RX63N Group

R01AN2098EJ0100
Rev. 1.00
Sep. 10, 2014

RX63N Group Flash Programmer (Boot Mode) Using the Renesas Starter Kit+ for RX63N

Abstract

This document describes a flash programmer for RX63N Group using the Renesas Starter Kit+ for RX63N (hereinafter referred to as RSK+RX63N).

The rewrite target is the RX63N Group. Boot mode is used for rewriting the user area in the RX63N Group.

Products

Flash programmer: RX63N Group (ROM size: 1 Mbyte to 2 Mbytes)

Target MCU: RX63N Group (ROM size: 256 Kbytes to 1 Mbyte)

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

The flash programmer runs on the RSK+RX63N. After starting the target RX63N Group MCU in boot mode, the flash programmer rewrites the user area in the RX63N Group using asynchronous serial communication.

Table 1.1 lists the Peripheral Functions and Their Applications, and Figure 1.1 shows a Flash Programmer Usage Example.

Channel 0 (SCI0) in the serial communications interface is used for asynchronous serial communication.

The communication data format and output format are as follows.

Start bit: 1 bit
Transfer data: 8 bits
Parity bit: None
Stop bit: 1 bit
Bit rate: 19,200 bps (until response to the new bit rate selection command)
1.5 Mbps (after the programming/erasure state transition command)
Output format: CMOS output

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
SCI0	Asynchronous serial transmission and reception
CMT0	Timer for wait time
I/O ports	Control for boot mode, LCD output

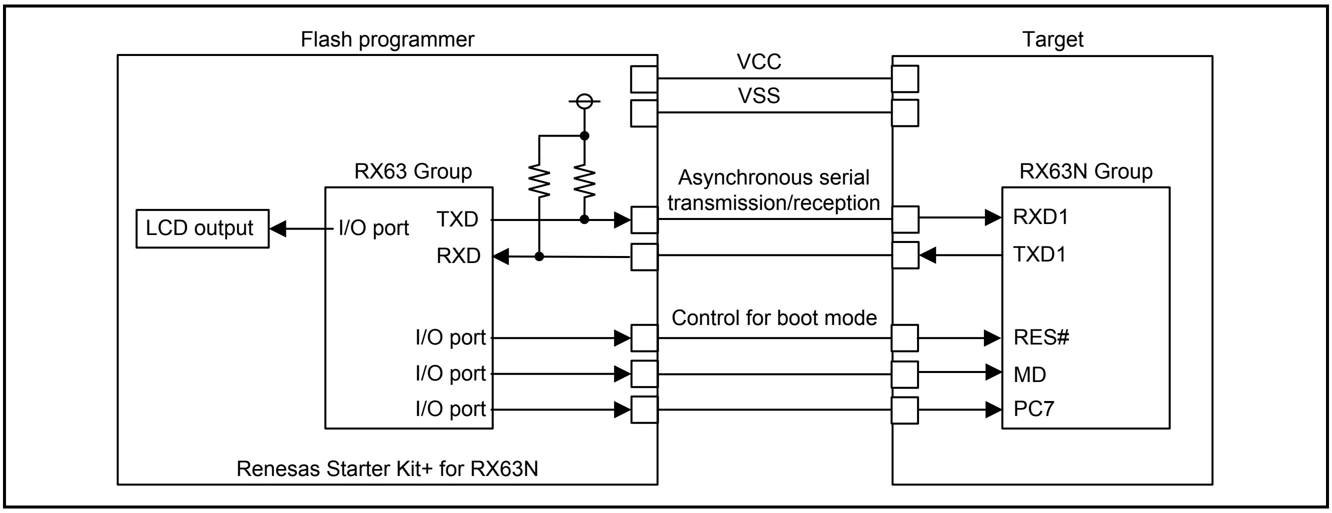
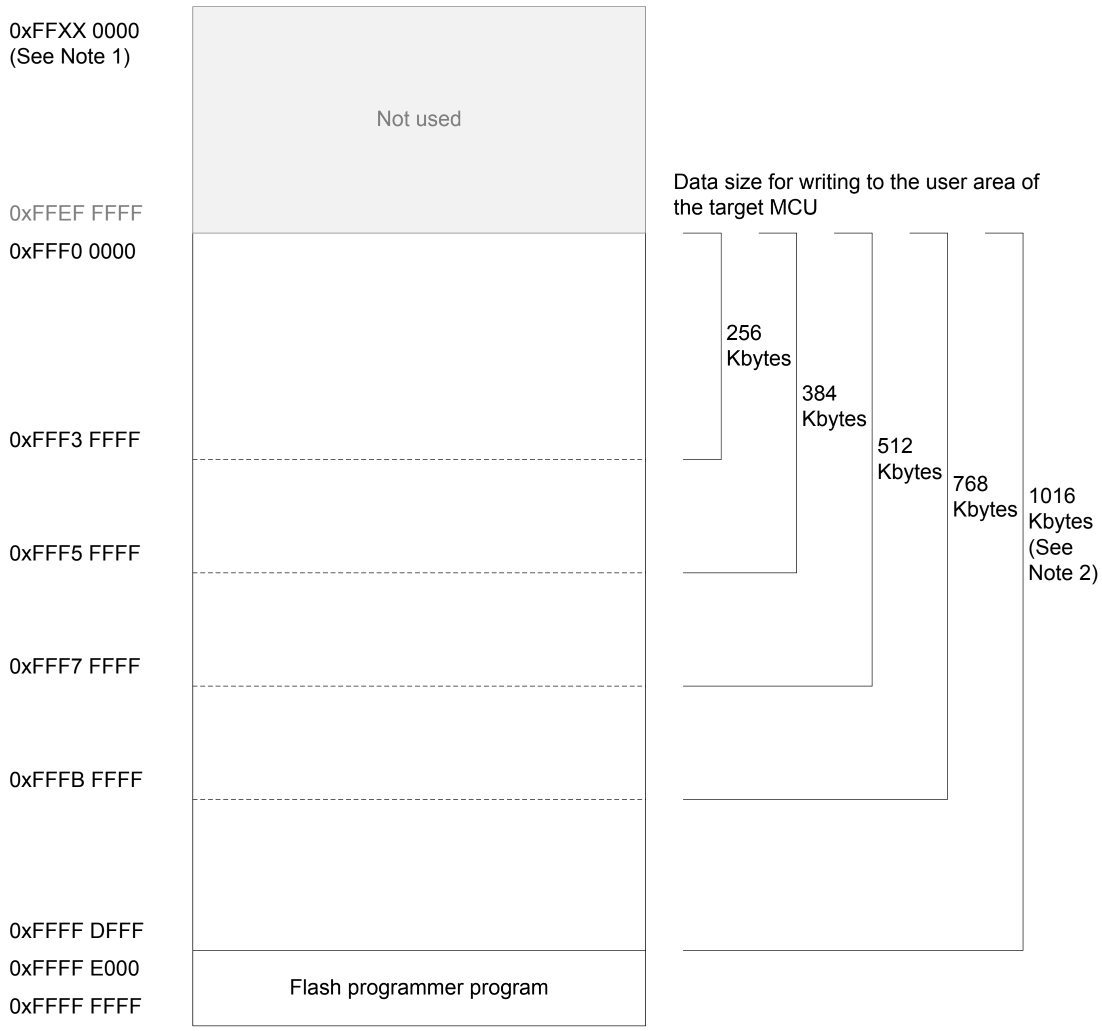


