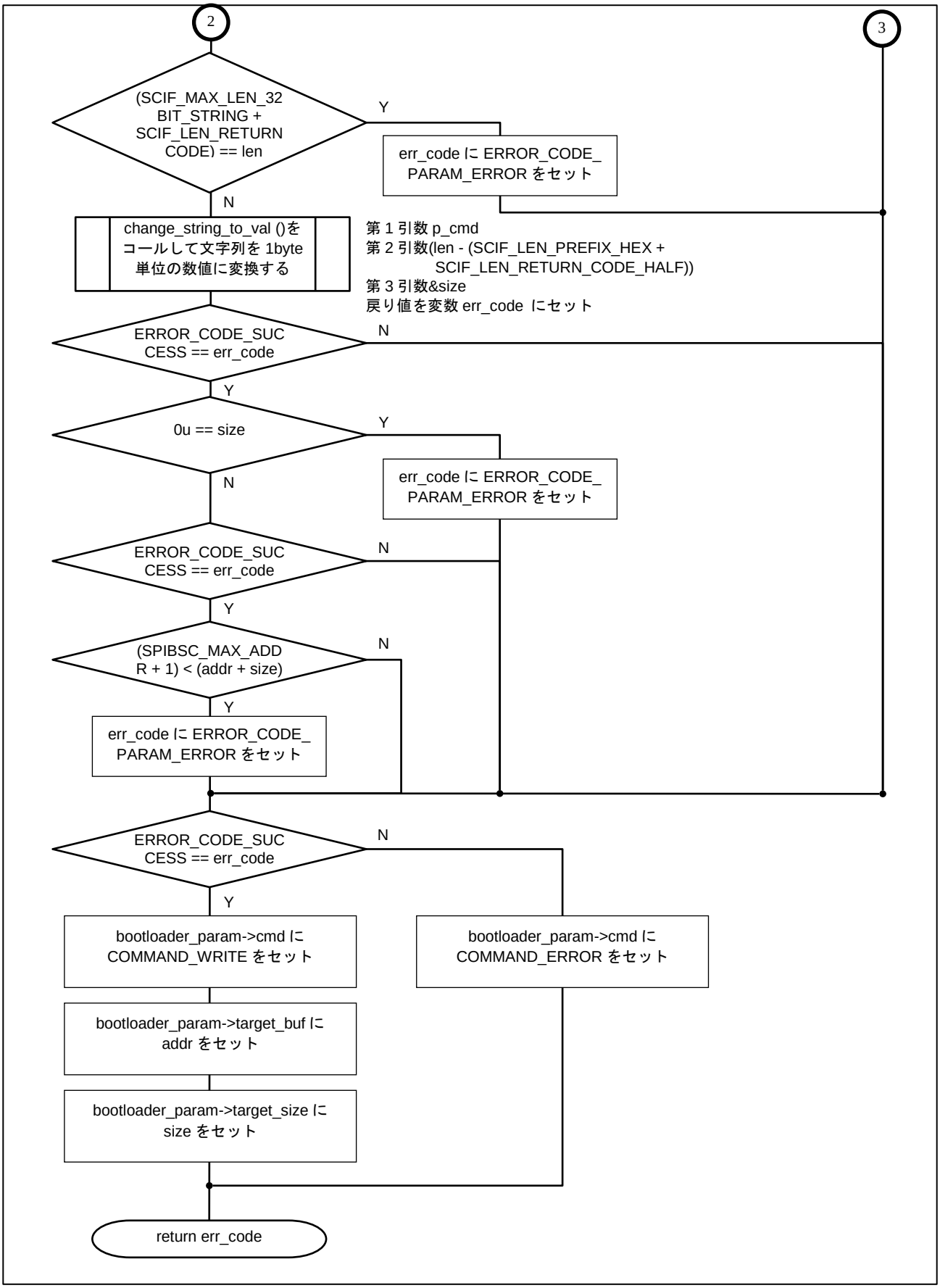
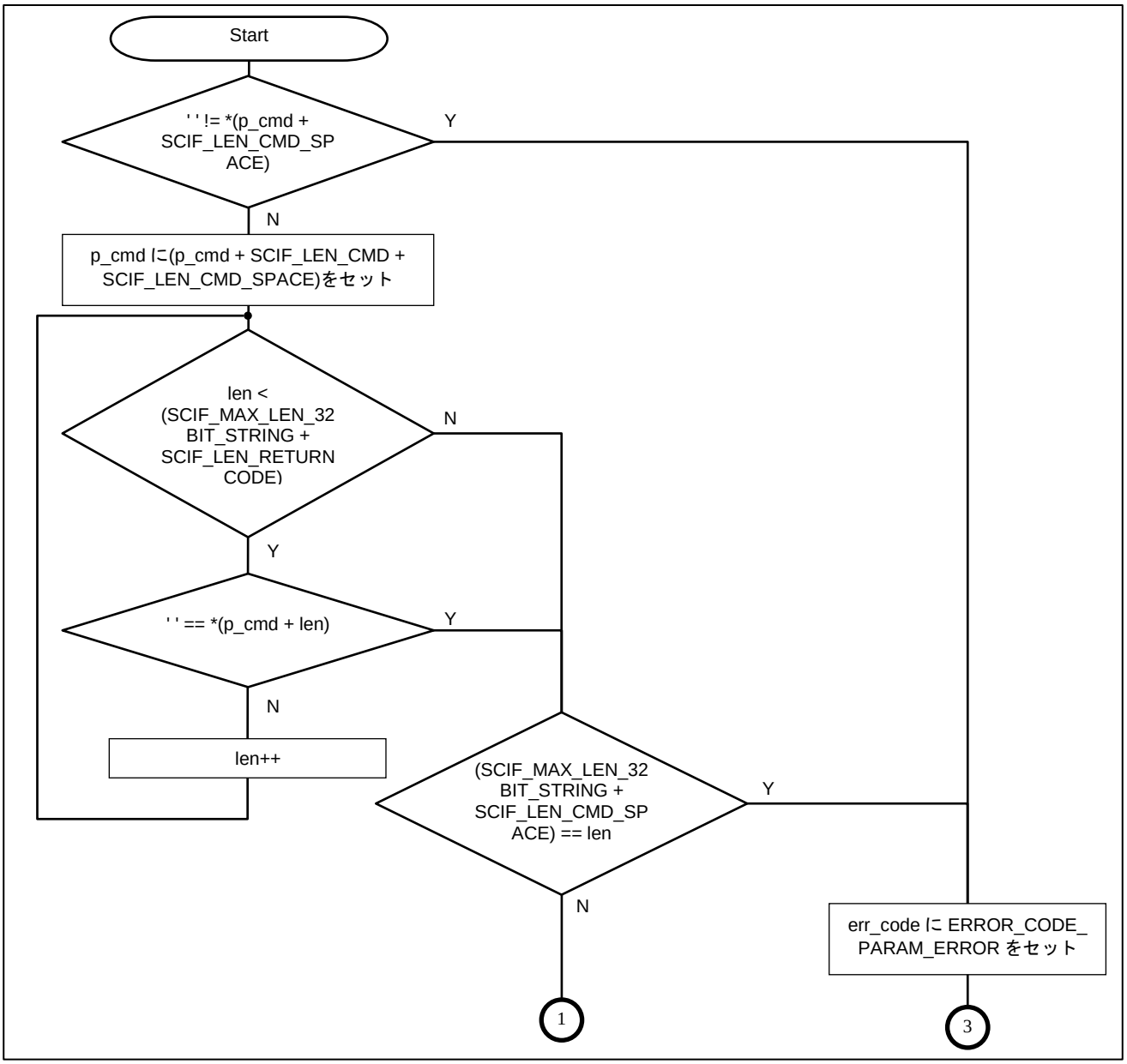
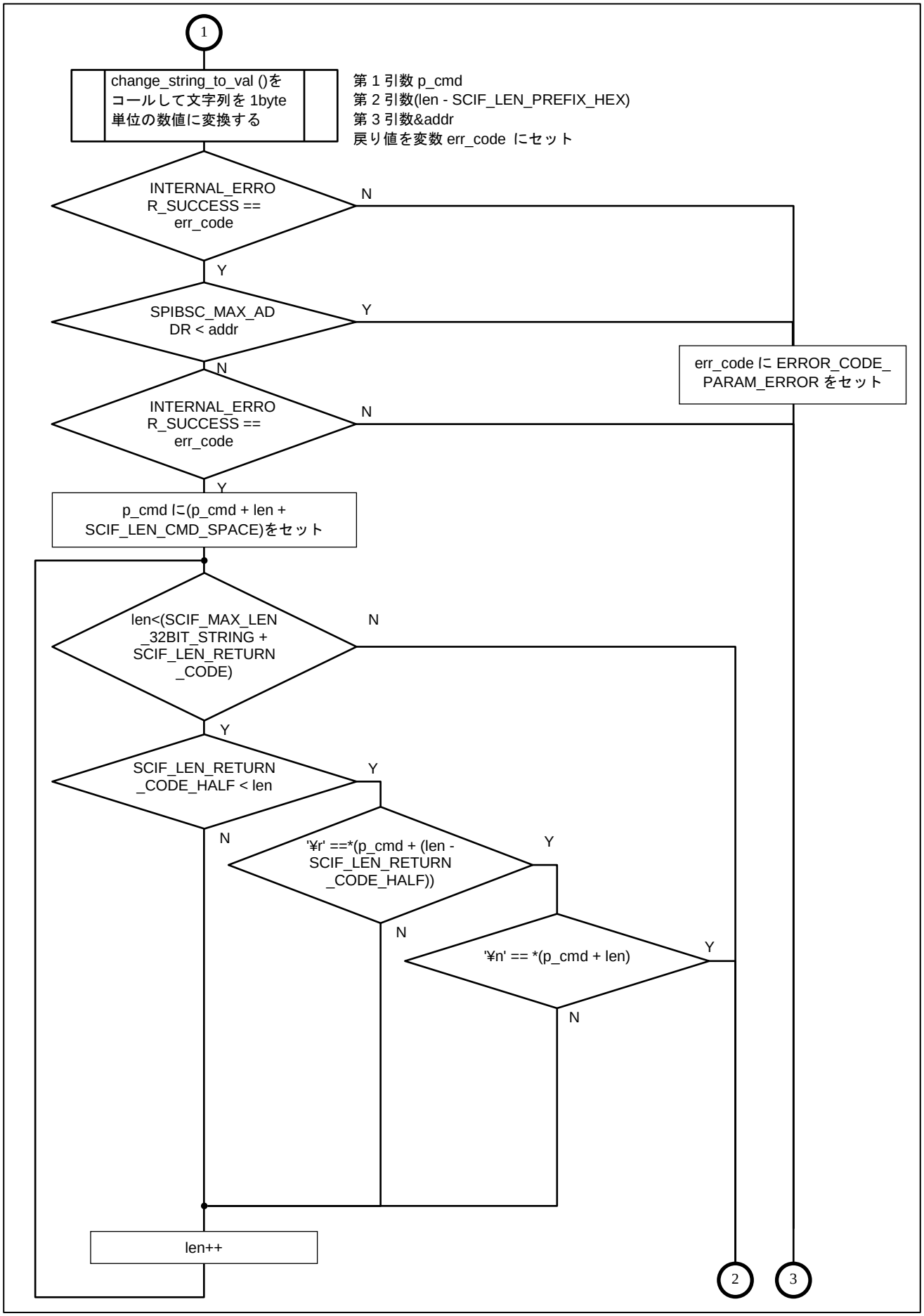


図 5.17 check_write_command 関数処理

(20) check_read_command

check_read_command		
概　要	read コマンドのフォーマットチェックを実行	
宣　言	static uint8_t check_read_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
説　明	下記にフローチャートを示します。	
引　数	bootloader_ctrl_t *bootloader_param	USB シリアル書き込みサンプル管理パラメータのポインタ
	uint8_t *p_cmd	受信したコマンドデータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_PARAM_ERROR：フォーマットエラー	
補足		




1
change_string_to_val ()をコールして文字列を 1byte 単位の数値に変換する
第 1 引数 p_cmd
第 2 引数(len - SCIF_LEN_PREFIX_HEX)
第 3 引数&addr
戻り値を変数 err_code にセット
INTERNAL_ERROR_SUCCESS == err_code
N
Y
SPIBSC_MAX_ADDR < addr
Y
N
err_code に ERROR_CODE_PARAM_ERROR をセット
INTERNAL_ERROR_SUCCESS == err_code
N
Y
p_cmd に(p_cmd + len + SCIF_LEN_CMD_SPACE)をセット
len<(SCIF_MAX_LEN_32BIT_STRING + SCIF_LEN_RETURN_CODE)
N
Y
SCIF_LEN_RETURN_CODE_HALF < len
Y
N
'¥r' ==*(p_cmd + (len - SCIF_LEN_RETURN_CODE_HALF))
Y
N
'¥n' == *(p_cmd + len)
Y
N
len++
2
3

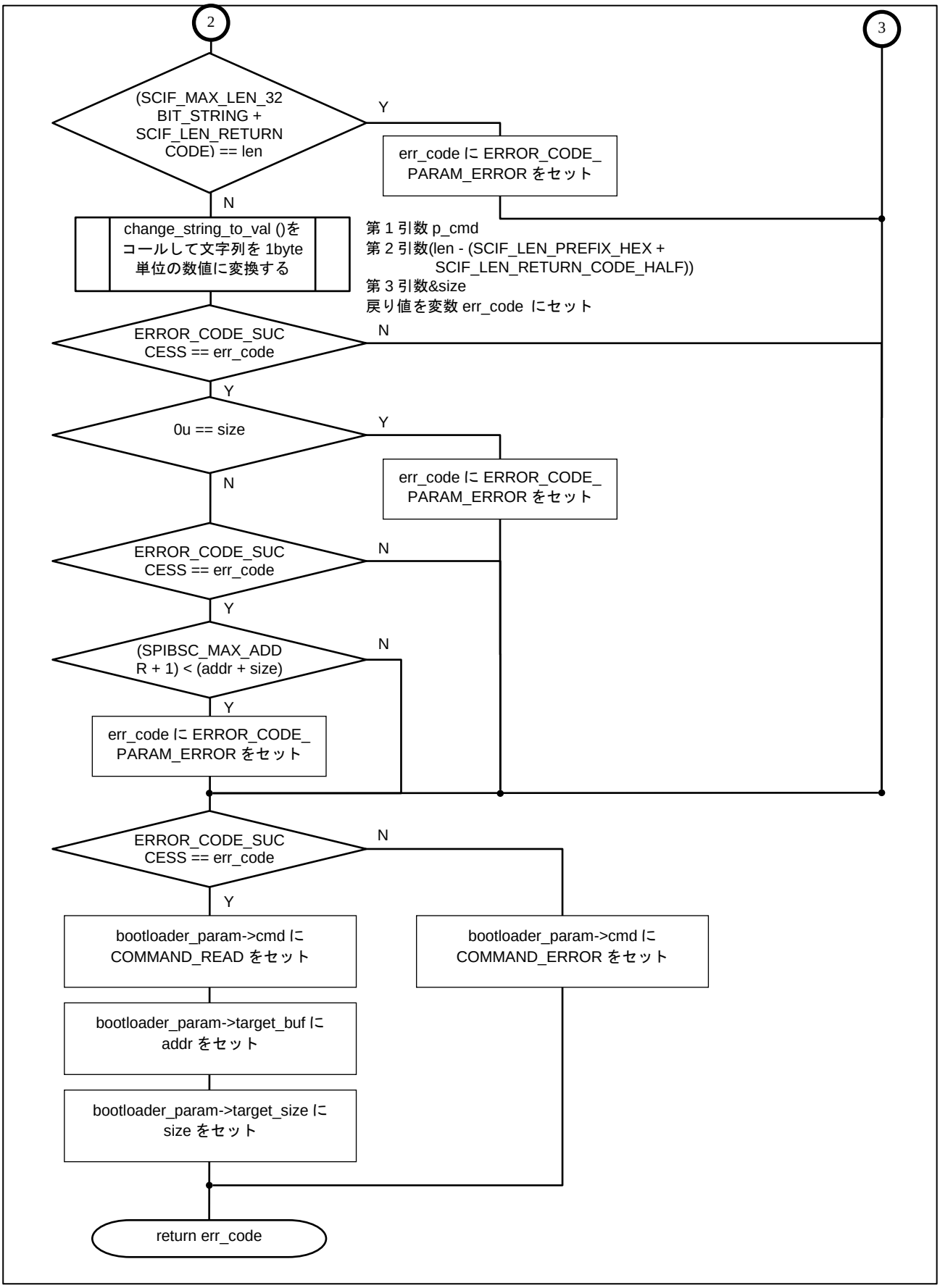
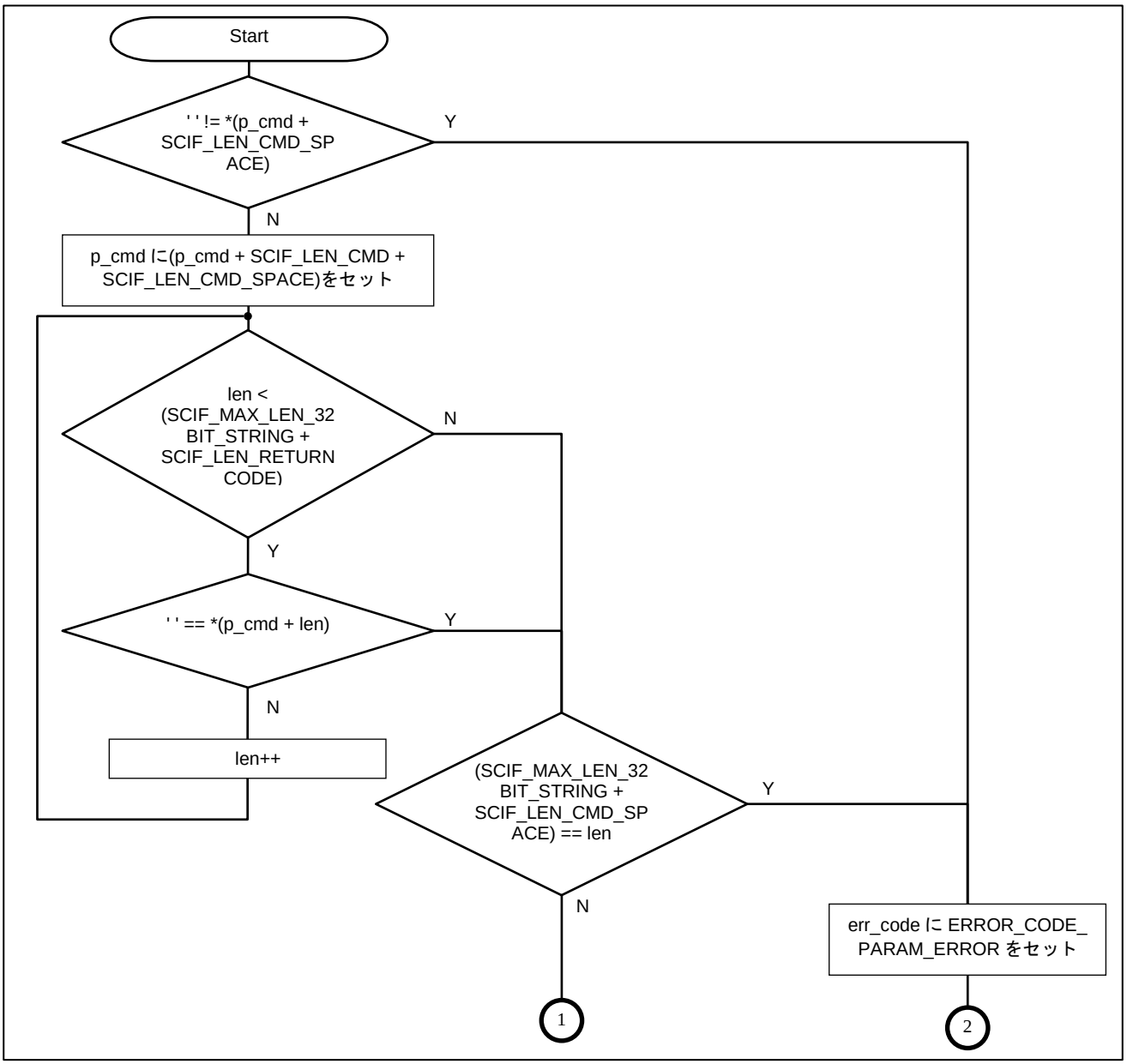


