RZ/T1 Group

Sample Program for Writing Serial Flash ROM via the USB
Microcomputers Incorporating the R-IN Engine

R01AN4432EJ0200
Rev.2.00
Sep. 30, 2019

Overview

This application note describes a sample program for writing the user application programs (hereinafter referred to as, "user program" or "user programs") to the serial flash ROM via the USB, utilizing the USB Peripheral Communications Device Class (PCDC) facility.

The major features of the sample program for writing serial flash ROM via the USB (hereinafter referred to as "USB serial writing sample program") are listed below.

- This sample program consists of three sections: the loader program, the user program, and the USB serial writing sample program.
- After booting of the RZ/T1, the loader program initializes the clock generator circuit and bus state controller, copies a user program or the USB serial writing sample program, written in the serial flash ROM, into the ATCM area, and starts the copied program.
  The level of the P44 pin on the board determines whether the loader program copies a user program or the USB serial writing sample program.
- The user program controls LEDs on the evaluation board. Pressing SW2 on the board makes the Arm® Cortex®-R4 core write the state of LED1 in terms of being on or off, as the LED data, to a shared memory area within the external pin interrupt handling routine. The Cortex-M3 core, on the other hand, reads the LED data in the shared memory and reflects its state through LED1 on the board.

  The user program in use is detailed in the RZ/T1 Group Application Note: Initial Settings of the Microcomputers Incorporating the R-IN Engine. For the specifications of the user program, refer to the RZ/T1 Group Application Note: Initial Settings of the Microcomputers Incorporating the R-IN Engine.
- The host PC sends commands with the use of terminal software such as TeraTerm. The USB serial writing sample program receives the commands via the USB and executes them. Tasks handled by the commands include downloading of the user program, writing to the serial flash ROM, and erasing sectors.

Target Device

RZ/T1 group microcomputers incorporating the R-IN engine

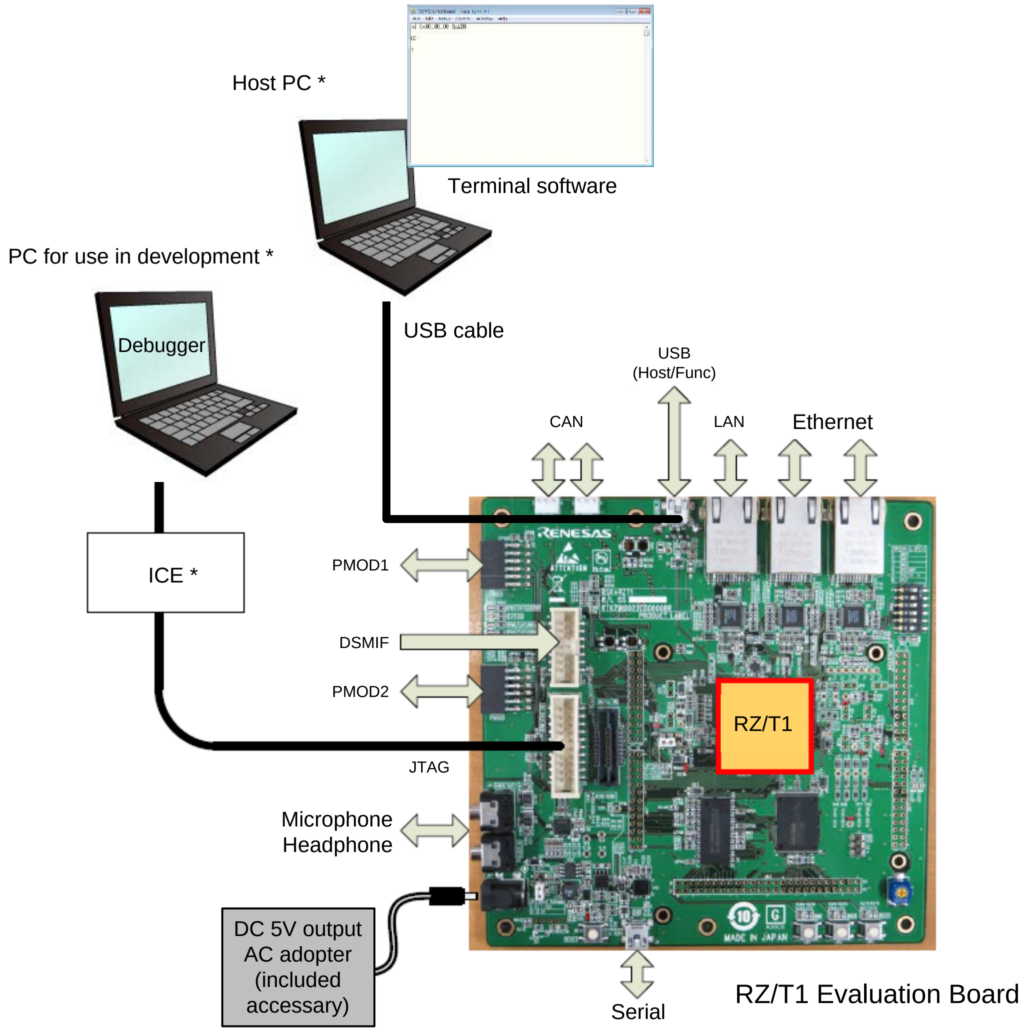
When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation and testing of the modified program.

Contents

1. System Specifications

1.1 System Configuration

Figure 1.1 shows system configuration of the sample program.



Note: Indicates the device that the user needs to prepare.

Figure 1.1 System Configuration

1.2 Operating Environment

The sample program described in this application note is for the environment below.

Table 1.1 Operating Environment

Item	Description
Board	RZ/T1 evaluation board RTK7910018C00000BE
MCU	RZ/T1 (incorporating the R-IN Engine) R7S910018
Operating frequency	CPU clock (CPUCLK): 450 MHz (Arm Cortex-R4) System clock (ICLK): 150 MHz (Arm Cortex-M3)
Operating voltage	3.3 V
Operating mode	SPI boot mode
Device	Serial flash ROM (64 Mbytes) MX25L51245GMI-10G from Macronix (sector size: 64 Kbytes)
Integrated development environment	Embedded Workbench® for Arm v8.30.1 from IAR systems e^2 studio 6.1.0 from RENESAS
Emulators	I-jet from IAR Systems J-Link from SEGGER
Terminal software	TeraTerm v4.97
Host PC	Windows 10 Enterprise Intel® Core™ i5-6300U processor running at 2.4 GHz or 2.5 GHz

1.3 Pins

1.3.1 Selecting Operating Mode

Operating modes of the RZ/T1 are selected with the external pins MD0, MD1, and MD2.

The table below is the levels of the individual pins for each operating mode.

Table 1.2 Operating Mode Selection

Mode Setting Pins			
MD2	**MD1**	**MD0**	**Operating Modes**
Low	Low	Low	SPI boot mode (serial flash ROM) Boot from the serial flash ROM which is connected to the SPI multi-I/O bus space.
Low	High	Low	16-bit bus boot mode (NOR flash) Boot from the 16-bit bus NOR flash memory which is connected to the CS0 space.
Low	High	High	32-bit bus boot mode (NOR flash) Boot from the 32-bit bus NOR flash memory which is connected to the CS0 space. (This setting is prohibited for the RZ/T1 evaluation boards.)
Other than above			Reserved (setting prohibited)

For selecting from among the operating modes listed above, DIP-SW switches SW4-1, 4-2, and 4-3 are provided on the RZ/T1 evaluation board to select the MD0, MD1, and MD2 pins. The operating mode is selected by the settings of SW4 on the evaluation board. This sample program runs in SPI boot mode and uses the combination of switch settings shaded in gray in the list below.

Table 1.3 Combination of SW4 Settings

Sample Program	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16-bit bus boot mode	ON	OFF	ON	ON	ON	OFF
SPI boot mode	ON	ON	ON	ON	ON	OFF

Select the operating mode before supplying power to the board.

1.3.2 Switching to the USB Serial Writing Sample Program

After the RZ/T1 is released from a reset, the level of the P44 pin (SW3 on the evaluation board) determines the program to be started, either a user program or the USB serial writing sample program.

Table 1.4 Pin Levels and Programs to be Started

P44 Pin Level	Program
Low (SW3 being pressed)	USB serial writing sample program
High	User program

LED10 on the evaluation board is lit up when the USB serial writing sample program is started.

1.4 Selecting Power Supply

The evaluation board is equipped with two power supply jumpers, JP2 and JP7.

Use them with the settings shaded in gray in the list below.

Table 1.5 JP2 and JP7 Settings

Jumper	Setting	Function
JP2 Power for the system is selected	1-2	The power supply is 7 to 12 V.
	2-3	The power supply is 5 V.
JP7 Source for VCCQ33B is selected	1-2	RZ/T1 digital 3.3 V power is supplied.
	2-3	RZ/T1 digital 1.2 V power is supplied.

Make the settings of jumpers before supplying power to the board.

2. Related Documents

The documents related to this application note are listed below for reference.

- RZ/T1 Group Application Note: Initial Settings of the Microcomputers Incorporating the R-IN Engine
  Document Number: R01AN2989EJxxxx
- RZ/T1 Group Application Note: USB Peripheral Communications Device Class Driver
  Document Number: R01AN2631EJxxxx
- RZ/T1 Group User's Manual: Hardware
  Document Number: R01UH0483EJxxxx
- RZ/T1 Group Application Note: Serial Flash Sample Program
  Document Number: R01AN3010EJxxxx
- Renesas Starter Kit+ for RZ/T1 User's Manual
  Document Number: R20UT3242EGxxxx

xxx: Revision

3. Peripheral Modules

See the RZ/T1 Group User's Manual: Hardware for basic descriptions for clock generator (CPG), interrupt controller (ICUA), error control module (ECM), extended internal RAM, general-purpose I/O ports, and USB 2.0 HS function module (USBf).

4. Hardware

Figure 4.1 shows an example of the hardware configuration.

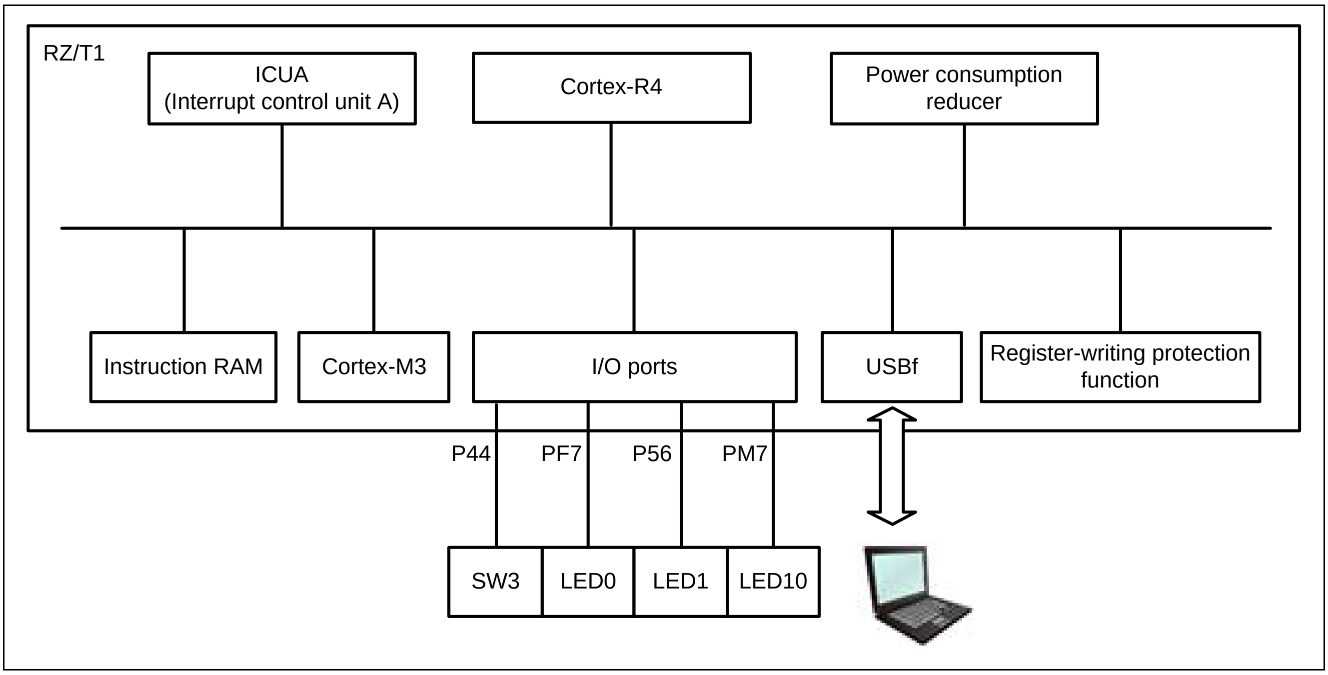


Figure 4.1 Hardware Configuration

5. Software

This section describes the case of EWARM from IAR systems unless otherwise stated.

5.1 Loader Program Operation Overview

After the RZ/T1 is booted after a reset, the loader program stored in the external serial flash ROM is copied into the internal RAM (BTCM).

After the boot processing, the loader program judges the device to have been reset, sets up the clock and bus, and copies the user program or the USB serial writing sample program from the serial flash ROM to the internal RAM (ATCM). When copying the user program to the internal RAM (ATCM), the loader program refers the user program information table in the serial flash ROM and copies to the internal RAM (ATCM). In this sample program, the user program information table is stored at address: 0x300A0000. Table 5.1 shows the user program information table.

Then, the loader program sets the MPU and cache, switches the exception vector to the low vector state, and branches to the point where the program copied into the internal RAM (ATCM) starts.

Table 5.1 User Program Information Table

No	Address(Offset)	Description
1	0x00	Start address of the user program area in the serial flash ROM
2	0x04	Start address of the user program area in the ATCM
3	0x08	End address of the user program area in the ATCM
4	0x0C	Start address of the user program area for Cortex-M3 in the serial flash ROM
5	0x10	Start address of the user program area for Cortex-M3 in the Instruction RAM
6	0x14	Size of the user program area for Cortex-M3 in the Instruction RAM (In case of e^2 studio, end address of the user program area for Cortex-M3 in the Instruction RAM)
7	0x18	Start address of the user program variable area in the serial flash ROM
8	0x1C	Start address of the user program variable area in the ATCM
9	0x20	End address of the user program variable area in the ATCM
10	0x24	Dummy
11	0x28	Start address of the bss area
12	0x2C	End address of the bss area

Figure 5.1 shows the operation overview after boot processing.

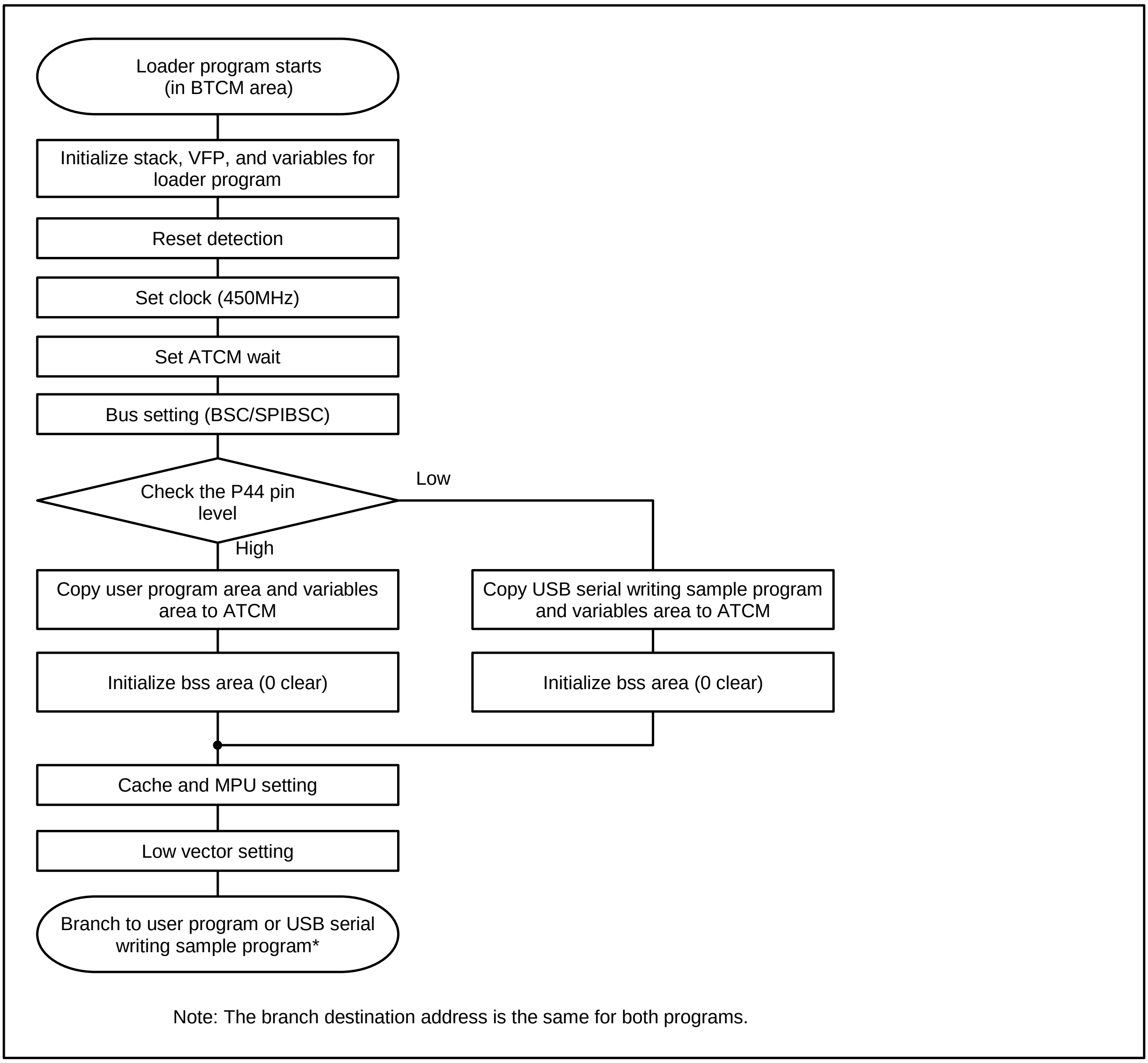


Figure 5.1 Operation Overview after Boot Processing

5.1.1 User program Information Table

The user program information tables for EWARM and e[2] studio are given below.

■EWARM

No	Address(Offset)	Description	Member	Value
1	0x00	Start address of the user program area in the serial flash ROM	user_prg_s_r_saddr	__section_begin ("USER_PRG_RBLOCK")
2	0x04	Start address of the user program area in the ATCM	user_prg_d_r_saddr	__section_begin ("USER_PRG_WBLOCK")
3	0x08	End address of the user program area in the ATCM	user_prg_d_r_eaddr	__section_end ("USER_PRG_WBLOCK")
4	0x0C	Start address of the user program area for Cortex-M3 in the serial flash ROM	cm3_s_saddr	__section_begin ("CM3_SECTION ")
5	0x10	Start address of the user program area for Cortex-M3 in the Instruction RAM	cm3_d_saddr	__section_begin ("CM3_SECTION_WBLOCK ")
6	0x14	Size of the user program area for Cortex-M3 in the Instruction RAM	cm3_d_size	__section_size ("CM3_SECTION")
7	0x18	Start address of the user program variable area in the serial flash ROM	user_prg_s_w_saddr	__section_begin ("USER_DATA_RBLOCK")
8	0x1C	Start address of the user program variable area in the ATCM	user_prg_d_w_saddr	__section_begin ("USER_DATA_WBLOCK")
9	0x20	End address of the user program variable area in the ATCM	user_prg_d_w_eaddr	__section_end ("USER_DATA_WBLOCK")
10	0x24	Dummy	dummy_addr	0x00000000
11	0x28	Start address of the bss area	user_prg_d_saddr	__section_begin ("USER_DATA_ZBLOCK")
12	0x2C	End address of the bss area	user_prg_d_eaddr	__section_end ("USER_DATA_ZBLOCK")

The user program information table structure is declared in user_prg_info_tbl.h and the user program information table is declared in loader_init2.c. The values of user program information table uses the section of the linker configuration file (\RZ_T1_R-IN_init\Cortex-R4\RZ_T1_init_boot\src\common\serial_boot\RZ_T1_init_serial_boot.icf) and are saved the address information of user program information table during build.