Figure 1.1 Flash Programmer Usage Example

1.1 RSK+RX63N User Area Memory Map

The program of the flash programmer and data to be written to the target MCU user area are stored in the RSK+RX63N User Area. Figure 1.2 shows the RSK+RX63N User Area Memory Map.

Refer to 5.1 Programming the RSK+RX63N for details on programming the RSK+RX63N user area.



Note 1. 0xFFE0 0000 for a 2-Mbyte ROM
0xFFE8 0000 for a 1.5-Mbyte ROM

Note 2. When the ROM size of the target MCU is 1 Mbyte, the maximum size for the user program written by the flash programmer is 1016 Kbytes (flash programmer program size is subtracted from 1 Mbyte).

Figure 1.2 RSK+RX63N User Area Memory Map

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F563NBDDFC (RX63N Group)
Operating frequencies	Main clock: 12 MHz PLL: 192 MHz (main clock divided by 1 and multiplied by 16) System clock (ICLK): 96 MHz (PLL divided by 2) Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Version 4.09.01
C compiler	Renesas Electronics C/C++ compiler package for RX Family V.1.02 Release 01
	Compile options -cpu=rx600 -output=obj="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -nologo (default settings of the integrated development environment are used)
iodefine.h version	Version 1.6A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit+ for RX63N (part number: R0K50563NC010BR)

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

— RX63N Group, RX631 Group Initial Setting Rev. 1.10 (R01AN1245EJ)
— RX63N Renesas Starter Kit Sample Code for Hi-performance Embedded Workshop Rev.1.00 (R01AN1395EG)

The initial setting functions and debug LCD output functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are current as of the publication of this application note. However, the latest version is always recommended. Visit the Renesas Electronics Corporation website to check for and download the latest version.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example.