図 5.18 check_read_command 関数処理

(21) check_sector_erase_command

check_sector_erase_command		
概　要	セクターイレースコマンドのフォーマットチェックを実行	
宣　言	static uint8_t check_sector_erase_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
説　明	下記にフローチャートを示します。	
引　数	bootloader_ctrl_t *bootloader_param	USB シリアル書き込みサンプル管理パラメータのポインタ
	uint8_t *p_cmd	受信したコマンドデータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_PARAM_ERROR：フォーマットエラー	
補足		



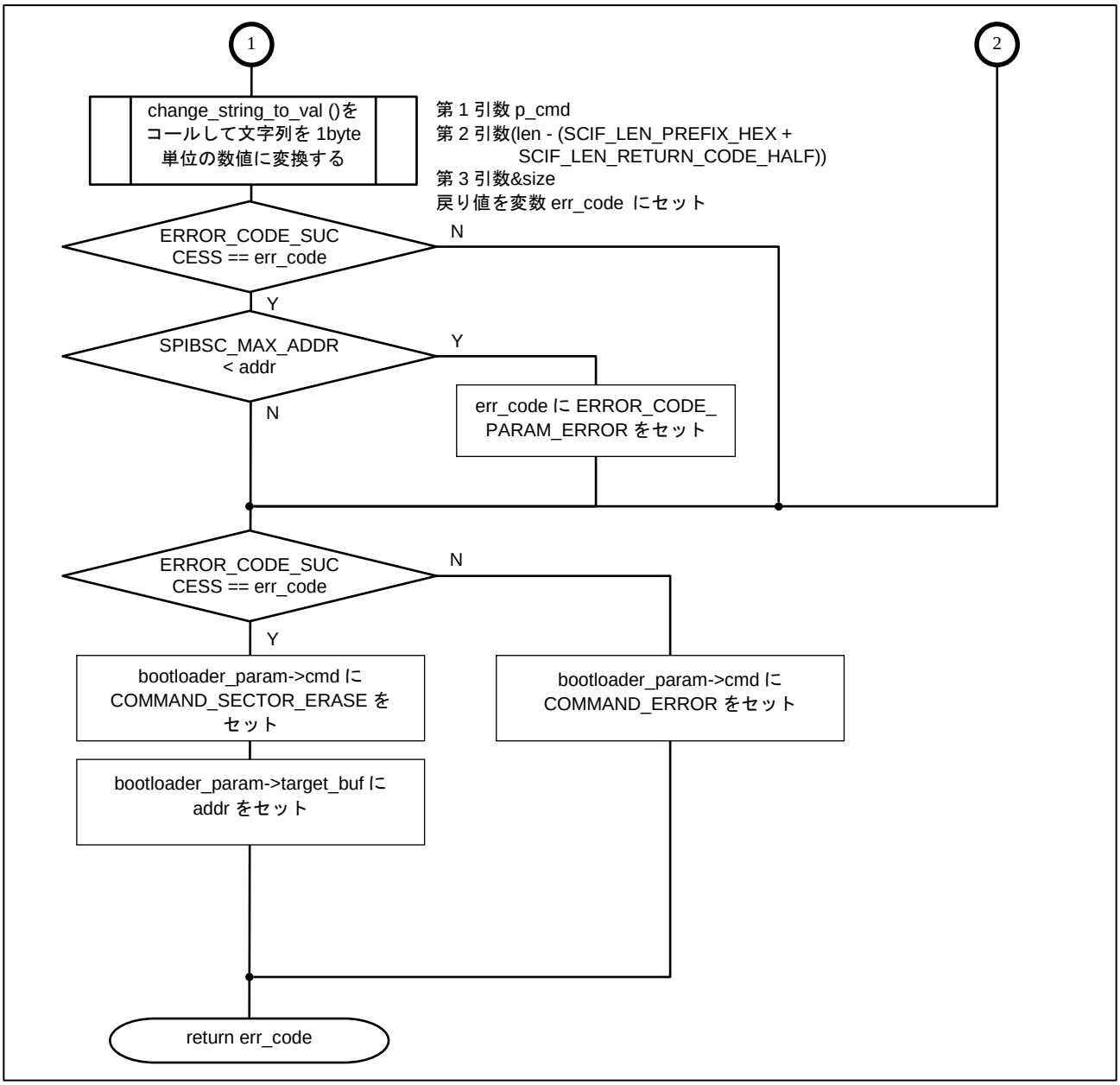


図 5.19 check_sector_erase_command 関数処理

(22) change_string_to_val

change_string_to_val		
概　要	文字列を 1byte 単位の数値に変換します。	
宣　言	static uint8_t change_string_to_val(uint8_t *p_inbuf, uint32_t column, uint32_t *p_outbuf)	
説　明	下記にフローチャートを示します。	
引　数	uint8_t *p_inbuf	変換元文字列データバッファのポインタ
	uint32_t column	変換文字列桁数
	uint32_t *p_outbuf	変換後の数値を格納するデータバッファのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_PARAM_ERROR：フォーマットエラー	

補足

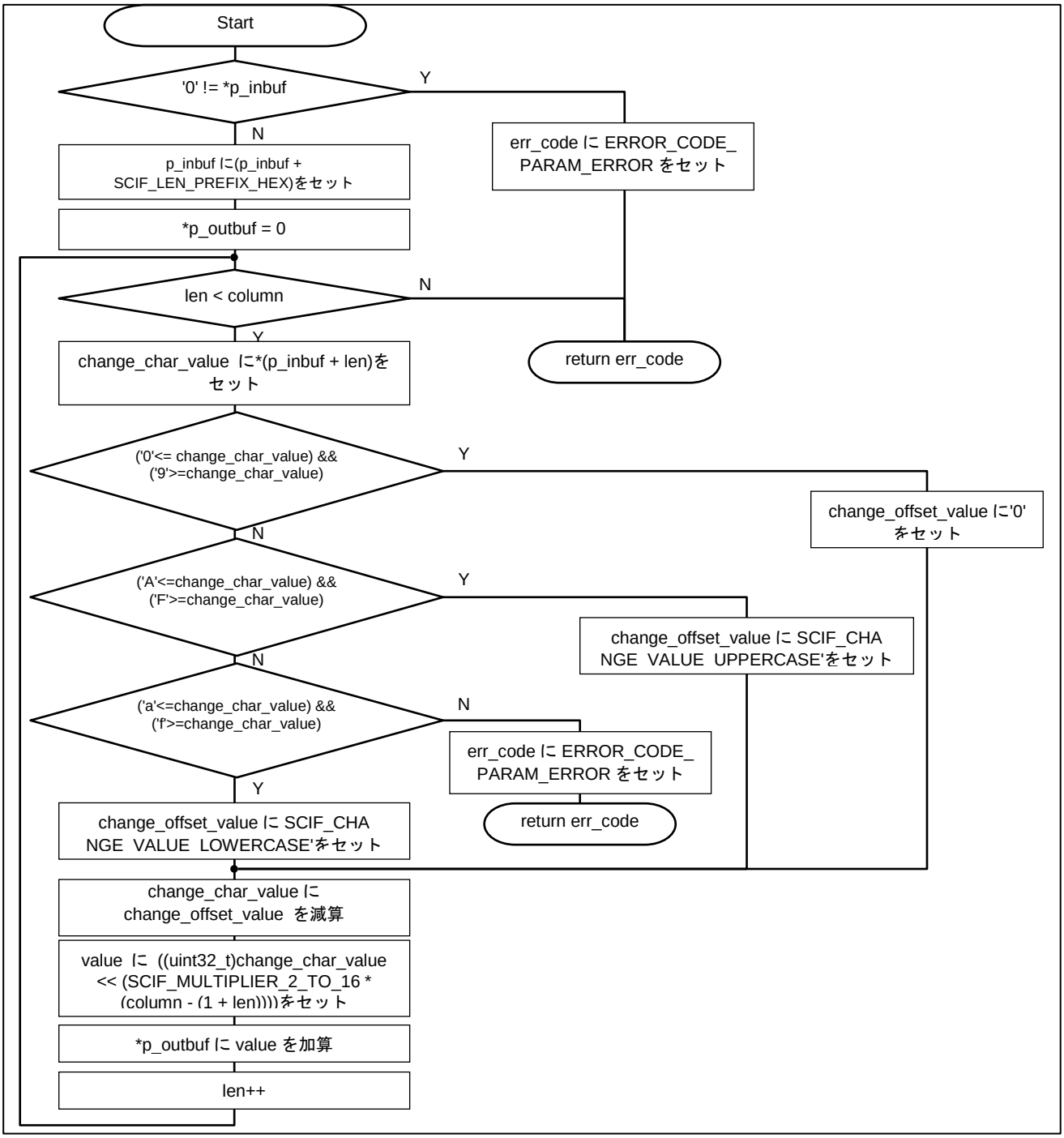


図 5.20 change_string_to_val 関数処理

(23) change_errcode_to_string

change_errcode_to_string		
概　要	数値を 16 進数表記の文字列に変換します。	
宣　言	static void change_errcode_to_string(uint32_t value, uint8_t *p_buf)	
説　明	下記にフローチャートを示します。	
引　数	uint32_t value	変換元データ
	uint8_t *p_buf	変換データの格納先バッファのポインタ
リターン値	None	
補足		

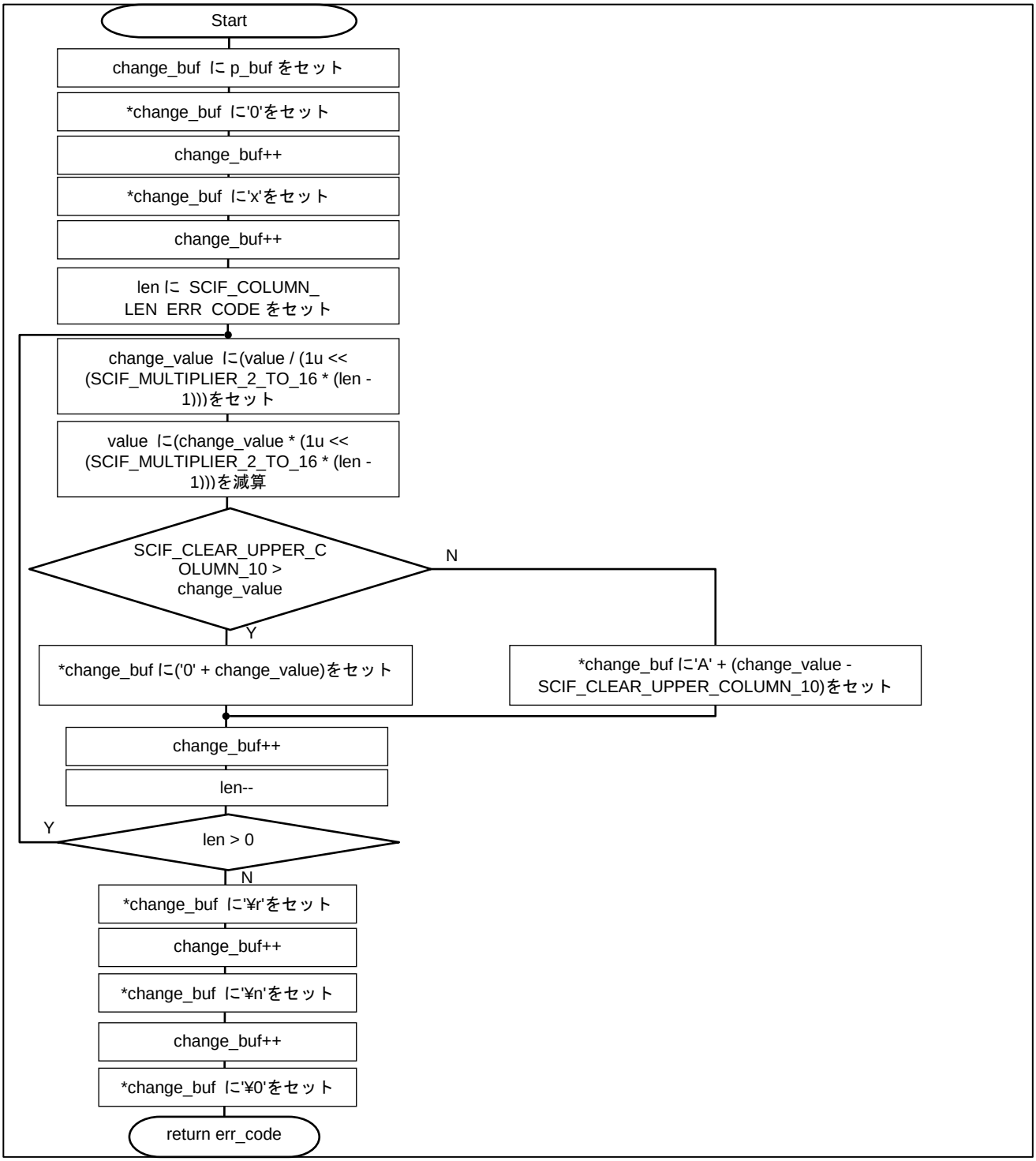


図 5.21 change_errcode_to_string 関数処理

(24) spibsc_init

spibsc_init	
概　要	SPIBSC の初期化
宣　言	static void spibsc_init(void)
説　明	本関数内で、Macronix 社製シリアル FlashROM（型名：MX25L51245G）に最適な SPIBSC 設定を行います。
引　数	None
リターン値	None
補足	本関数の詳細は、「RZ/T1 グループ シリアルフラッシュサンプルプログラム」を参照してください。