【User Program Information Table Structure (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;
```

【User Program Information Table (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 (user_prog_info_tbl.asm)

No	Address(Offset)	Description	Value
1	0x00	Start address of the user program area in the serial flash ROM	_main_text
2	0x04	Start address of the user program area in the ATCM	_main_text_start
3	0x08	End address of the user program area in the ATCM	_text_end
4	0x0C	Start address of the user program area for Cortex-M3 in the serial flash ROM	_cm3
5	0x10	Start address of the user program area for Cortex-M3 in the Instruction RAM	_cm3_start
6	0x14	End address of the user program area for Cortex-M3 in the Instruction RAM	_cm3_end
7	0x18	Start address of the user program variable area in the serial flash ROM	_mdata
8	0x1C	Start address of the user program variable area in the ATCM	_data_start
9	0x20	End address of the user program variable area in the ATCM	_data_end
10	0x24	Dummy	0x00000000
11	0x28	Start address of the bss area	__bss_start__
12	0x2C	End address of the bss area	__bss_end__

The user program information table is declared in user_prog_info_tbl.asm. The values of user program information table uses the section of the linker scription file (\RZ_T1_R-IN_init_sflash\sample_cr4\src\linker_scriptHardwareDebug.ld) and are saved the address information of user program information table during build.

【User Program Information Table (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 Memory Maps

The address space of the RZ/T1 group is described in the RZ/T1 Group User's Manual: Hardware.

5.3 Section Assignment for the Sample Program

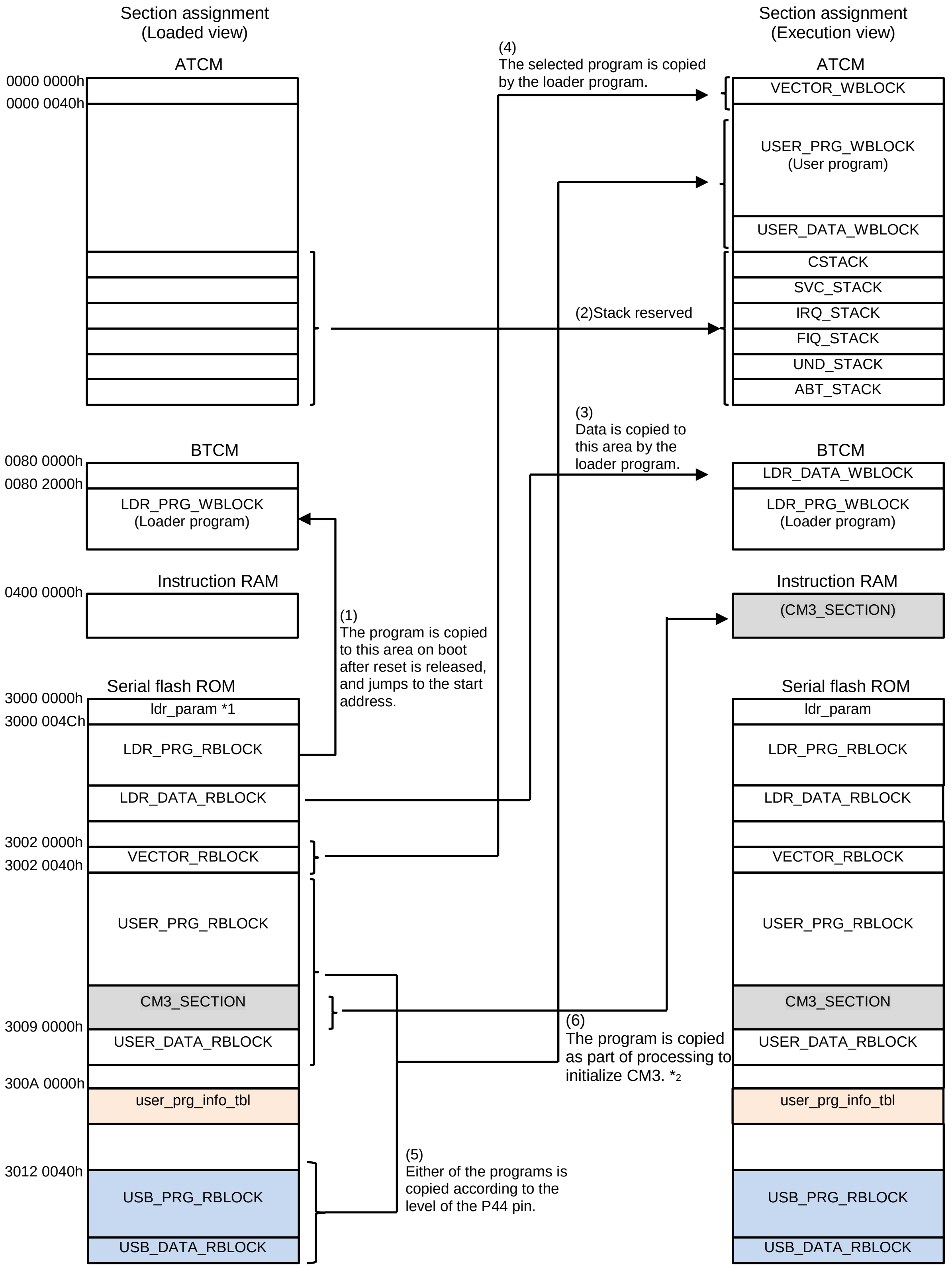
Table 5.2 and Table 5.3 list the sections for use with the Cortex-R4 and M3 cores, respectively. Figure 5.2 and Figure 5.3 show section assignments for the Cortex-R4 and M3 cores, respectively.

Table 5.2 Sections for Use with Cortex-R4

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_WBLOCK	Reset and exception processing vector table	Code	—	ATCM
USER_PRG_WBLOCK	User program area (for execution)	Code	—	ATCM
USER_DATA_WBLOCK	User program variable area (for execution)	Data	—	ATCM
CSTACK	Stack area	Data	—	ATCM
SVC_STACK	Supervisor (SVC) mode stack area	Data	—	ATCM
IRQ_STACK	IRQ mode stack area	Data	—	ATCM
FIQ_STACK	FIQ mode stack area	Data	—	ATCM
UND_STACK	Undefined (UND) mode stack area	Data	—	ATCM
ABT_STACK	Abort (ABT) mode stack area	Data	—	ATCM
LDR_DATA_WBLOCK	Loader program variable area (for execution)	Data	—	BTCM
LDR_PRG_WBLOCK	Loader program area (for execution)	Code	—	BTCM
ldr_param	Loader parameters	Data	FLASH	—
LDR_PRG_RBLOCK	Loader program area (for storing)	Code	FLASH	—
LDR_DATA_RBLOCK	Loader program variable area (for storing)	Data	FLASH	—
VECTOR_RBLOCK	Reset and exception processing vector table area (for storing)	Code	FLASH	—
USER_PRG_RBLOCK	User program area (for storing) User program area for Cortex-M3 (for storing)	Code	FLASH	—
USER_DATA_RBLOCK	User program variable area (for storing)	Data	FLASH	—
user_prg_info_tbl	User Program Information Table	Data	FLASH	—
USB_PRG_RBLOCK	USB serial writing sample program area (for storing)	Code	FLASH	—
USB_DATA_RBLOCK	USB serial writing sample program variable area (for storing)	Data	FLASH	—

Table 5.3 Sections for Use with Cortex-M3

Area Name	Description	Type	Loading Area	Execution Area
vectors	Vector area	Code	—	Instruction RAM
readonly	User program area	Code	—	Instruction RAM
_SHARED_MEM	Shared memory area	Data	—	Data RAM
readwrite	User program variable area	Data	—	Data RAM
HEAP	Heap area	Data	—	Data RAM
CSTACK	Stack area	Data	—	Data RAM



Notes 1. The loader parameters in ldr_param section are used in boot processing. For details, see the RZ/T1 Group User's Manual: Hardware.

2. For details on the CM3 initialization processing, see the RZ/T1 Group Application Note: Initial Settings of the Microcomputers Incorporating the R-IN Engine.

Figure 5.2 Section Assignment for Cortex-R4

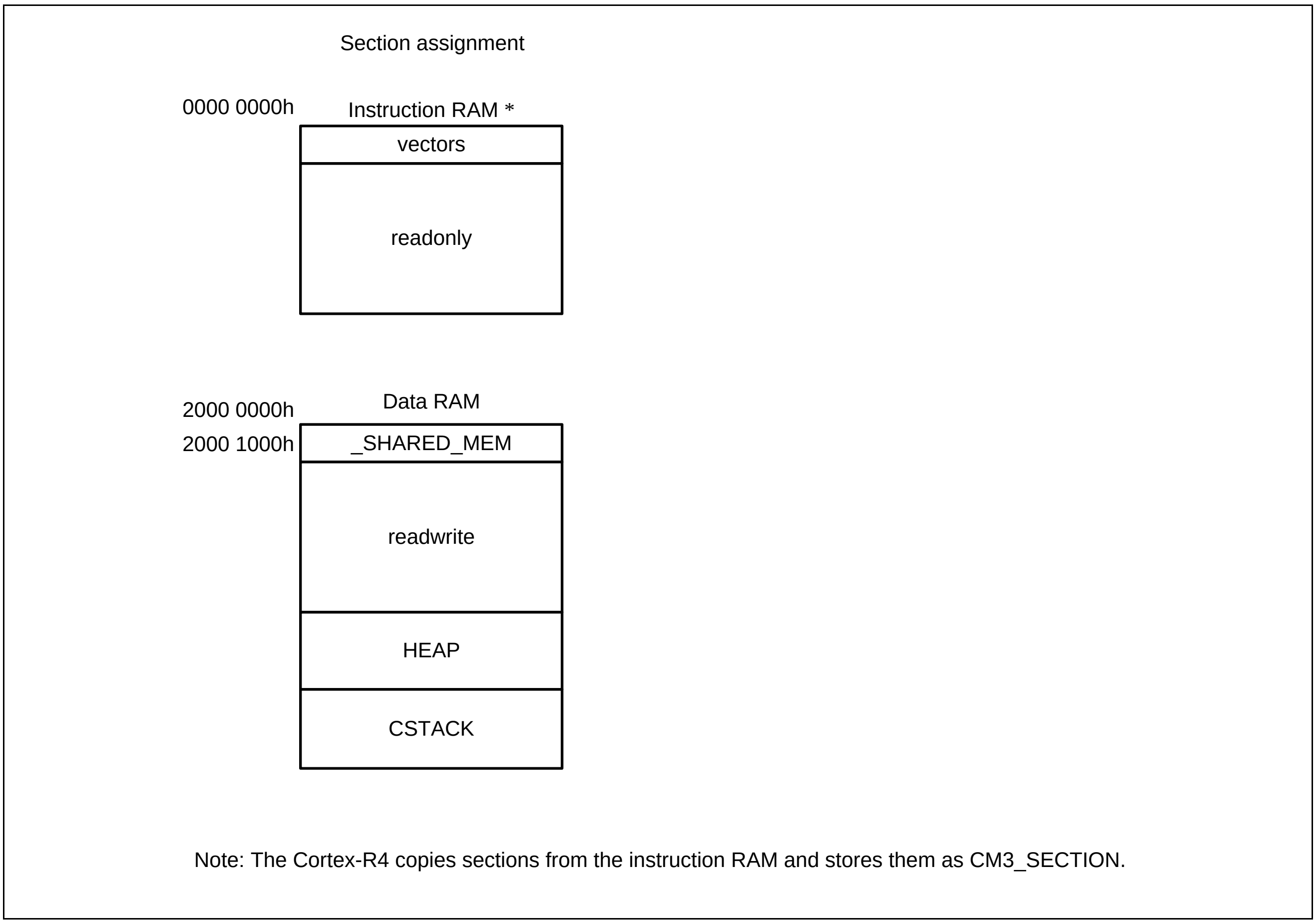


Figure 5.3 Section Assignment for Cortex-M3

5.4 USB Serial Writing Sample Program

5.4.1 Overview

The program runs the initial settings of the USB and waits for an input (any data) from the terminal software for a handshake with the host PC. After an input from the terminal software, the program handles the operation according to the command from the host PC.

The USB serial writing sample program handles the following operations.

1. Writing a user program to the serial flash ROM area
2. Reading data from the serial flash ROM area
3. Sector erase
4. Protection control

Figure 5.4 shows an overview of the USB serial writing sample program.

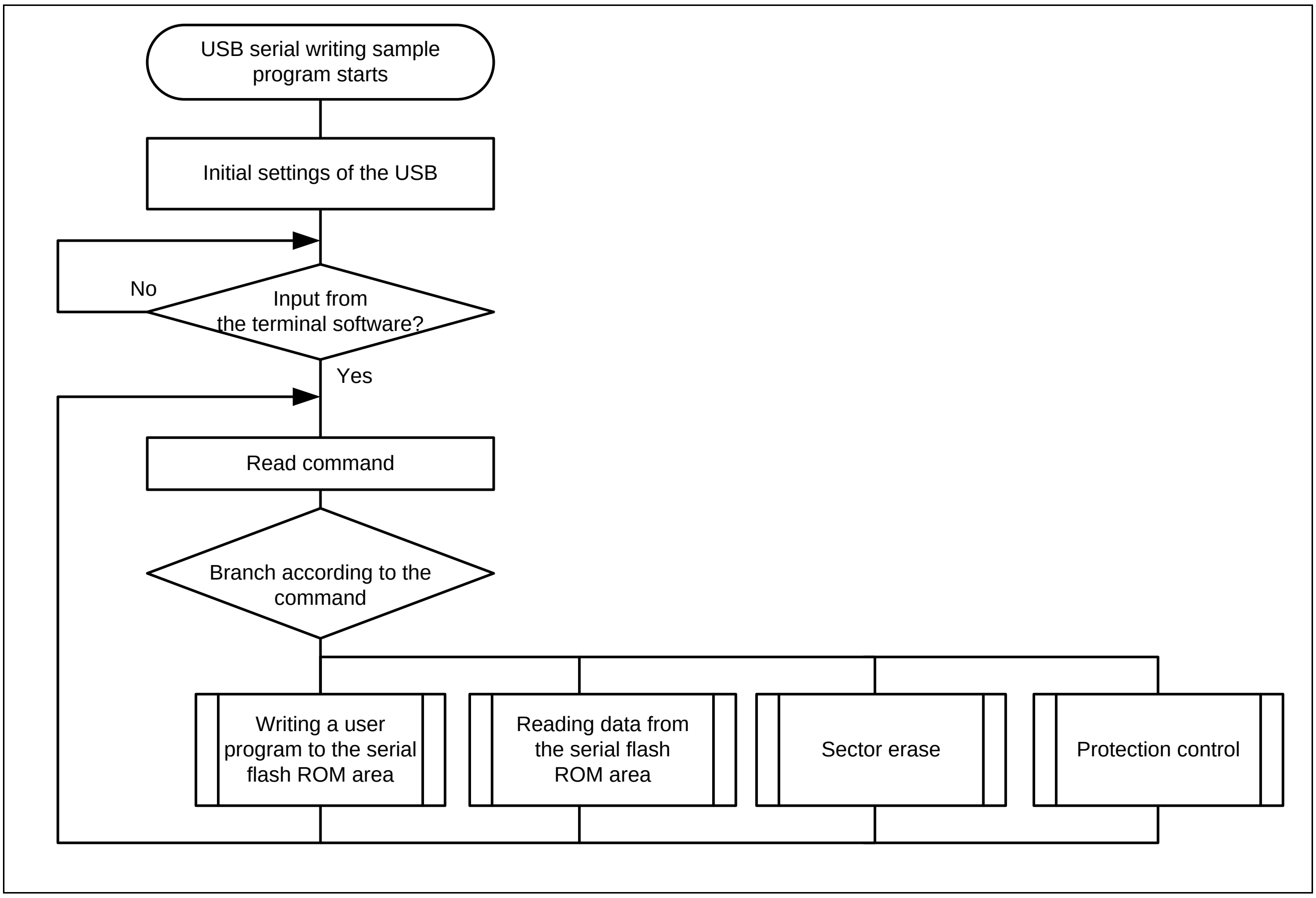


Figure 5.4 Overview of the USB Serial Writing Sample Program

5.4.2 Software Configuration

Figure 5.5 shows the software configuration of the USB serial writing sample program.

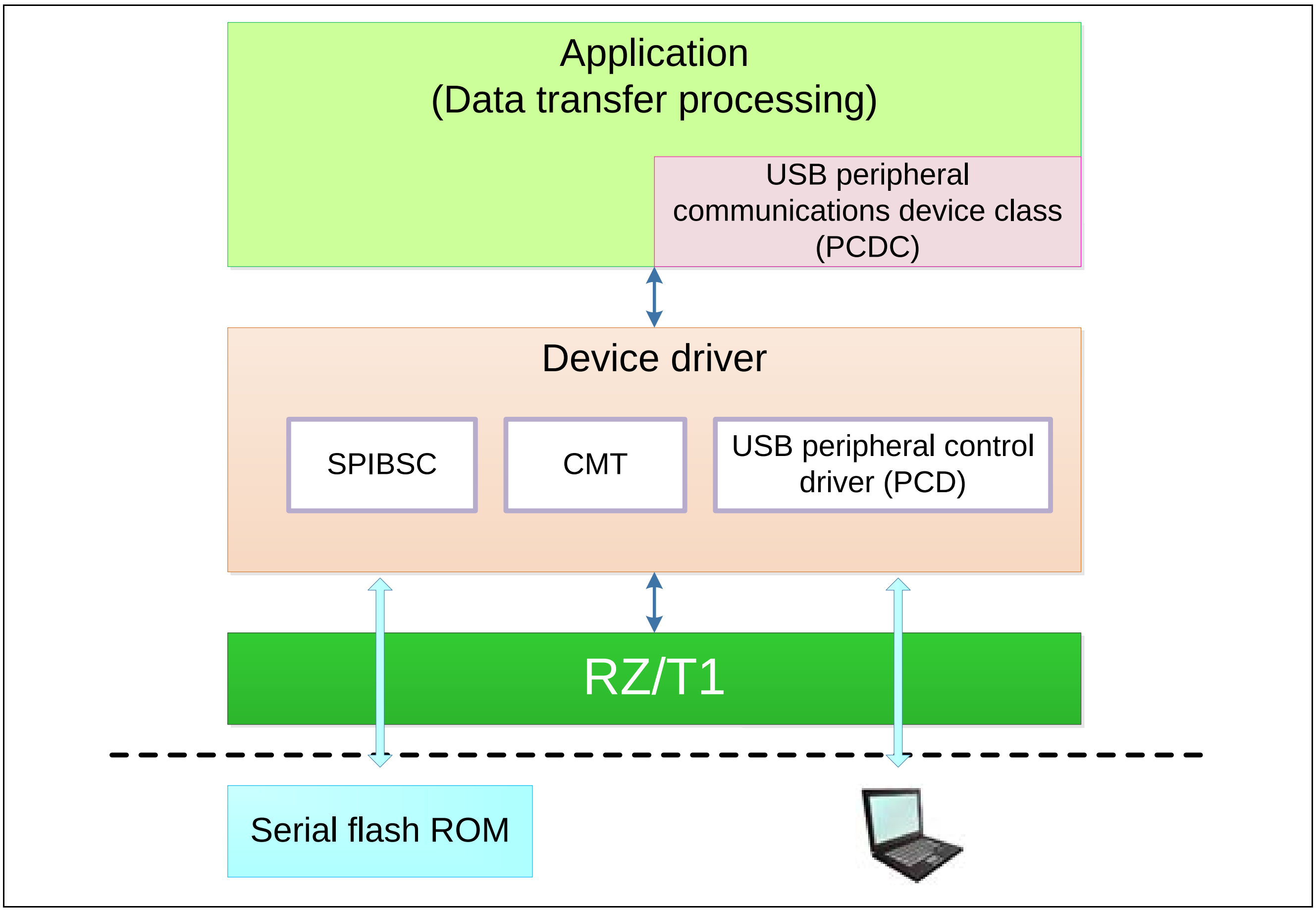


Figure 5.5 Software Configuration

5.4.3 Software Description

(1) List of Constants

Table 5.4 lists constants for use in the USB serial writing sample program.

Table 5.4 List of Constants

Constant	Setting Value	Description
CDC_DATA_LEN	1024	Length of data for transfer via the USB
EVENT_MAX	5	Maximum number of events
TASK_LOOPS_BETWEEN_INSTRUCTIONS	0x5000000	Address where the initial connection message is stored
USB_CDC_DATA_INVALID	0x00	USB reception buffer does not hold data
USB_CDC_DATA_VALID	0x01	USB reception buffer holds data
ALIGN_SIZE	0x20	Alignment size
CDC_REV_DATA_MAX	CDC_DATA_LEN	Maximum data length receivable via the USB
CDC_REV_BUF_DATA_MAX	1024+512	Maximum size of the USB reception buffer
R_SFALSH_WRITE_SZ	0x400	Data size to be written to the serial flash ROM
R_USB_SEND_WAIT	59u	Wait time after transfer via the USB
R_SFALSH_SEND_SZ	511u	Maximum number of data for transfer via the USB
SPIBSC_MIN_ADDR	0x00000000	SPIBSC start address (Address 0x30000000 is treated as 0x00000000.)
SPIBSC_MAX_ADDR	0x03FFFFFF	SPIBSC end address (Address 0x30000000 is treated as 0x00000000.)
PRCR_ACCESS_UNLOCK	0x0000A503	Enable writing to the MSTPCR register
PRCR_ACCESS_LOCK	0x0000A500	Disable writing to the MSTPCR register
LENGTH_PROTECT_CODE	4	SPIBSC protection code buffer size
COMMAND_PROTECT_CONTROL	2	SPIBSC protection control command
COMMAND_WRITE	3	SPIBSC write command
COMMAND_READ	4	SPIBSC read command
COMMAND_SECTOR_ERASE	6	SPIBSC sector erase command
COMMAND_ERROR	11	Command error
SCIF_CMD_PROTECT_LOWER	"p"	Protection control command (lower case)
SCIF_CMD_PROTECT_UPPER	"P"	Protection control command (upper case)
SCIF_CMD_WRITE_LOWER	"w"	Write command (lower case)
SCIF_CMD_WRITE_UPPER	"W"	Write command (upper case)
SCIF_CMD_READ_LOWER	"r"	Read command (lower case)
SCIF_CMD_READ_UPPER	"R"	Read command (upper case)
SCIF_CMD_SECTOR_ERASE_LOWER	"e"	Sector erase command (lower case)
SCIF_CMD_SECTOR_ERASE_LOWER	"E"	Sector erase command (upper case)
SCIF_COLUMN_LEN_ERR_CODE	2	Number of digits in error codes
SCIF_LEN_ERR_CODE_STRING	6	Error code string length
SCIF_LEN_CMD	1	Command string length
SCIF_MAX_LEN_COMMAND	115	Maximum length of the command string to be received
SCIF_LEN_CMD_SPACE	1	Length of sequences of spaces in commands
SCIF_LEN_RETURN_CODE	2	Newline length (CR+LF)
SCIF_LEN_RETURN_CODE_HALF	1	Length of either the carriage return (CR) or the linefeed (LF) code

Constant	Setting Value	Description
SCIF_LEN_PREFIX_HEX	2	Length of prefix (0x) in a hexadecimal string
SCIF_MULTIPLIER_2_TO_16	4	Power of 2 to produce 16
SCIF_CLEAR_UPPER_COLUMN_10	10	Complement for use in the conversion of strings longer than 0x0A
SCIF_LEN_1BYTE_DATA	4	String length of 1-byte data (0xXX)