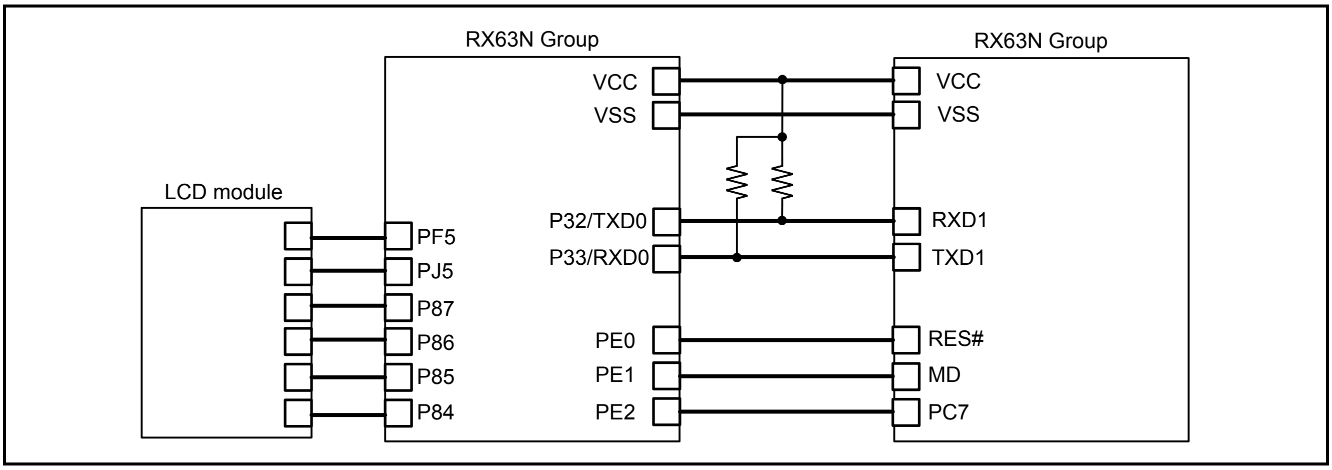


Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P87	Output	Debug LCD data 7 output
P86	Output	Debug LCD data 6/backlight output
P85	Output	Debug LCD data 5/Y drive output
P84	Output	Debug LCD data 4/X drive output
PF5	Output	Debug LCD enable output
PJ5	Output	Debug LCD register select output
P33/RXD0	Input	Input pin for SCI0 receive data
P32/TXD0	Output	Output pin for SCI0 transmit data
PE0	Output	RES# pin control
PE1	Output	MD pin control
PE2	Output	PC7 pin control

5. Software

5.1 Programming the RSK+RX63N

Data to be programmed in the RSK+RX63N user area is as follows:

- User program to be programmed in the target MCU user area
- Flash programmer program

This document describes an example of using the Renesas Flash Development Toolkit (hereinafter referred to as FDT).

Data to be programmed in the RSK+RX63N user area are merged using the editor function for S-Record files or hexadecimal files in the FDT. Also, the merged data is programmed in the RSK+RX63N user area using the FDT.

Refer to the User's Manual of the FDT (R20UT0508EJ) for details on using the FDT.

Figure 5.1 shows the Flow of Programming the RSK+RX63N.