(25) r_cdc_read_complete

r_cdc_read_complete		
概　要	USB 受信完了コールバック関数	
宣　言	static void r_cdc_read_complete(USB_UTR_t *mess)	
説　明	USB 受信バッファに受信データを格納します	
引　数	USB_UTR_t *mess	USB 通信構造体のポインタ
リターン値	None	
補足		

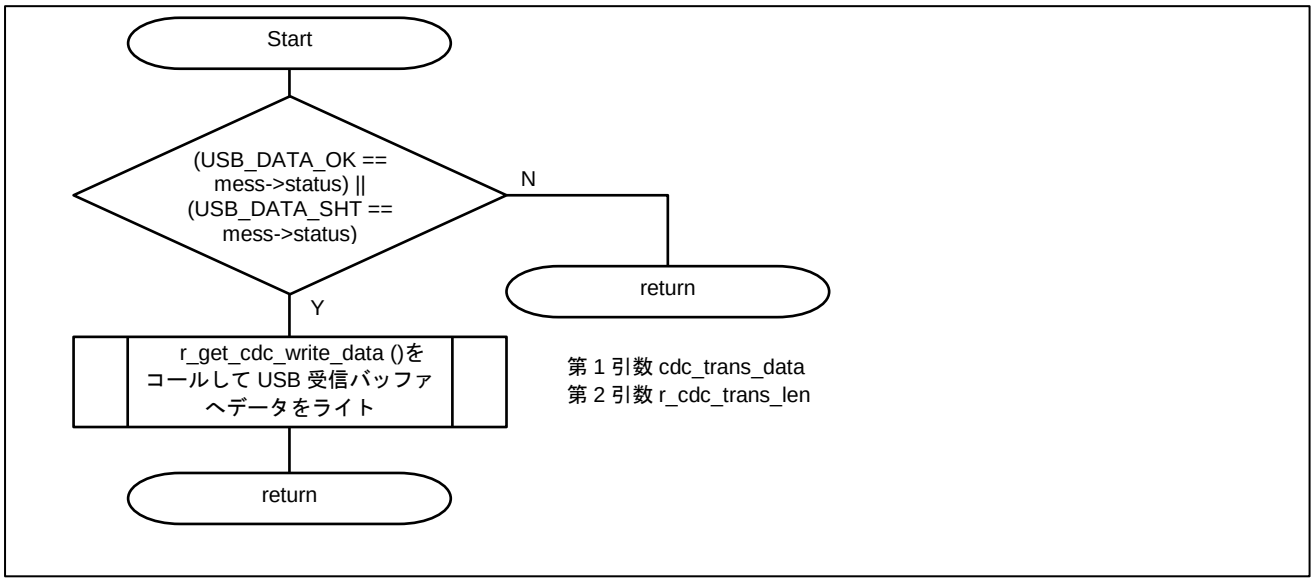


図 5.22 r_cdc_read_complete 関数処理

(26) r_cdc_write_complete

r_cdc_write_complete		
概　要	USB 送信完了コールバック関数	
宣　言	static void r_cdc_write_complete(USB_UTR_t *mess)	
説　明	usbf_wfin_flag を true にします	
引　数	USB_UTR_t *mess	USB 通信構造体のポインタ
リターン値	None	
補足	USB 通信構造体に関しては、「RZ/T1 グループ USB Peripheral Communications Device Class Driver」を参照してください。	

(27) r_cdc_rev_data_is_valid

r_cdc_rev_data_is_valid	
概　要	USB 受信バッファにデータの有無を確認
宣　言	static uint8_t r_cdc_rev_data_is_valid(void)
説　明	USB 受信バッファにデータがない場合は、USB_CDC_DATA_INVALID を返します。データがある場合は、USB_CDC_DATA_VALID を返します。
引　数	None
リターン値	USB_CDC_DATA_INVALID(0u)：USB 受信バッファにデータなし USB_CDC_DATA_VALID(1u)：USB 受信バッファにデータあり
補足	

(28) r_cdc_rev_data_clear

r_cdc_rev_data_clear	
概　要	USB 受信バッファをクリア
宣　言	static void r_cdc_rev_data_clear(void)
説　明	cdc_rev_data_pr と cdc_rev_data_pw を 0 クリアします。
引　数	None
リターン値	None
補足	

(29) r_get_cdc_write_data

r_get_cdc_write_data	
概　要	USB 受信バッファへデータをライト
宣　言	static uint8_t r_get_cdc_write_data(uint8_t* pbuf, uint32_t sz)
説　明	sz で指定したサイズのデータを USB 受信バッファに格納します。
引　数	uint8_t* pbuf　　バッファのポインタ uint32_t sz　　ライトサイズ
リターン値	INTERNAL_ERROR_SUCCESS：正常終了 INTERNAL_ERROR_SCIF_ERROR：ライトサイズが 0
補足	

(30) r_get_cdc_rev_data

r_get_cdc_rev_data	
概　要	USB 受信バッファのデータをリード
宣　言	static uint8_t r_get_cdc_rev_data(uint8_t* pbuf, uint32_t sz)
説　明	リードバッファへリードサイズ分の USB 受信データを格納します。
引　数	uint8_t* pbuf　　リードバッファのポインタ uint32_t sz　　リードサイズ
リターン値	INTERNAL_ERROR_SUCCESS：正常終了 INTERNAL_ERROR_SCIF_ERROR：USB 受信バッファに格納されているデータサイズが 0 または、リードサイズ分のデータがない
補足	

(31) r_get_cdc_rev_1Bdata

r_get_cdc_rev_1Bdata	
概　要	USB 受信バッファのデータを 1 バイトリード
宣　言	static uint8_t r_get_cdc_rev_1Bdata(void)
説　明	下記にフローチャートを示します。
引　数	None
リターン値	USB 受信バッファに格納されているデータ
補足	

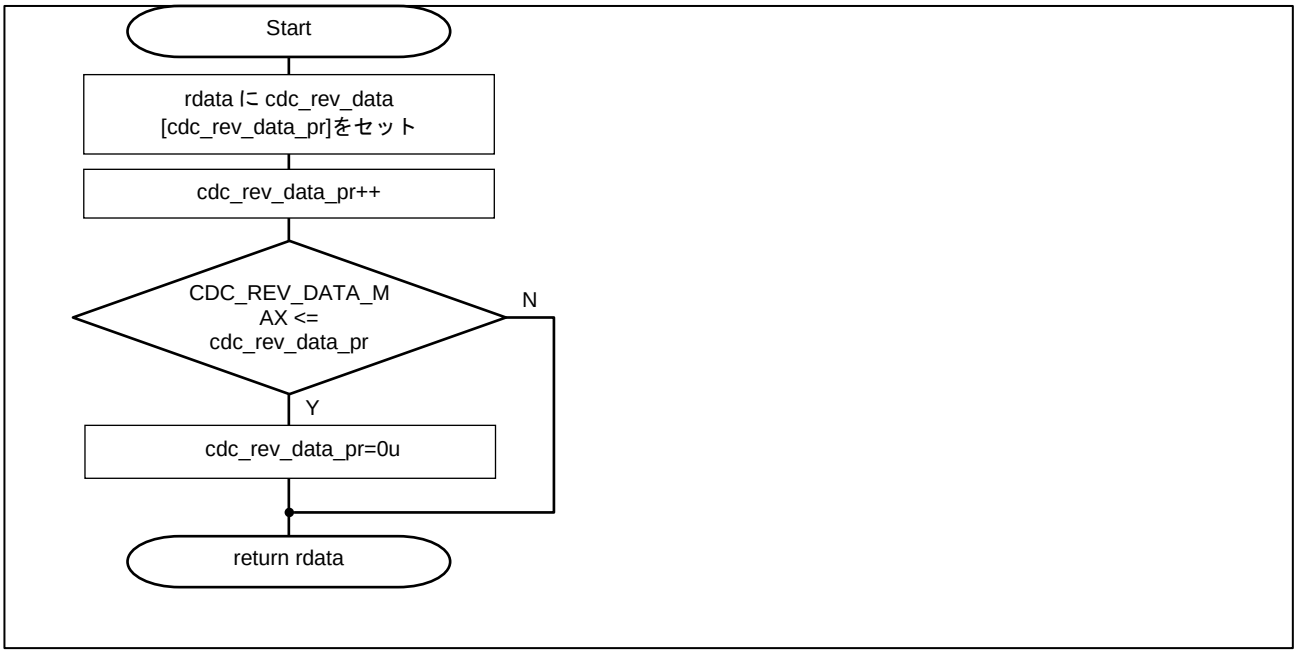


図 5.23 r_get_cdc_rev_1Bdata 関数処理

(32) r_cdc_start

r_cdc_start	
概　要	USB CDC スタート
宣　言	static void r_cdc_start(void)
説　明	下記にフローチャートを示します。
引　数	None
リターン値	None
補足	

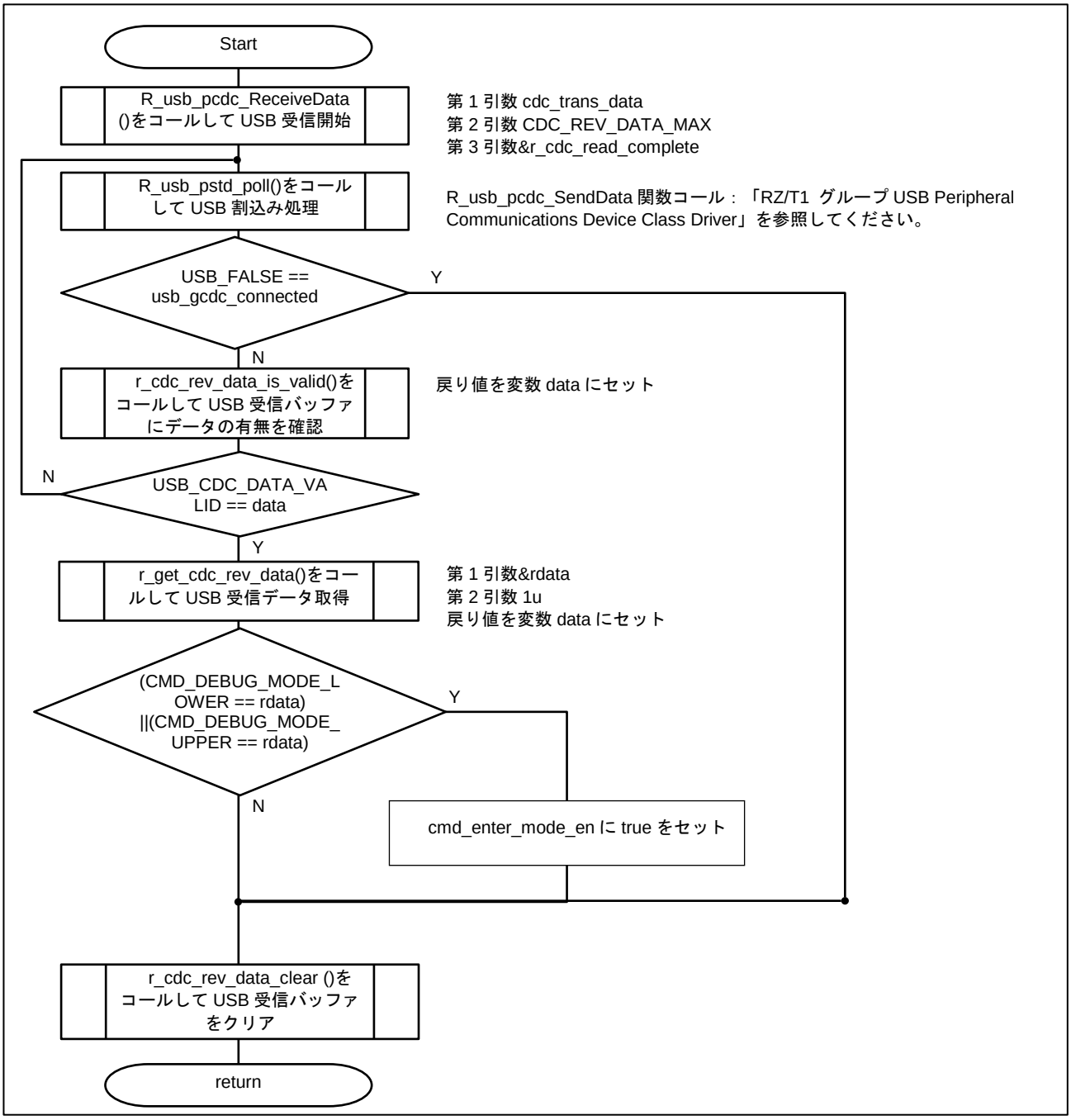


図 5.24 r_cdc_start 関数処理

(33) r_usb_write_data_to_sFlash

r_usb_write_data_to_sFlash	
概　要	データ受信とシリアル FlashROM への書き込みを実施
宣　言	static uint8_t r_usb_write_data_to_sFlash(bootloader_ctrl_t *bootloader_param)
説　明	下記にフローチャートを示します。
引　数	bootloader_ctrl_t *bootloader_param　USB シリアル書き込みサンプル管理パラメータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_TIMEOUT：USB 受信タイムアウトエラー(10s) ERROR_CODE_VERIFY：ベリファイエラー ERROR_CODE_HW_ERROR：HW エラー発生 ERROR_CODE_FILE_TRANSFER：シリアル FlashROM へのデータ転送失敗
補足	

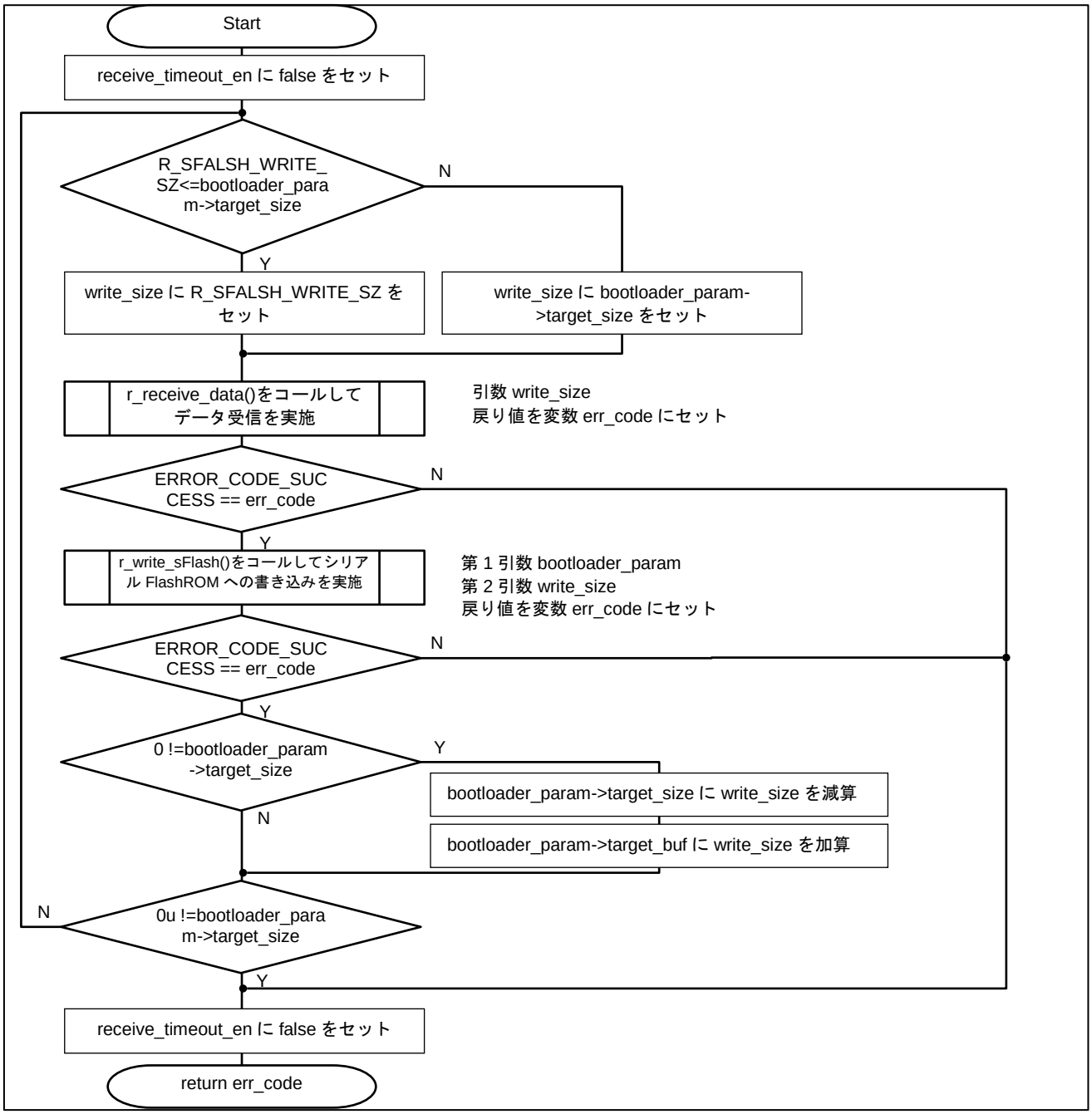


図 5.25　r_usb_write_data_to_sFlash 関数処理

(34)　r_usb_read_data_from_sFlash

r_usb_read_data_from_sFlash	
概　要	シリアル FlashROM の読み出しとデータ送信を実施
宣　言	static uint8_t r_usb_read_data_from_sFlash(bootloader_ctrl_t *bootloader_param)
説　明	下記にフローチャートを示します。
引　数	bootloader_ctrl_t *bootloader_param　USB シリアル書き込みサンプル管理パラメータのポインタ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_HW_ERROR：HW エラー発生 ERROR_CODE_FILE_TRANSFER：シリアル FlashROM へのデータ転送失敗
補足	