SCIF_DIGIT_1BYTE_DATA	2	Number of digits for 1-byte data
SCIF_MAX_LEN_32BIT_STRING	10	Maximum length of strings (prefixed by 0x) of 32-bit data
SCIF_CHANGE_VALUE_UPPER CASE	0x37	Value for use in the conversion of upper-case characters to numeric values
SCIF_CHANGE_VALUE_LOWER CASE	0x57	Value for use in the conversion of lower-case characters to numeric values
CMD_DEBUG_MODE_LOWER	"d"	Command to indicate the start-of-command character (lower case)
CMD_DEBUG_MODE_UPPER	"D"	Command to indicate the start-of-command character (upper case)
CMTW_TIMEOUT_REC_DATA	1464843	Timeout period for data reception (10 s)
CMTW_TIMEOUT_MESSAGE	1464843	Timeout period for data transmission (10 s)
CMTW0_CMWI0_NUM	25	Interrupt vector number assigned to the CMTW
ERROR_CODE_SUCCESS	0x00	Normal end
ERROR_CODE_FILE_TRANSFER	0xFF	File transfer failed
ERROR_CODE_PARAM_ERROR	0xFE	Abnormal parameter
ERROR_CODE_VERIFY	0xFC	Verification of the written data failed
ERROR_CODE_NO_CORRESPOND	0xFB	A non-supported command was received
ERROR_CODE_TIMEOUT	0xFA	Timeout error occurred
ERROR_CODE_HW_ERROR	0xF8	HW error occurred
INTERNAL_ERROR_SUCCESS	0	Normal end (return value for internal use by the program)
INTERNAL_ERROR_SCIF_ERROR	-1	HW error occurred (return value for internal use by the program)
INTERNAL_ERROR_SCIF_TIMEOUT	-2	Timeout error occurred (return value for internal use by the program)

(2) Structures/Unions/Enumerated Types

The structures, unions, and enumerated types used in the USB serial writing sample program are listed in the tables below.

Table 5.5 Structure DEV_INFO_t

Member Name	Content
uint16_t state	State of the application
uint16_t event_cnt	Event count
uint16_t event[EVENT_MAX]	Event number

Table 5.6 Enumerated Type STATE_t

Member Name	Content
STATE_ATTACH	Attach processing
STATE_DATA_TRANSFER	Data transfer processing
STATE_DETACH	Detach processing

Table 5.7 Enumerated Type EVENT_t

Member Name	Content
EVENT_NONE	No event
EVENT_CONFIGERD	USB device has been connected
EVENT_USB_READ_START	Request data reception via the USB
EVENT_USB_READ_COMPLETE	Data has been received via the USB
EVENT_USB_WRITE_START	Request data transmission via the USB
EVENT_USB_WRITE_COMPLETE	Data has been transmitted via the USB
EVENT_COM_NOTIFY_START	Request transmission of the class notification "SerialState"
EVENT_COM_NOTIFY_COMPLETE	Class notification "SerialState" has been transmitted

Table 5.8 Enumerated Type USB_PCDC_APL_STATE

Member Name	Content
APP_STATE_IDLE	IDLE state
APP_STATE_ECHO_MODE	ECO mode

Table 5.9 Structure bootloader_ctrl_t

Member Name		Content
uint8_t	cmd	Command to be executed
uint32_t	timeout_err_flag	State in terms of timeout errors. This member being true indicates that an error has occurred.
uint8_t	*target_buf	Pointer to the buffer subject to execution of the command
uint8_t	*src_buf	Pointer to the buffer that contains the source data for copying
uint32_t	target_size	Amount of data to be handled by the command
uint8_t	*protect_code	Pointer to the protection code area
uint32_t	protect_code_size	Size of the protection code area

(3) List of Global Variables

Table 5.10 lists the global variables.

Table 5.10 Global Variables List

Type	Variable Name	Content
static uint8_t	gb_rec_buf [R_SFALSH_WRITE_SZ]	Buffer in which data from the serial flash ROM are stored
static uint8_t	gb_spibsc_protect_code [LENGTH_PROTECT_CODE]	Buffer in which the SPIBSC protection code is stored
static uint32_t	g_cmtw_start_flag	CMTW overlapping start checking flag
static bootloader_ctrl_t	bootloader_param	Control parameters for the USB serial writing sample program
static volatile uint8_t	usbf_wfin_flag	Flag indicating completion of transmission via the USB
static uint8_t	cmd_enter_mode_en	Start-of-command character flag: This flag being true indicates that the start-of-command character is in use and being false indicates that it is not in use.
static uint8_t	receive_timeout_en	Flag indicating that timeout for data reception via the USB is enabled.
uint8_t	cdc_trans_data_base [CDC_DATA_LEN + ALIGN_SIZE]	Buffer for use in data transfer via the USB
uint8_t*	cdc_trans_data	Pointer to the buffer for use in data transfer via the USB
static uint8_t	cdc_rev_data [CDC_REV_BUF_DATA_MAX]	Buffer for the data received via the USB
static uint32_t	cdc_rev_data_pw	Point in the buffer where the data received via the USB are to be written to
static uint32_t	cdc_rev_data_pr	Point in the buffer where the data received via the USB are to be read from

(4) List of Functions

Table 5.11 lists functions.

Table 5.11 List of Functions

Function Name	Description	Scope	Definition File
main	Main processing of the user program	global	main.c
port_init	Port setting	global	main.c
ecm_init	ECM initial setting	global	main.c
icu_init	Interrupt setting	global	main.c
usbf_main	Main processing for the USB peripherals	global	r_usb_pcdc_apl.c
cdc_connect_wait *1	Waits for a connection via the USB	global	r_usb_pcdc_apl.c
cdc_detach_device *1	Detaching	global	r_usb_pcdc_apl.c
cdc_configured *1	Callback for the device configured processing	global	r_usb_pcdc_apl.c
cdc_detach *1	Callback for detach processing	global	r_usb_pcdc_apl.c
cdc_default *1	Callback for default processing	global	r_usb_pcdc_apl.c
cdc_suspend *1	Callback for suspension processing	global	r_usb_pcdc_apl.c
cdc_resume *1	Callback for resume processing	global	r_usb_pcdc_apl.c
cdc_interface *1	Callback for interface processing	global	r_usb_pcdc_apl.c
cdc_registration *1	Register the device driver	global	r_usb_pcdc_apl.c
apl_init *1	Initialization during the main processing of the USB peripheral facility	global	r_usb_pcdc_apl.c
cdc_event_set *1	Issues an event	global	r_usb_pcdc_apl.c
cdc_event_get *1	Gets an event	global	r_usb_pcdc_apl.c
r_data_trans_sFlash_writing	Main processing for data transfer and writing to the serial flash ROM	global	r_usb_data_trans_sFlash_writing.c
get_boot_param_pointer	Gets the pointer to the control parameters for the USB serial writing sample program	global	r_usb_data_trans_sFlash_writing.c
scif_putc	Sends characters via the USB	global	r_usb_data_trans_sFlash_writing.c
scif_getc	Receives characters via the USB	static	r_usb_data_trans_sFlash_writing.c
scif_puts	Sends strings via the USB	static	r_usb_data_trans_sFlash_writing.c
spibsc_set_protect_code	Sends the SPIBSC protection control code	static	r_usb_data_trans_sFlash_writing.c
spibsc_verify_data	Verifies the data written to the SPIBSC	static	r_usb_data_trans_sFlash_writing.c
scif_get_cmd	Gets a command	static	r_usb_data_trans_sFlash_writing.c
check_protect_control_command	Checks the format of the protection control command	static	r_usb_data_trans_sFlash_writing.c
check_write_command	Checks the format of the write command	static	r_usb_data_trans_sFlash_writing.c
check_read_command	Checks the format of the read command	static	r_usb_data_trans_sFlash_writing.c
check_sector_erase_command	Checks the format of the sector erase command	static	r_usb_data_trans_sFlash_writing.c
change_string_to_val	Converts a string to 1-byte unit numerical values	static	r_usb_data_trans_sFlash_writing.c

Function Name	Description	Scope	Definition File
change_errcode_to_string	Converts a numerical value to hexadecimal strings	static	r_usb_data_trans_sFlash_writing.c
spibsc_init	Initializes the SPIBSC	static	r_usb_data_trans_sFlash_writing.c
r_cdc_read_complete	Callback for completion of data reception via the USB	static	r_usb_data_trans_sFlash_writing.c
r_cdc_write_complete	Callback for completion of data transmission via the USB	static	r_usb_data_trans_sFlash_writing.c
r_cdc_rev_data_is_valid	Checks if data exists in the USB reception buffer	static	r_usb_data_trans_sFlash_writing.c