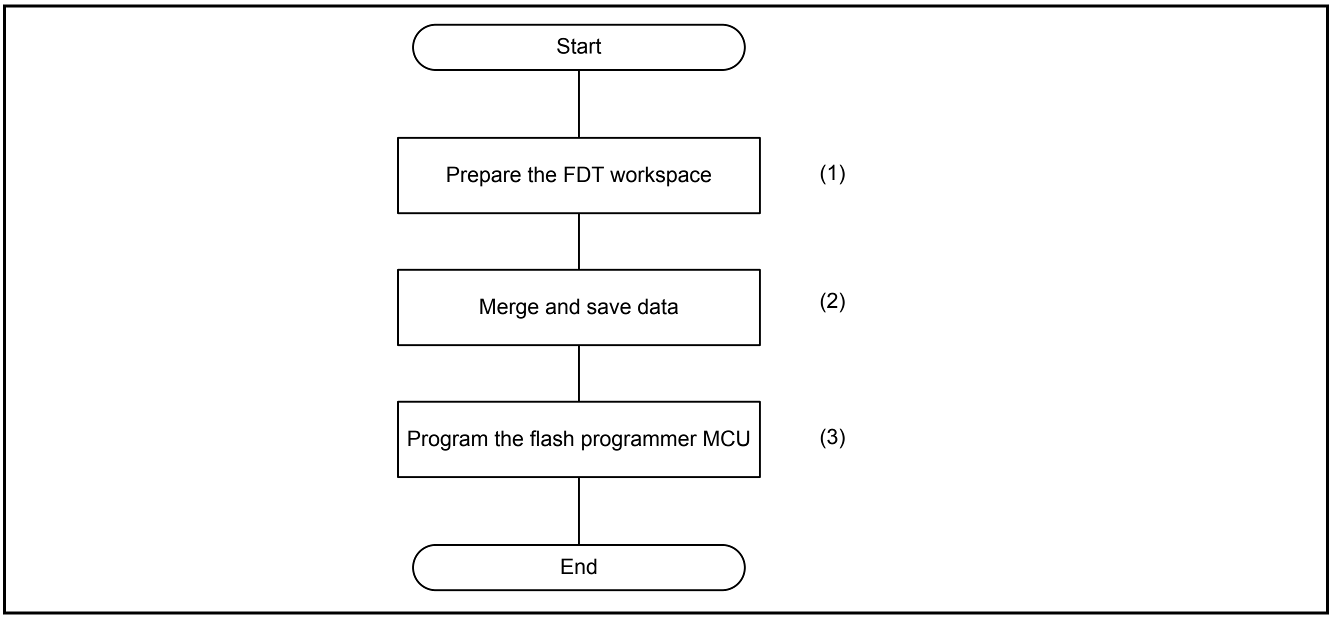


Figure 5.1 Flow of Programming the RSK+RX63N

(1) Refer to 5.1.1 Prepare the FDT Workspace for details.

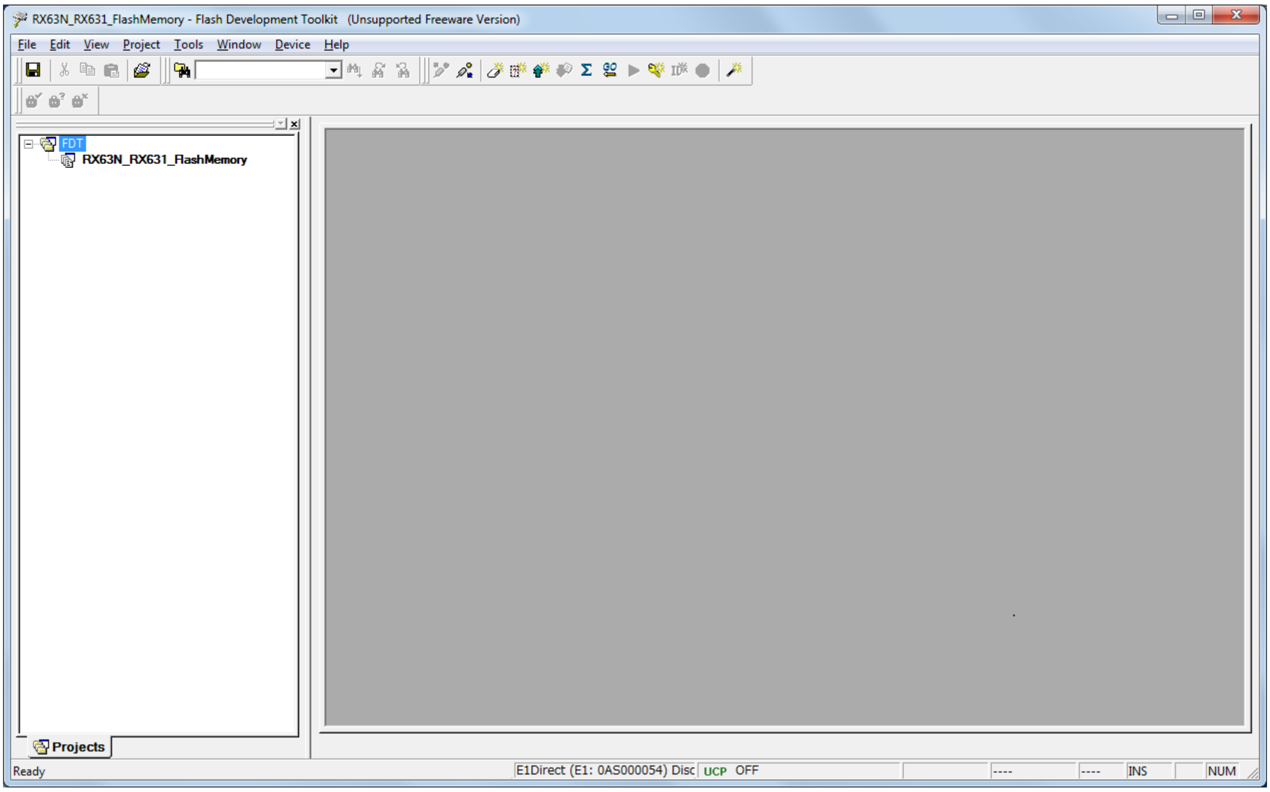
(2) Refer to 5.1.2 Merge and Save Data for details.