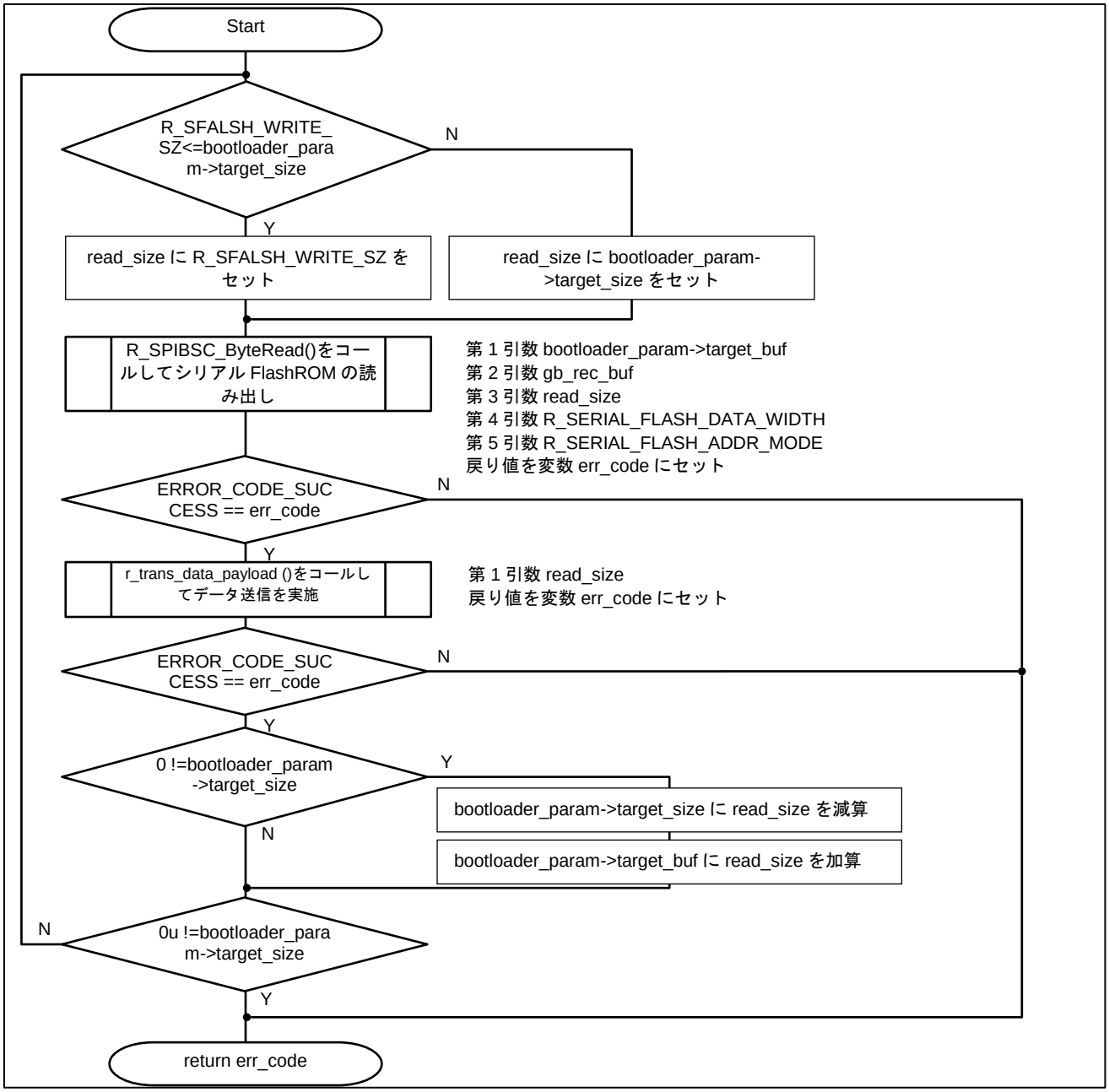


図 5.26　r_usb_read_data_from_sFlash 関数処理

(35)　r_receive_data

r_receive_data	
概　要	データ受信を実施
宣　言	static uint8_t r_receive_data(uint32_t rec_size)
説　明	下記にフローチャートを示します。
引　数	uint32_t rec_size　　データ受信サイズ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_TIMEOUT：USB 受信タイムアウトエラー(10s) ERROR_CODE_HW_ERROR：HW エラー発生
補足	

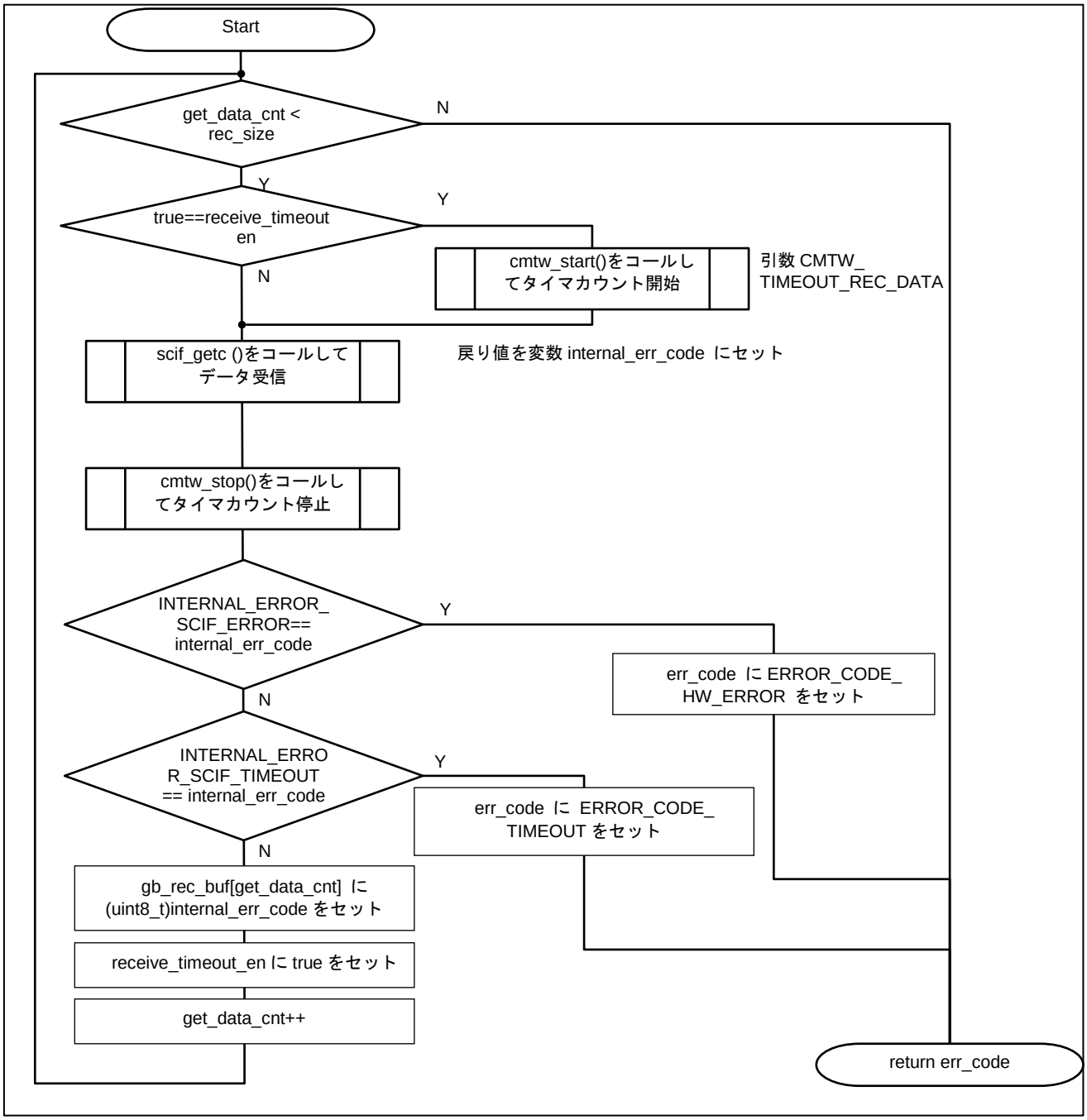


図 5.27　r_receive_data 関数処理

(36)　r_write_sFlash

r_write_sFlash		
概　要	シリアル FlashROM への書き込みを実施	
宣　言	static uint8_t r_write_sFlash(bootloader_ctrl_t *bootloader_param, uint32_t write_size)	
説　明	下記にフローチャートを示します。	
引　数	bootloader_ctrl_t *bootloader_param	USB シリアル書き込みサンプル管理パラメータのポインタ
	uint32_t write_size	書き込みサイズ
リターン値	ERROR_CODE_SUCCESS：正常終了	
	ERROR_CODE_VERIFY：ベリファイエラー	
	ERROR_CODE_FILE_TRANSFER：シリアル FlashROM へのデータ転送失敗	
補足		

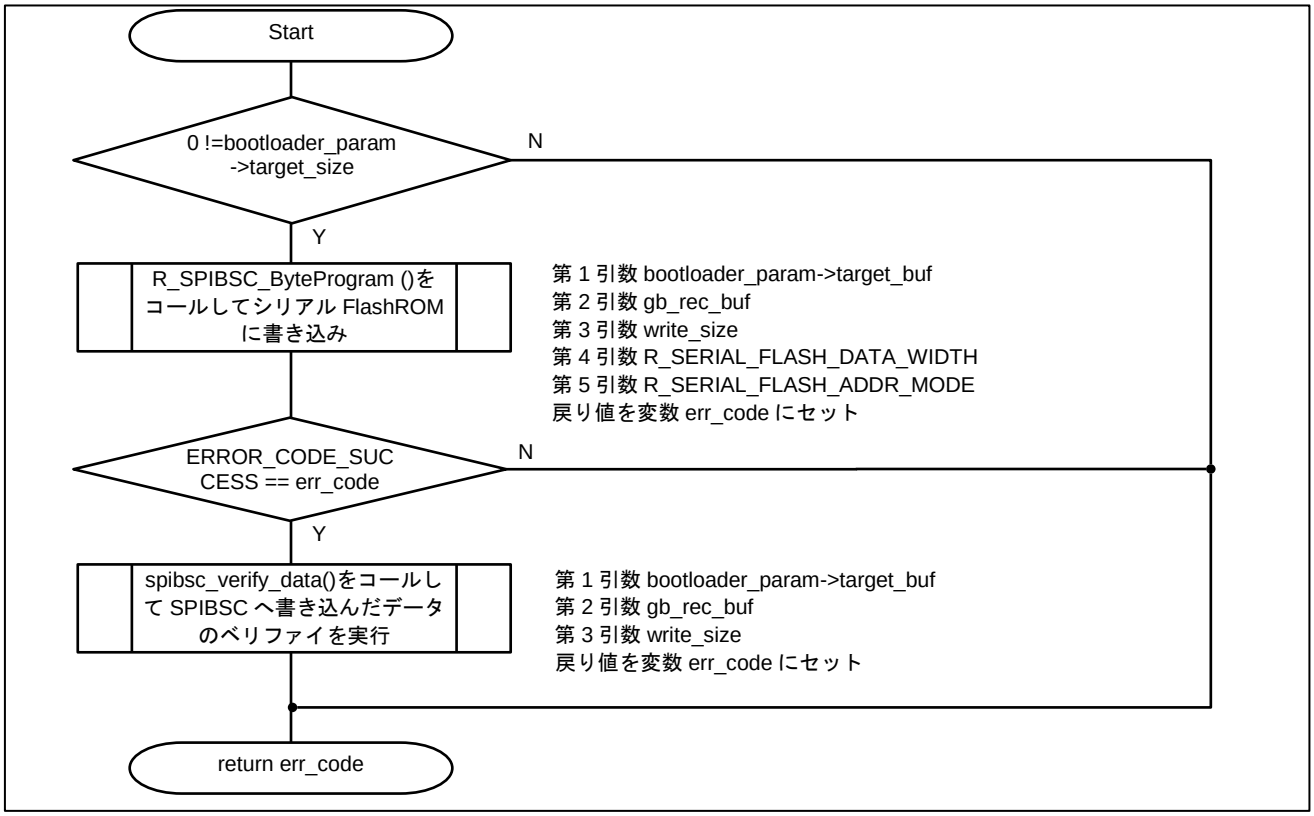
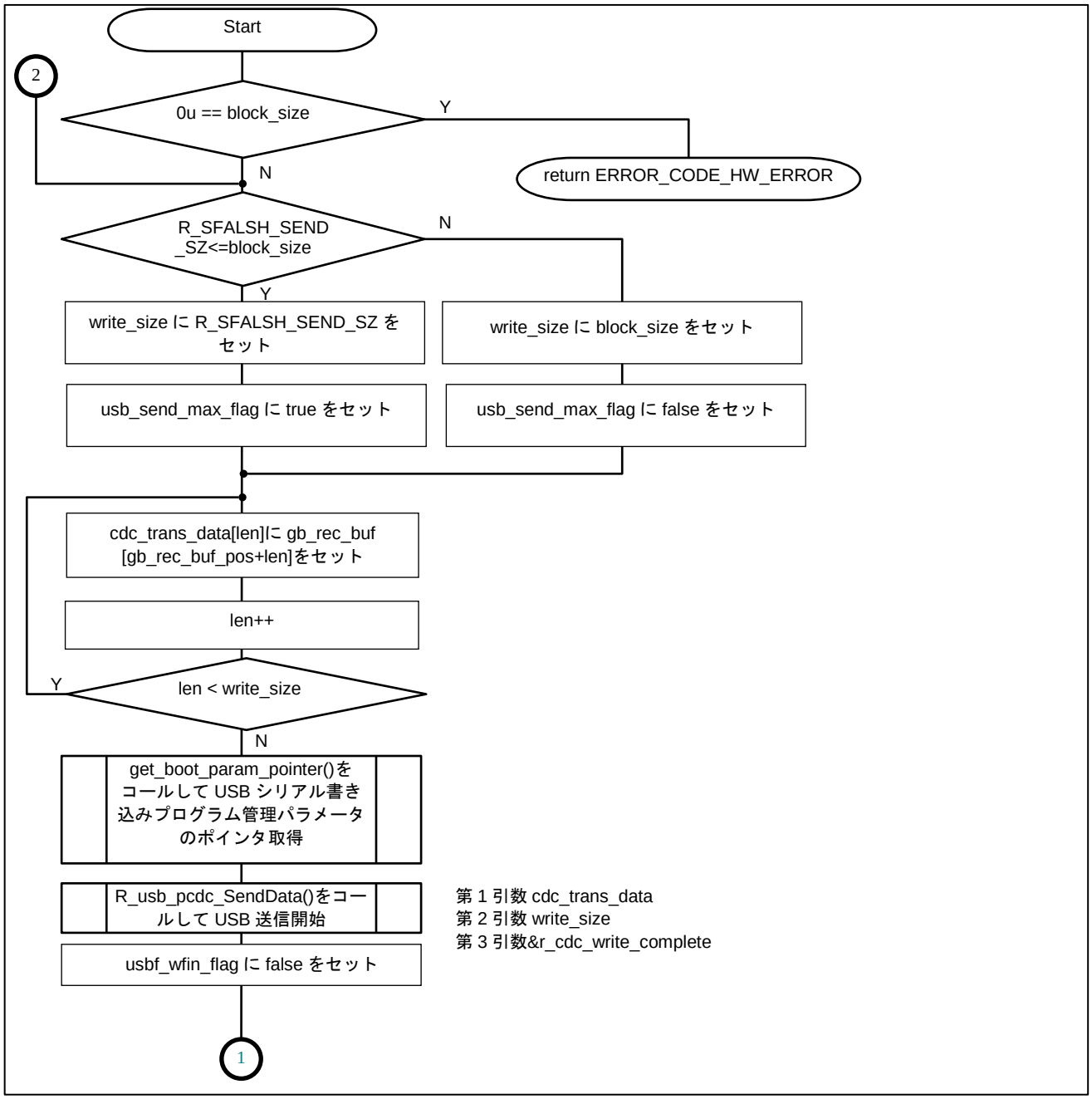