r_cdc_rev_data_clear	Clears the USB reception buffer	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_write_data	Writes data to the USB reception buffer	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_rev_data	Reads data from the USB reception buffer	static	r_usb_data_trans_sFlash_writing.c
r_get_cdc_rev_1Bdata	Reads 1 byte of data from the USB reception buffer	static	r_usb_data_trans_sFlash_writing.c
r_cdc_start	Starts communications via the USB with the CDC protocol	static	r_usb_data_trans_sFlash_writing.c
r_usb_write_data_to_sFlash	Receives data and writes them to the serial flash ROM	static	r_usb_data_trans_sFlash_writing.c
r_usb_read_data_from_sFlash	Reads data from the serial flash ROM and sends them	static	r_usb_data_trans_sFlash_writing.c
r_receive_data	Receives data	static	r_usb_data_trans_sFlash_writing.c
r_write_sFlash	Writes data to the serial flash ROM	static	r_usb_data_trans_sFlash_writing.c
r_trans_data_payload	Sends data	static	r_usb_data_trans_sFlash_writing.c

Note 1. See the RZ/T1 Group Application Note: USB Peripheral Communications Device Class Driver, for details of the functions.

(5) main

main	
Synopsis	Main processing of the user program
Declaration	void main(void)
Description	See the flowchart given below.
Argument	None
Returned value	None
Remark	This function is mapped to address: 0x00000040.

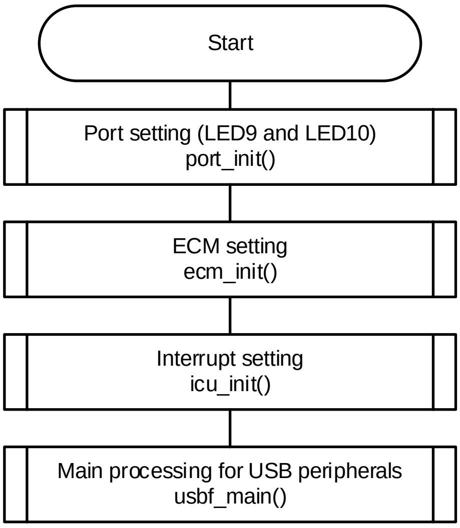


Figure 5.6 Main Processing (Cortex-R4)

(6) port_init

port_init	
Synopsis	Port setting
Declaration	void port_init(void)
Description	This function sets port pins PM7 and PM6 to output mode, and for output of the low level.
Argument	None
Returned value	None
Remark	

(7) ecm_init

ecm_init	
Synopsis	ECM initial setting
Declaration	void ecm_init(void)
Description	This function initializes the ECM.
Argument	None
Returned value	None
Remark	

(8) icu_init

icu_init	
Synopsis	Interrupt setting
Declaration	void icu_init(void)
Description	This function enables interrupts.
Argument	None
Returned value	None
Remark	

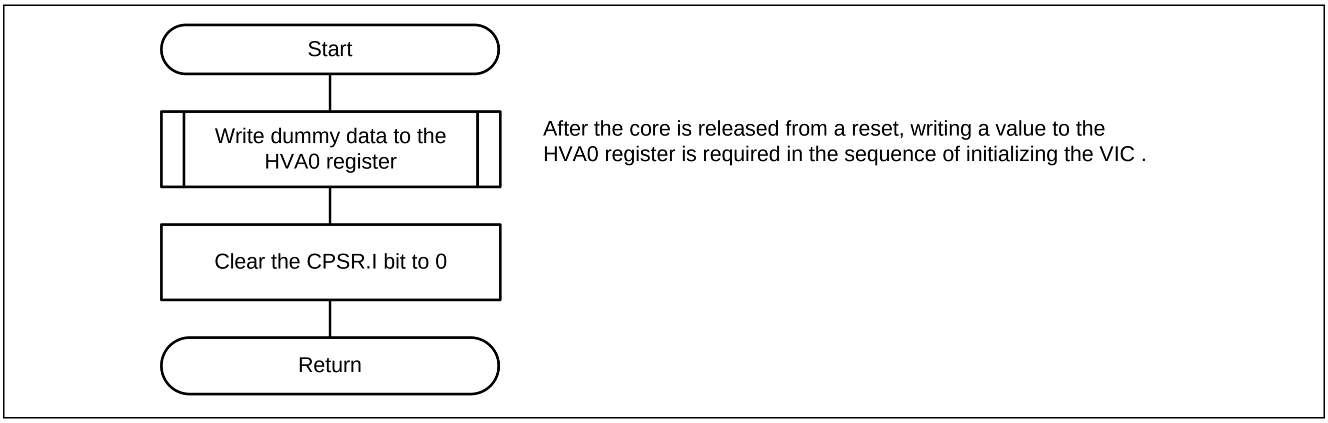


Figure 5.7 icu_init Function Processing

(9) usbf_main

usbf_main	
Synopsis	Main processing for the USB peripherals
Declaration	void usbf_main(void)
Description	This is the main processing of the sample program. A flowchart is given below. Operation of the program is based on the states of communications and related events. More specifically, the program checks events related to the states of communications and performs required processing in response. After processing an event, the program changes to the relevant state if this is required. • STATE_ATTACH: Processing for attachment • STATE_DATA_TRANSFER: Processing of data transfer and writing to the serial flash ROM • STATE_DETACH: Processing for detachment
Argument	None
Returned value	None
Remark	The main processing for data transfer and writing to the serial flash ROM is an infinite loop and does not return to this function.

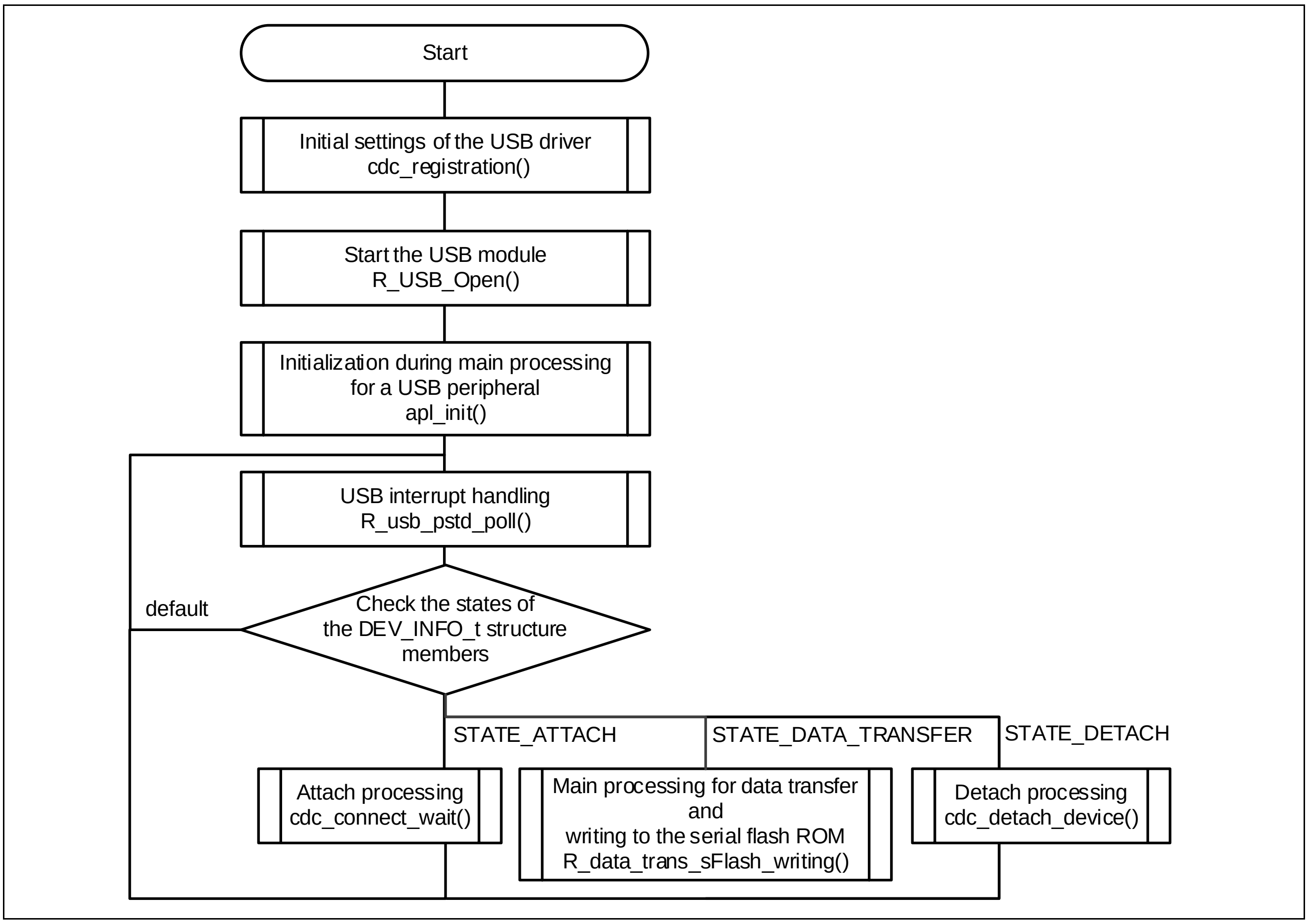


Figure 5.8 usbf_main Function Processing

(10) r_data_trans_sFlash_writing

r_data_trans_sFlash_writing	
Synopsis	Main processing for data transfer and writing to the serial flash ROM
Declaration	void r_data_trans_sFlash_writing(void)
Description	See the flowcharts given below.
Argument	None
Returned value	None
Remark	

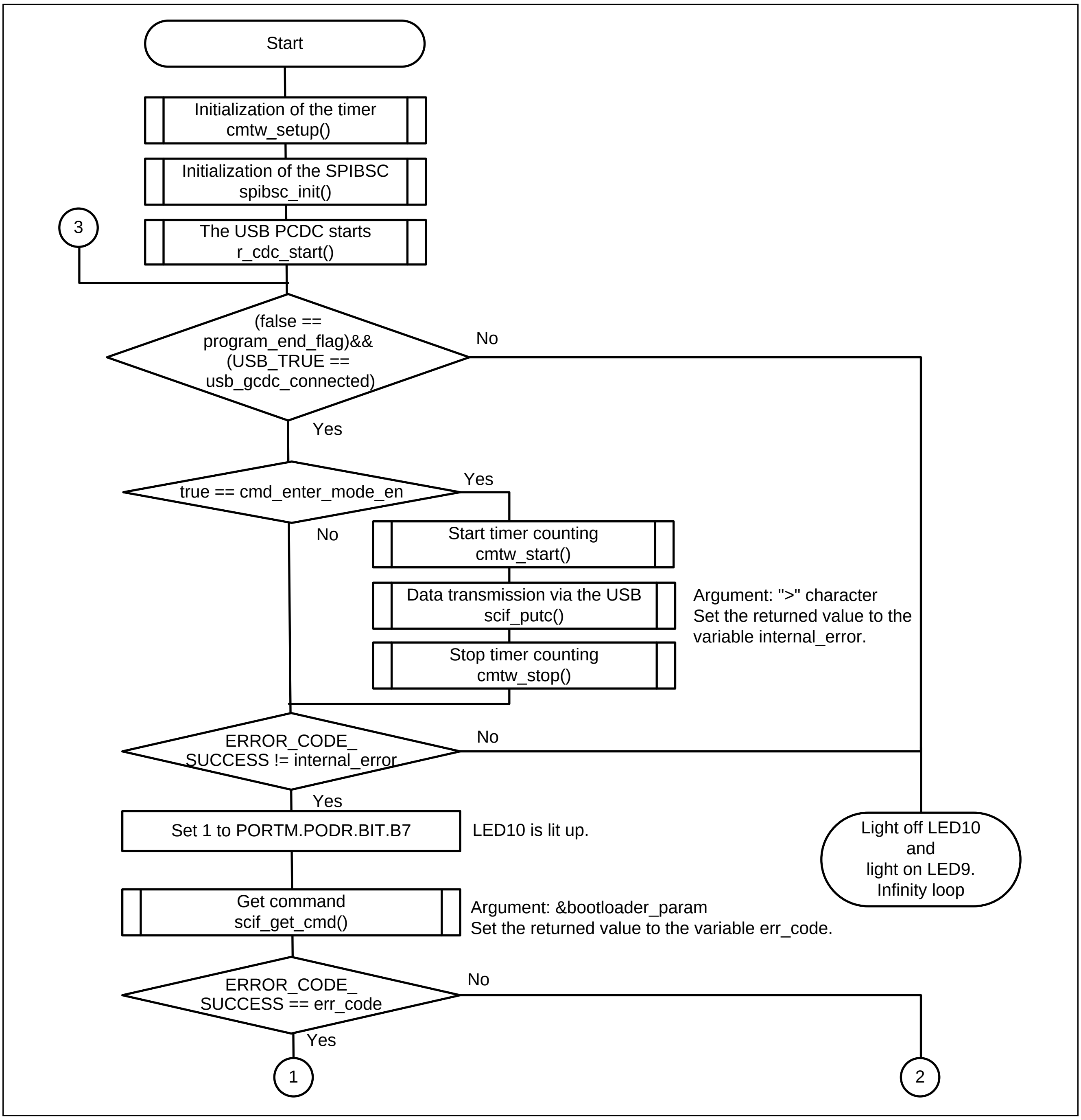


Figure 5.9 r_data_trans_sFlash_writing Function Processing (1/2)

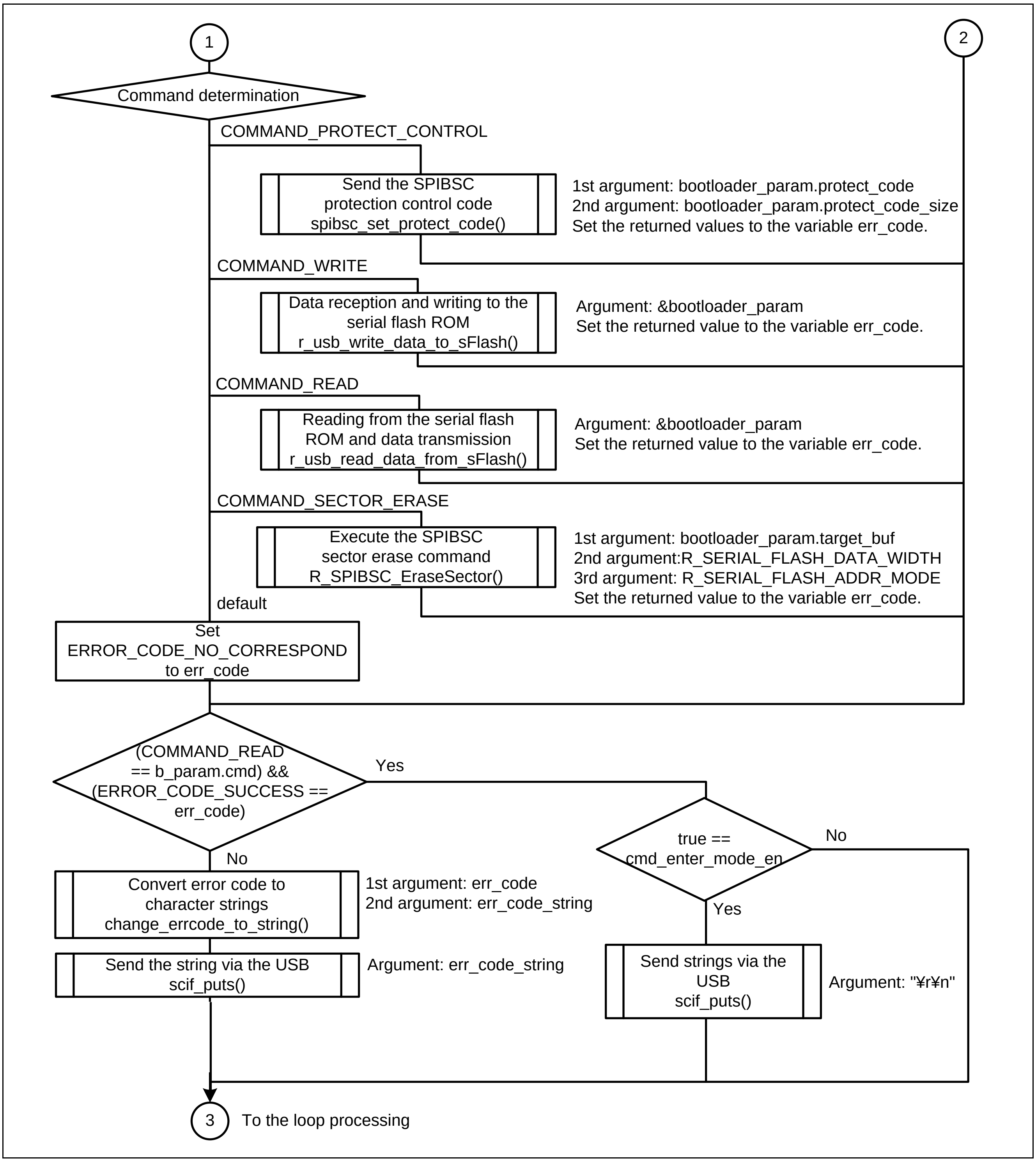


Figure 5.9 r_data_trans_sFlash_writing Function Processing (2/2)

(11) get_boot_param_pointer

get_boot_param_pointer	
Synopsis	Gets the pointer to the control parameters for the USB serial writing sample program.
Declaration	bootloader_ctrl_t *get_boot_param_pointer(void)
Description	This function passes the pointer to the variable bootloader_param.
Argument	None
Returned value	Pointer to the variable bootloader_param
Remark	

(12) scif_putc

scif_putc	
Synopsis	Sends characters via the USB.
Declaration	int32_t scif_putc(uint8_t cha)
Description	This function sends characters via the USB.
Argument	uint8_t cha Character
Returned value	INTERNAL_ERROR_SUCCESS: Normal end INTERNAL_ERROR_SCIF_TIMEOUT: A timeout error occurred. INTERNAL_ERROR_SCIF_ERROR: The USB device is detached.
Remark	

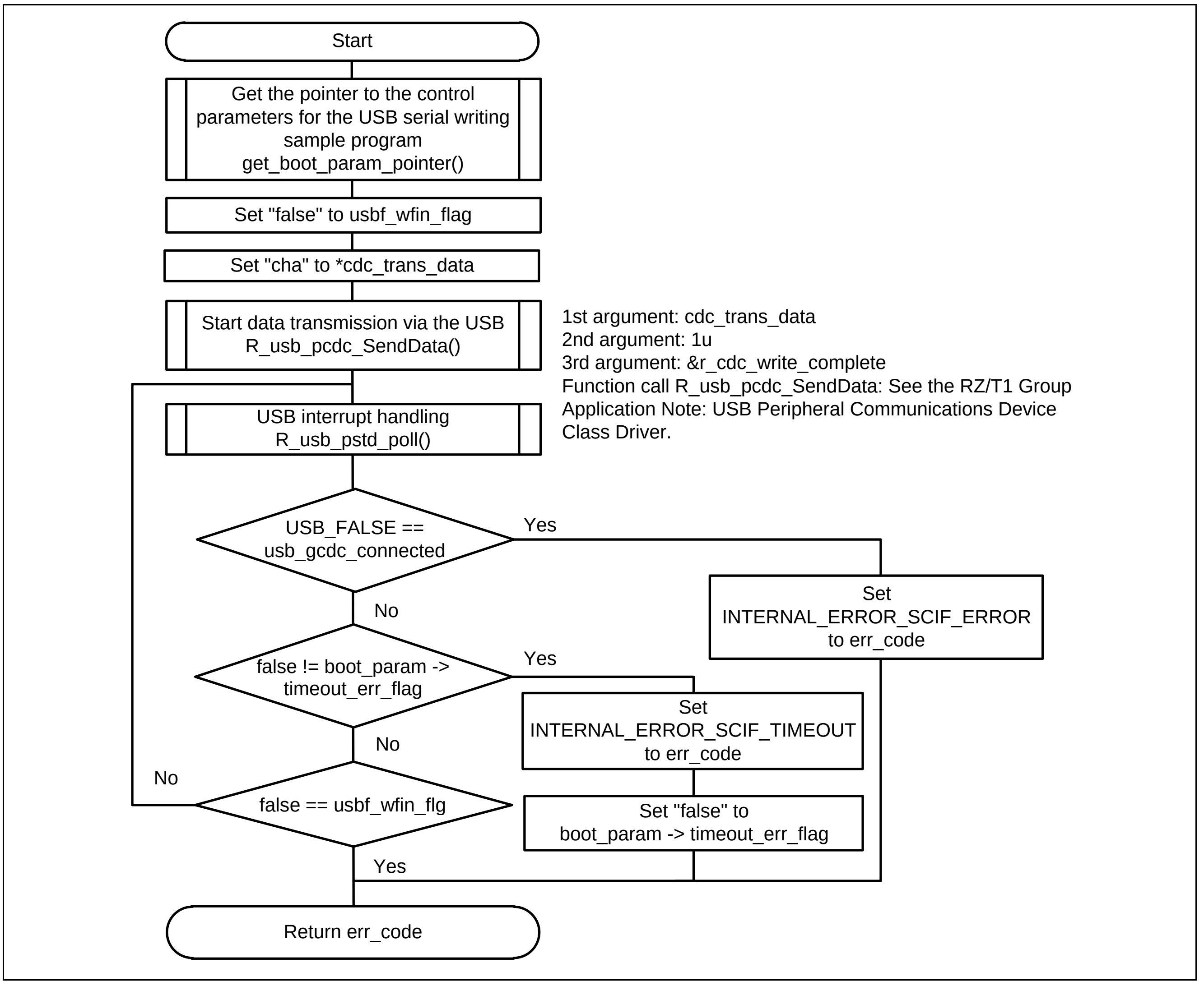


Figure 5.10 scif_putc Function Processing

(13) scif_getc

scif_getc	
Synopsis	Receives data via the USB.
Declaration	static int32_t scif_getc(void)
Description	This function receives data via the USB.
Argument	None
Returned value	Normal value: Received data INTERNAL_ERROR_SCIF_TIMEOUT: Timeout occurred. INTERNAL_ERROR_SCIF_ERROR: A reception error occurred, the USB device was detached.
Remark	

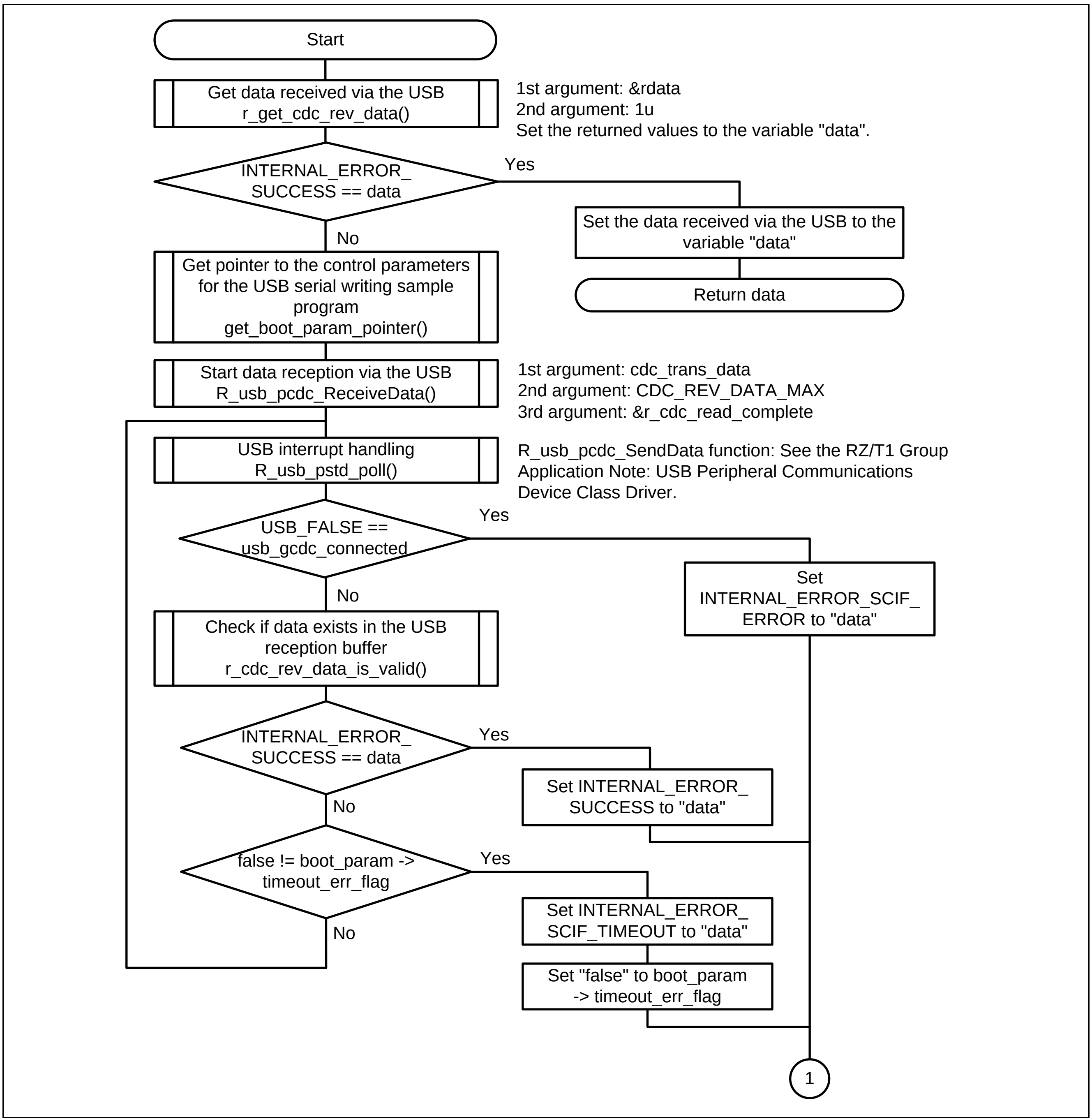


Figure 5.11 scif_getc Function Processing (1/2)

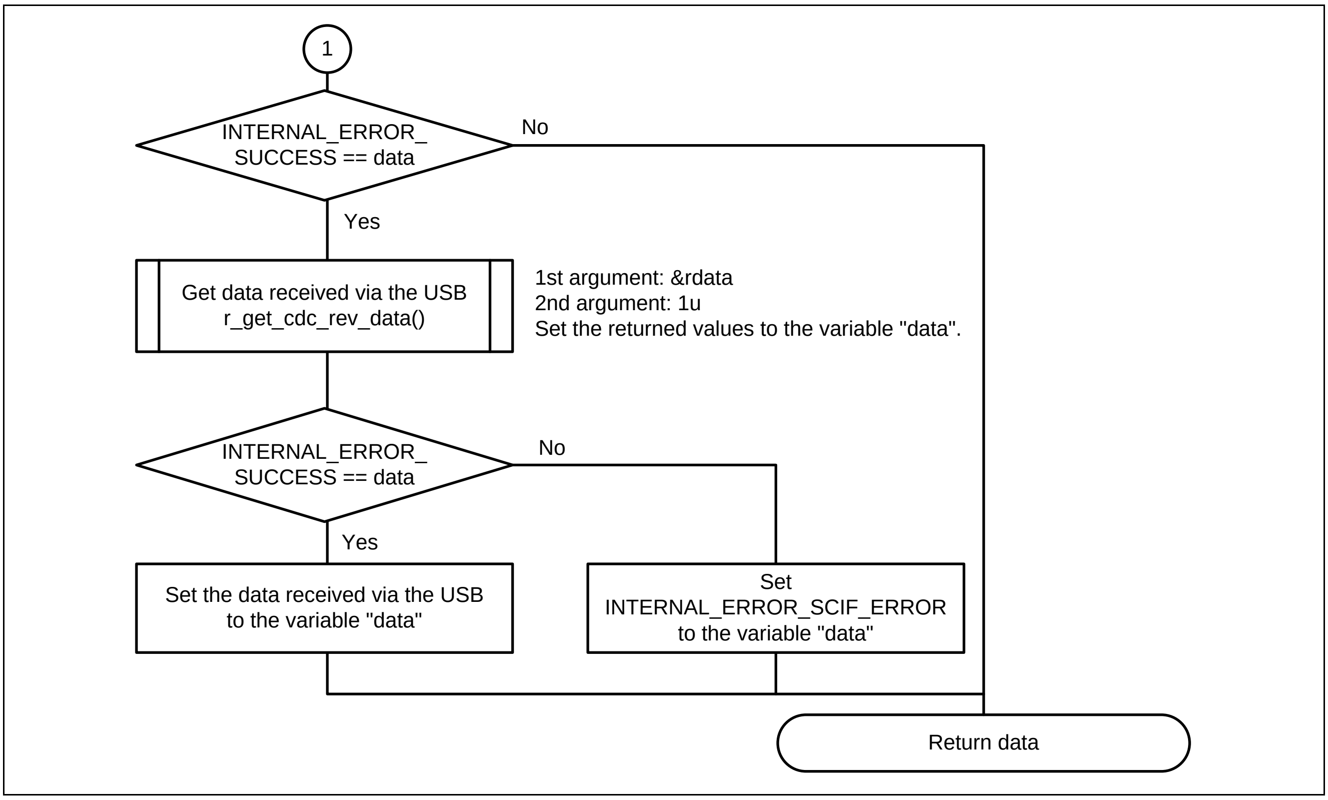


Figure 5.11 scif_getc Function Processing (2/2)

(14) scif_puts

scif_puts	
Synopsis	Sends strings via the USB.
Declaration	static int32_t scif_puts(uint8_t *p_str)
Description	This function sends strings via the USB.
Argument	uint8_t *p_str String data
Returned value	INTERNAL_ERROR_SUCCESS: Normal end INTERNAL_ERROR_SCIF_ERROR: Argument error
Remark	

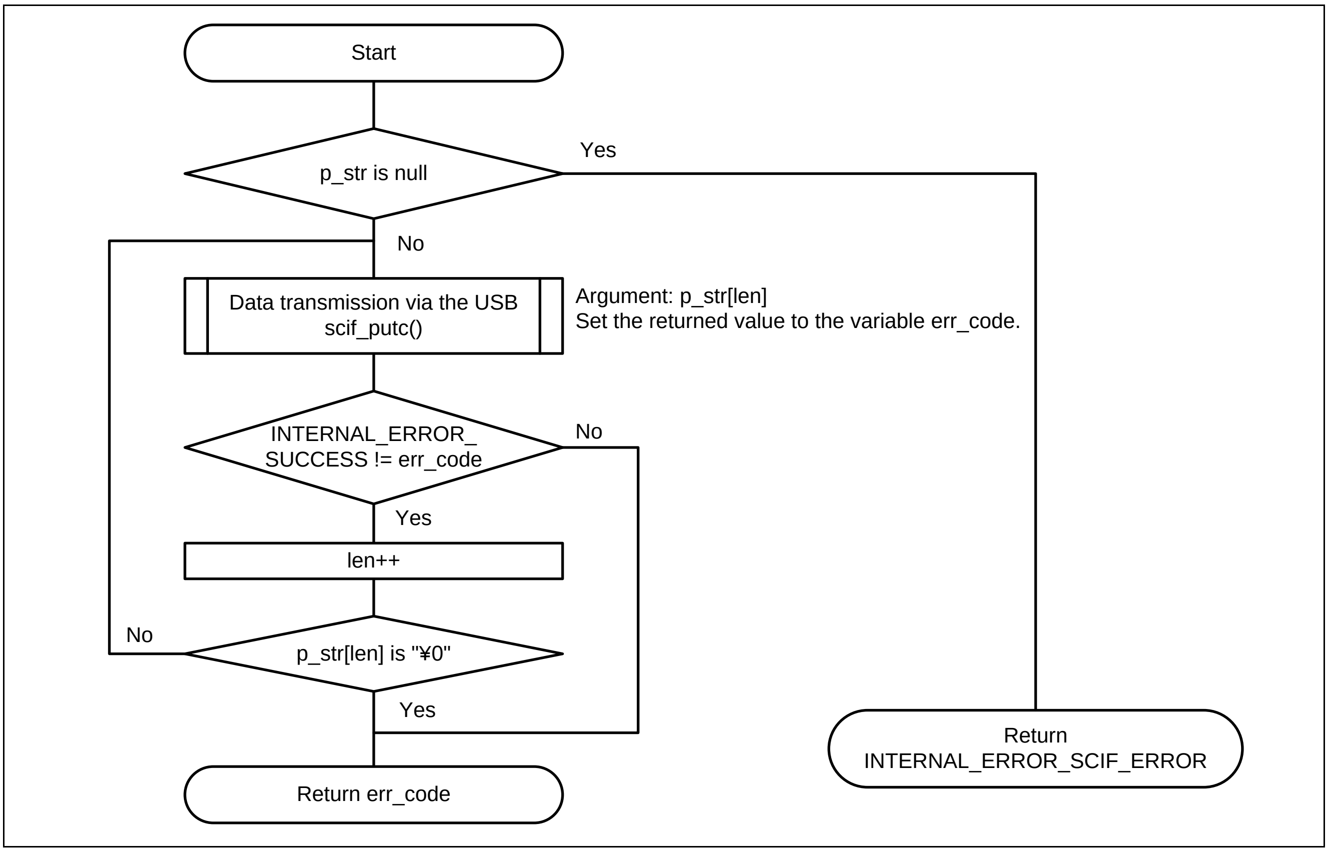


Figure 5.12 scif_puts Function Processing

(15) spibsc_set_protect_code

spibsc_set_protect_code		
Synopsis	Sends the SPIBSC protection control code.	
Declaration	static uint8_t spibsc_set_protect_code(uint8_t *protect_code, uint32_t size)	
Description	See the flowcharts given below.	
Argument	uint8_t *protect_code	Pointer to the protection control code
	uint32_t size	Length of the protection control code
Returned value	ERROR_CODE_SUCCESS: Normal end	
	ERROR_CODE_FILE_TRANSFER: Data transfer to the serial flash ROM failed.	
	ERROR_CODE_TIMEOUT: Timeout (10 s) occurred.	
Remark		

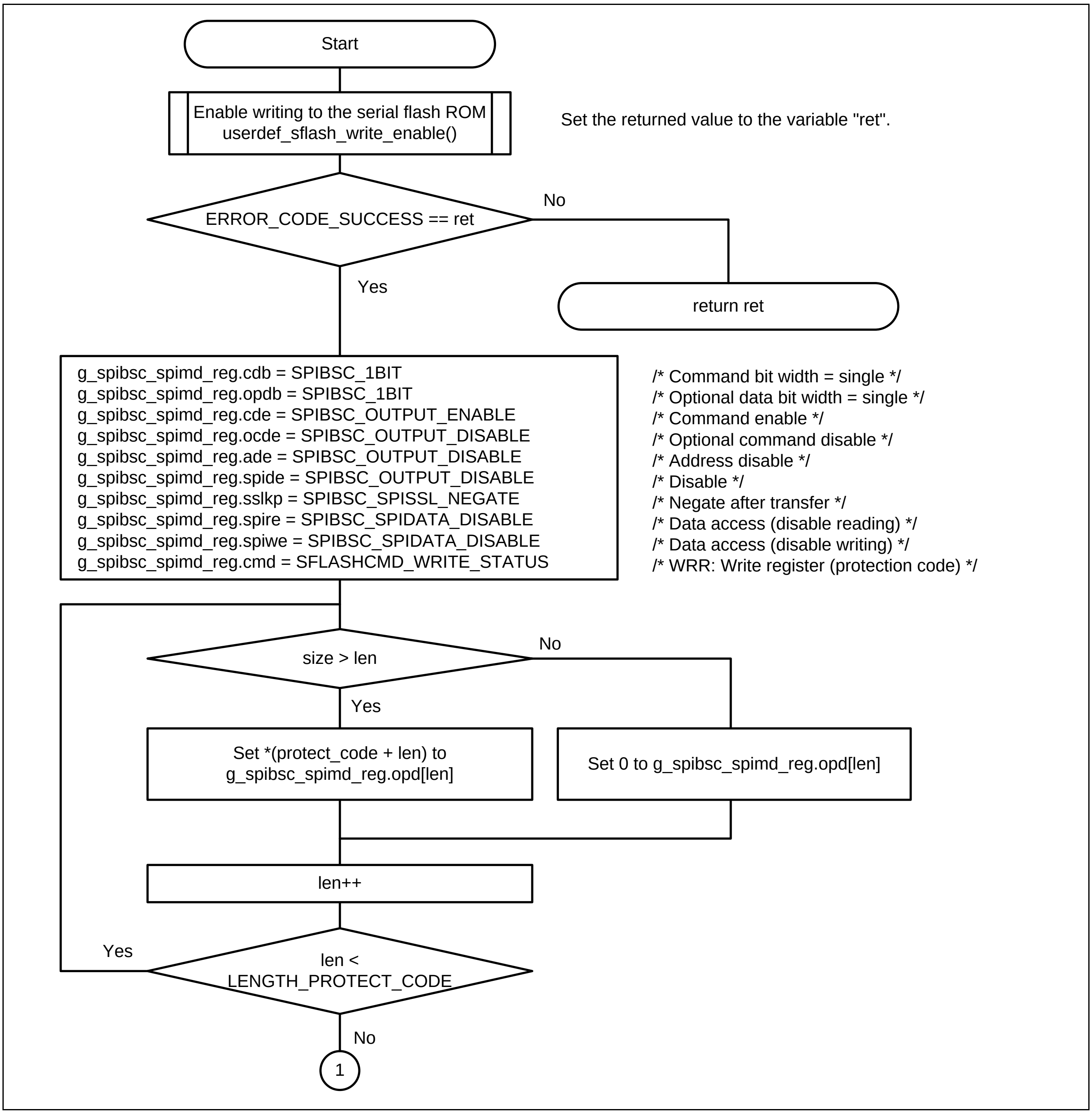


Figure 5.13 spibsc_set_protect_code Function Processing (1/2)

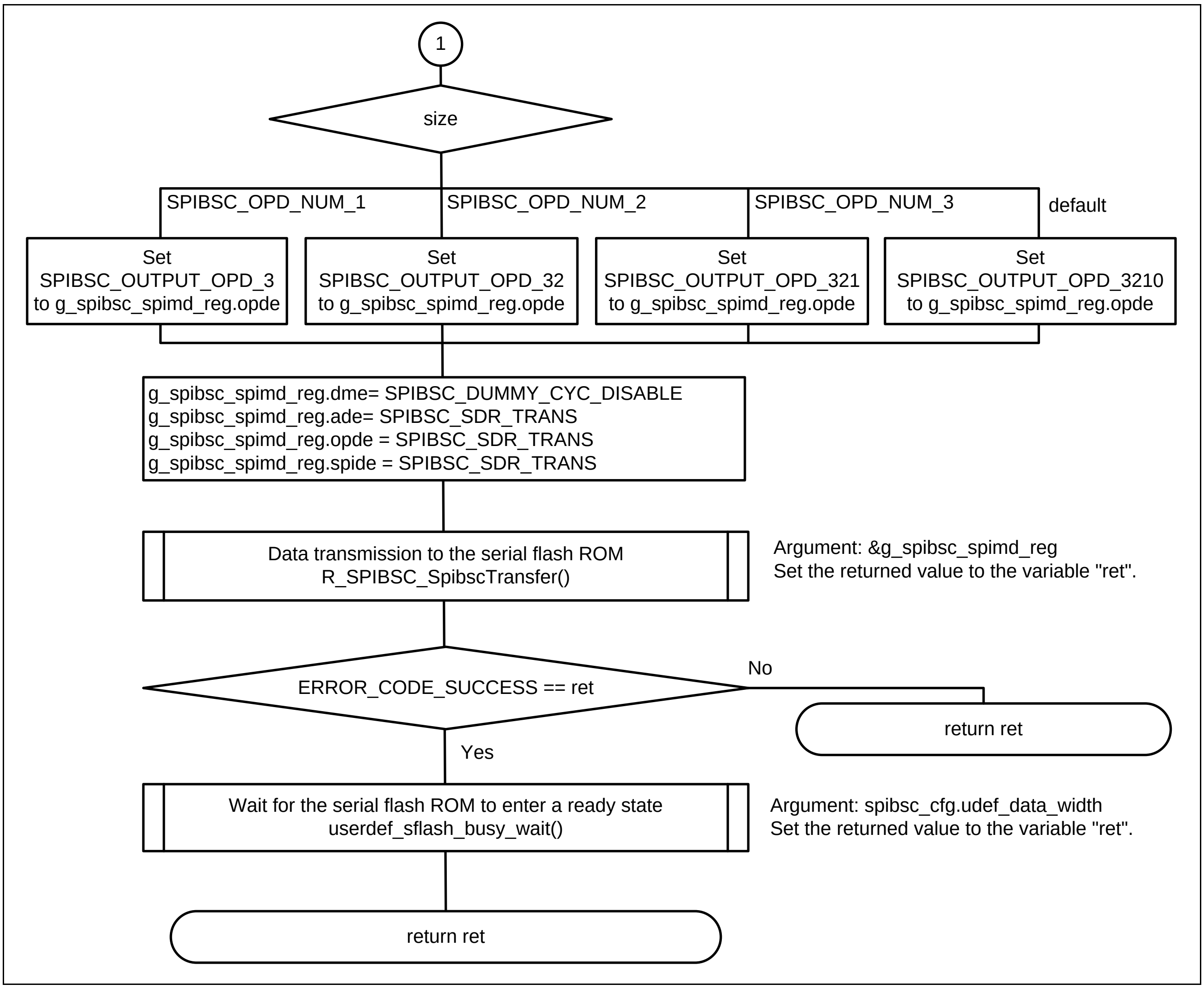


Figure 5.13 spibsc_set_protect_code Function Processing (2/2)

(16) spibsc_verify_data

spibsc_verify_data		
Synopsis	Verifies the data written to the SPIBSC.	
Declaration	static uint8_t spibsc_verify_data(uint32_t target_addr, uint8_t *src_addr, int32_t size)	
Description	See the flowchart given below.	
Argument	uint32_t target_addr	Write destination address
	uint8_t *src_addr	Pointer to the source buffer address
	uint32_t size	Write data size
Returned value	ERROR_CODE_SUCCESS: Normal end	
	ERROR_CODE_FILE_TRANSFER: Data transfer to the serial flash ROM failed.	
	ERROR_CODE_VERIFY: Verification error	
Remark		

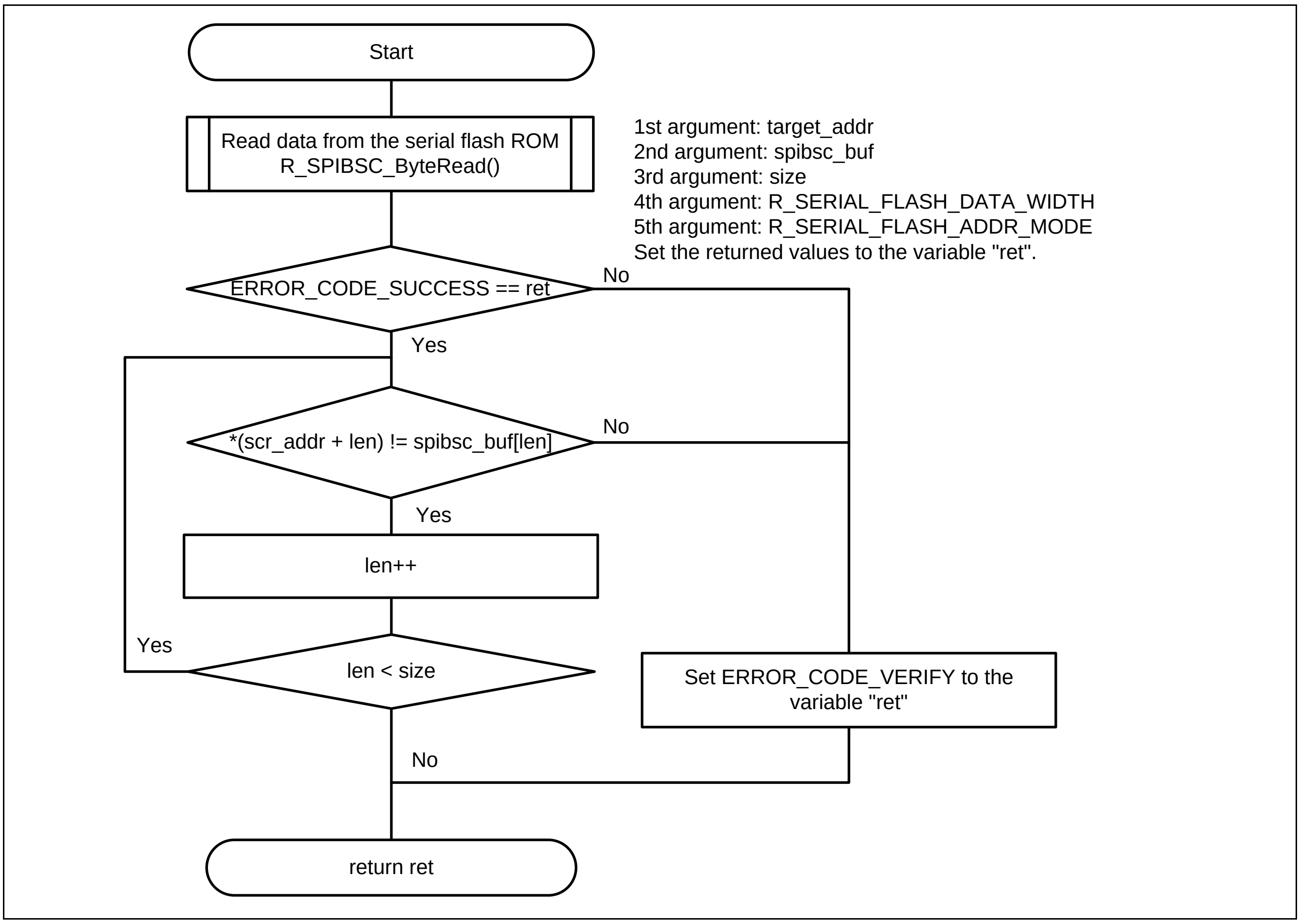


Figure 5.14 spibsc_verify_data Function Processing

(17) scif_get_cmd

scif_get_cmd		
Synopsis	Gets the command.	
Declaration	static uint8_t scif_get_cmd (bootloader_ctrl_t *bootloader_param)	
Description	See the flowcharts given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_PARAM_ERROR: Format error of the received command ERROR_CODE_NO_CORRESPOND: Received a non-supported command. ERROR_CODE_HW_ERROR: HW error occurred. ERROR_CODE_TIMEOUT: Timeout error occurred.	
Remark		

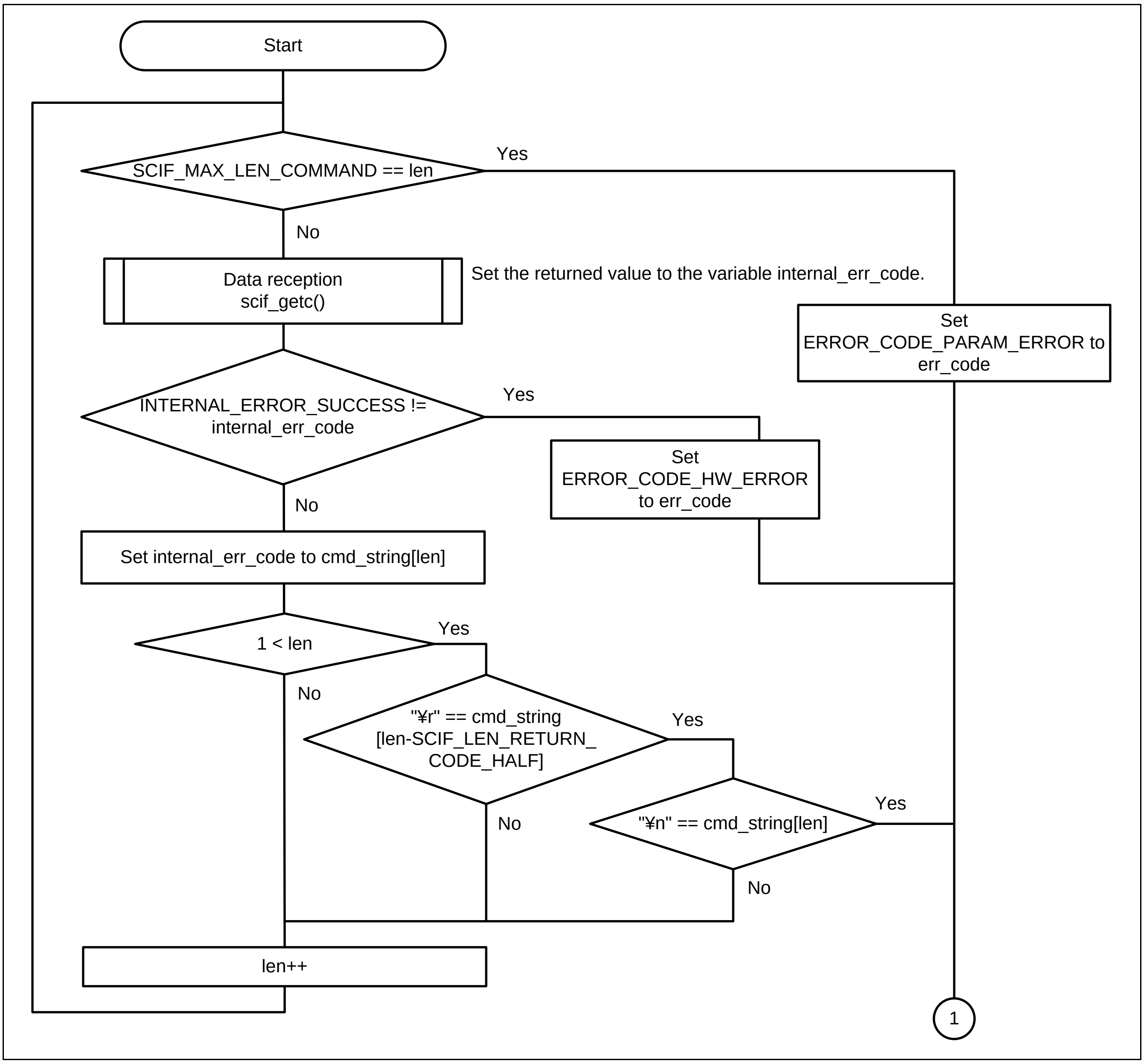


Figure 5.15 scif_get_cmd Function Processing (1/2)

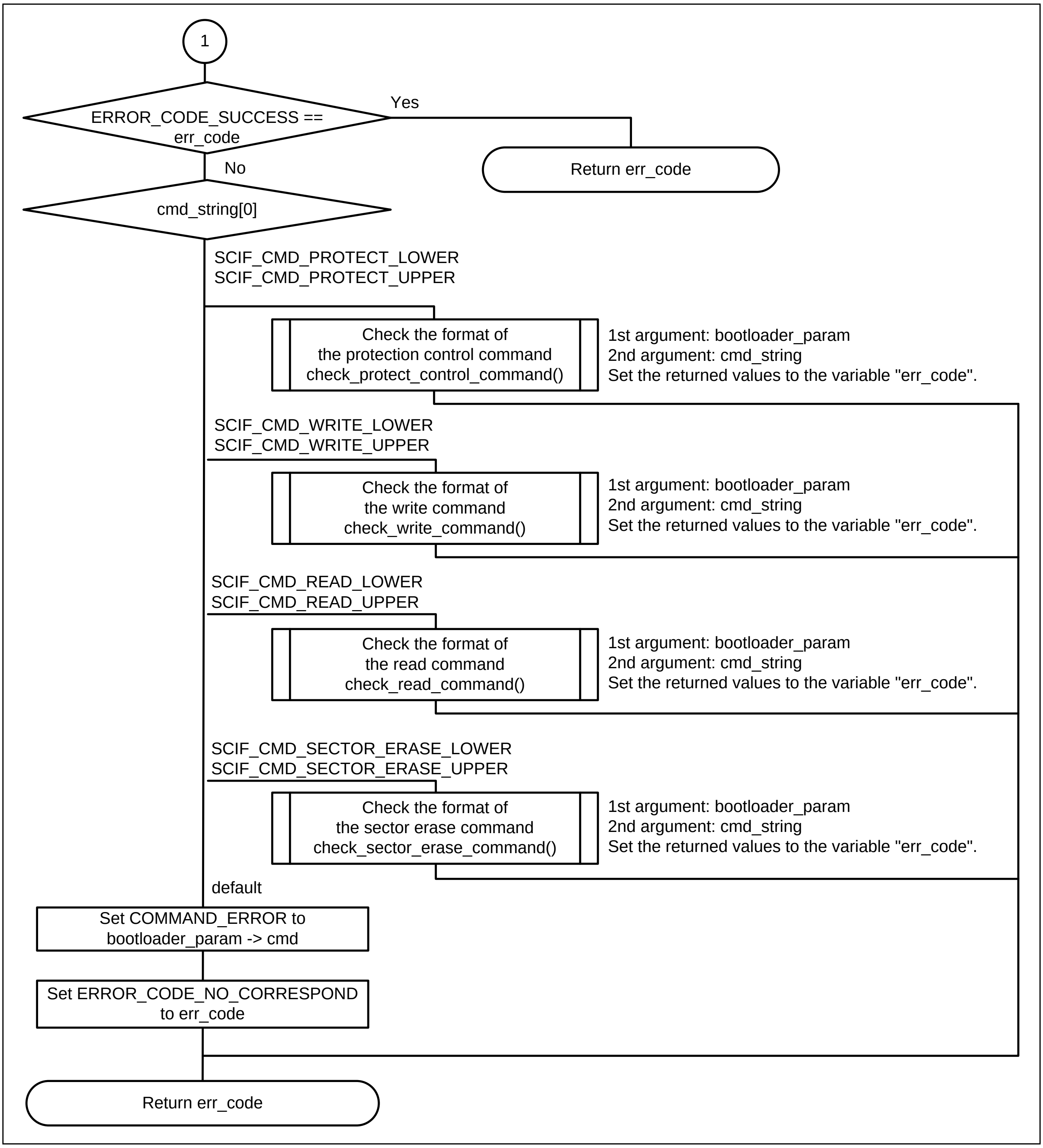


Figure 5.15 scif_get_cmd Function Processing (2/2)

(18) check_protect_control_command

check_protect_control_command		
Synopsis	Performs format check to the protection control command.	
Declaration	static uint8_t check_protect_control_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
Description	See the flowchart given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