図 5.28　r_write_sFlash 関数処理

(37) r_trans_data_payload

r_trans_data_payload	
概　要	データ送信を実施
宣　言	static uint8_t r_trans_data_payload(uint32_t block_size)
説　明	下記にフローチャートを示します。
引　数	uint32_t block_size　　データ送信サイズ
リターン値	ERROR_CODE_SUCCESS：正常終了 ERROR_CODE_TIMEOUT：USB 送信タイムアウトエラー(10s) ERROR_CODE_HW_ERROR：HW エラー発生
補足	



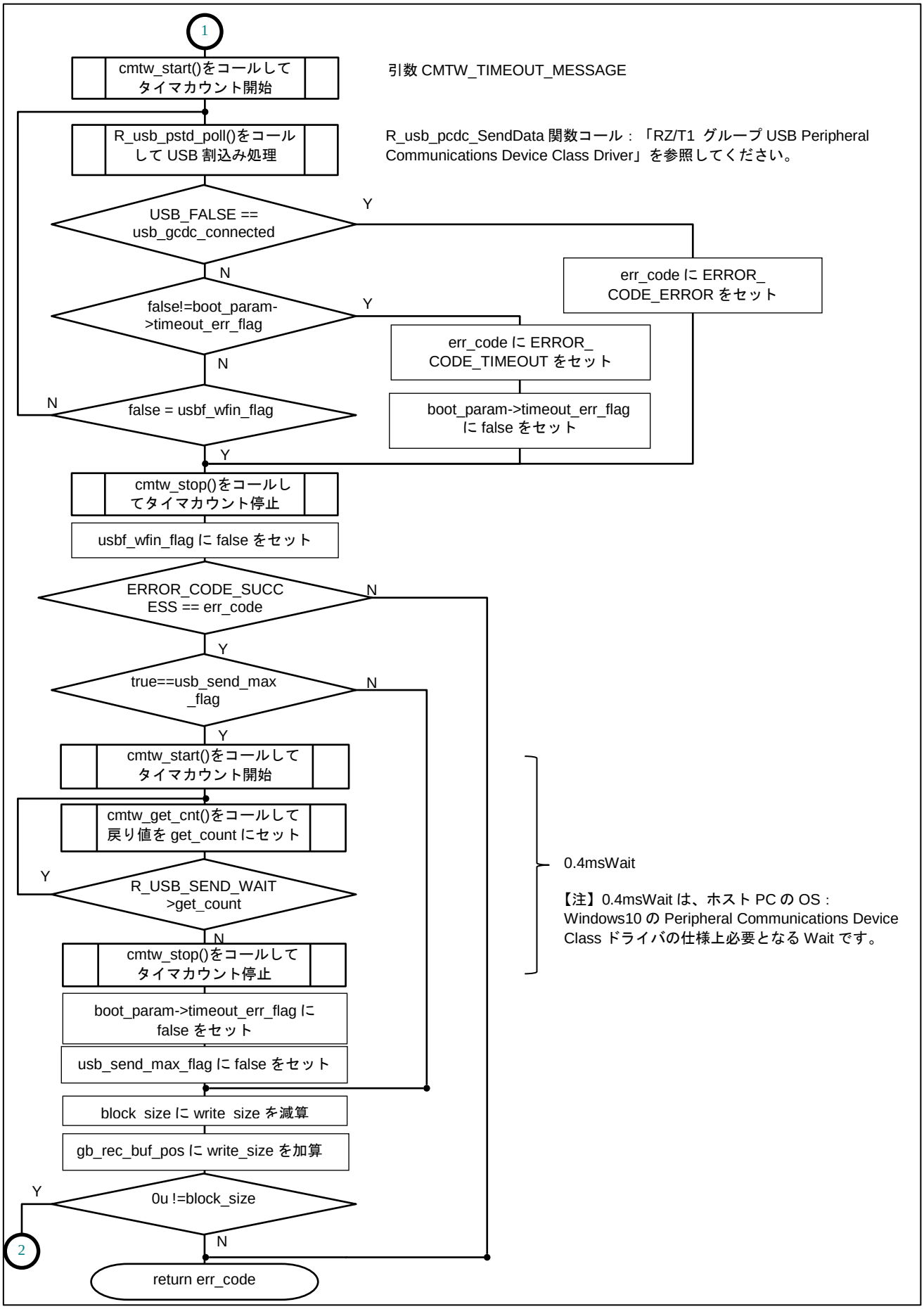


図 5.29 r_trans_data_payload 関数処理

(38) USB Peripheral Communications Device Class(PCDC)

USB Peripheral Communications Device Class(PCDC)については、「RZ/T1 グループ USB Peripheral Communications Device Class Driver」を参照してください。

(39) USB Peripheral Control Driver (PCD)

USB Peripheral Control Driver(PCD)については、「RZ/T1 グループ USB Peripheral Communications Device Class Driver」を参照してください。

(40) SPIBSC

以下に USB シリアル書き込みサンプルの SPIBSC の仕様を示します

・Address area :　4bytes (max memory size is 64MB)

・SPI data width :　Single (1bit)

・SPBCLK :　75MHz

USB シリアル書き込みサンプルは以下のコマンドを使用しています。

Description	Instruction	Note
Write Enable	WREN (0x06)	-
Write Register	WRR (0x01)	-
Page Program	PP4B (0x12)	-
Read	FASTREAD4B (0x0C)	-
Sector Erase	BE4B (0xDC)	-
Read Status Register	RDSR (0x05)	-

SPIBSC のプログラムの詳細な仕様に関しては、「RZ/T1 グループ シリアルフラッシュサンプルプログラム」を参照してください。

5.5　ユーザプログラム

ユーザプログラムは、「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」のユーザアプリケーションを使用しています。ユーザプログラムの詳細な仕様に関しては、「RZ/T1 グループ R-IN Engine 搭載製品　初期設定」を参照してください。

ユーザプログラムは、Cortex-M3 用ユーザプログラム領域をシリアル FlashROM から Instruction RAM へコピーする時に、「5.1 ローダプログラム動作概要」の表 5.1 のユーザプログラム情報テーブルを参照しています。図 5.30 に Cortex-M3 コア初期化処理(init_cm3 関数)のフローチャート(EWARM 版)を示します。

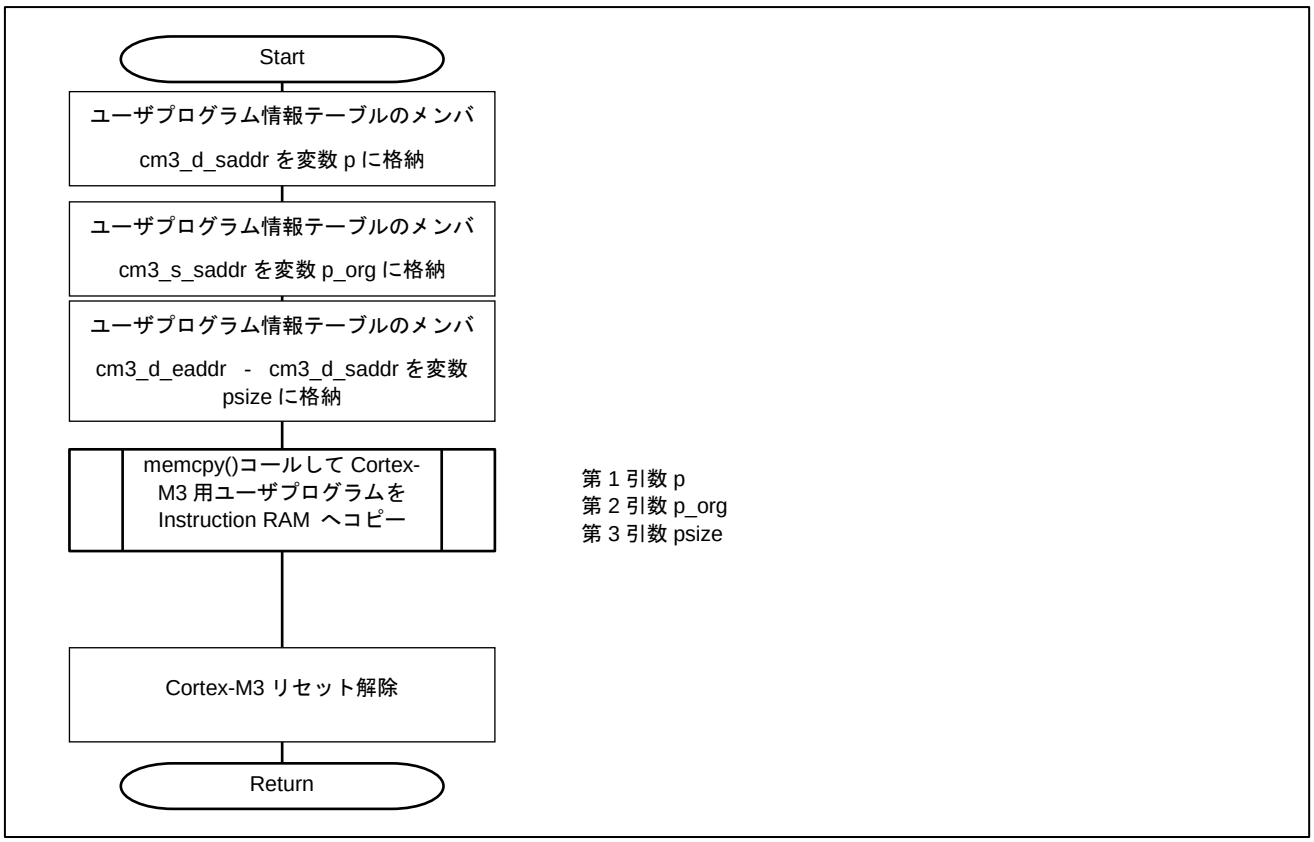


図 5.30 init_cm3 関数処理

6. USB シリアル書き込みサンプルの書き込み手順

6.1 RZ/T1 評価ボード接続

① ホスト PC(Tera Term インストール済み)と RZ/T1 評価ボードの USB コネクタ(J6)を USB ケーブルで接続します。

② ICE の JTAG コネクタを J10(ARM JTAG20)に接続し、開発用 PC と USB 接続します。

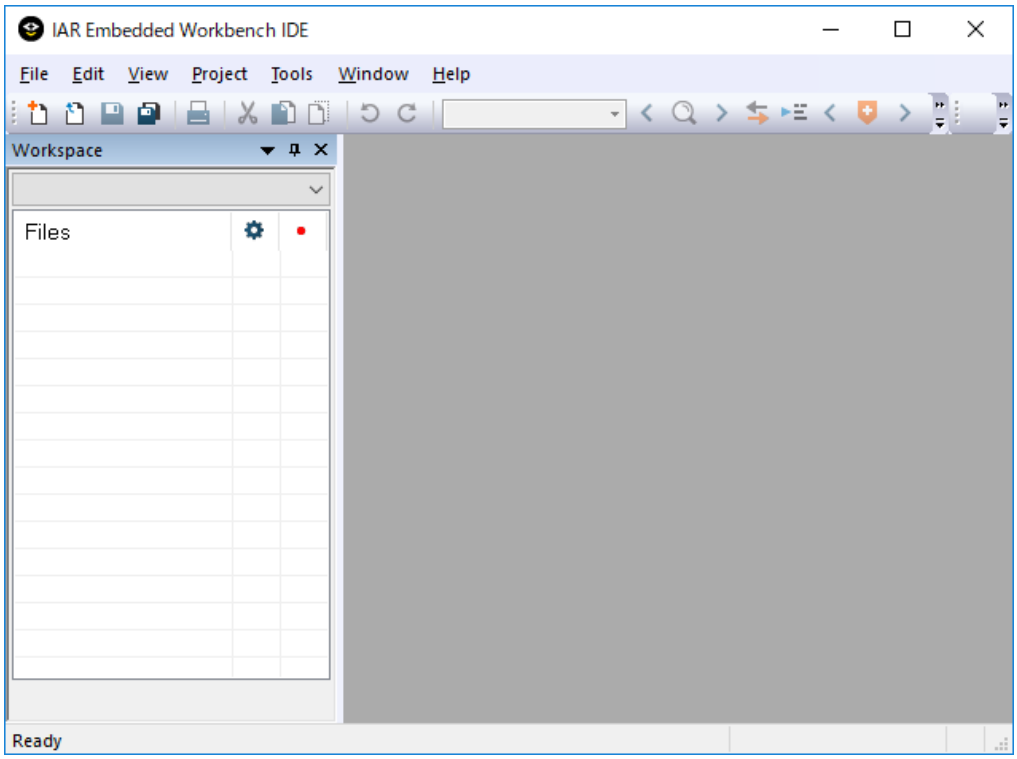
③ DC5V 出力 AC アダプターを J17 に接続し、電源を投入します。

6.2 USB シリアル書き込みサンプルの書き込み

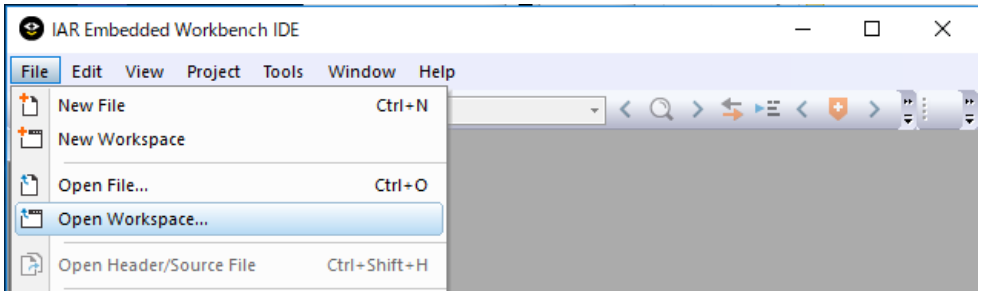
6.2.1 EWARM の場合

(1) USB シリアル書き込みサンプルのビルド

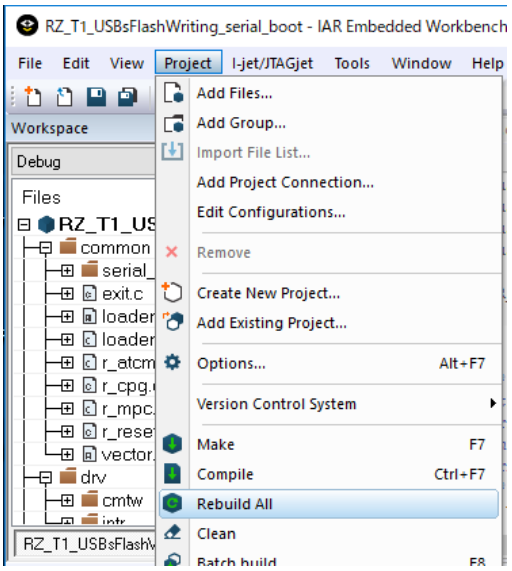
①Windows のスタートメニューから、[すべてのプログラム] → [IAR Embedded Workbench for Arm 8.30.1] → [IAR Embedded Workbench] をクリックし、IAR Embedded Workbench を起動します。



② [File] → [Open Workspace…] で“¥workspace¥iccarm¥RZ_T1_R-IN_init¥RZ_T1_R-IN_usb¥Cortex-R4¥RZ_T1_init_boot ¥RZ_T1_USBsFlashWriting_serial_boot.eww”ファイルをダブルクリックし、ワークスペースを開きます。

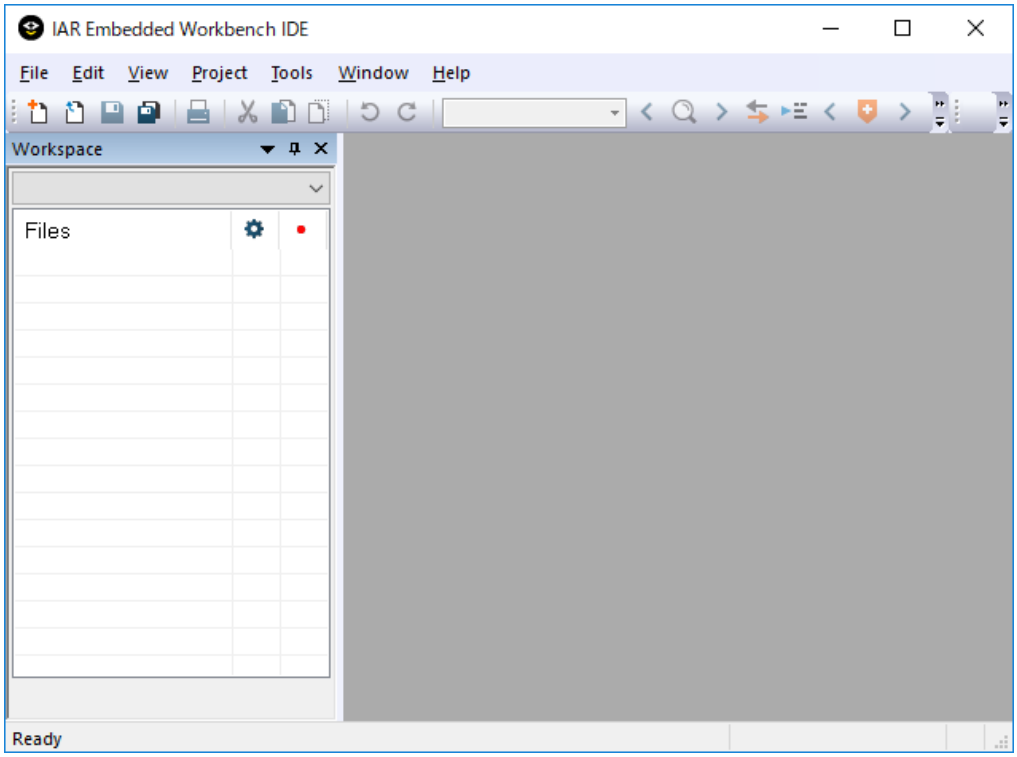


③ [Project] → [Rebuild All] でビルドを実行します。

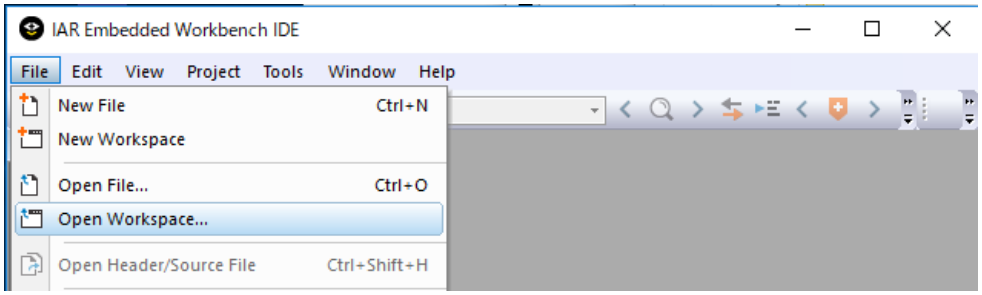


(2)　ユーザプログラムと USB シリアル書き込みサンプルのビルドと書き込み

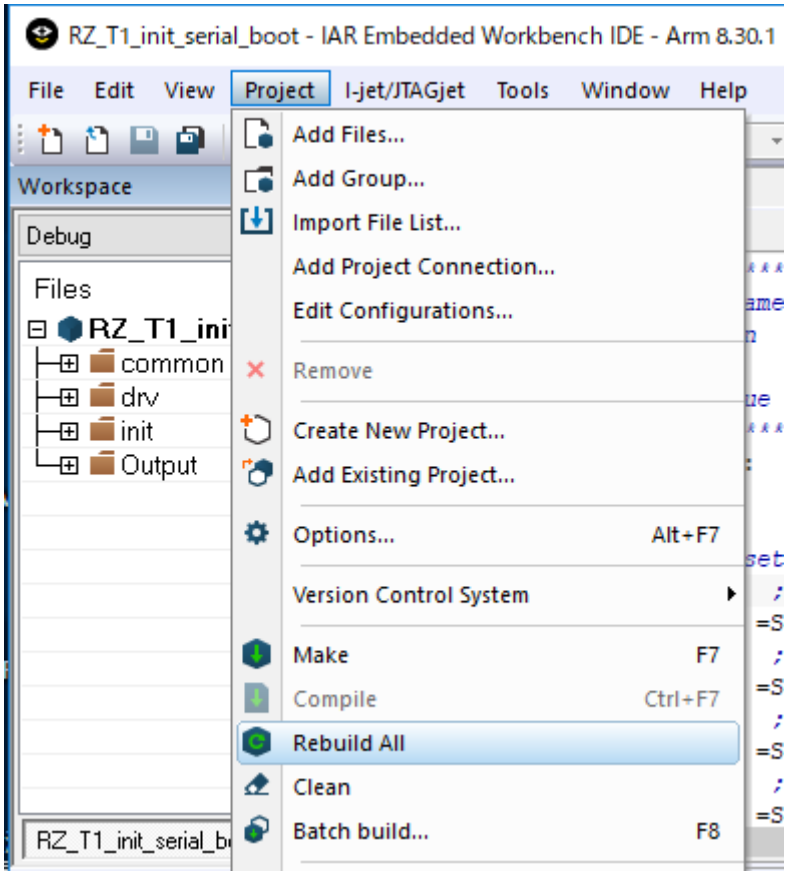
① Windows のスタートメニューから、[すべてのプログラム] → [IAR Embedded Workbench for Arm 8.30.1] → [IAR Embedded Workbench] をクリックし、IAR Embedded Workbench を起動します。



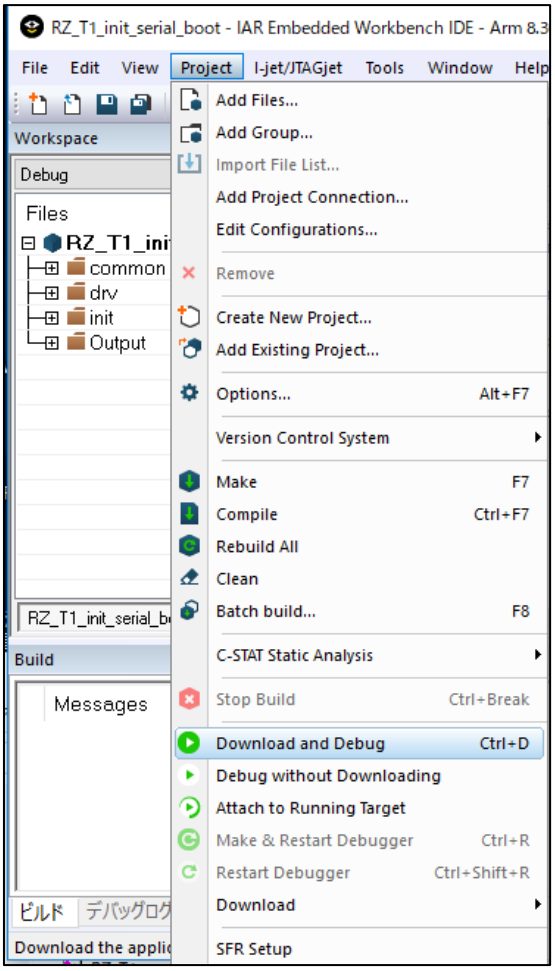
② [File] → [Open Workspace…] で“¥workspace¥iccarm¥RZ_T1_R-IN_init¥RZ_T1_R-IN_init¥Cortex-R4¥RZ_T1_init_boot¥RZ_T1_init_serial_boot.eww”ファイルをダブルクリックし、ワークスペースを開きます。



③ [Project] → [Rebuild All] でビルドを実行します。

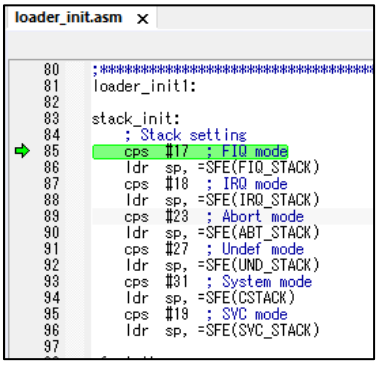


④ [Project] → [Download and Debug] で FlashROM への書き込みを行います。



FlashROM への書き込みが完了すると stack_init 関数の先頭でブレークします。これでプログラムの書き込みは完了です。デバッガを立ち下げて ICE を取り外して、RZ/T1 評価ボードをスタンドアロンで立ち上げます。

stack_init 関数の先頭でブレークしない場合、ボードの接続を確認して、「6. USB シリアル書き込みサンプルの書き込み手順」を再度実行してください。

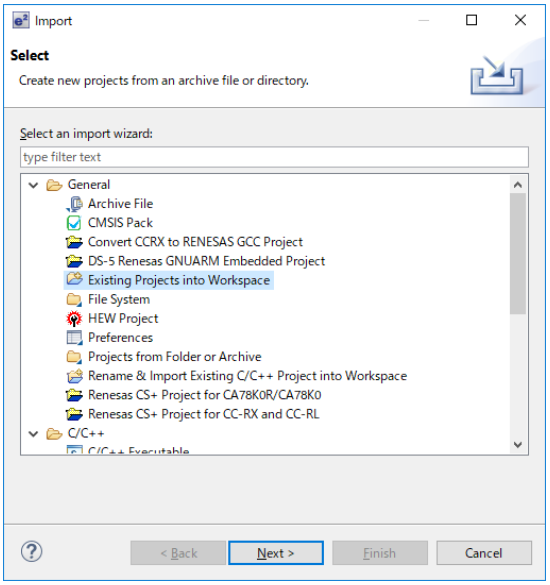


6.2.2　e² studio の場合

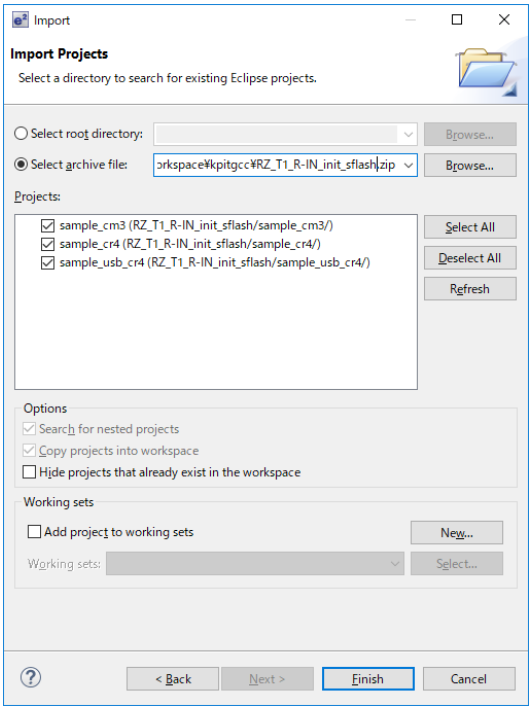
①ご利用 PC 上にサンプルプログラムを格納するためのワークスペースとして空のフォルダを作成します。

② e² studio を起動し、ワークスペースに①で作成したフォルダを指定します。

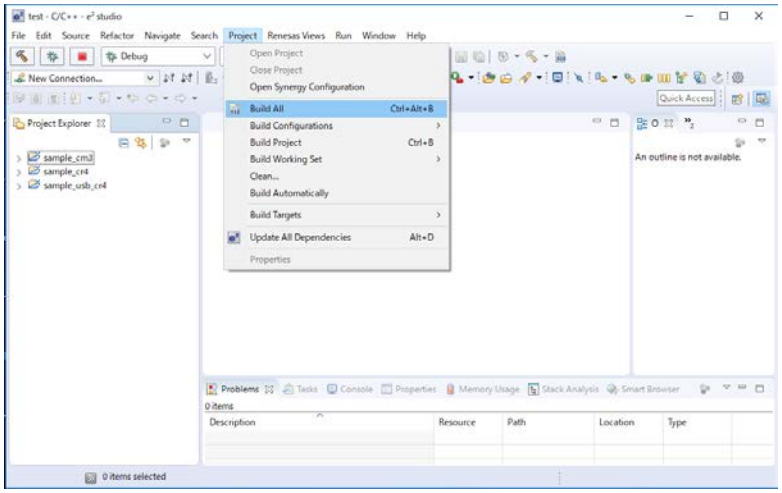
③ [File] → [Import…] でインポート画面を開き、[General] → [Existing Projects into Workspace] を選択して [Next」を押します。



④ [Select archive file:] のラジオボタンにチェックを付け、[Browse…] を選択し、圧縮されたサンプルプログラム（¥workspace¥kpitgcc¥RZ_T1_R-IN_init_sflash.zip）を選択し、[Finish] を押します。

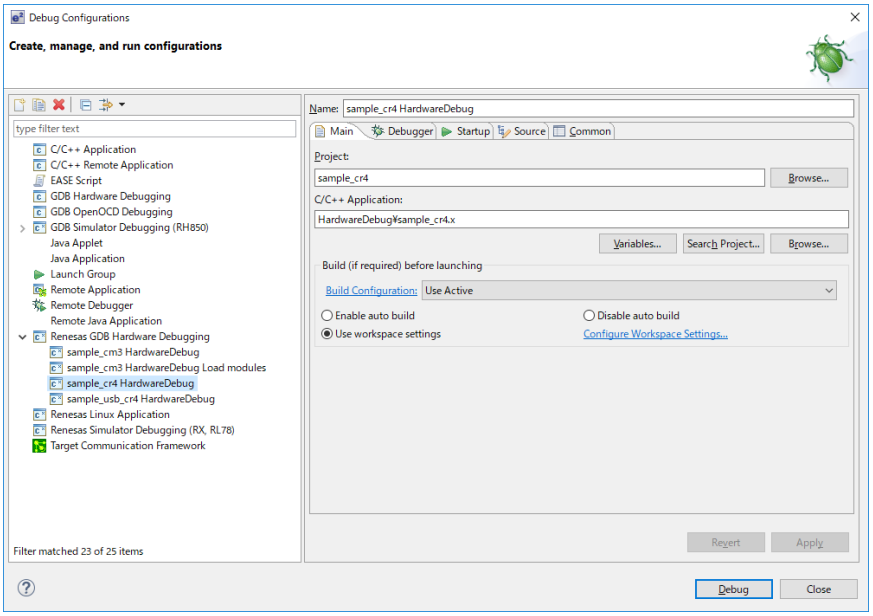


⑤ ProjectExplorer の”sample_cr4”をクリックして、[Project] → [Build All] を実行します。



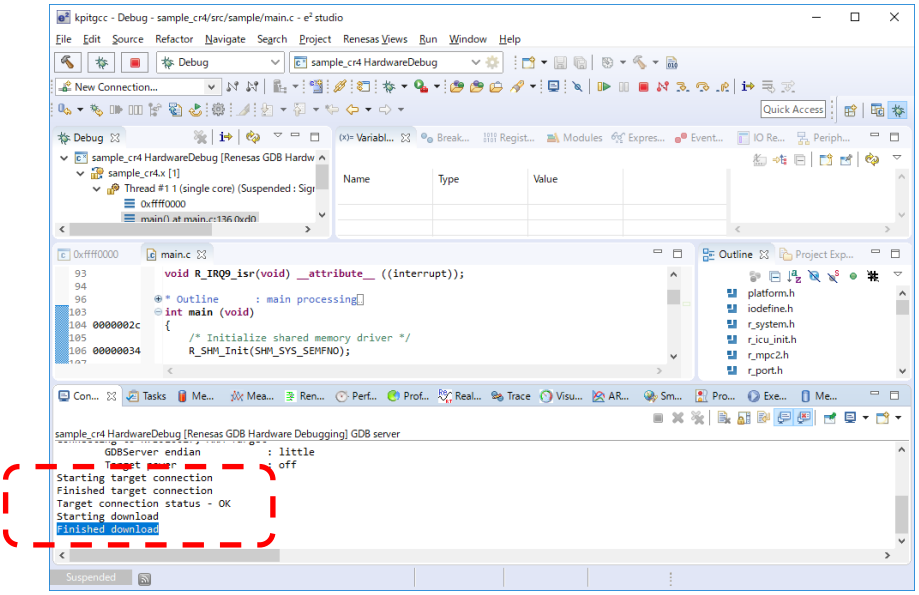
⑥[Run] → [Debug Configurations…] を選択し、Debug Configurations 画面を開きます。