	uint8_t *p_cmd	Pointer to the received command data
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_PARAM_ERROR: Format error	
Remark		

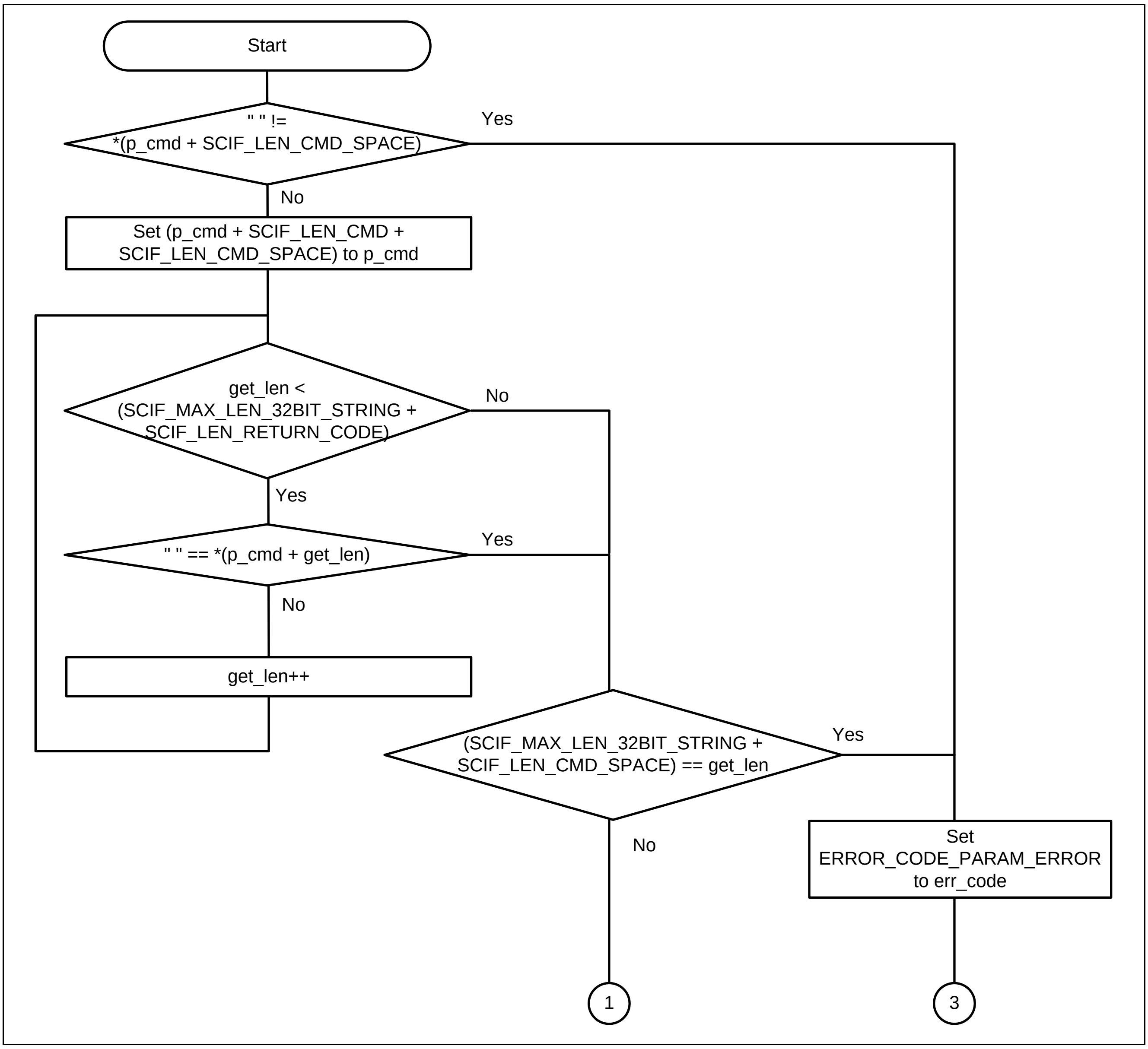


Figure 5.16 check_protect_control_command Function Processing (1/3)

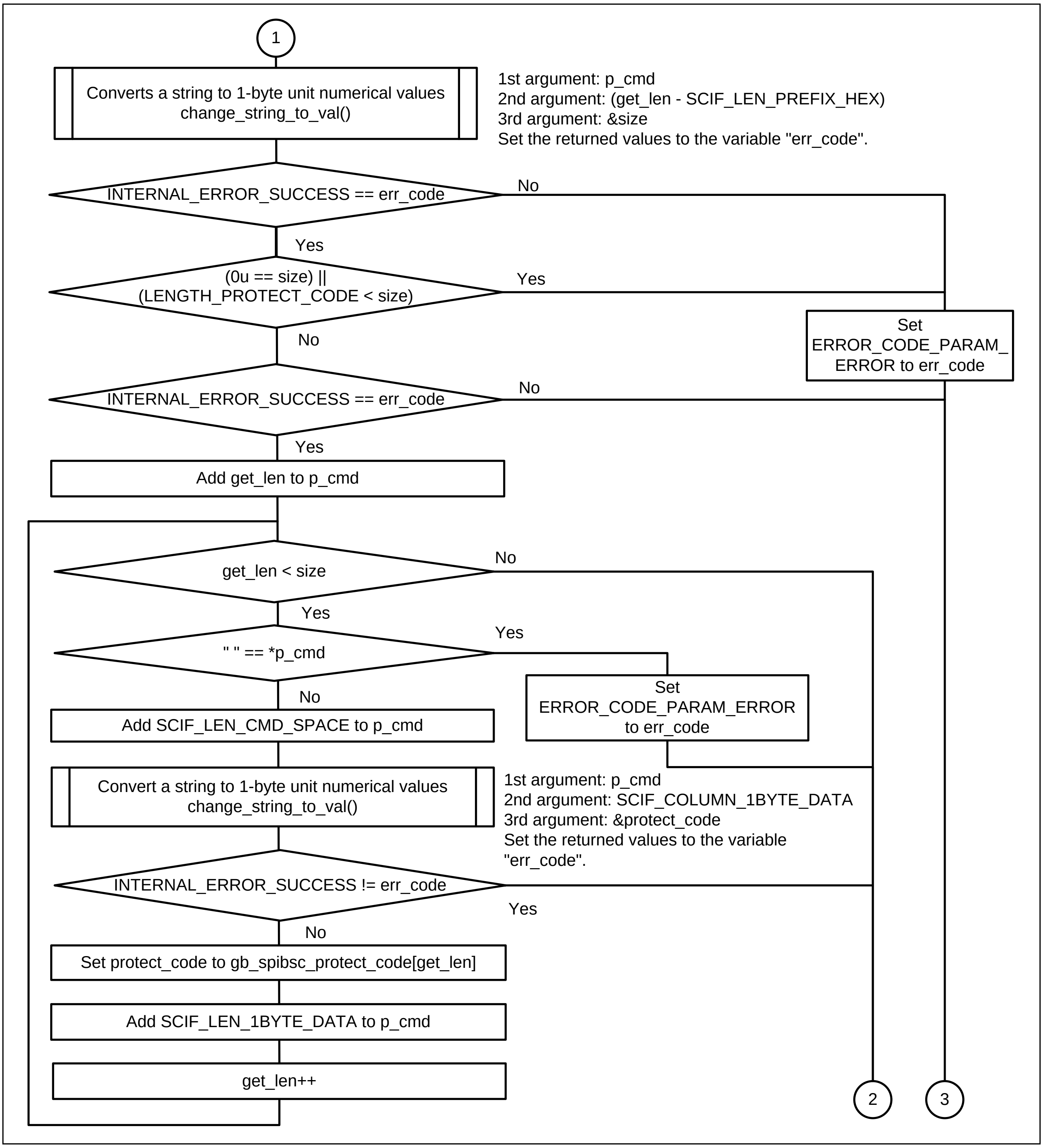


Figure 5.16 check_protect_control_command Function Processing (2/3)

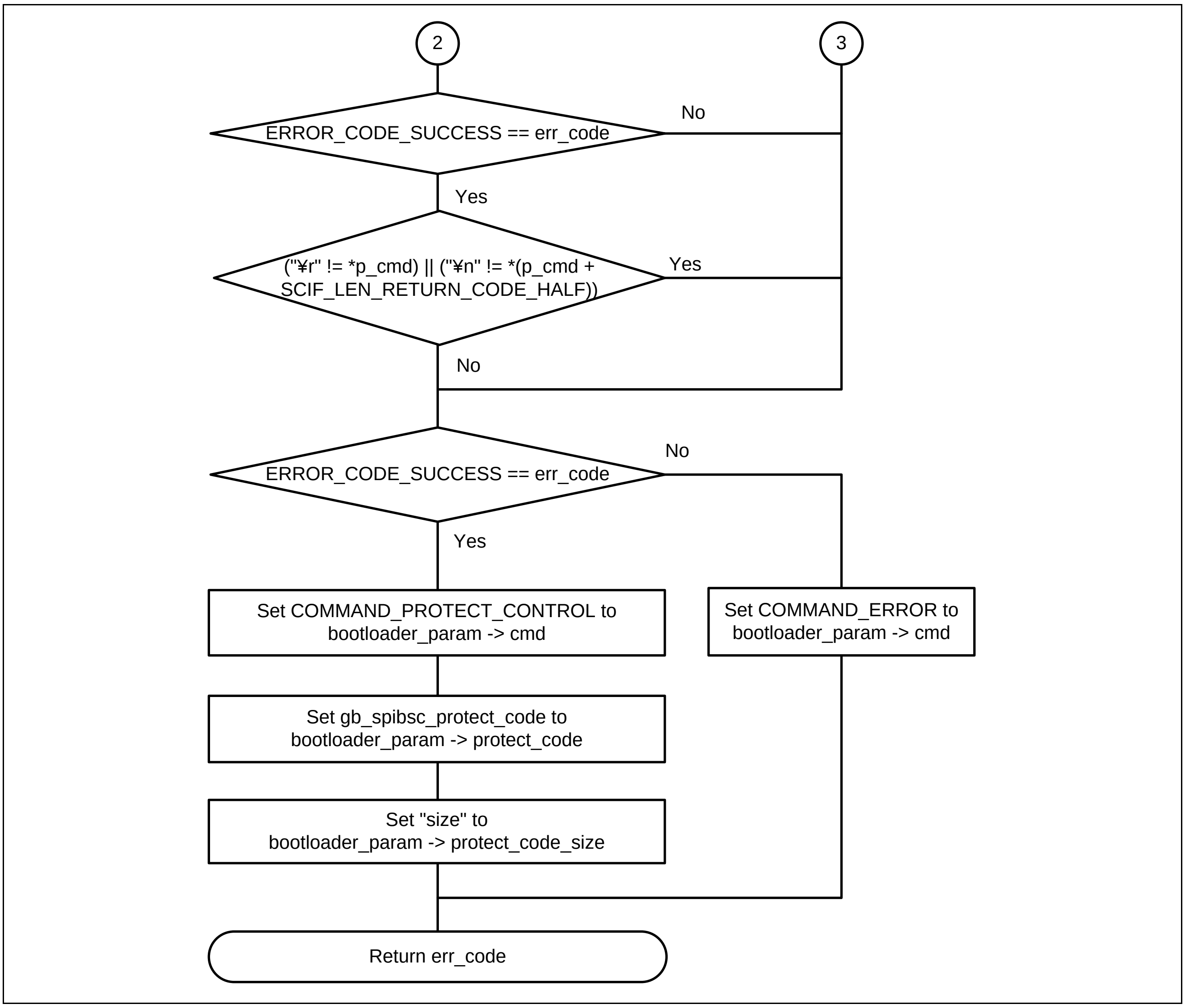


Figure 5.16 check_protect_control_command Function Processing (3/3)

(19) check_write_command

check_write_command		
Synopsis	Performs format check to the write command.	
Declaration	static uint8_t check_write_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
Description	See the flowcharts given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
	uint8_t *p_cmd	Pointer to the received command data
Returned value	ERROR_CODE_SUCCESS: Normal end	
	ERROR_CODE_PARAM_ERROR: Format error	
Remark		

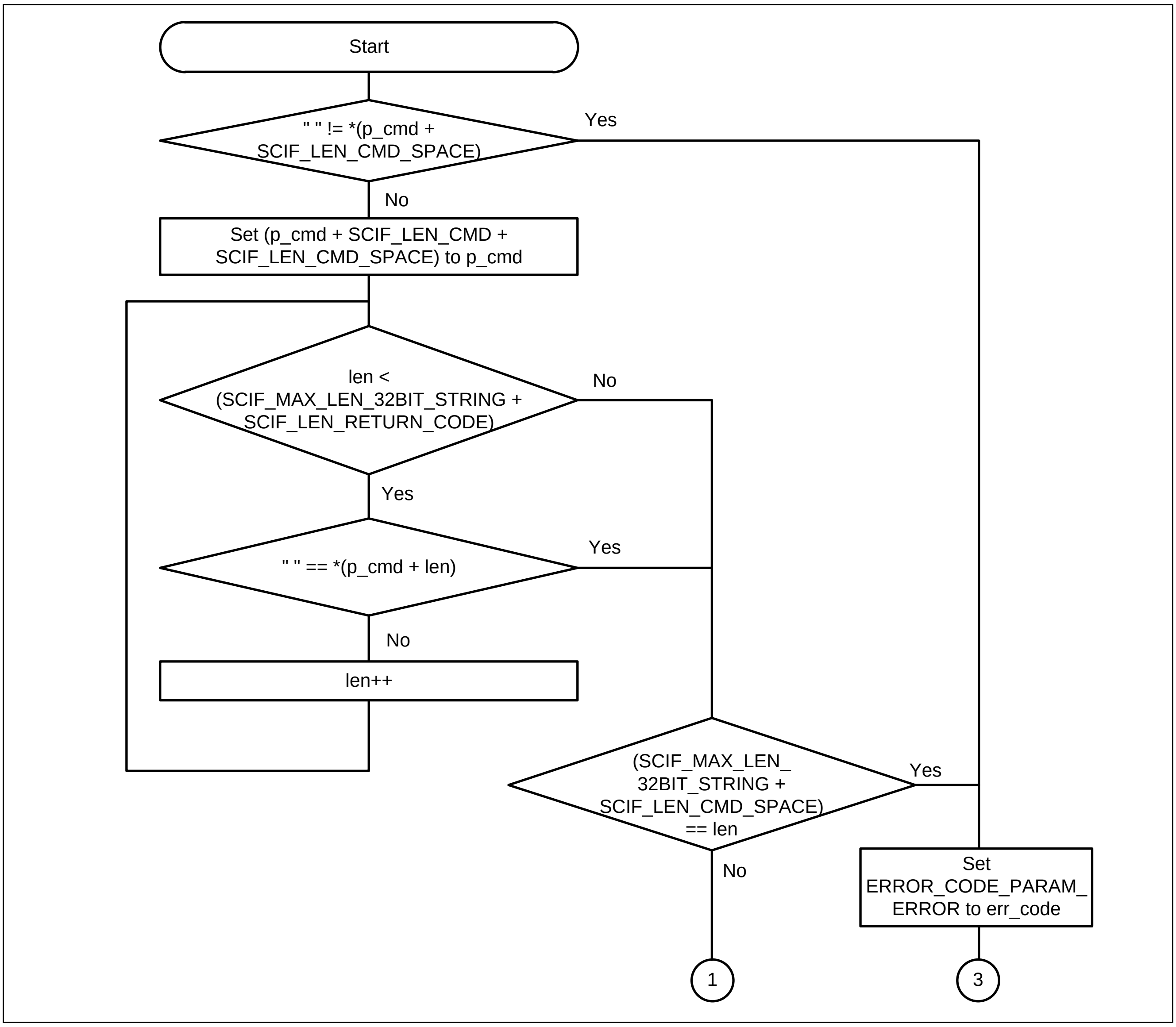


Figure 5.17 check_write_command Function Processing (1/3)

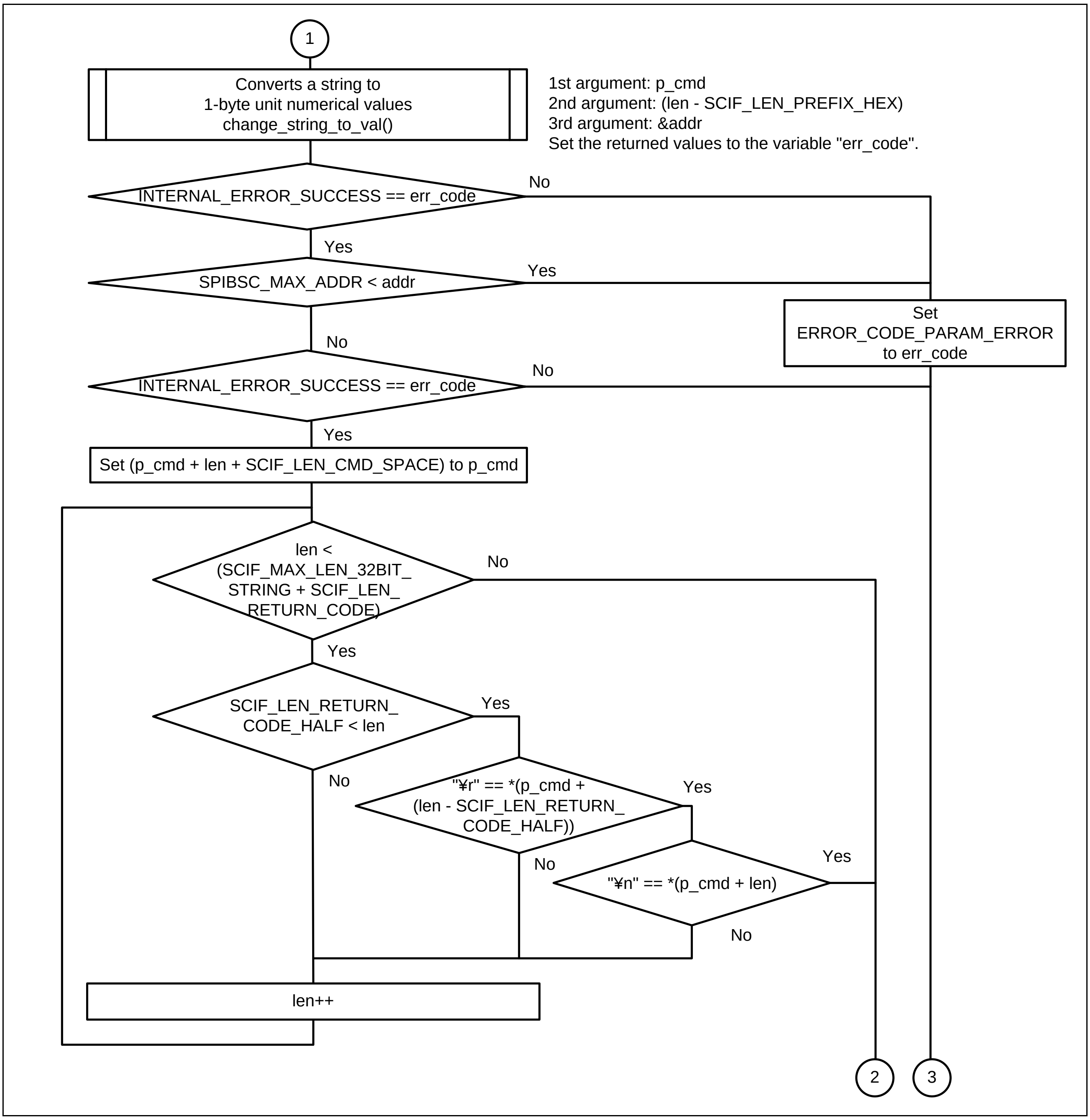


Figure 5.17 check_write_command Function Processing (2/3)

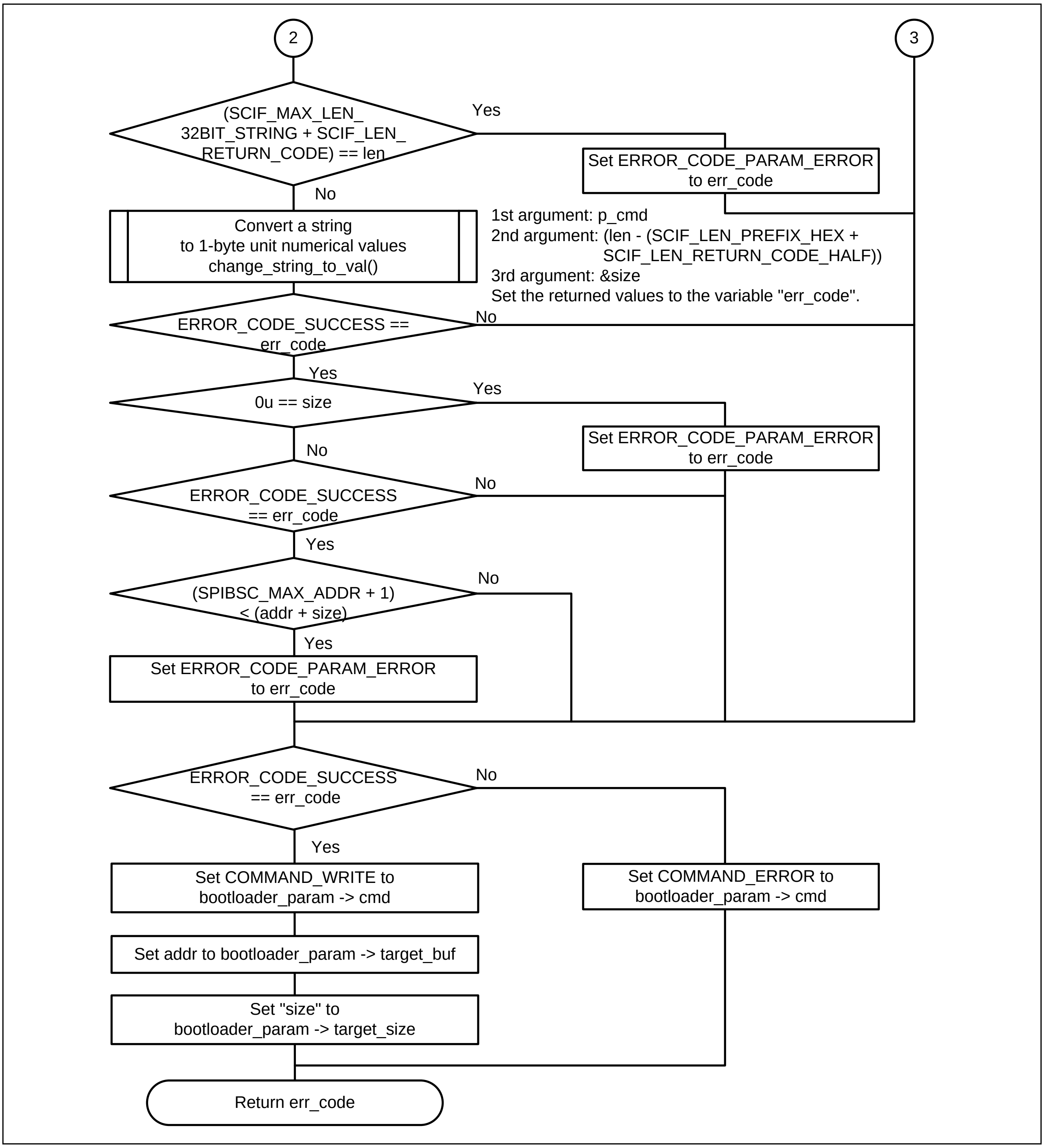


Figure 5.17 check_write_command Function Processing (3/3)

(20) check_read_command

check_read_command		
Synopsis	Performs format check to the read command.	
Declaration	static uint8_t check_read_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
Description	See the flowcharts given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
	uint8_t *p_cmd	Pointer to the received command data
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_PARAM_ERROR: Format error	
Remark		

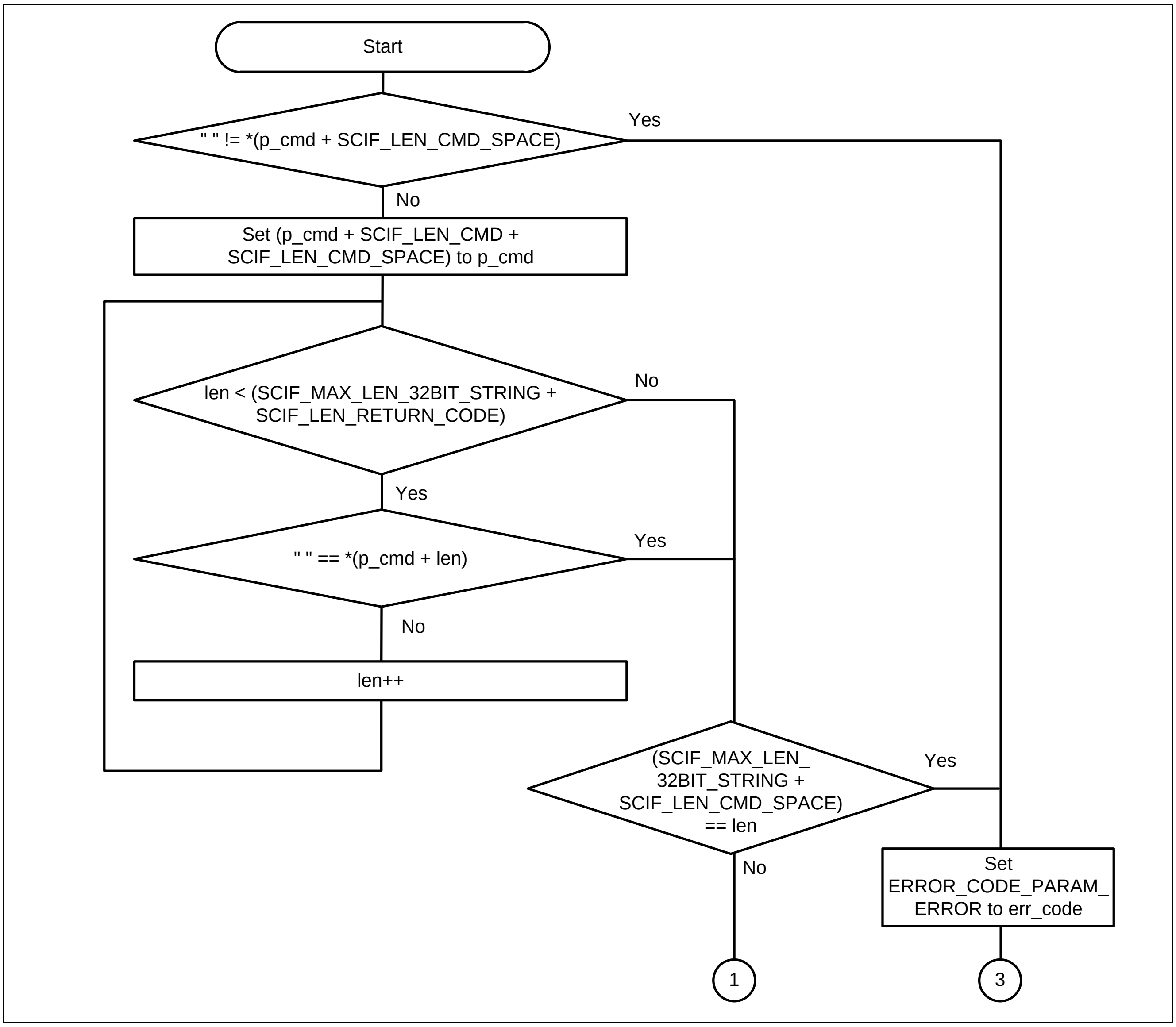


Figure 5.18 check_read_command Function Processing (1/3)

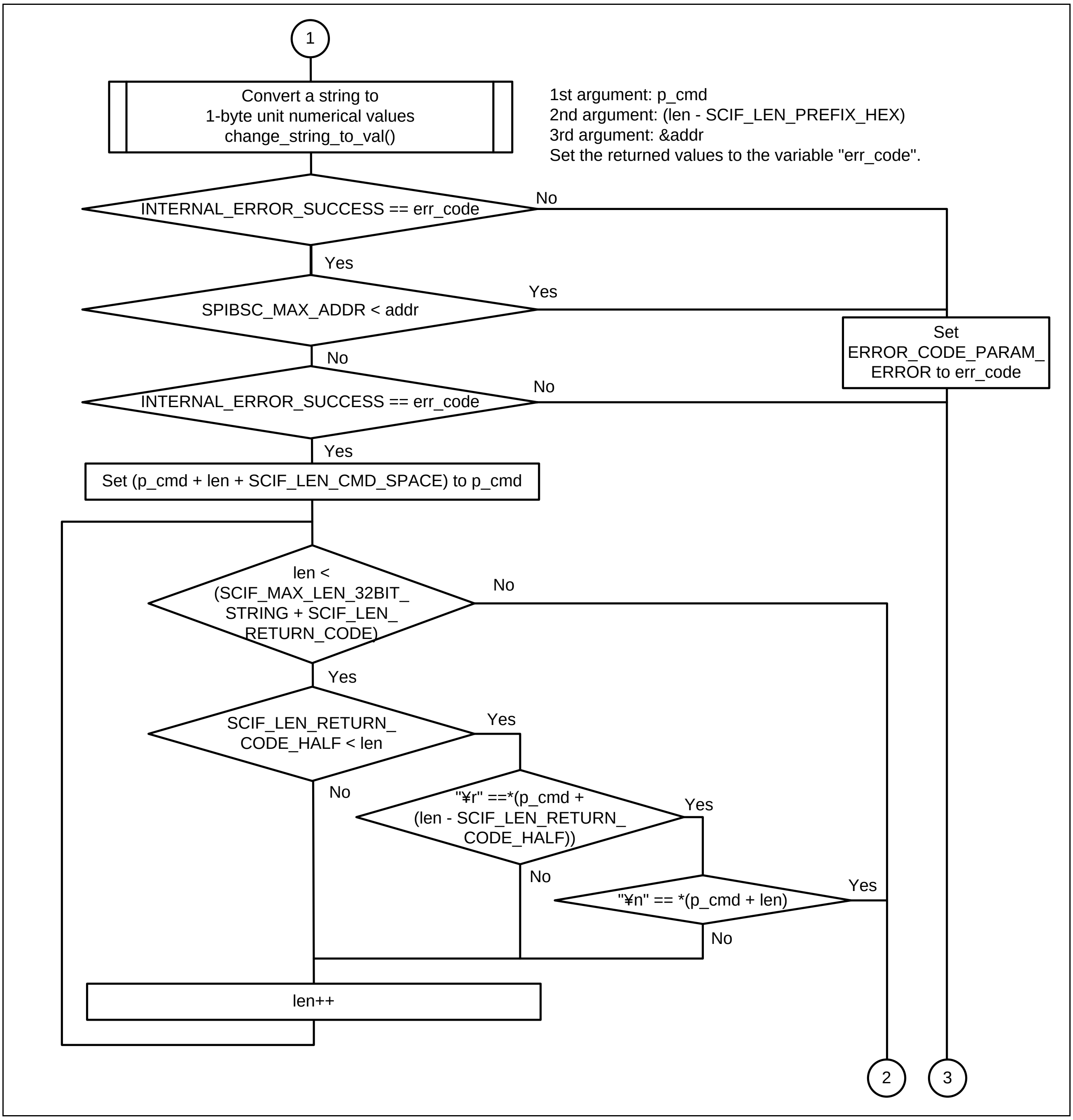


Figure 5.18 check_read_command Function Processing (2/3)

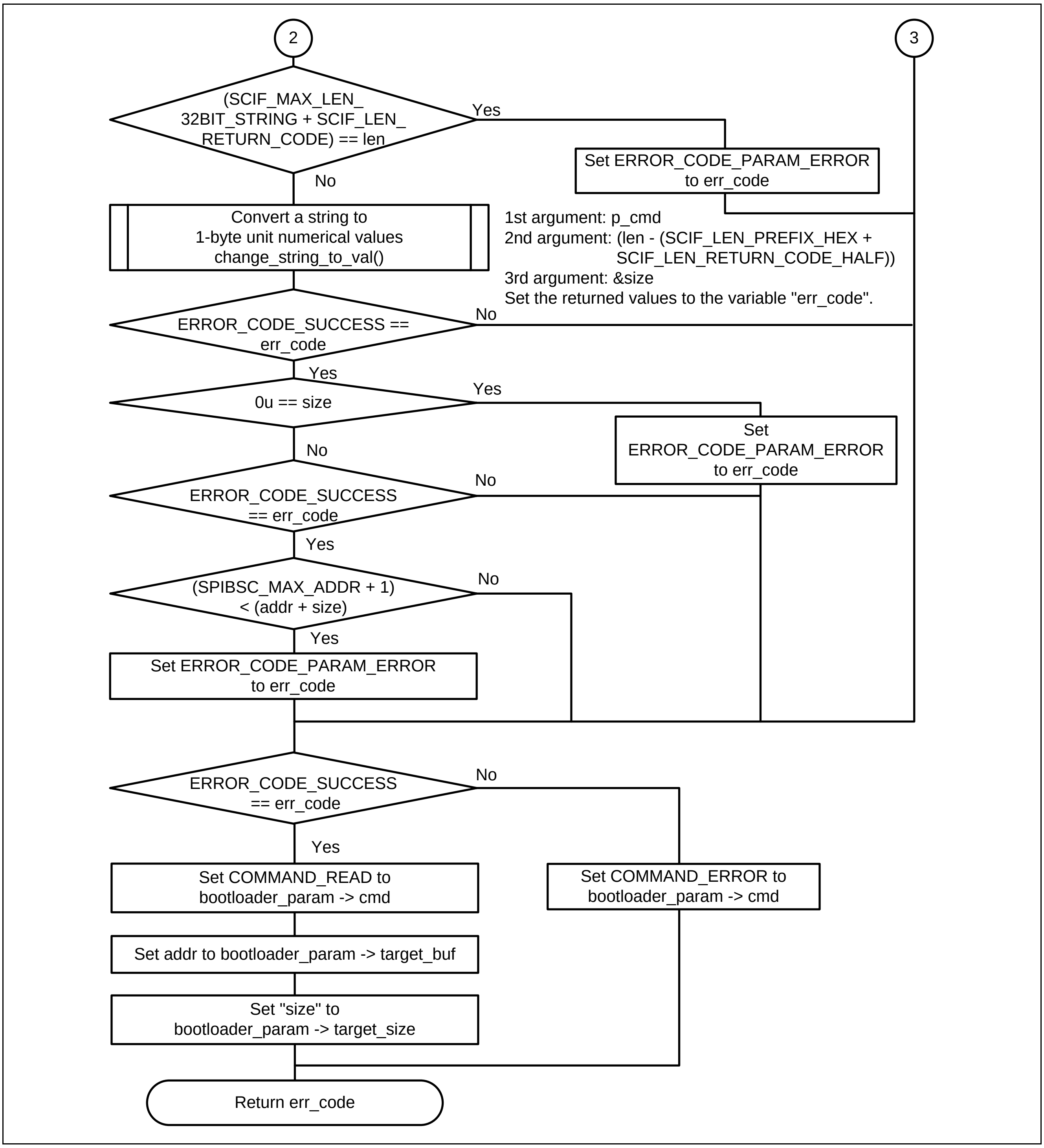


Figure 5.18 check_read_command Function Processing (3/3)

(21) check_sector_erase_command

check_sector_erase_command		
Synopsis	Performs format check to a sector erase command.	
Declaration	static uint8_t check_sector_erase_command(bootloader_ctrl_t *bootloader_param, uint8_t *p_cmd)	
Description	See the flowcharts given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
	uint8_t *p_cmd	Pointer to the received command data
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_PARAM_ERROR: Format error	
Remark		

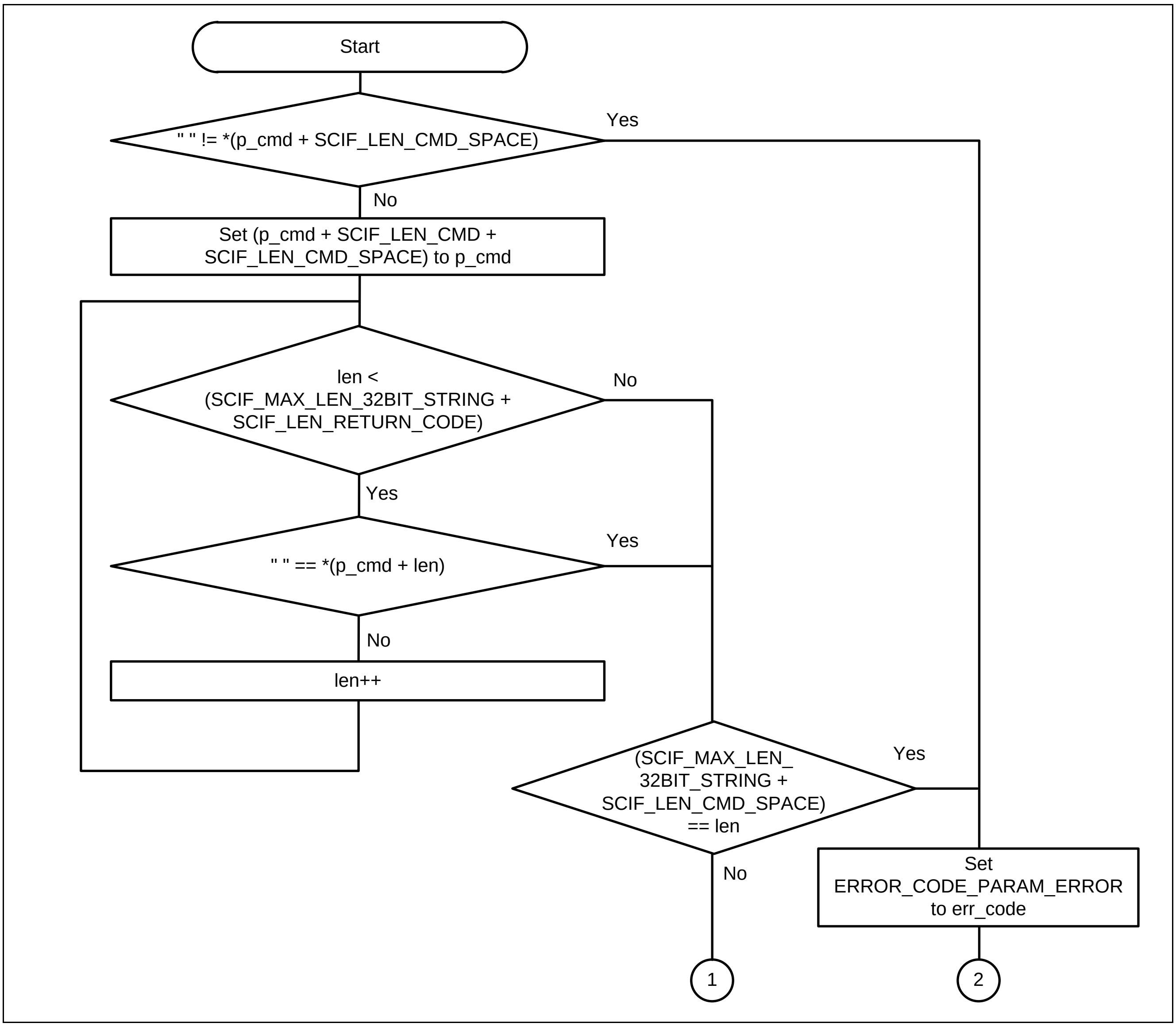


Figure 5.19 check_sector_erase_command Function Processing (1/2)

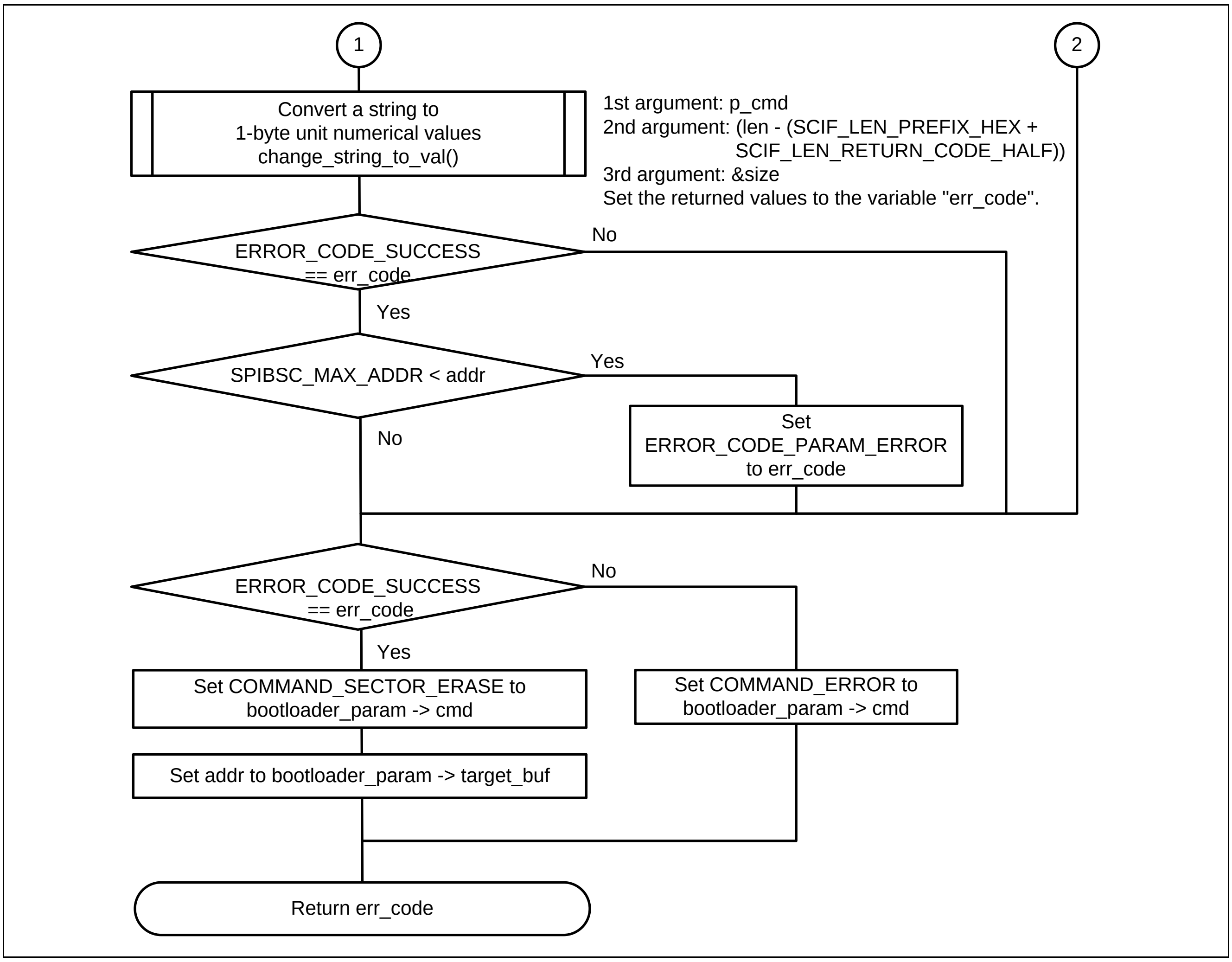


Figure 5.19 check_sector_erase_command Function Processing (2/2)

(22) change_string_to_val

change_string_to_val		
Synopsis	Converts a string to 1-byte unit numerical values.	
Declaration	static uint8_t change_string_to_val(uint8_t *p_inbuf, uint32_t column, uint32_t *p_outbuf)	
Description	See the flowchart given below.	
Argument	uint8_t *p_inbuf	Pointer to the buffer which holds the string data to be converted
	uint32_t column	Number of digits of the string to be converted
	uint32_t *p_outbuf	Pointer to the buffer to which the converted numerical value will be stored
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_PARAM_ERROR: Format error	
Remark		

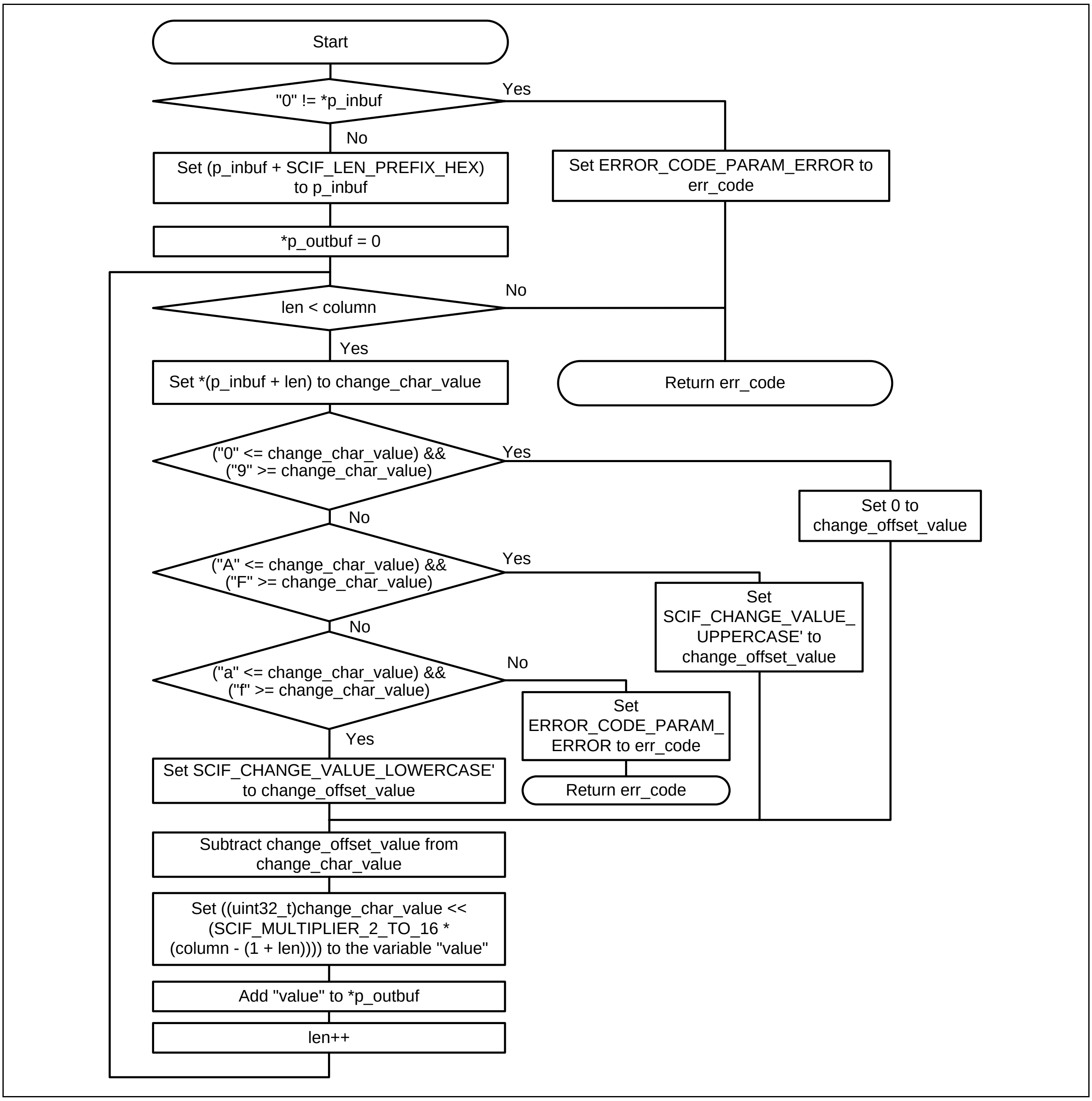


Figure 5.20 change_string_to_val Function Processing

(23) change_errcode_to_string

change_errcode_to_string		
Synopsis	Converts a numerical value to hexadecimal strings.	
Declaration	static void change_errcode_to_string(uint32_t value, uint8_t *p_buf)	
Description	See the flowchart given below.	
Argument	uint32_t value	Data to be converted
	uint8_t *p_buf	Pointer to the buffer in which the converted data will be stored
Returned value	None	
Remark		

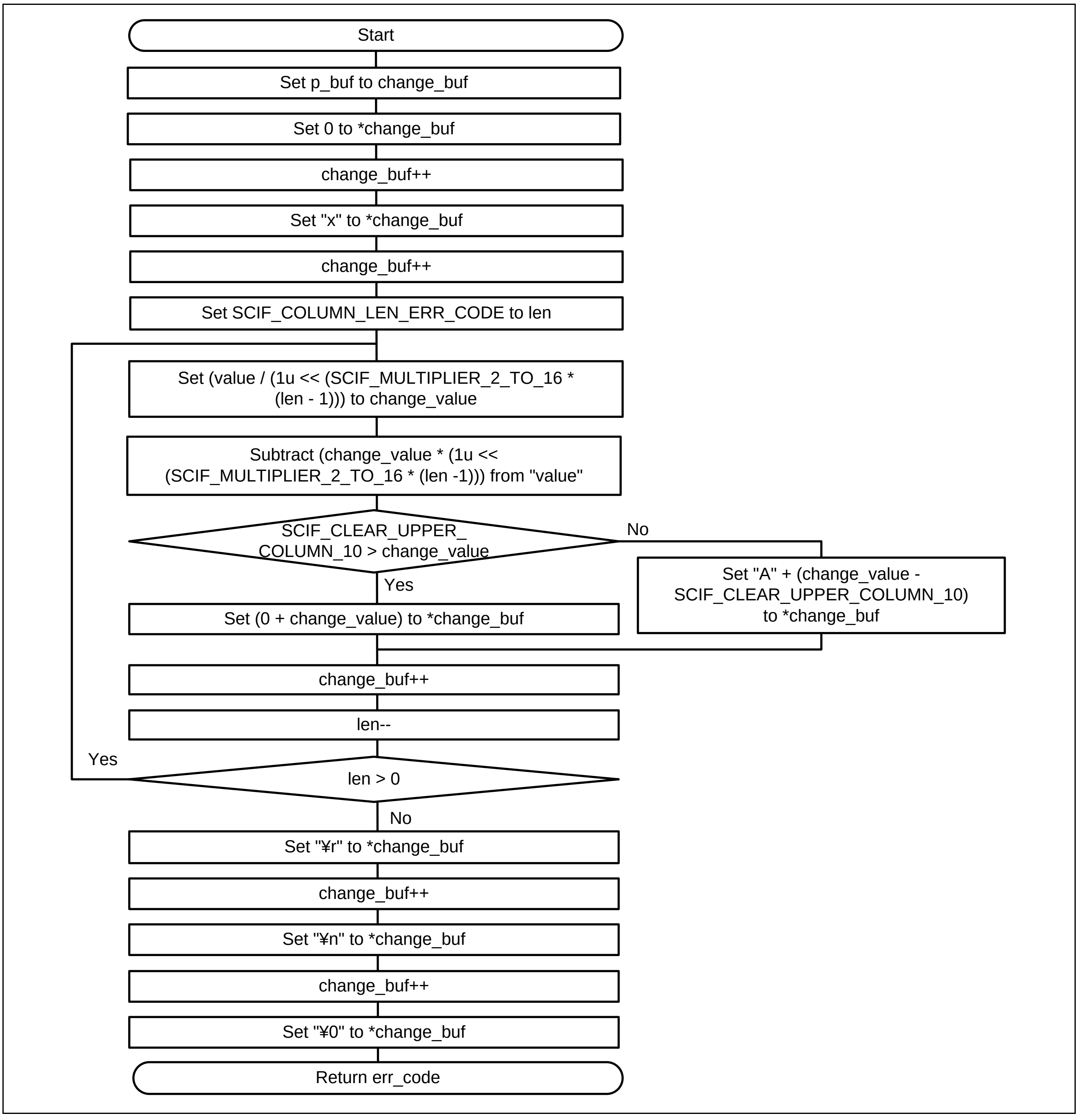


Figure 5.21 change_errcode_to_string Function Processing

(24) spibsc_init

spibsc_init	
Synopsis	Initialization of the SPIBSC
Declaration	static void spibsc_init(void)
Description	This function optimally configures the SPIBSC for a serial flash ROM made by Macronix (MX25L51245G).
Argument	None
Returned value	None
Remark	For details of this function, see the RZ/T1 Group Application Note: Serial Flash Sample Program.

(25) r_cdc_read_complete

r_cdc_read_complete		
Synopsis	Callback for completion of data reception via the USB	
Declaration	static void r_cdc_read_complete(USB_UTR_t *mess)	
Description	This functions stores data to the USB reception buffer.	
Argument	USB_UTR_t *mess	Pointer to the USB communications structure
Returned value	None	
Remark		

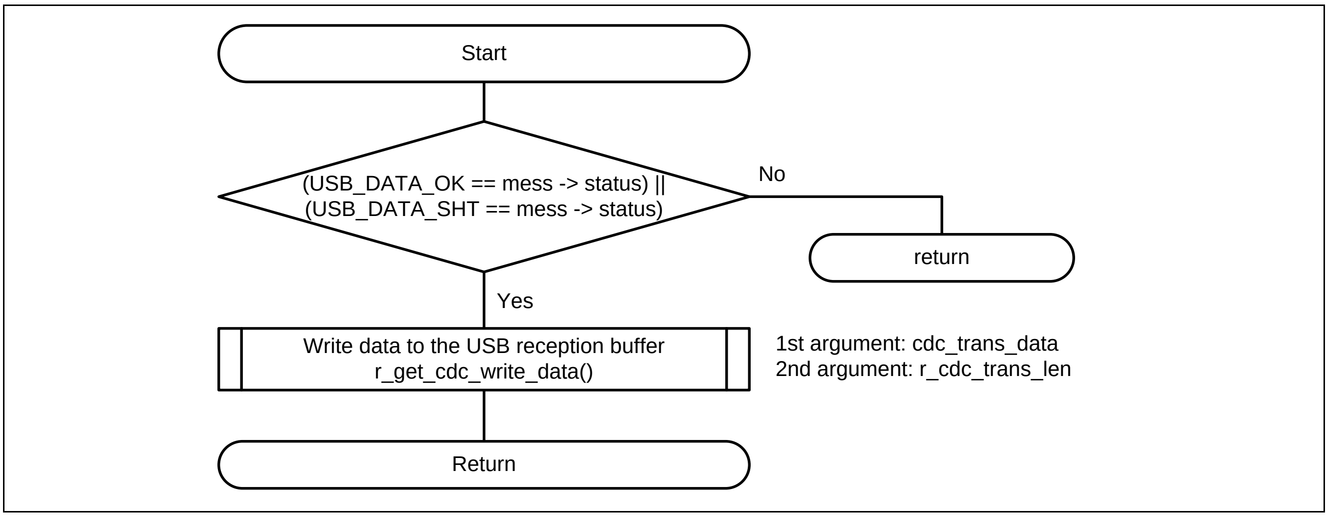


Figure 5.22 r_cdc_read_complete Function Processing

(26) r_cdc_write_complete

r_cdc_write_complete		
Synopsis	Callback for completion of data transmission via the USB	
Declaration	static void r_cdc_write_complete(USB_UTR_t *mess)	
Description	This function sets usbf_wfin_flag to "true".	
Argument	USB_UTR_t *mess	Pointer to the USB communications structure
Returned value	None	
Remark	For details of the USB communications structure, see the RZ/T1 Group Application Note: USB Peripheral Communications Device Class Driver.	

(27) r_cdc_rev_data_is_valid

r_cdc_rev_data_is_valid	
Synopsis	Checks if data exists in the USB reception buffer.
Declaration	static uint8_t r_cdc_rev_data_is_valid(void)
Description	This function returns USB_CDC_DATA_INVALID if data do not exist in the USB reception buffer and USB_CDC_DATA_VALID if data exists.
Argument	None
Returned value	USB_CDC_DATA_INVALID(0u): Data do not exist in the USB reception buffer. USB_CDC_DATA_VALID(1u): Data exists in the USB reception buffer.
Remark	

(28) r_cdc_rev_data_clear

r_cdc_rev_data_clear	
Synopsis	Clears the USB reception buffer.
Declaration	static void r_cdc_rev_data_clear(void)
Description	Clear cdc_rev_data_pr and cdc_rev_data_pw to 0.
Argument	None
Returned value	None
Remark	

(29) r_get_cdc_write_data

r_get_cdc_write_data		
Synopsis	Writes data to the USB reception buffer.	
Declaration	static uint8_t r_get_cdc_write_data(uint8_t* pbuf, uint32_t sz)	
Description	This function stores an amount of data specified in sz to the USB reception buffer.	
Argument	uint8_t* pbuf	Pointer to the buffer
	uint32_t sz	Write size
Returned value	INTERNAL_ERROR_SUCCESS: Normal end INTERNAL_ERROR_SCIF_ERROR: Write size is 0	
Remark		

(30) r_get_cdc_rev_data

r_get_cdc_rev_data		
Synopsis	Reads data from the USB reception buffer.	
Declaration	static uint8_t r_get_cdc_rev_data(uint8_t* pbuf, uint32_t sz)	
Description	This function stores data received via the USB of the predetermined size for reading in the USB read buffer.	
Argument	uint8_t* pbuf	Pointer to the read buffer
	uint32_t sz	Read size
Returned value	INTERNAL_ERROR_SUCCESS: Normal end INTERNAL_ERROR_SCIF_ERROR: Size of the data stored in the USB reception buffer is 0 or smaller than the amount specified in the parameter "read size".	
Remark		

(31) r_get_cdc_rev_1Bdata

r_get_cdc_rev_1Bdata	
Synopsis	Reads 1 byte of data from the USB reception buffer.
Declaration	static uint8_t r_get_cdc_rev_1Bdata(void)
Description	See the flowchart given below.
Argument	None
Returned value	Data stored in the USB reception buffer
Remark	

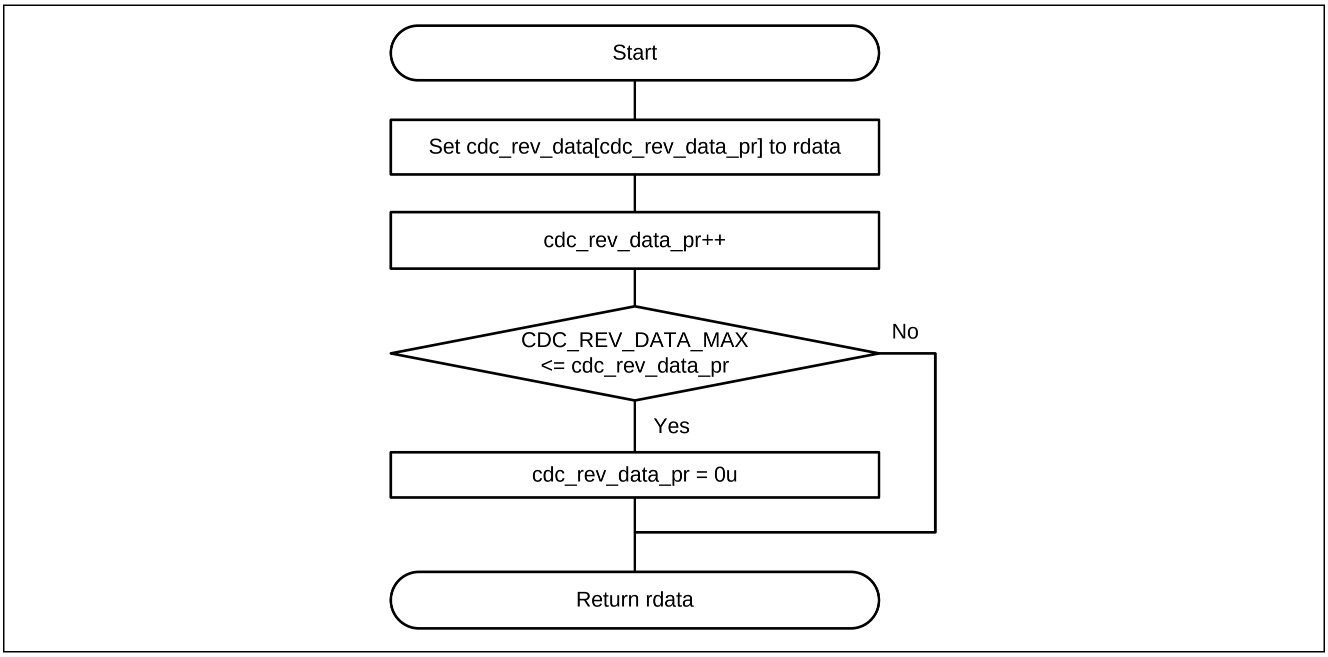


Figure 5.23 r_get_cdc_rev_1Bdata Function Processing

(32) r_cdc_start

r_cdc_start	
Synopsis	Starts the USB CDC.
Declaration	static void r_cdc_start(void)
Description	See the flowchart given below.
Argument	None
Returned value	None
Remark	

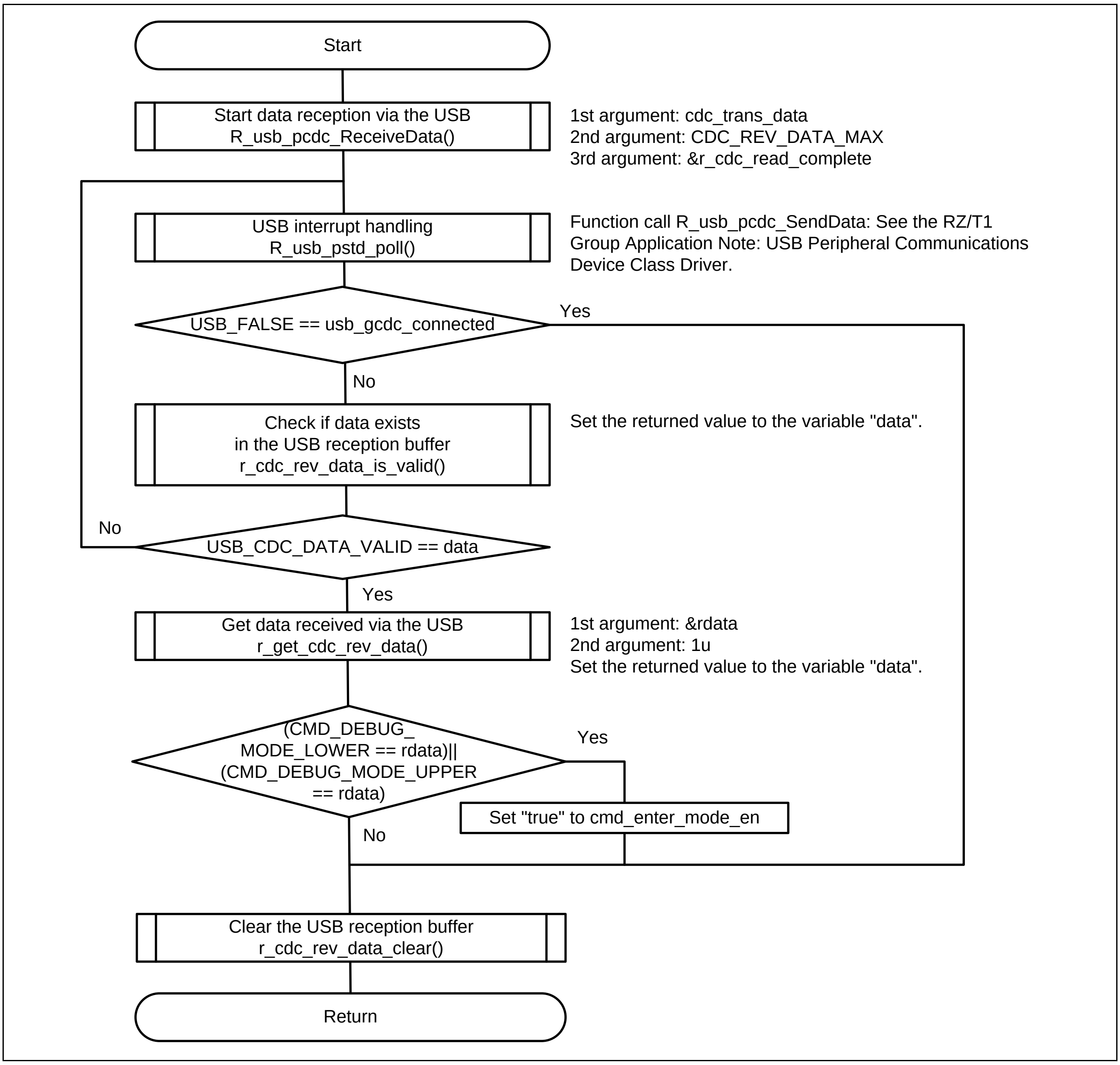


Figure 5.24 r_cdc_start Function Processing

(33) r_usb_write_data_to_sFlash

r_usb_write_data_to_sFlash		
Synopsis	Data reception and writing to the serial flash ROM	