⑦ RZ/T1 評価ボードと J-Link を接続した状態で、[Renesas GDB Hardware Debugging] → [sample_cr4 HardwareDebug] を選択後、[Debug] を押すことで、Cortex-R4 コアのデバッグを開始します。



FlashROM への書き込みが完了すると Console 画面にて「Finished download」が表示されます。これでプログラムの書き込みは完了です。デバッガを立ち下げて ICE を取り外して、RZ/T1 評価ボードをスタンドアロンで立ち上げます。

「Finished download」が表示されない場合、ボードの接続を確認して、「6. USB シリアル書き込みサンプルの書き込み手順」を再度実行してください。



7.　ユーザプログラムをシリアル FlashROM へ書き込む方法

以降、特に明記しない場合は EWARM（IAR システムズ社製）を使用した場合について説明します。

本サンプルプログラムのパッケージの中に、LED2 が点滅するユーザプログラム(以下、書き込み対象プログラム)「¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.bin」を用意しています。

本章では、シリアル FlashROM 書き込み方法として、以下 2 通りの書き込み方法を説明します。

・TeraTerm マクロを使用したシリアル FlashROM 書き込み方法

・TeraTerm マクロを使用せずにシリアル FlashROM へ書き込む方法

7.1　TeraTerm マクロを使用したシリアル FlashROM 書き込み方法

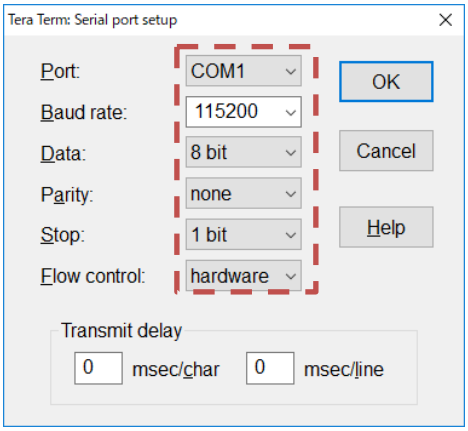
7.1.1　RZ/T1 評価ボード接続

① ホスト PC(Tera Term インストール済み)と RZ/T1 評価ボードの USB コネクタ(J6)を USB ケーブルで接続します。

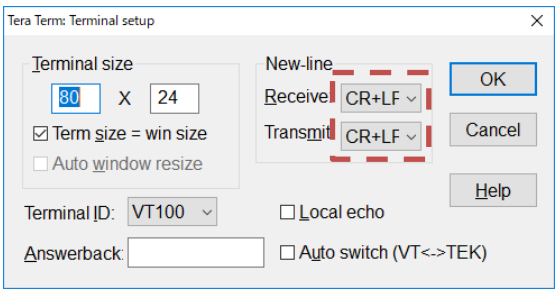
② DC5V 出力 AC アダプターを J17 に接続し電源を投入し、SW3 を押下したまま、リセットボタンを押下します。

③ ホスト PC にインストールされている Tera Term を起動します。

④ 通信設定を行います。Tera Term のメニューから“Setup” -> “Serial port…”を選択します。

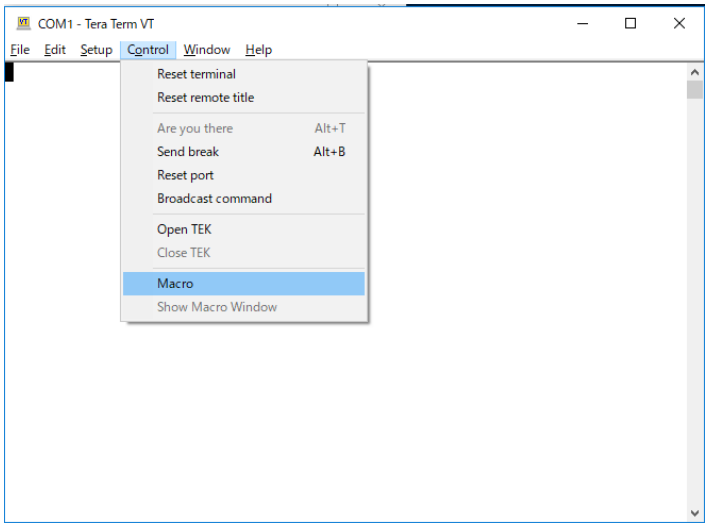


⑤ ターミナル設定を行います。Tera Term のメニューから“Setup” -> “Terminal…”を選択します。「New line」の「Receive」と「Transmit」を「CR+LF」にします。

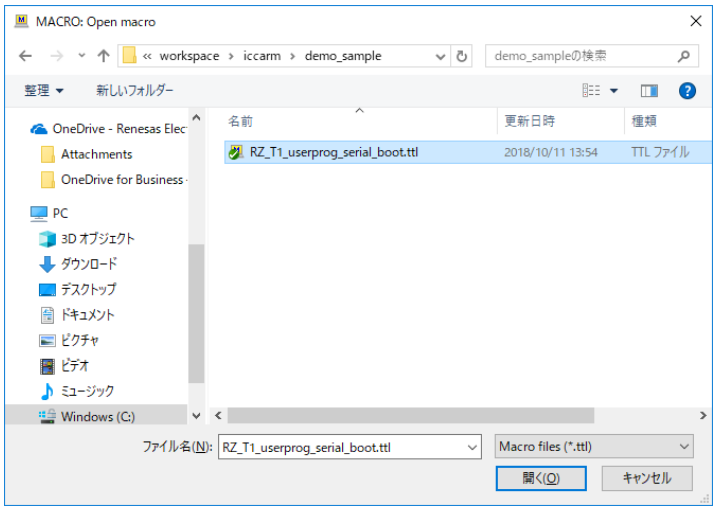


7.1.2　TeraTerm マクロの実行

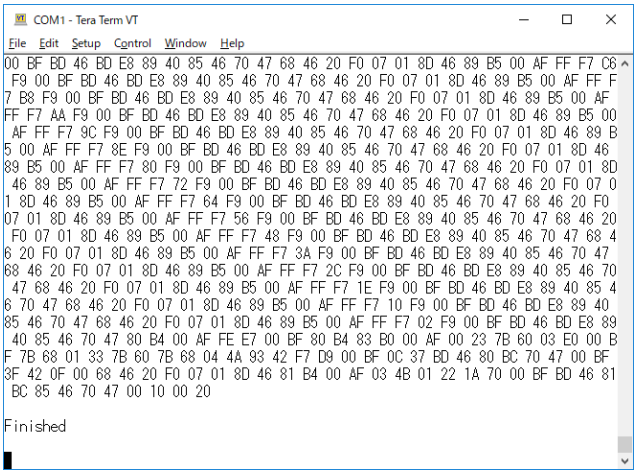
① Tera Term のメニューから“Control” -> “Macro”を選択します。



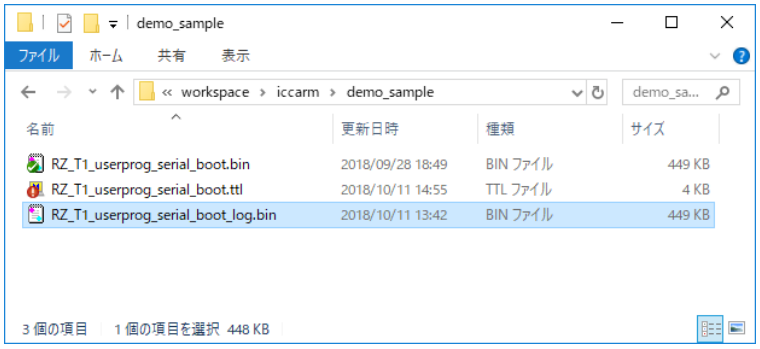
② TeraTerm マクロ(¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.ttl)を選択すると TeraTerm マクロが実行されます。



③ “Finished”が表示されると、書き込み完了です。



③実行後シリアル FlashROM 領域から読み出したバイナリファイルが、RZ_T1_userprog_serial_boot.bin と同じフォルダに(RZ_T1_userprog_serial_boot_log.bin)生成されます。書き換え対象プログラム(RZ_T1_userprog_serial_boot.bin)と比較して一致していることを確認し、RZ_T1_userprog_serial_boot.bin が正常にシリアル FlashROM に書き込まれたことを確認します。



シリアル FlashROM に書き込んだ書き換え対象プログラムを起動する場合、評価ボードの電源を OFF にして、再び電源を ON にしてリセットを押下して書き換え対象プログラムを起動します。この時 SW3 を押下しないでリセットを押下します。

7.2　TeraTerm マクロを使用せずにシリアル FlashROM へ書き込む方法

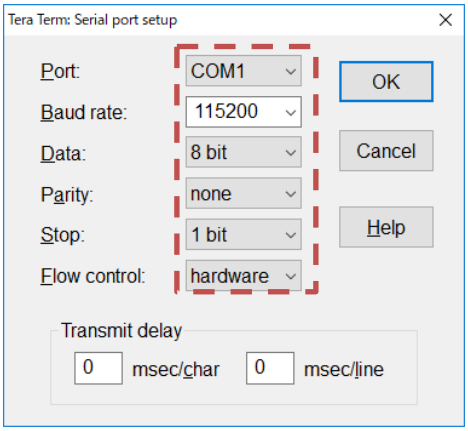
7.2.1　RZ/T1 評価ボード接続

① ホスト PC(Tera Term インストール済み)と RZ/T1 評価ボードの USB コネクタ(J6)を USB ケーブルで接続します。

② DC5V 出力 AC アダプターを J17 に接続し電源を投入し、SW3 を押下したまま、リセットボタンを押下します。

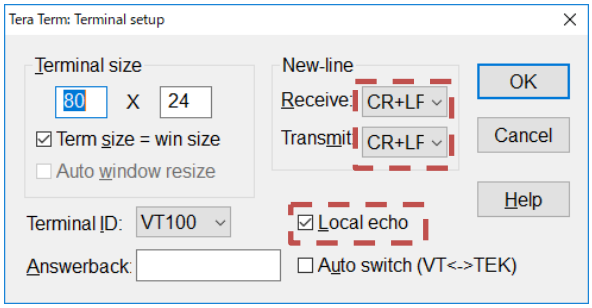
③ ホスト PC にインストールされている Tera Term を起動します。

④ 通信設定を行います。Tera Term のメニューから“Setup” -> “Serial port…”を選択します。



⑤ ターミナル設定を行います。Tera Term のメニューから“Setup” -> “Terminal…”を選択します。

「New line」の「Receive」と「Transmit」を「CR+LF」にします。入力コマンドを確認したい場合は、「Localecho」にチェックをします。



⑥　表示モードを選択します。

USB シリアル書き込みサンプルは下記 2 種類の表示モードを用意しており、ターミナルソフトからの入力データにより、表示モードを選択します。

○コマンド入力開始文字ありモード

’D’または、’d’を入力した場合、コマンド入力開始文字の’>’が表示されます。

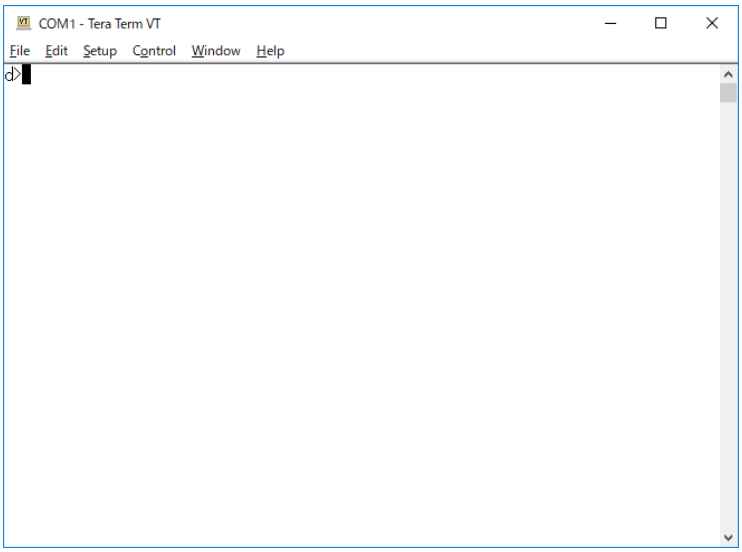
○コマンド入力開始文字なしモード

‘D’または、’d’以外を入力した場合、コマンド入力開始文字の’>’が表示されません。

TeraTerm の Log 機能を使用してシリアル FlashROM に格納されているバイナリを読み出す時には、コマンド入力開始文字の’>’が Log に含まれてしまうため、コマンド入力開始文字なしモードにします。

TeraTerm の Log 機能を使用してシリアル FlashROM に格納されているバイナリを読み出す方法については、「8.2.2 シリアル FlashROM 領域の読み出し」を参照してください。

下記にコマンド入力開始文字ありモード時のターミナルソフトの画面を示します。



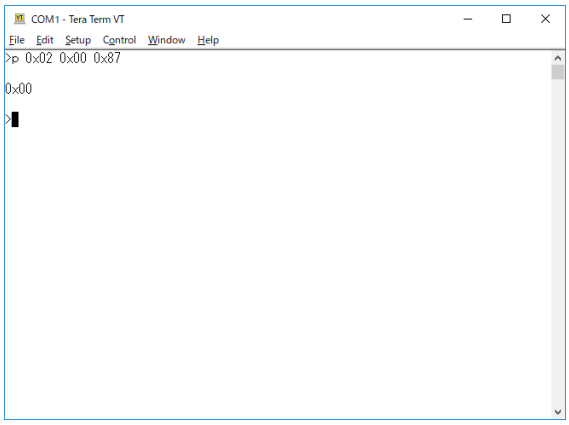
USB シリアル書き込みサンプルが正常に起動されると LED10 が点灯されます。LED10 が点灯されない場合、ボードの電源を OFF にして、①から再度実行してください。

7.2.2　ユーザプログラムの書き込み

以降のターミナルソフトの画面は、コマンド入力開始文字ありモード(‘>’を表示)の画面です。

(1)　シリアル FlashROM のライトプロテクション解除

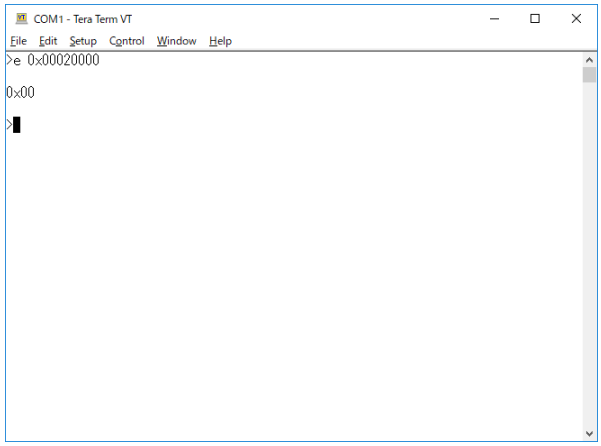
"p 0x02 0x00 0x87”を入力して、シリアル FlashROM のライトプロテクションを解除します。解除が完了すると”0x00”が表示されます。



(2)　セクターイレース

"e 0x00020000”を入力して、シリアル FlashROM 領域をセクターイレースします。セクターイレースが完了すると”0x00”が表示されます。セクターイレースコマンドを使用して、書き換え対象プログラムのサイズ以上の領域を消去します。セクターサイズに関しては、シリアル FlashROM のデータシートをご参考ください。

セクターイレース中は、評価ボードの電源を OFF にしないでください。

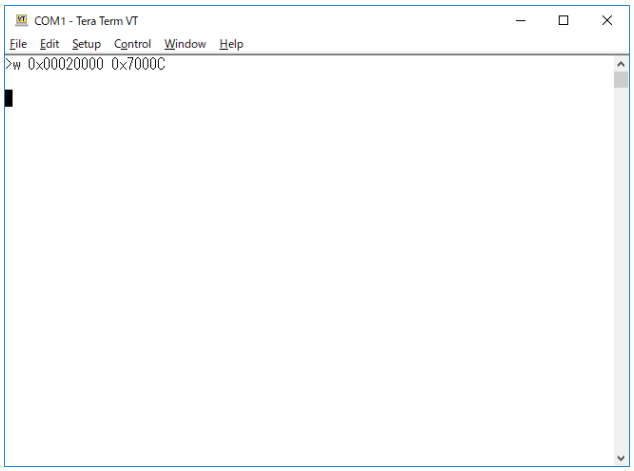


(3)　シリアル FlashROM のライトプロテクション解除

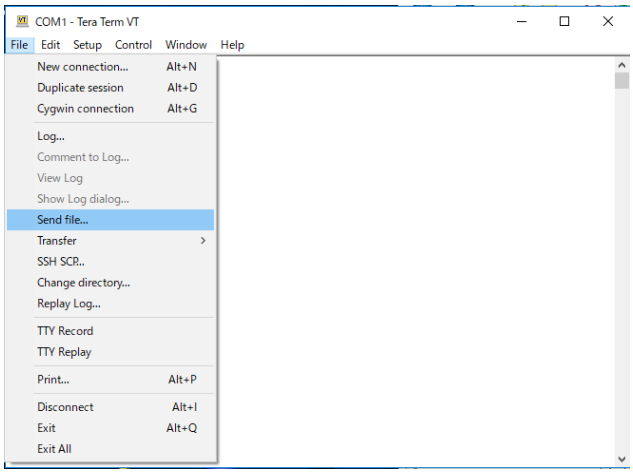
再びライトプロテクションを解除します。詳細は、「(1) シリアル FlashROM のライトプロテクション解除」を参照してください。

(4)　シリアル FlashROM への書き込み

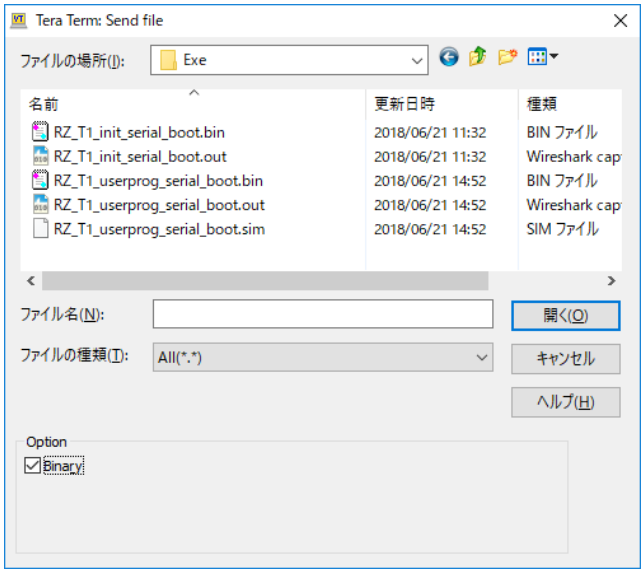
①　“w 0x00020000 0x7000C(書き換え用プログラムサイズ)”を入力して、書き換え対象プログラムをシリアル FlashROM に書き込みます。シリアル FlashROM への書き込み中は、評価ボードの電源を OFF にしないでください。



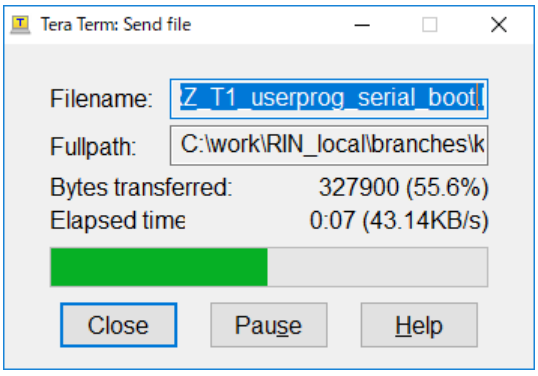
②　Tera Term のメニューから“File” -> “Send file…”を選択します。



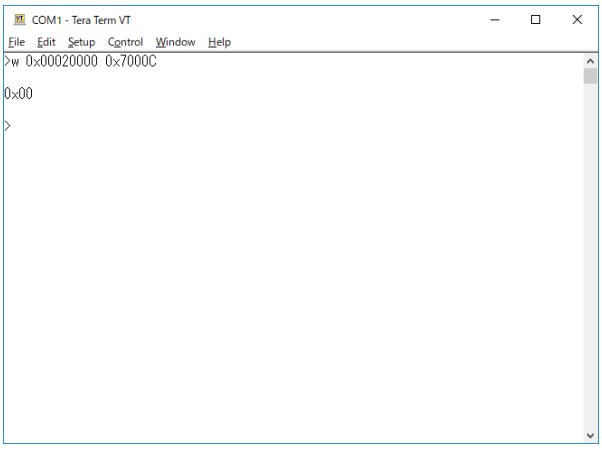
③　書き換え対象プログラム(“¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.bin”)を選択します。この時「Option」の「Binary」にチェックをします。



④　送信を開始します。



⑤　書き込みが完了すると、”0x00”が表示されます。シリアル FlashROM に書き込んだ書き換え対象プログラムを起動する場合、評価ボードの電源を OFF にして、再び電源を ON にしてリセットを押下して書き換え対象プログラムを起動します。この時 SW3 を押下しないでリセットを押下します。



8. USB シリアル書き込みサンプルのコマンド仕様

以降のターミナルソフトの画面は、コマンド入力開始文字ありモード(‘>’を表示)の画面です。

8.1 コマンド一覧

USB シリアル書き込みサンプルが受付可能なコマンドの一覧を示します。ターミナルソフトから本コマンドを実行してください。

表 8.1 コマンド一覧表

No	コマンド	名称	概要
1	‘w’ or ‘W’	シリアル FlashROM 領域にユーザプログラムを書き込み	PP4B(0x12)コマンドを使用してシリアル FlashROM 領域にユーザプログラムを書き込みます。
2	‘r’ or ‘R’	シリアル FlashROM 領域の読み出し	FASTREAD4B(0x0C)コマンドを使用してシリアル FlashROM 領域に格納されているユーザプログラムを読み出します。
3	‘e’ or ‘E’	セクターイレース	BE4B(0xDC)コマンドを使用して指定アドレスが含まれるセクターをセクターイレースします。
4	‘p’ or ‘P’	Protect Control	WRR(0x01)コマンドを使用してシリアル FlashROM のライトプロテクションを解除します。

コマンドで使用するアドレスはシリアル FlashROM 内(0x00000000~0x040000000)のアドレスとなります。

8.2 コマンド詳細

8.2.1 シリアル FlashROM 領域にユーザプログラムを書き込み

ホスト PC のターミナルソフトからユーザプログラムのバイナリを受信してシリアル FlashROM に書き込みます。このコマンドを実行する前に、「セクターイレース」、「Protect Control」を実行します。

① コマンドフォーマット

w^address^size[CR+LF]

^：スペース、[CR+LF]：改行コード

② 使用例

w 0x00000000 0x00000001

W 0X00000000 0X00000001

③ 引数

address：ベースアドレス(“0x”を含む 16 進数で記載します。(e.g. 0x00000000))

size：ライトサイズ(“0x”を含む 16 進数で記載します。(e.g. 0x00000001))

(“w“コマンド実行後にユーザプログラムをバイナリで送ります。)

④　戻り値

"0x00" : 正常終了

"0xFE" : パラメータエラー、またはコマンドフォーマットエラー

"0xFC" : ベリファイエラー

"0xFB" : 受信したコマンドが未サポート

"0xFA" : バイナリ受信待ちのタイムアウトエラー(10s)

⑤　Note

・address 設定範囲：0x00000000 ~ 0x03FFFFFF(シリアル FlashROM アドレス)

・size 設定範囲：0x00000001 ~ 0x04000000

・指定した size よりも、ユーザプログラムのバイナリサイズの方が大きい場合、指定した size 分までのデータがシリアル FlashROM に書き込まれ、超過分は書き込まれません。なお、超過分のデータを送信した場合、USB シリアル書き込みサンプルは、超過分のデータを次のコマンドのデータと認識します。必ず、指定した size と同じサイズのバイナリデータを送信してください。

・address+size が 0x04000000 より大きい場合は、パラメータエラーとなります。

8.2.2　シリアル FlashROM 領域の読み出し

シリアル FlashROM に格納されているバイナリを読み出し、ホスト PC へ送信します。このコマンドを実行する前に、「Protect Control」を実行します。

① コマンドフォーマット

r^address^size[CR+LF]

^：スペース、[CR+LF]：改行コード

② 使用例

r 0x00000000 0x00000001

R 0X00000000 0X00000001

③ 引数

address：ベースアドレス(“0x”を含む 16 進数で記載します。(e.g. 0x00000000))

size：リードサイズ(“0x”を含む 16 進数で記載します。(e.g. 0x00000001))

④ 戻り値

正常終了した場合は、戻り値を表示しません。

"0xFE"：パラメータエラー、またはコマンドフォーマットエラー

"0xFB"：受信したコマンドが未サポート

⑤ Note

・address 設定範囲：0x00000000 ~ 0x03FFFFFF(シリアル FlashROM アドレス)

・size 設定範囲：0x00000001 ~ 0x04000000

・address+size が 0x04000000 より大きい場合は、パラメータエラーとなります。

以下にシリアル FlashROM アドレス：0x00000000 からサイズ：0x100 のデータを読み込む例を示します。

- Tera Term からコマンドを入力します。

  以下にコマンドフォーマットを示します。

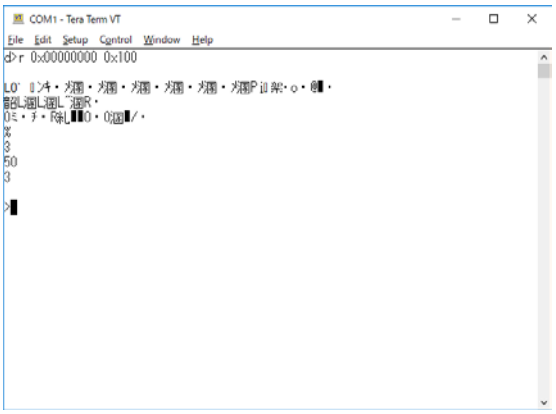
  “r␣0x00000000␣0x100”⏎

  　　↑アドレス　　↑サイズ

【コマンド入力開始文字ありモード】

コマンド入力開始文字ありモード時の TeraTerm の画面を以下に示します。

TeraTerm への送信が完了すると、コマンド入力開始文字の’>’が表示されます。



【コマンド入力開始文字なしモード】

コマンド入力開始文字なしモード時の TeraTerm の画面を以下に示します。

コマンド入力開始文字の'>'が表示されないため、指定したサイズのバイナリが表示されたか判断できません。コマンド入力開始文字なしモード時は TeraTerm の Log 機能(“File”->”Log…”)を使用して、バイナリをファイルに保存することをお奨めします。以下に TeraTerm の Log 機能(“File”->”Log…”)を使用して、バイナリをファイルに保存する方法を示します。

① TeraTerm の受信速度を上げるため、TeraTerm を 16 進表示モードにします。TeraTerm 上で「Shift」+「Esc」を入力します。

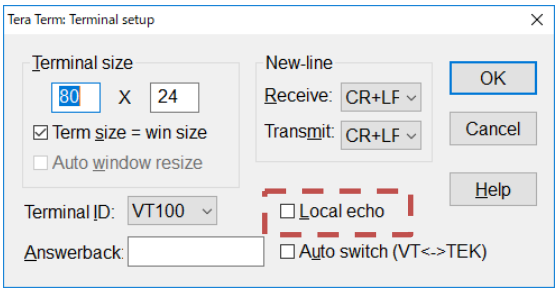
16 進表示モードを有効にするには、TeraTerm を起動する前に設定ファイル(デフォルト：C:¥Program Files (x86)¥teraterm¥TERATERM.INI)を開き、Debug=on、DebugModes=hex に変更します。

```
; Display all characters (debug mode)↵
Debug=on↵
; Debug mode type which can be selected by user.↵
;   on|all    = All types↵
;   off|none  = Disabled debug mode↵
;   normal    = usual teraterm debug mode↵
;   hex       = hex output↵
;   noout     = disable output completely↵
DebugModes=hex↵
```

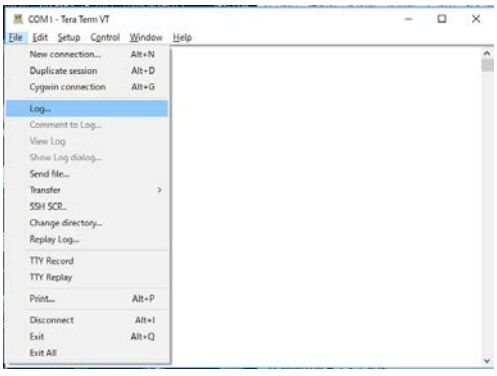
図 8.1 TERATERM.INI

② ターミナル設定を行います。Tera Term のメニューから“Setup” -> “Terminal…”を選択します。

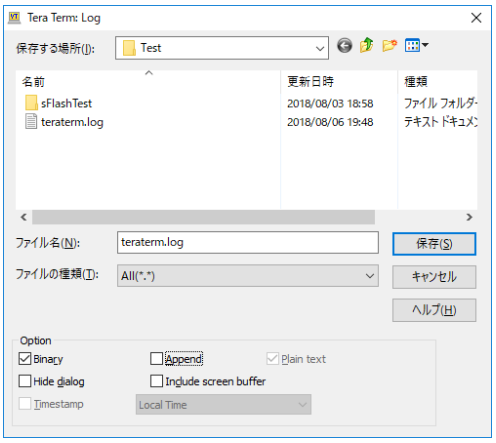
入力コマンドが Log に保存されてしまうため、「Localecho」のチェックを外します。



③ R コマンド実行前に TeraTerm の Log 機能(“File”->”Log…”)を選択します。

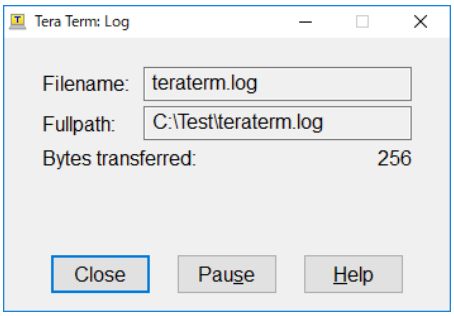


④　バイナリを保存するファイル(“teraterm.log”)を選択します。「Option」の「Binary」にチェックをします。



⑤　R コマンドを実行します。

⑥　保存されたバイナリのサイズを確認して“Close“を押下します。



TeraTerm の Log 機能(“File”->”Log…”)を使用してバイナリをファイルに保存する場合は、コマンド入力開始文字ありモードで Log を実行するとコマンド入力開始文字の’>’が保存されてしまうため、コマンド入力開始文字なしモードで Log を実行します。

8.2.3　セクターイレース

シリアル FlashROM をセクターイレースします。このコマンドを実行する前に、「Protect Control」を実行します。

① コマンドフォーマット

e^address[CR+LF]

^：スペース, [CR+LF]：改行コード

② 使用例

e 0x00000000

E 0X00000000

③ 引数

address：セクターアドレス(“0x”を含む 16 進数で記載します。(e.g. 0x00000000))

④ 戻り値

"0x00"：正常終了

"0xFE"：パラメータエラー、またはコマンドフォーマットエラー

"0xFB"：コマンド範囲外

⑤ Note

・address 設定範囲：0x00000000 ~ 0x03FFFFFF(シリアル FlashROM アドレス)

・address が 0x04000000 以上の場合は、パラメータエラーとなります。

8.2.4 Protect Control

シリアル FlashROM のライトプロテクションを解除します。

① コマンドフォーマット

p^size^data1^data2^....^data4[CR+LF]

^ : スペース、 [CR+LF] : 改行コード

② 使用例

p 0x02 0x00 0x87

P 0X04 0X00 0X01 0X02 0X03

③ 引数

size：data 引数の Total 数(“0x”を含む 16 進数で記載します。(e.g. 0x00000001))

data：シリアル FlashROM のライトデータ(“0x”と 2 桁の 16 進数で記載します。(e.g. 0x00000001))

data はシリアル FlashROM の仕様に従って指定できます。

④ 戻り値

"0x00" : 正常終了

"0xFE" : パラメータエラー、またはコマンドフォーマットエラー

"0xFB" : コマンド範囲外

⑤ Note

・size 設定範囲：0x01 ~ 0x04(引数：data の数(data1 data2....data4)と等しくする必要があります。)

・データは Protect Control データとしてシリアル FlashROM に書き込まれます。

・data は、ユーザが実際に使用するシリアル FlashROM の仕様に依存します。 お使いになるシリアル FlashROM のデータシートを参照してください。

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
1.0	2018.10.31	—	初版作成
2.0	2019.9.30	P.10, P.12-P.13, P.14-P.15, P.65	ユーザプログラムのみ書き換え可能にするために、ユーザプログラム情報テーブルを追加 5.1 ローダプログラム動作概要の修正 5.1.1 ユーザプログラム情報テーブルの追加 5.3 サンプルプログラムのセクション配置の修正 5.5 ユーザプログラムの修正
		P.23, P.25, P.71-P.72	USB シリアル書き込みサンプルの Cortex-M3 のプロジェクト削除 5.4.3 ソフトウェア詳細の(4)の修正 5.4.3 ソフトウェア詳細の(5)の修正 6.2.2 e2 studio の場合の修正

製品ご使用上の注意事項

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

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