Declaration	static uint8_t r_usb_write_data_to_sFlash(bootloader_ctrl_t *bootloader_param)	
Description	See the flowchart given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_TIMEOUT: Timeout error (10 s) for data reception via the USB ERROR_CODE_VERIFY: Verification error ERROR_CODE_HW_ERROR: HW error occurred. ERROR_CODE_FILE_TRANSFER: Data transfer to the serial flash ROM failed.	
Remark		

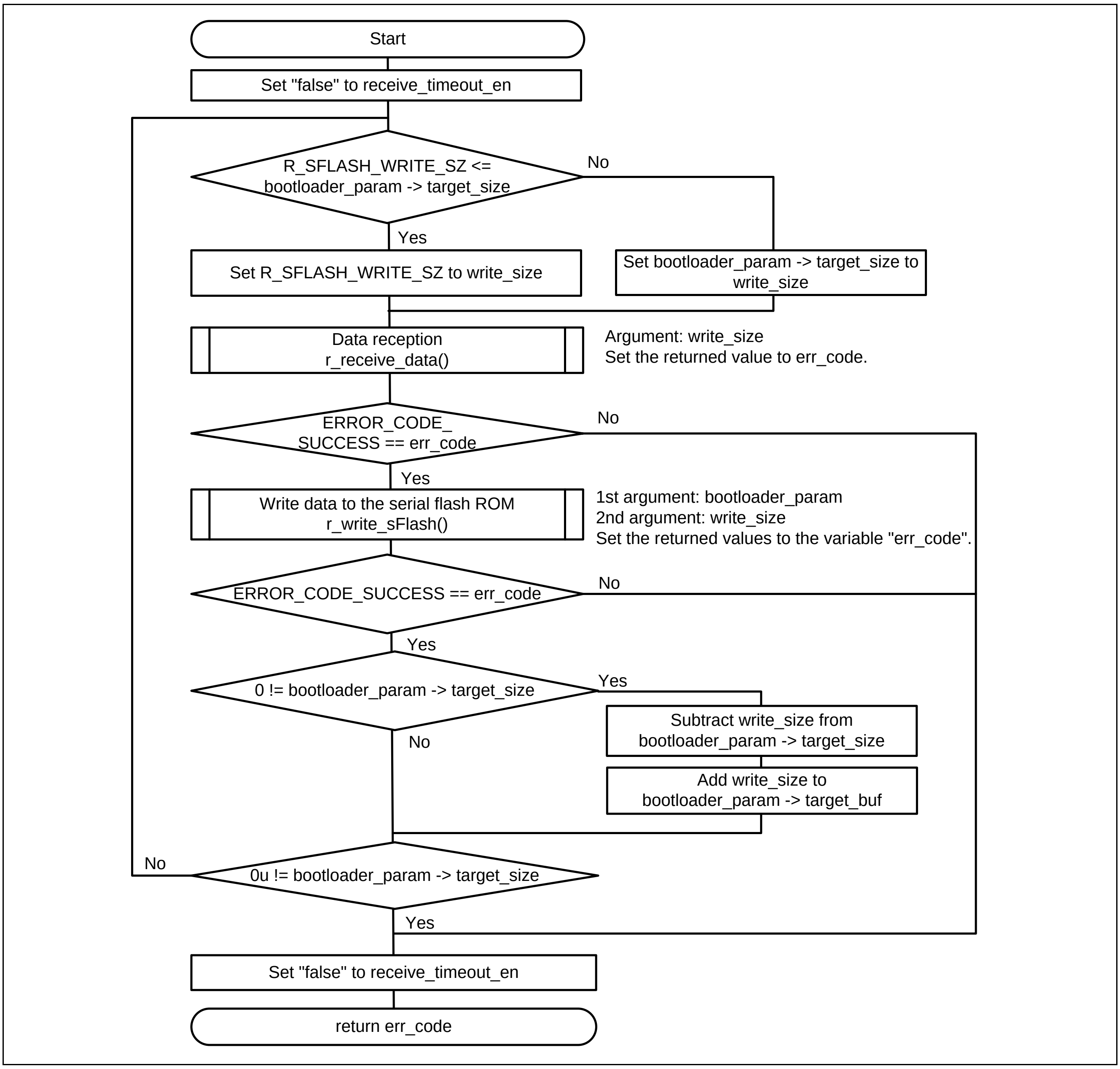


Figure 5.25 r_usb_write_data_to_sFlash Function Processing

(34) r_usb_read_data_from_sFlash

r_usb_read_data_from_sFlash		
Synopsis	Reads data from the serial flash ROM and sends them.	
Declaration	static uint8_t r_usb_read_data_from_sFlash(bootloader_ctrl_t *bootloader_param)	
Description	See the flowchart given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program.
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_HW_ERROR: HW error occurred. ERROR_CODE_FILE_TRANSFER: Data transfer to the serial flash ROM failed.	
Remark		

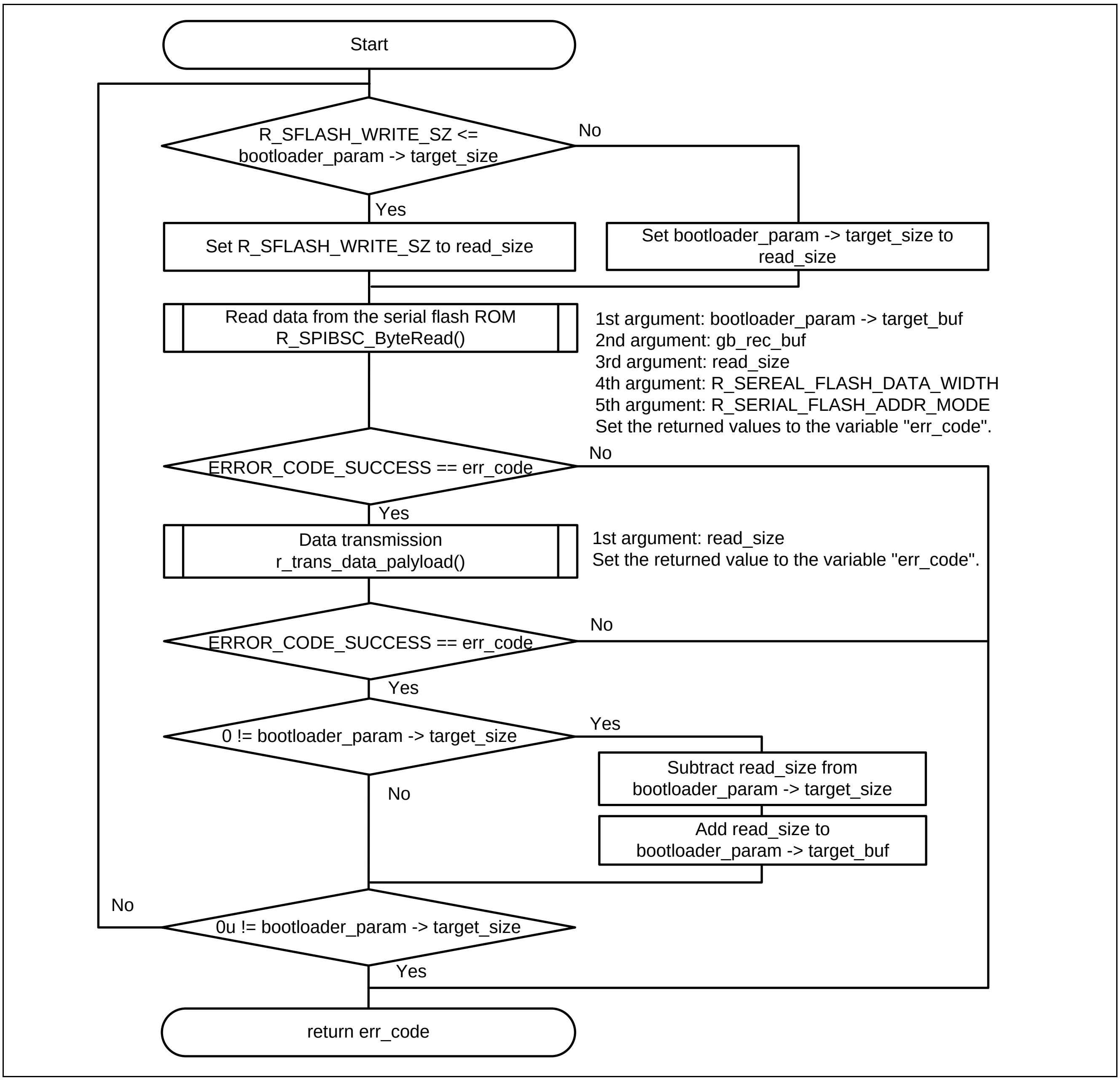


Figure 5.26 r_usb_read_data_from_sFlash Function Processing

(35) r_receive_data

r_receive_data	
Synopsis	Receives data.
Declaration	static uint8_t r_receive_data(uint32_t rec_size)
Description	See the flowchart given below.
Argument	uint32_t rec_size Size of data to be received
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_TIMEOUT: Timeout error (10s) for data reception via the USB ERROR_CODE_HW_ERROR: HW error occurred.
Remark	

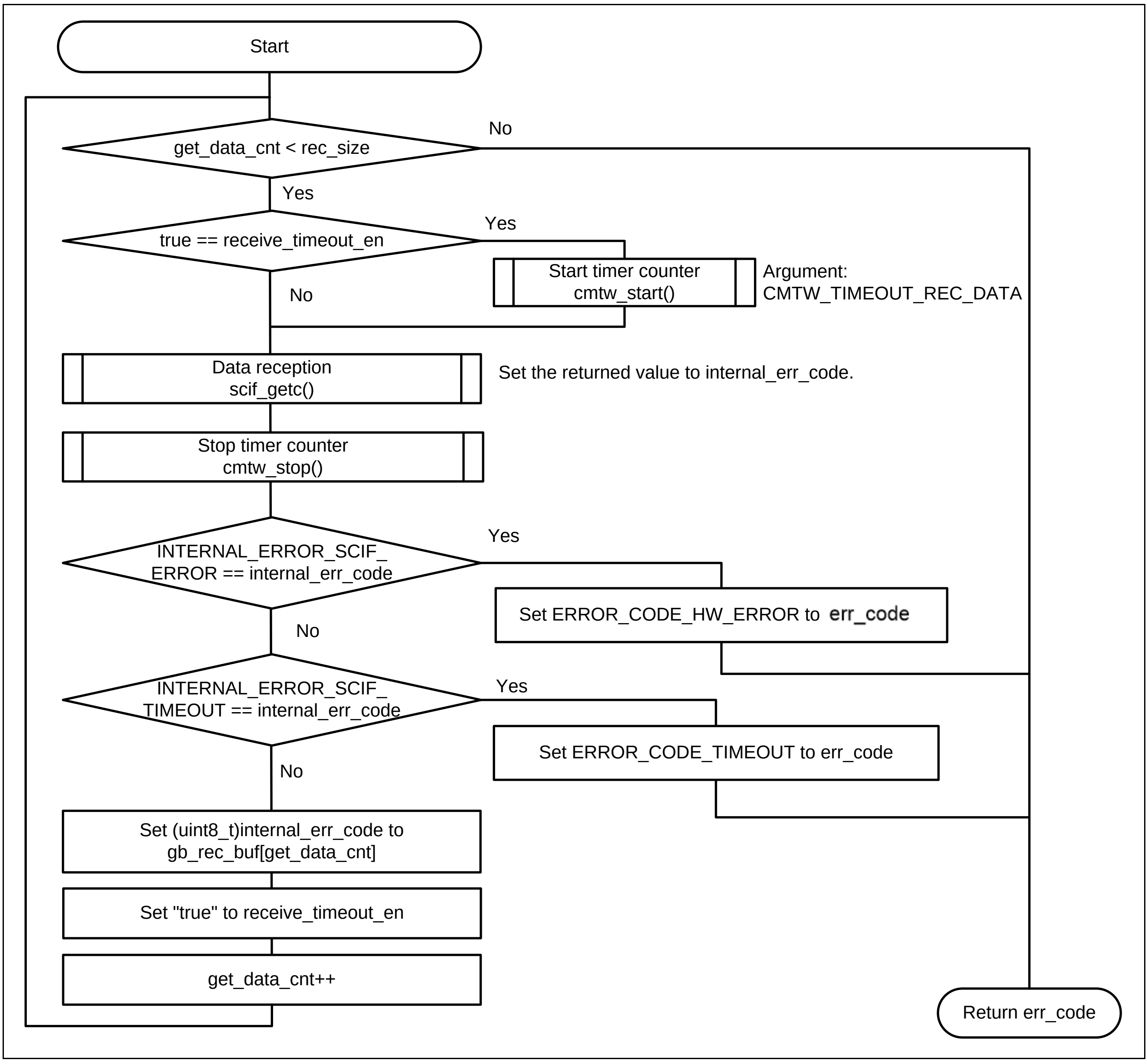


Figure 5.27 r_receive_data Function Processing

(36) r_write_sFlash

r_write_sFlash		
Synopsis	Writes data to the serial flash ROM.	
Declaration	static uint8_t r_write_sFlash(bootloader_ctrl_t *bootloader_param, uint32_t write_size)	
Description	See the flowchart given below.	
Argument	bootloader_ctrl_t *bootloader_param	Pointer to the control parameters for the USB serial writing sample program
	uint32_t write_size	Size of data to be written
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_VERIFY: Verification error ERROR_CODE_FILE_TRANSFER: Data transfer to the serial flash ROM failed.	
Remark		

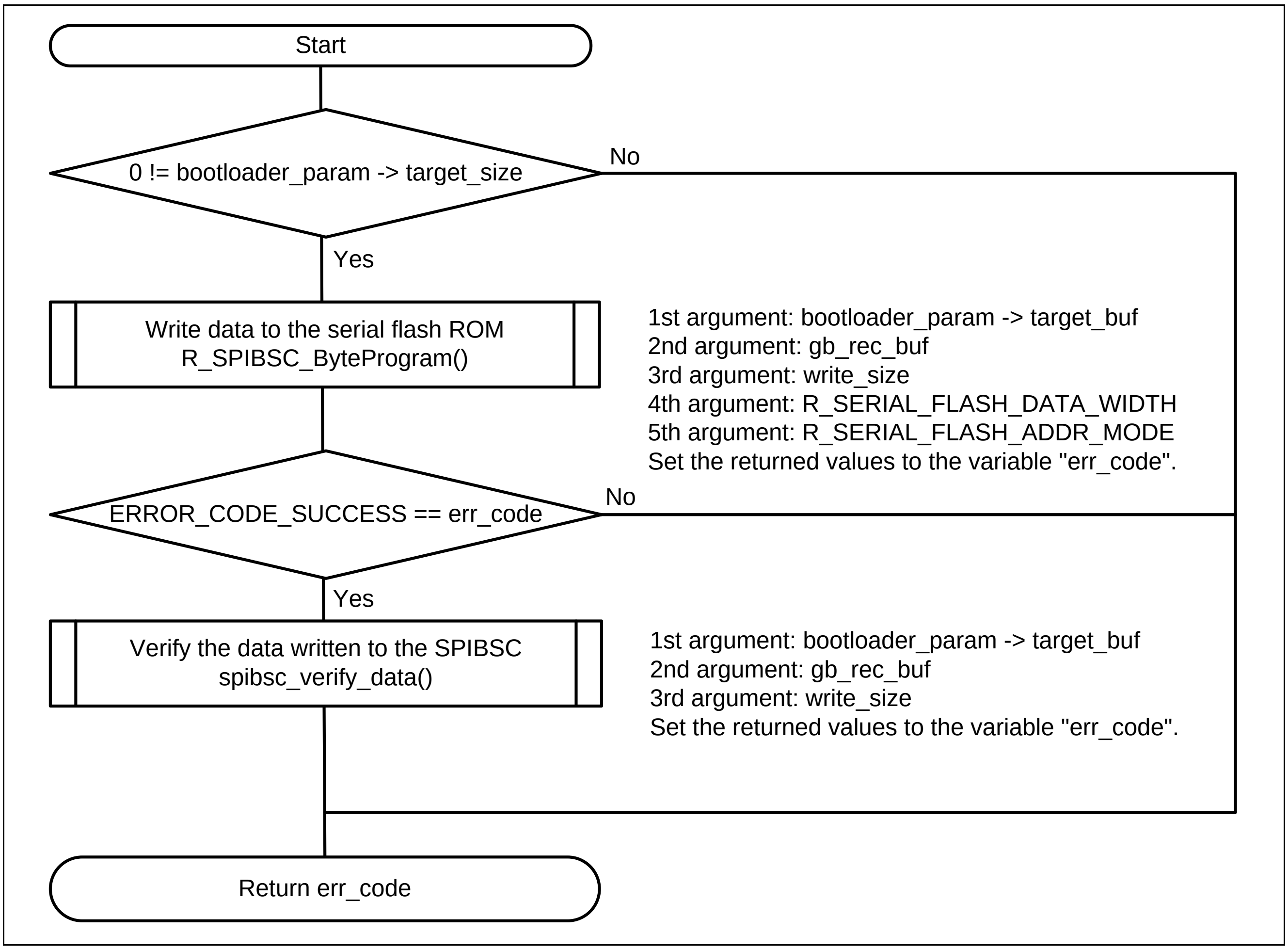


Figure 5.28 r_write_sFlash Function Processing

(37) r_trans_data_payload

r_trans_data_payload		
Synopsis	Sends data.	
Declaration	static uint8_t r_trans_data_payload(uint32_t block_size)	
Description	See the flowcharts given below.	
Argument	uint32_t block_size	Size of data for transmission
Returned value	ERROR_CODE_SUCCESS: Normal end ERROR_CODE_TIMEOUT: Timeout error (10 s) for data transmission via the USB ERROR_CODE_HW_ERROR: HW error occurred	
Remark		

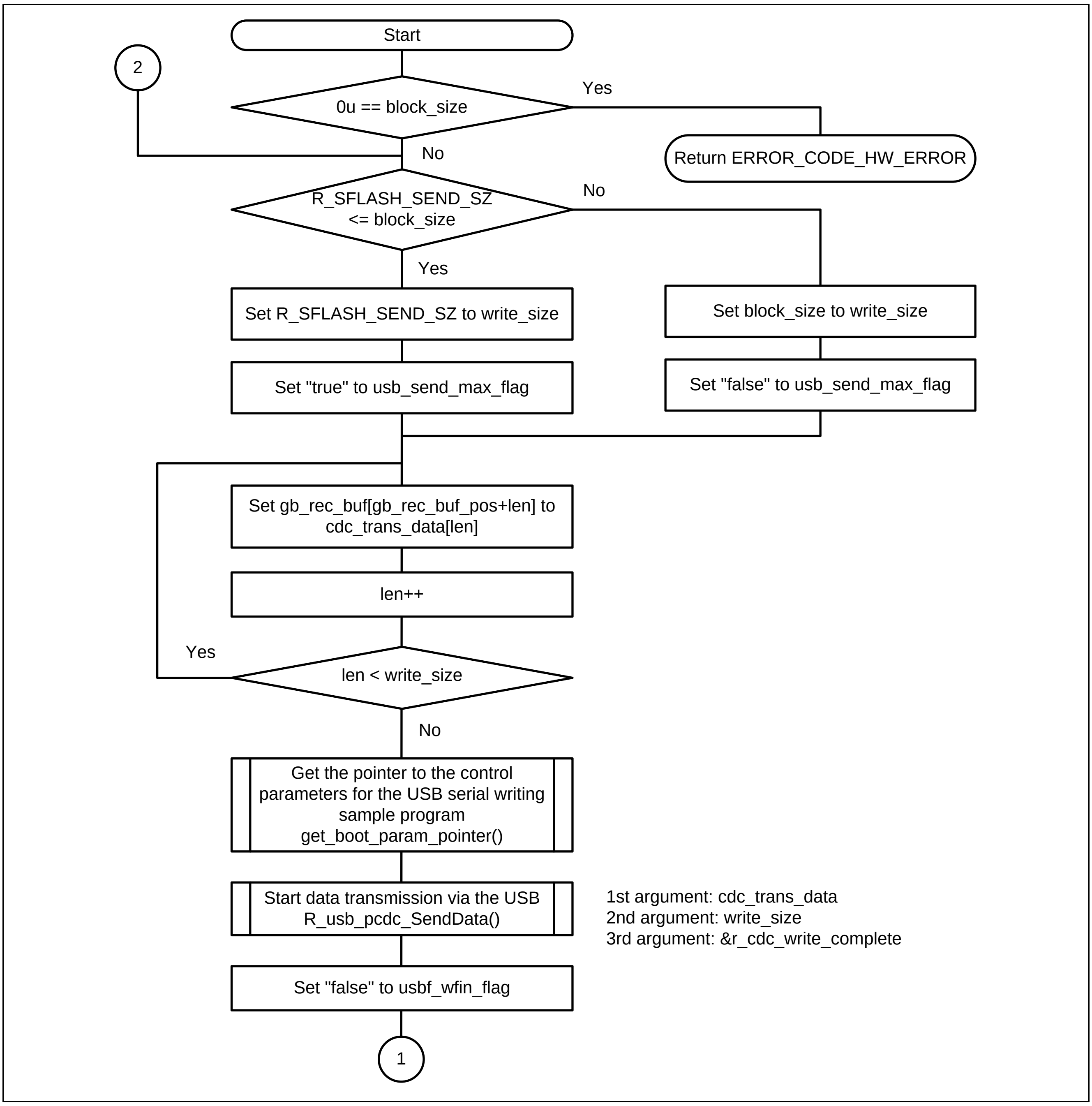


Figure 5.29 r_trans_data_payload Function Processing(1/2)

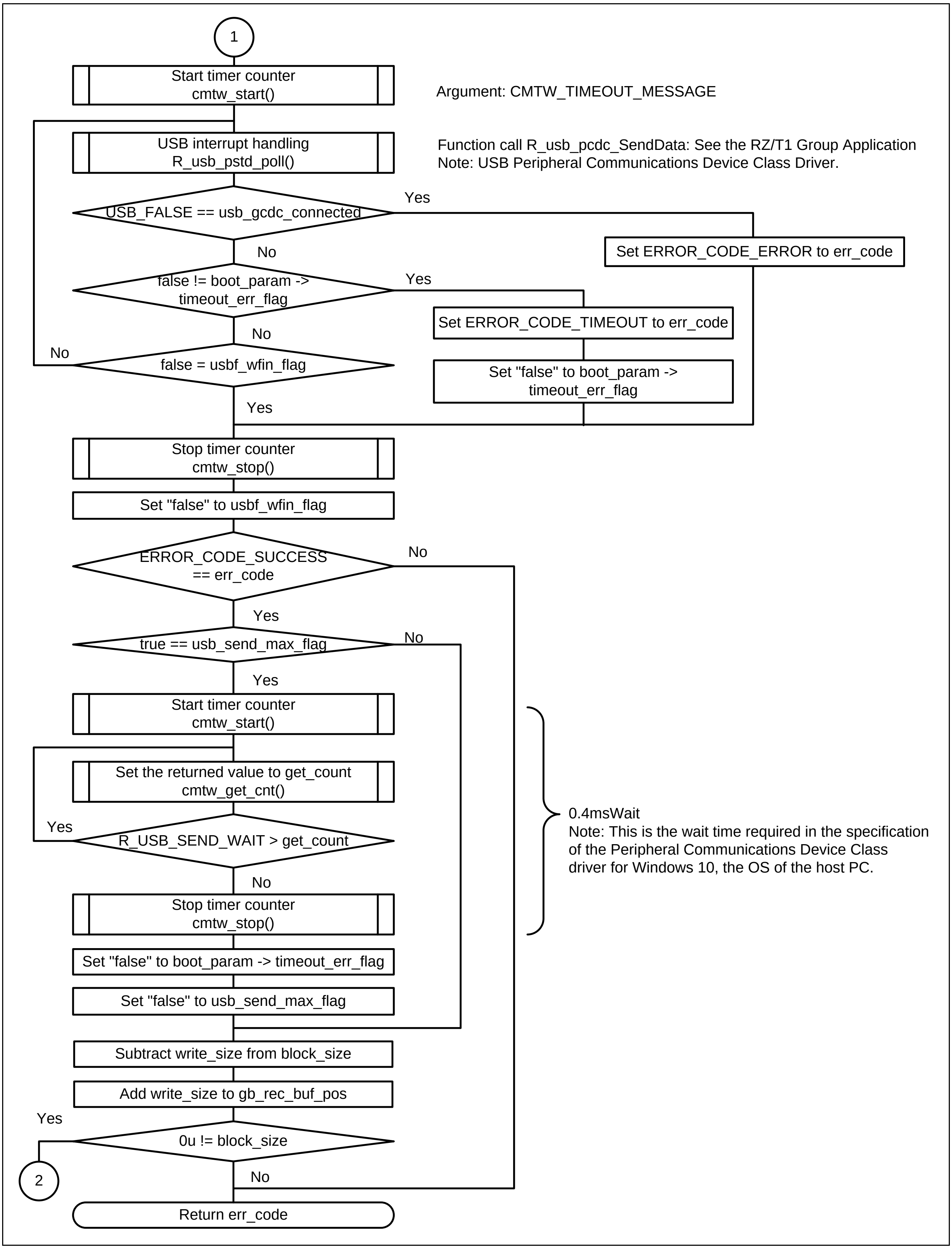


Figure 5.29 r_trans_data_payload Function Processing(2/2)

(38) USB Peripheral Communications Device Class (PCDC)

For the USB Peripheral Communications Device Class (PCDC), see the RZ/T1 Group Application Note: USB Peripheral Communications Device Class Driver.

(39) USB Peripheral Control Driver (PCD)

For the USB Peripheral Control Driver (PCD), see the RZ/T1 Group Application Note: USB Peripheral Communications Device Class Driver.

(40) SPIBSC

The specifications of the SPIBSC for use in the USB serial writing sample program are listed below.

- Address area: 4 bytes (maximum memory size is 64 MB)
- SPI data width: Single (1 bit)
- SPBCLK: 75 MHz

The USB serial writing sample program uses the following commands.

Description	Instruction	Note
Write enable	WREN (0x06)	—
Write register	WRR (0x01)	—
Page programming	PP4B (0x12)	—
Read	FASTREAD4B (0x0C)	—
Sector erase	BE4B (0xDC)	—
Read status register	RDSR (0x05)	—

For detailed specifications of the SPIBSC for use in this sample program, see the RZ/T1 Group Application Note: Serial Flash Sample Program.

5.5 User Program

The user program for use in this sample program is that described in the RZ/T1 Group Application Note: Initial Settings of the Microcomputers Incorporating the R-IN Engine. See that document for detailed specifications.

When the user program copies the user program area for Cortex-M3 from the serial flash ROM to the instruction RAM, the user program refer to the user program information table (Refer to Table 5.1). **Figure 5.30** shows the flowchart of Initialization of the Cortex-M3 core (init_cm3 function). (For EWARM)

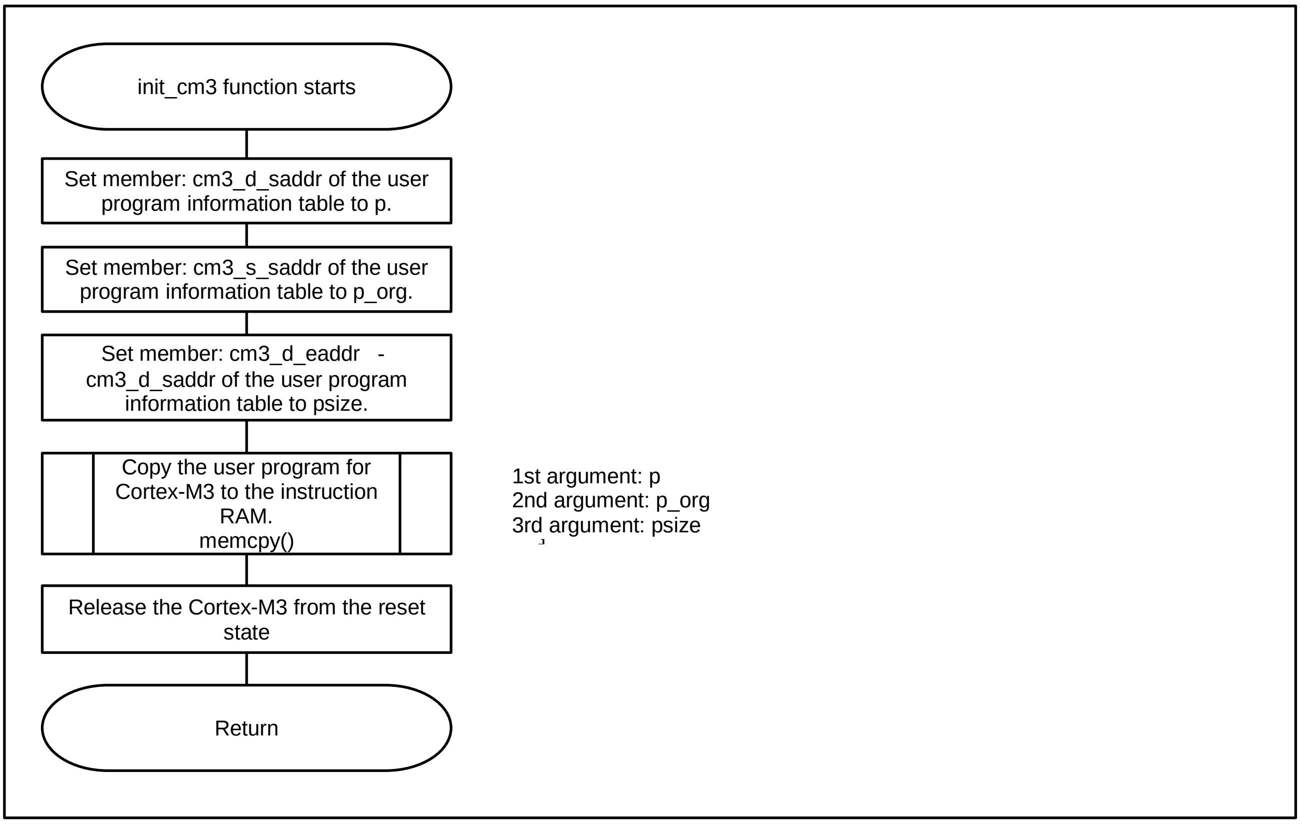


Figure 5.30 init_cm3 Function Processing

6. Procedure of Writing the USB Serial Writing Sample Program

6.1 Connecting the RZ/T1 Evaluation Board

①Connect the host PC, to which TeraTerm has been installed, with the USB connector J6 on the RZ/T1 evaluation board via an USB cable.

②Connect the JTAG connector of the ICE to J10 (ARM JTAG20) and connect the PC for use in development with a USB cable.

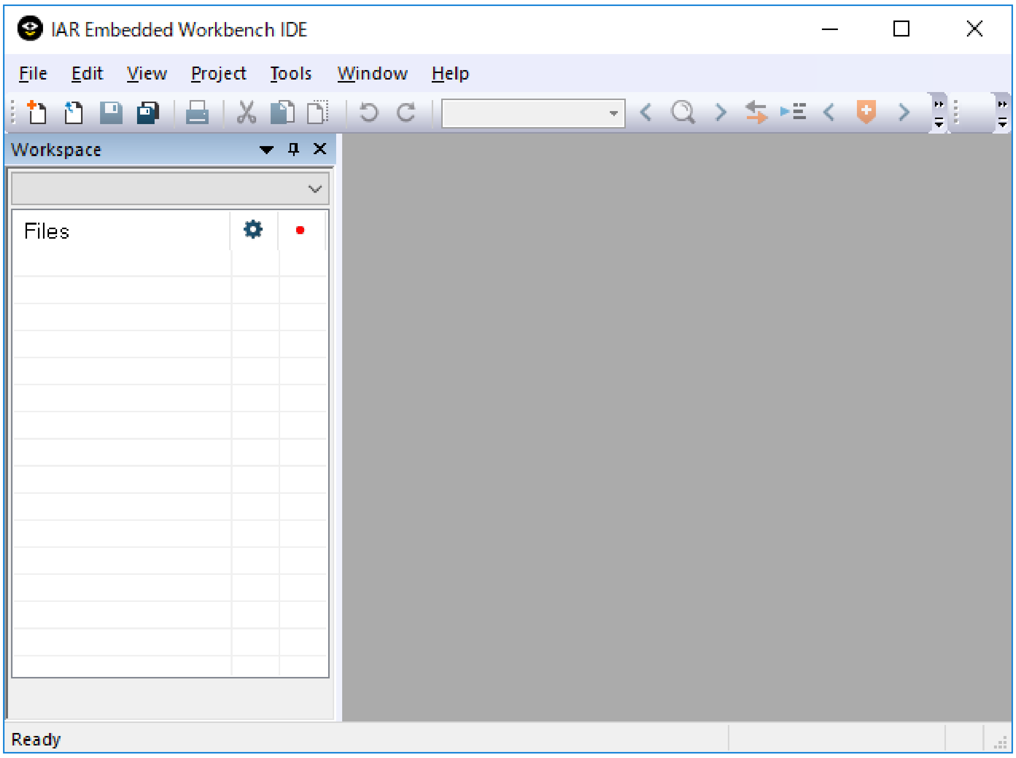
③Connect the DC5V output AC adopter to J17 and supply power.

6.2 Writing the USB Serial Writing Sample Program

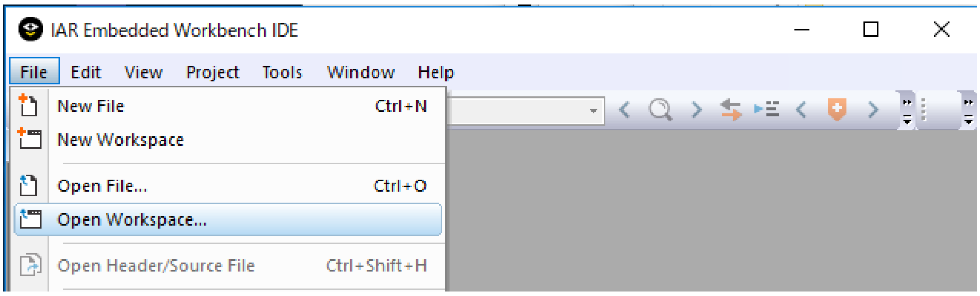
6.2.1 For EWARM

(1) Building the USB Serial Writing Sample Program

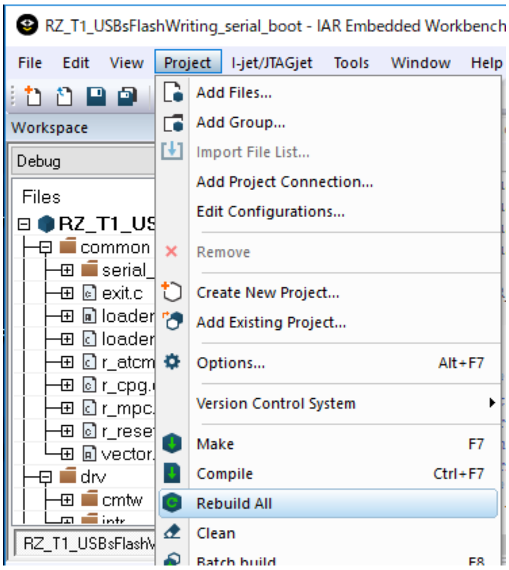
①Start up the IAR Embedded Workbench. From the Windows start menu, select [All programs] > [IAR Embedded Workbench for Arm 8.30.1] > [IAR Embedded Workbench].



②Open a workspace. Select [File] > [Open Workspace…], then double-click on the file "¥workspace¥iccarm¥RZ_T1_R-IN_init¥RZ_T1_R-IN_usb¥CortexR4¥RZ_T1_init_boot¥ RZ_T1_USBsFlashWriting_serial_boot.eww".

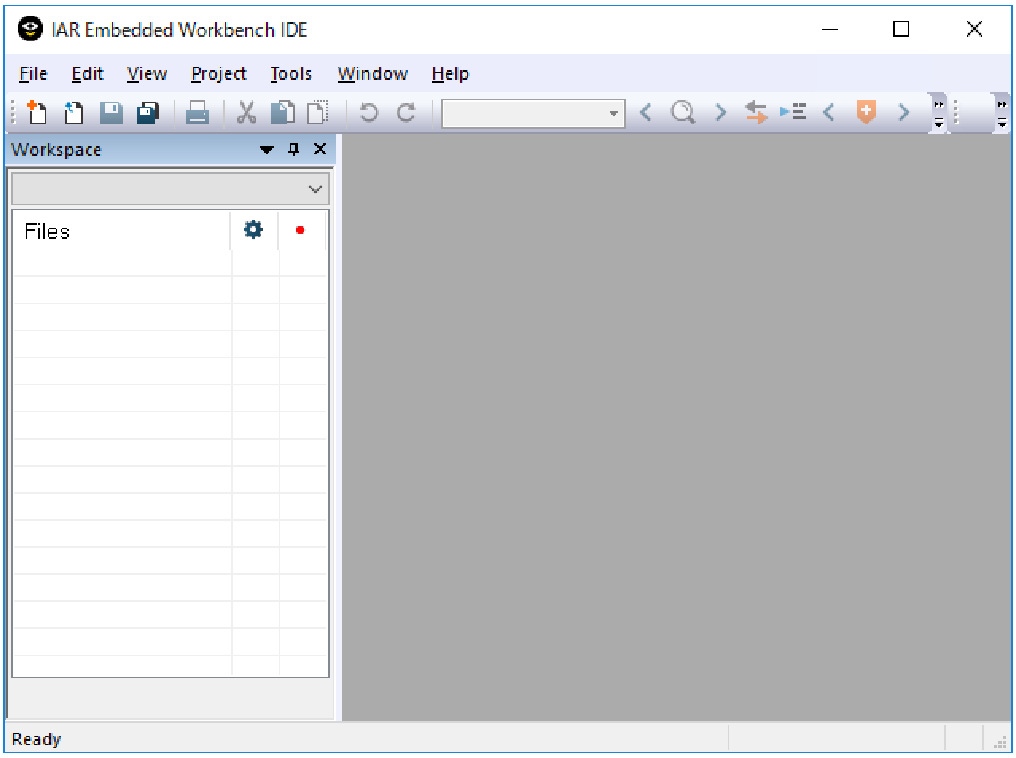


③Execute build. Select [Project] > [Rebuild All].

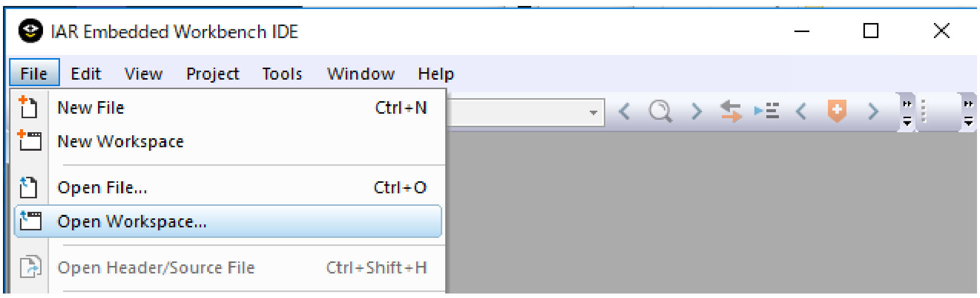


(2) Building and Writing the User Program and the USB Serial Writing Sample Program

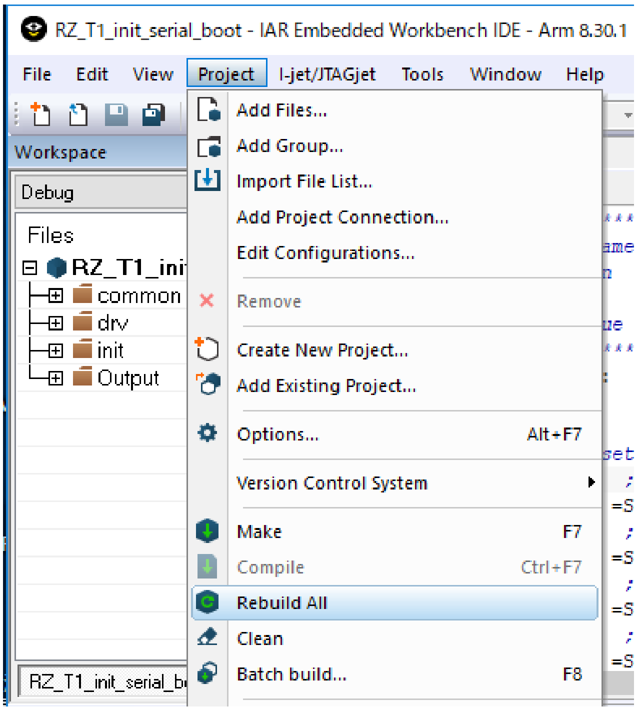
①Start up the IAR Embedded Workbench. From the Windows start menu, select [All programs] > [IAR Embedded Workbench for Arm 8.30.1] > [IAR Embedded Workbench].



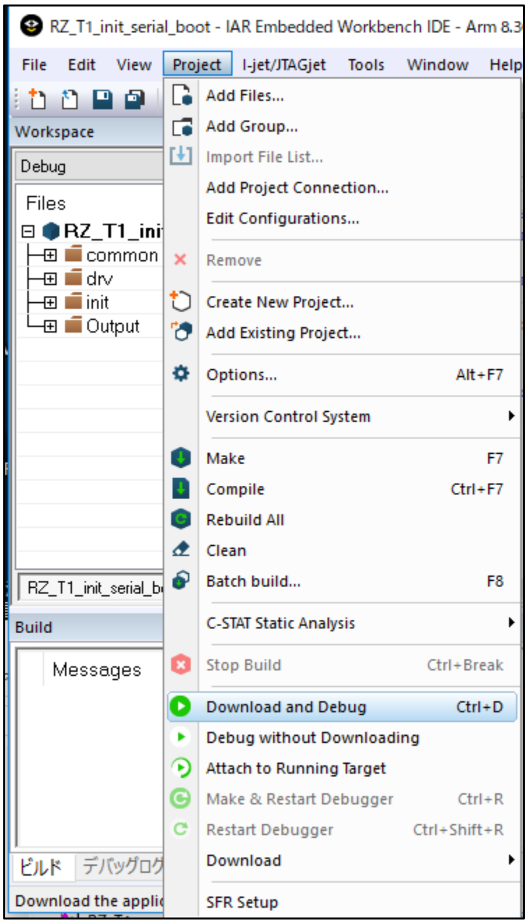
②Open a workspace. Select [File] > [Open Workspace…], then double-click on the file "¥workspace¥iccarm¥RZ_T1_R-IN_init¥RZ_T1_R-IN_init¥CortexR4¥RZ_T1_init_boot¥RZ_T1_init_serial_boot.eww".



③Execute build. Select [Project] > [Rebuild All].

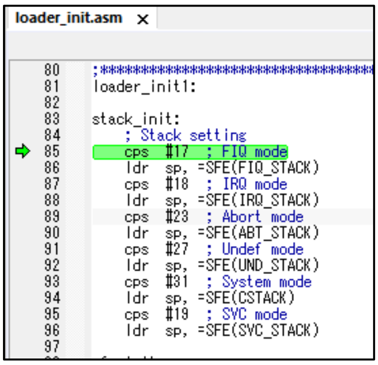


④Write the program to the flash ROM. Select [Project] > [Download and Debug].



On completion of writing of the program to the flash ROM, a break point appears where the stack_init function starts. Terminate the debugger, remove the ICE, and run the RZ/T1 evaluation board in stand-alone mode.

If a breakpoint does not appear where the stack_init function starts, check the connections of the board and repeat the procedure described in section 6, Procedure of Writing the USB Serial Writing Sample Program.

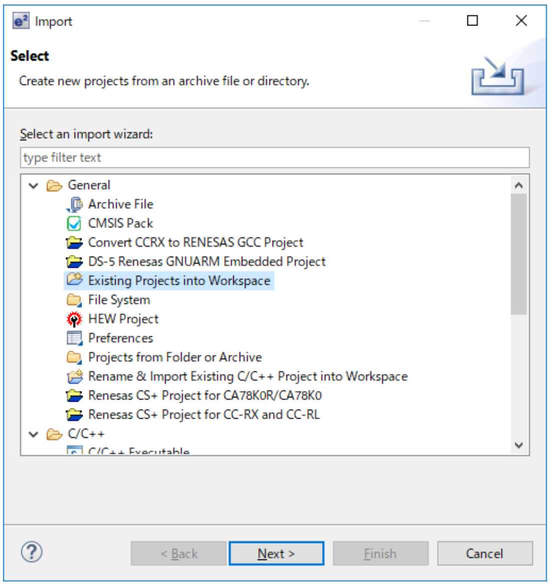


6.2.2 For e[2] studio

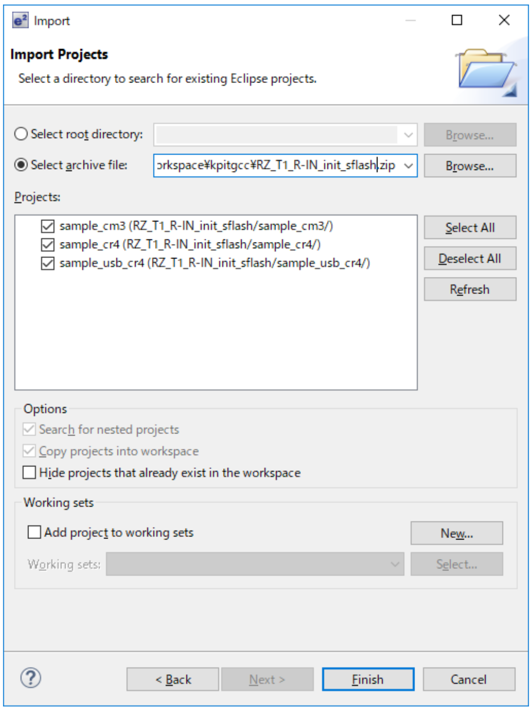
①Create an empty folder on your PC to be used as a workspace, where the sample program will be stored.

②Run e[2] studio. Specify the folder created in step ① as the workspace.

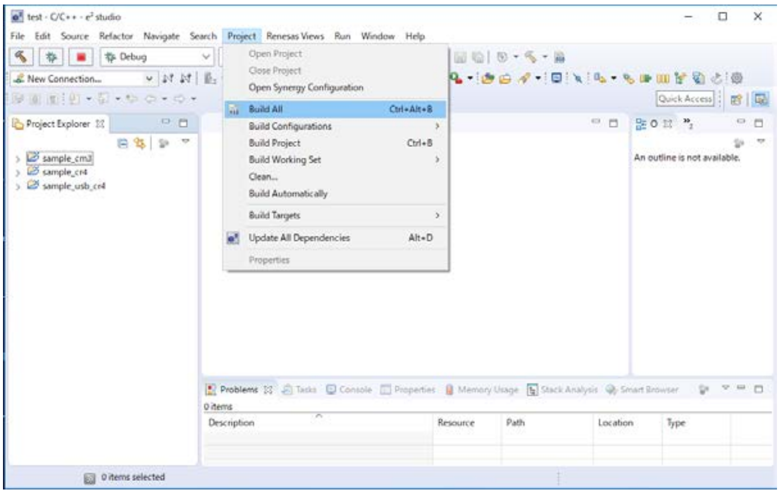
③Open the Import window by selecting [File] then [Import]. In the window, select [General] > [Existing Projects into Workspace], then [Next].



④Select the radio button [Select archive file:]. Then, click [Browse…] and select the compressed sample program "¥workspace¥kpitgcc¥RZ_T1_R-IN_init_sflash.zip " and click [Finish].

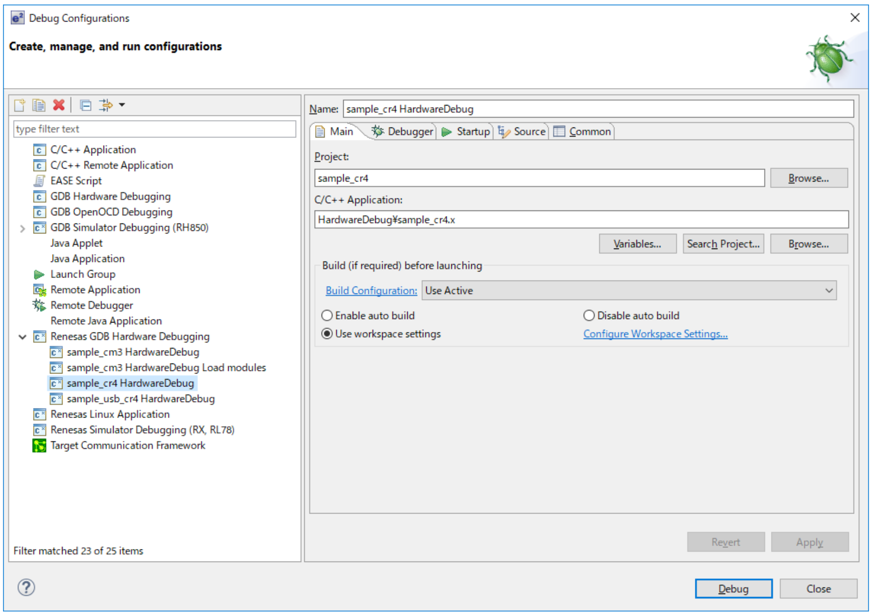


⑤Click [sample_cr4] of the Project Explorer window and go to [Project] and run [Build All].



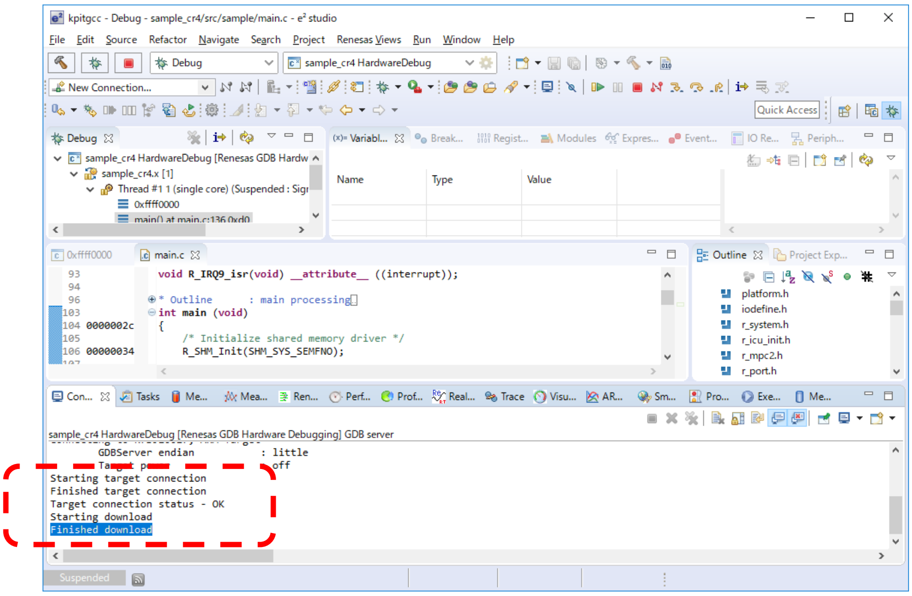
⑥Open the debug configurations window by selecting [Run] > [Debug Configurations…].

⑦While the board is connected to J-LINK, select [Renesas GDB Hardware Debugging] > [sample_cr4 HardwareDebug]. Then click [Debug] and start debugging of the Cortex-R4 core.



On completion of writing program to the flash ROM, the text "Finished download" appears on the console window. Terminate the debugger, remove the ICE, and run the RZ/T1 evaluation board in stand-alone mode.

If "Finished download" text does not appear on the window, check the connections of the board and repeat the procedure described in section 6, Procedure of Writing the USB Serial Writing Sample Program.



7. Procedure for Writing User Program to the Serial Flash ROM

This section describes the case of EWARM from IAR systems unless otherwise stated.

In this package, the user programs that blink LED2 (hereinafter referred to as “target user program”) (¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.bin) are prepared.

This section describes the procedure for writing of the following two patterns.

- Procedure for Writing to the Serial Flash ROM with Using TeraTerm Macro
- Procedure for Writing to the Serial Flash ROM without Using TeraTerm Macro

7.1 Procedure for Writing to the Serial Flash ROM with Using TeraTerm Macro

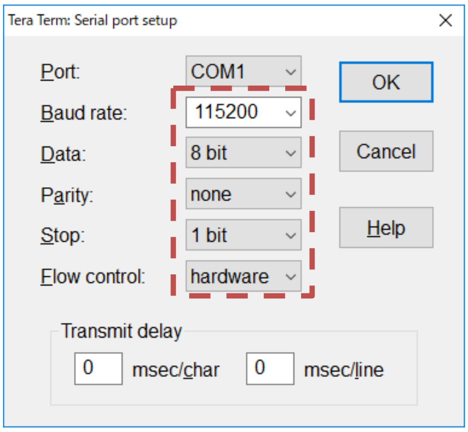
7.1.1 Connecting the RZ/T1 Evaluation Board

①Connect the host PC, to which TeraTerm has been installed, with the USB connector J6 on the RZ/T1 evaluation board via an USB cable.

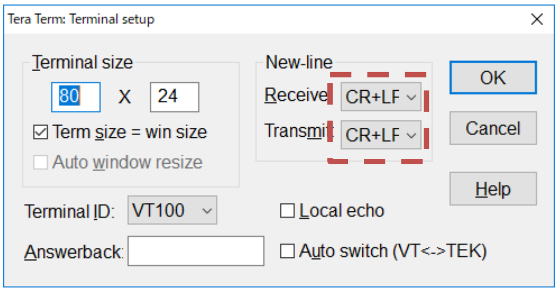
②Connect the DC5V output AC adopter to J17 and supply power. Then, press the reset button while SW3 is being pressed.

③Start the TeraTerm from the host PC.

④Configure communications settings. Select [Setup] then [Serial port…] from the TeraTerm menu.

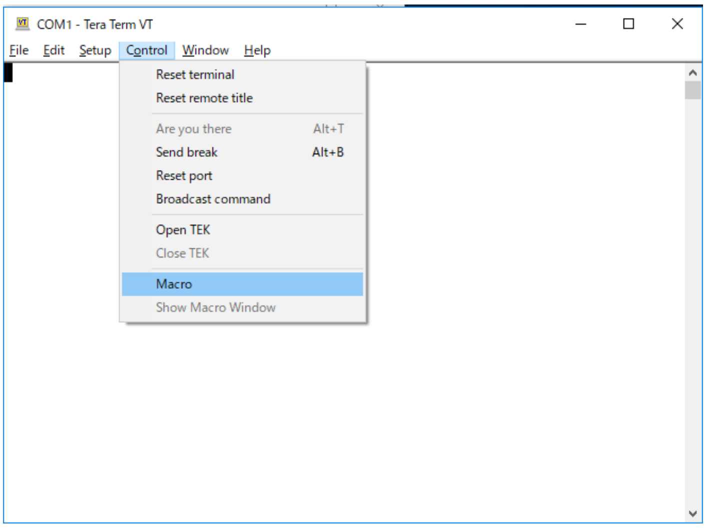


⑤Configure terminal settings. Select [Setup] then [Terminal…] from the TeraTerm menu. In the [New-line] group, select "CR+LF" for both [Receive] and [Transmit] settings.

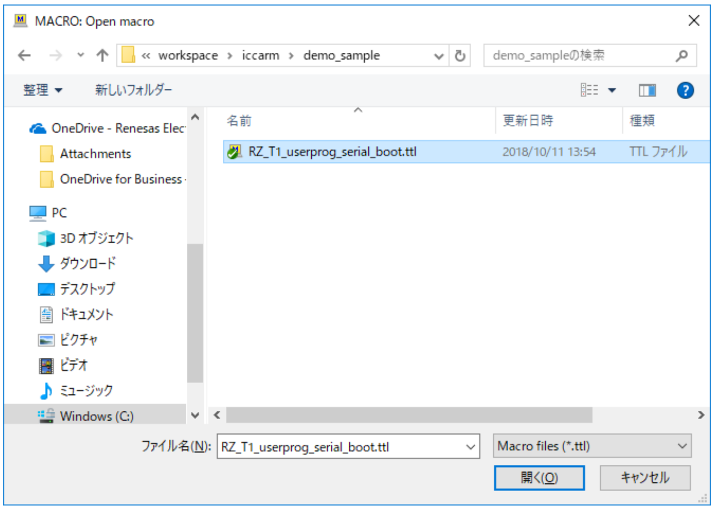


7.1.2 Procedure for Running the TeraTerm Macro

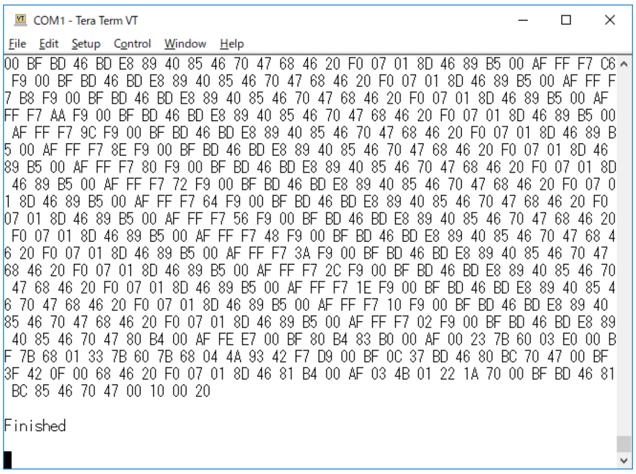
①Select [Control] then [Macro] from the TeraTerm menu.



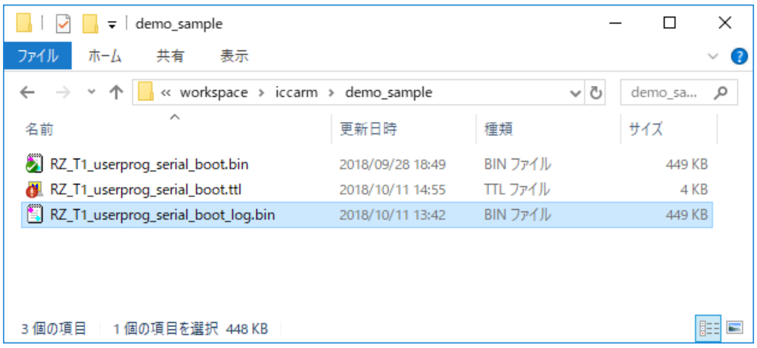
②Select the TeraTerm macro (¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.ttl).



③"Finished" appears on the screen on completion of writing of the program.



After ③ execution, the binary file read from the serial Flash ROM area is generated (RZ_T1_userprog_serial_boot_log.bin) in the same folder as RZ_T1_userprog_serial_boot.bin.
You compare with target user program (RZ_T1_userprog_serial_boot.bin), check that target user program was successfully written to the serial flash ROM.



To start up the user program written to the serial flash ROM, reset the board by turning it off and on, then press the reset button, but not SW3 at this time.

7.2 Procedure for Writing to the Serial Flash ROM without Using TeraTerm Macro

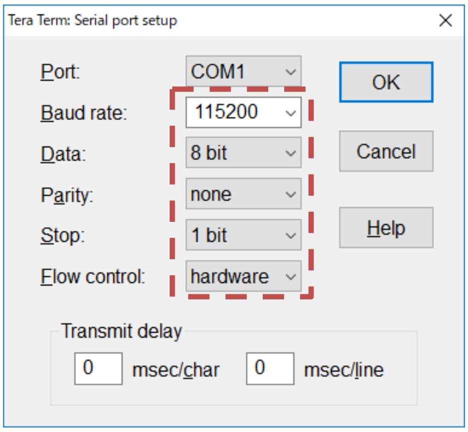
7.2.1 Connecting the RZ/T1 Evaluation Board

①Connect the host PC, to which TeraTerm has been installed, with the USB connector J6 on the RZ/T1 evaluation board via an USB cable.

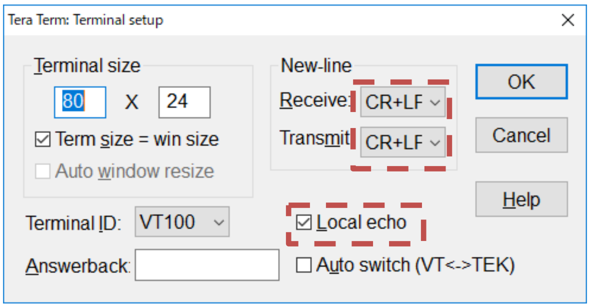
②Connect the DC5V output AC adopter to J17 and supply power. Then, press the reset button while SW3 is being pressed.

③Start the TeraTerm from the host PC.

④Configure communications settings. Select [Setup] then [Serial port…] from the TeraTerm menu.



⑤Configure terminal settings. Select [Setup] then [Terminal…] from the TeraTerm menu. In the [New-line] group, select "CR+LF" for both [Receive] and [Transmit] settings. Put a checkmark on [Local echo] to check input commands.



⑥Select display mode.

The USB serial writing sample program provides two display modes. Select the mode based on the type of input data from the terminal software.

- The start-of-command character is in use

  In this mode, ">" will appear as the start-of-command character. This mode is selected by entering "D" or "d".

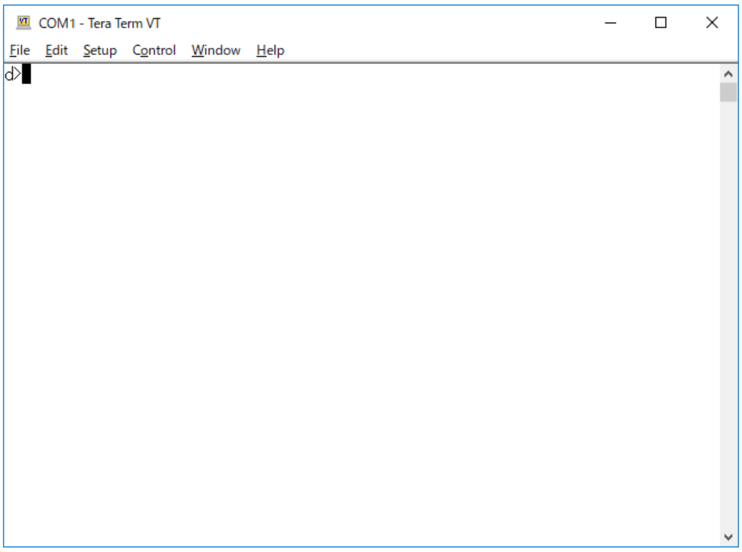
- The start-of-command character is not in use

  In this mode, ">" will not appear as the command start character. This mode is selected by entering any key other than "D" and "d".

Note that TeraTerm treats the start-of-command character ">" as part of the log when it reads binary data from the serial flash ROM by using the Log function. In this case, select the mode in which the start-of-command character is not in use.

How to read binary data stored in the serial flash ROM by using the Log function is detailed in section 8.2.2, Reading from the Serial Flash ROM Area.

This is the screen of the terminal software when the start-of-command character is in use.



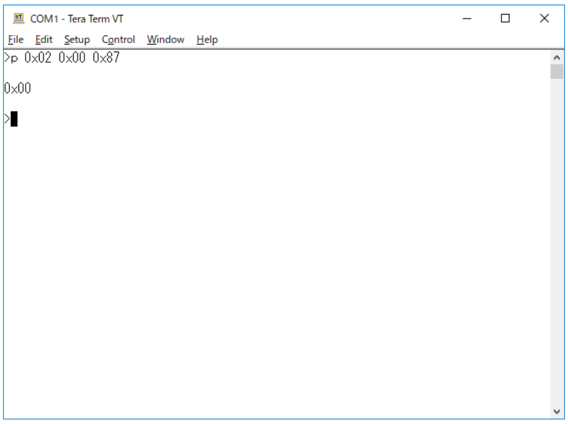
A successful startup of the USB serial writing sample program lights up LED10. If the LED is not lit up, turn off the board and restart from step ①.

7.2.2 Writing User Programs

Throughout this section, terminal software screens are those with the start-of-command character ">".

(1) Releasing Write Protection on the Serial Flash ROM

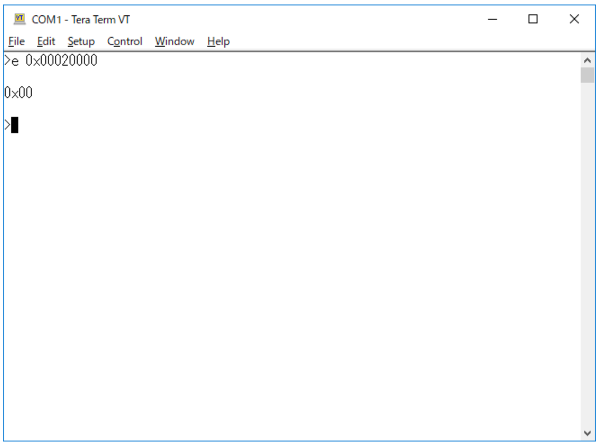
Entering "p 0x02 0x00 0x87" releases a write protection on the serial flash ROM. "0x00" appears on completion of the task.



(2) Sector Erase

Entering "e 0x00020000"commands a sector erase on the area of the serial flash ROM. "0x00" appears on completion of the task. Use a sector erase command to erase the area, larger than the size of the target user program. Refer to the datasheet for the serial flash ROM in use for the sector size.

Do not turn off the board while sector erase is in progress.

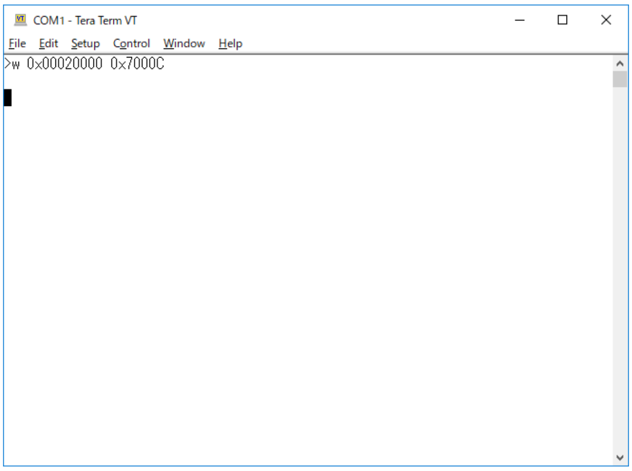


(3) Releasing Write Protection on the Serial Flash ROM

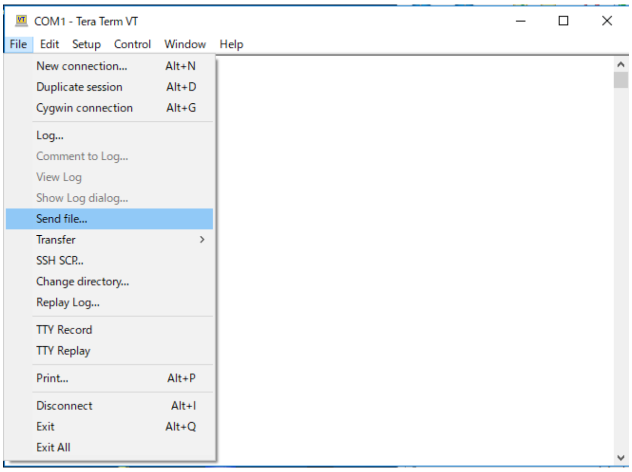
Again, release the write protection on the flash ROM by referring to section (1), Releasing Write Protection on the Serial Flash ROM.

(4) Writing to the Serial Flash ROM

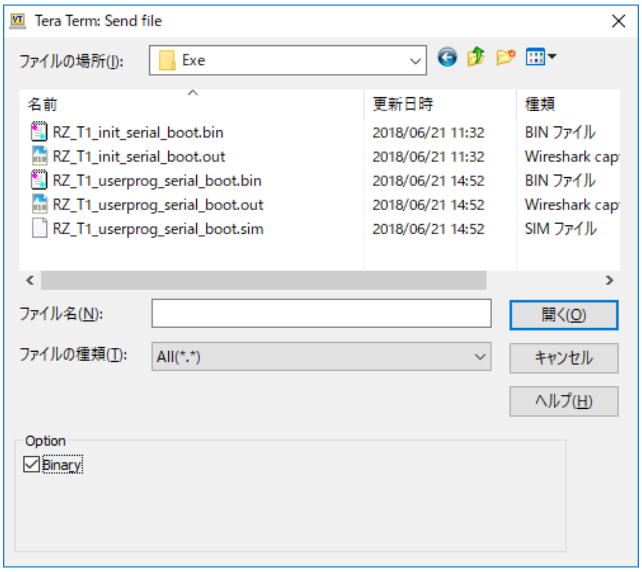
①Entering "w 0x00020000 0x7000C (size of the target user program)" commands writes the target user program to the serial flash ROM. Do not turn off the board while writing to the ROM is in progress.



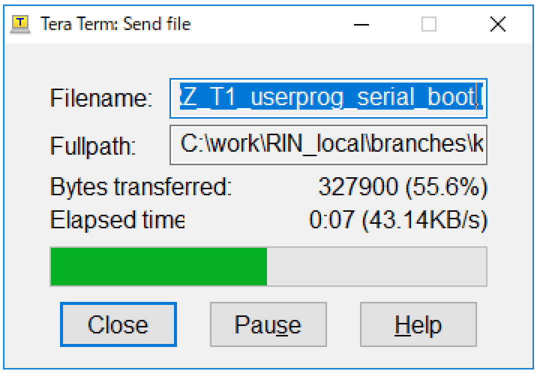
②Select [File] > [Send file…] from the TeraTerm menu.



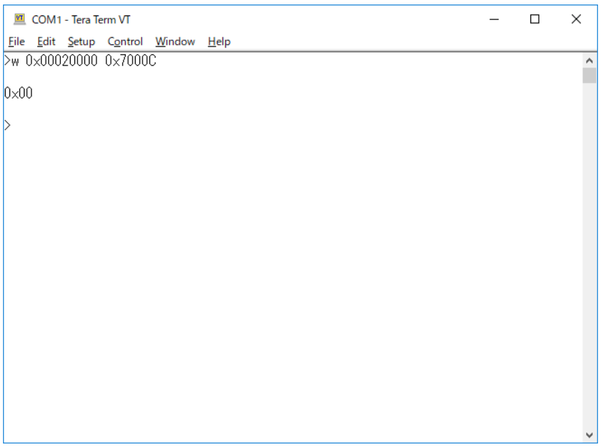
③Select the file for transmission, "¥workspace¥iccarm¥demo_sample¥RZ_T1_userprog_serial_boot.bin". Here, put a checkmark on [Binary] in the option section.



④Transmission starts.



⑤"0x00" appears on the screen on completion of writing of the program. To start up the target user program written to the serial flash ROM, reset the board by turning it off and on, then press the reset button, but not SW3 at this time.



8. Specifications of Commands for use with USB Serial Writing Sample Program

Throughout this section, terminal software screens are those with the start-of-command character ">".

8.1 List of Commands

The commands available for the program are listed in the table below. Execute these commands on the terminal software.

Table 8.1 List of Commands

No.	Command	Name	Description
1	"w" or "W"	Writing user program to the serial flash ROM area	Writes the user program to the serial flash ROM area by using the PP4B (0x12) command.
2	"r" or "R"	Reading data from the serial flash ROM area	Reads the user program from the serial flash ROM area by using the FASTREAD4B (0x0C) command.
3	"e" or "E"	Sector erase	Erases the sector which includes the specified address by using the BE4B (0xDC) command.
4	"p" or "P"	Protection control	Releases write protection on the serial flash ROM by using the WRR (0x01) command.

The address used in the command is treated as the address in the serial flash ROM (0x00000000 to 0x040000000).

8.2 Detail of Commands

8.2.1 Writing User Program to the Serial Flash ROM Area

On reception of the binary data of the user program from the terminal software of the host PC, this command writes the user program to the serial flash ROM. Before executing this command, execute the sector erase and protection control commands.

(1) Command format

w^address^size[CR+LF]

^: space

[CR+LF]: Newline code

(2) Usage examples

w 0x00000000 0x00000001

W 0X00000000 0X00000001

(3) Arguments

address: Base address in hexadecimal notation prefixed by 0x, e.g. 0x00000000

size: Write size in hexadecimal notation prefixed by 0x, e.g. 0x00000001

The binary data of the user program will be sent after the "w" command is executed.

(4) Returned values

0x00: Normal end

0xFE: Parameter error or command format error

0xFC: Verification error

0xFB: Non-supported command is received.

0xFA: Timeout error (10 s) while waiting for binary data

(5) Note

Range of setting values for the arguments are as follows.

address: 0x00000000 to 0x03FFFFFF (the address assigned to the serial flash ROM)

size: 0x00000001 to 0x04000000

If the amount of binary data for the user program is larger than that specified in the argument "size", the excess data will not be written to the ROM. If you need to send the excess as well, the data must be handled as that for a subsequent command. Be sure to send binary data with the size specified in "size".

If address + size is larger than 0x04000000, Returned value is parameter error.

8.2.2 Reading from the Serial Flash ROM Area

This command reads binary data stored in the serial flash ROM and send it to the host PC. Execute the protection control command before executing this command.

(1) Command format

r^address^size[CR+LF]

^: space

[CR+LF]: Newline code

(2) Usage examples

r 0x00000000 0x00000001

R 0X00000000 0X00000001

(3) Arguments

address: Base address in hexadecimal notation prefixed by 0x, e.g. 0x00000000

size: Read size in hexadecimal notation prefixed by 0x, e.g. 0x00000001

(4) Returned values

A returned value will not appear for a normal end.

0xFE: Parameter error or command format error

0xFB: Non-supported command is received

(5) Note

Range of setting values for the arguments are as follows.

address: 0x00000000 to 0x03FFFFFF (the address assigned to the serial flash ROM)

size: 0x00000001 to 0x04000000

If address + size is larger than 0x04000000, Returned value is parameter error.

An example is given below for reading data of the size 0x100 from the serial flash ROM of the address 0x00000000.

- Entering the command on TeraTerm

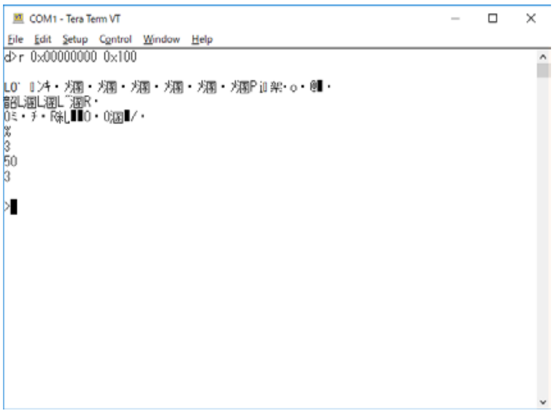
  Here is the command format:

  "r␣0x00000000␣0x100"⏎
  (address) (size)

[When the start-of-command character is in use]

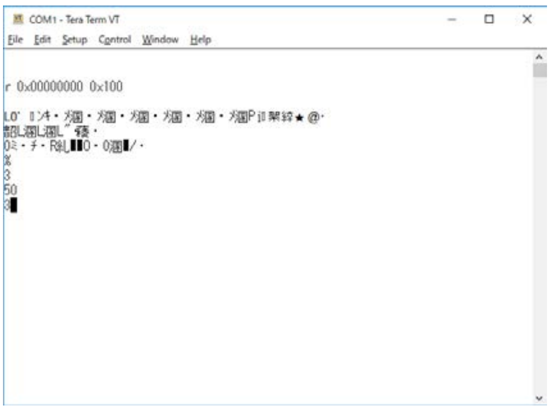
The TeraTerm screen in this mode is given below.

The start-of-command character ">" appears on the screen on completion of data transmission to TeraTerm.



[When the start-of-command character is not in use]

The TeraTerm screen in this mode is given below.



In this mode, you don't have a start-of-command character ">" to tell you if the specified amount of binary data is being displayed or not. In this case, use the TeraTerm Log function (go to [File] then [Log…]) when you store binary data in a file.

How to store binary data in a file by using the Log function is described below.

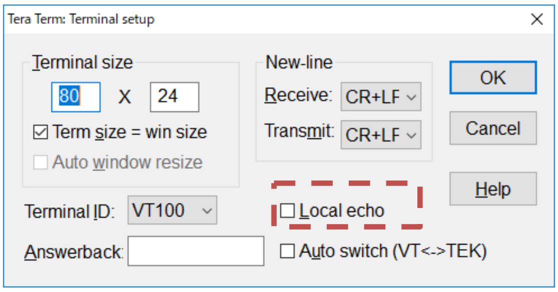
①Change screen of the TeraTerm to hex output mode for increasing the reception speed of TeraTerm. Enter the [Shift+ESC] on the screen of TeraTerm.

To enable hex output mode, [TERATERM.INI] file (default: C:¥Program Files (x86)¥teraterm¥TERATERM.INI) is opened before TeraTerm is started. And [TERATERM.INI] file is rewritten to Debug=on, DebugModes=hex.

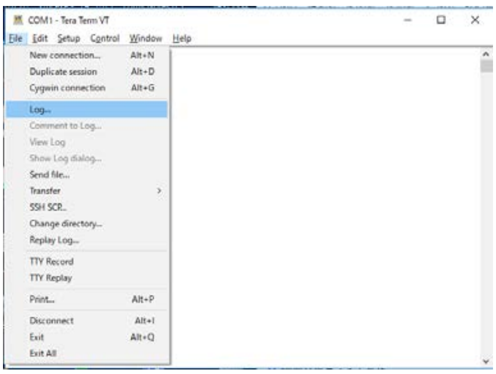
```
; Display all characters (debug mode)
Debug=on
; Debug mode type which can be selected by user.
;   on|all    = All types
;   off|none  = Disabled debug mode
;   normal    = usual teraterm debug mode
;   hex       = hex output
;   noout     = disable output completely
DebugModes=hex
```

Figure 8.1 TERATERM.INI

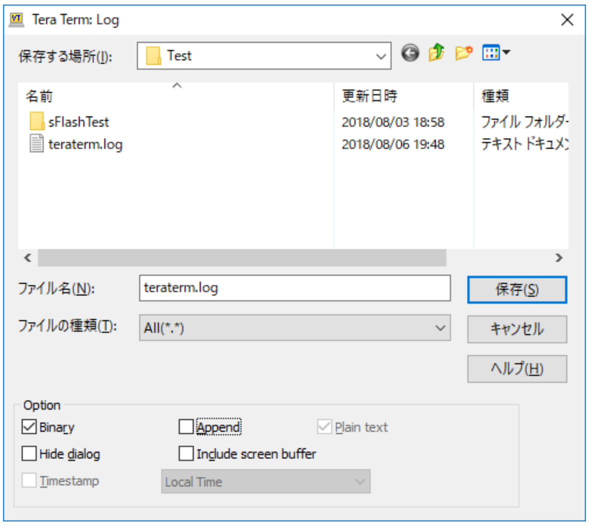
②Configure terminal settings. Select [Setup] then [Terminal…] from the TeraTerm menu. Uncheck a checkmark on [Local echo] in order not to save input commands to file.



③Select [File] then [Log] to use the TeraTerm Log function before executing "R" command.

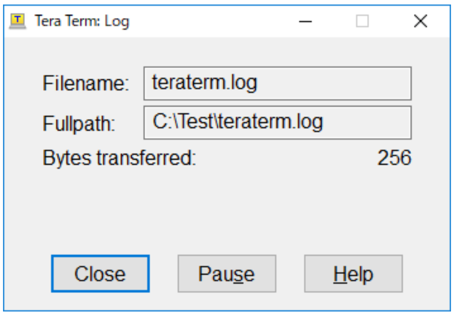


④Select the file, "teraterm.log", to which the binary data will be stored. Here, put a checkmark on [Binary] in the option section.



⑤Execute "R" command.

⑥Check the size of the binary data and click [Close].



Always perform logging without a start-of-command character when you store binary data to a file through TeraTerm. Otherwise, ">" characters will be treated as part of the binary data and stored in the file.

8.2.3 Sector Erase

This command erases sectors of the serial flash ROM. Execute the protection control command before executing this command.

(1) Command format

e^address[CR+LF]

^: space

[CR+LF]: Newline code

(2) Usage examples

e 0x00000000

E 0X00000000

(3) Argument

address: Sector address in hexadecimal notation prefixed by 0x, e.g. 0x00000000

(4) Returned values

0x00: Normal end

0xFE: Parameter error or command format error

0xFB: Outside the valid range for the command

(5) Note

Range of setting values for the argument is as follows.

address: 0x00000000 to 0x03FFFFFF (the address assigned to the serial flash ROM)

If address + size is larger than 0x03FFFFFF, Returned value is parameter error.

8.2.4 Protection Control

This command releases write protection on the serial flash ROM.

(1) Command format

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

^: space

[CR+LF]: Newline code

(2) Usage examples

p 0x02 0x00 0x87

P 0X03 0X00 0X01 0X02 0X03

(3) Arguments

size: The total number of "data" arguments in hexadecimal notation prefixed by 0x, e.g. 0x00000001

data: Data to be written to the serial flash ROM in double-digit hexadecimal notation including 0x, e.g. 0x00000001.

"data" can be specified in accordance with the specification of the flash ROM in use.

(4) Returned values

0x00: Normal end

0xFE: Parameter error or command format error

0xFB: Outside the valid range for the command

(5) Notes

- Range of values for "size": 0x01 to 0x04. This should correspond with the number of arguments "data", i.e., 0x01 for only data1.
- Data will be written into the serial flash ROM as that of the protection control command.

  "data" depends on the specification of the serial flash ROM for use. Refer to the datasheet of your serial flash ROM.

Website and Support
Renesas Electronics website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision History	RZ/T1 Group Application Note: Sample Program for Writing Serial Flash ROM via the USB

Rev.	Date	Description	
		Page	**Summary**
1.00	Oct. 31, 2018	—	First Edition issued
2.00	Sep. 30, 2019	P.10, P.14 - P.15, P.65	Add User Program Information Table so that only user program area can be rewritten. 5.1 Loader Program Operation Overview 5.3 Section Assignment for the Sample Program 5.5 User Program
		P.23, P.25, P.71-P.72	Delete the Cortex-M3 project in the USB serial writing sample program project. 5.4.3 Software Description (4) 5.4.3 Software Description (5) 6.2.2 For e^2 studio

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.