(3) Refer to 5.1.3 Program the RSK+RX63N User Area for details.

5.1.1 Prepare the FDT Workspace

Create a workspace and project to use the FDT. Set the MCU used for the flash programmer as the target device.

In the example, Workspace Name is FDT, and Project Name is RX63N_RX631_FlashMemory.



5.1.2 Merge and Save Data

Perform steps (1) to (7) to merge and save data.

(1) Add data files to be merged to the project

In the example, folders FlashMemoryPrograma and UserProgram are added to the RX63N_RX631_FlashMemory project.

The main.mot file of the flash programmer's program is added to the FlashMemoryPrograma folder.

The following data files are added to the UserProgram folder for each size of the user program to be programmed in the target MCU user area:

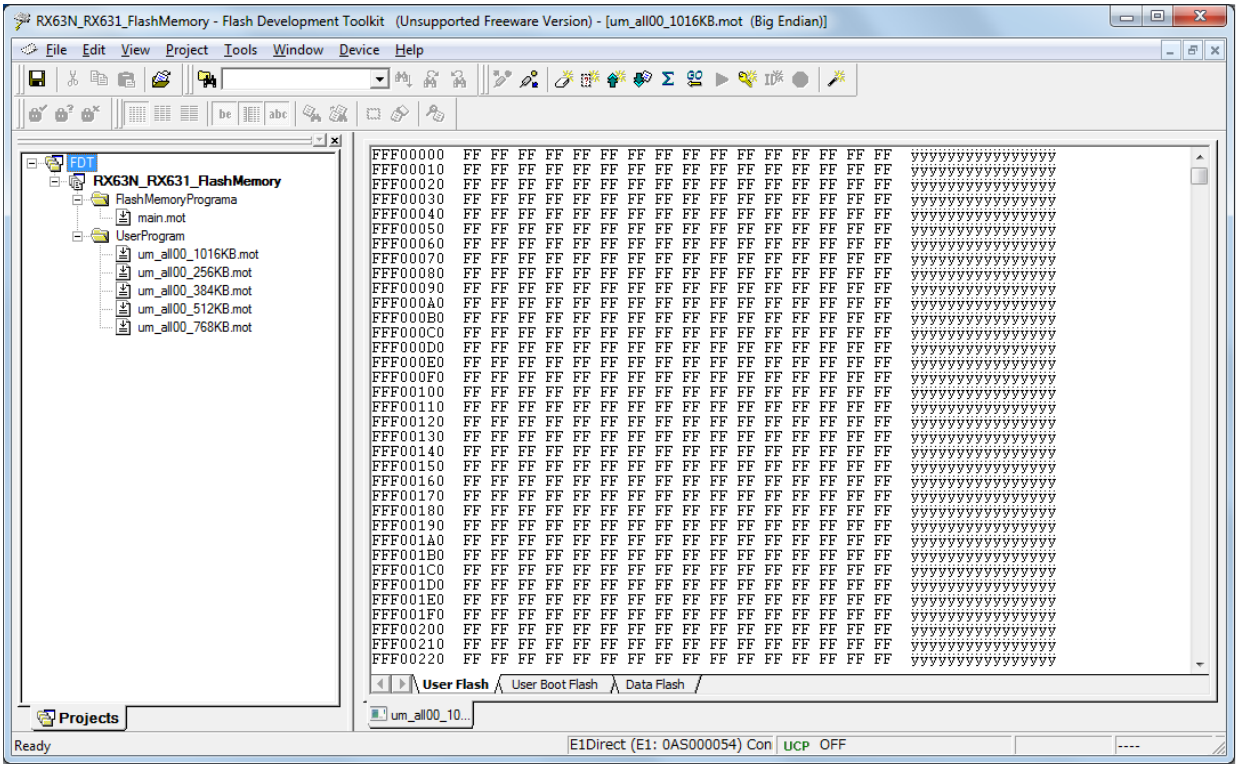
um_all00_256KB.mot file when the user program size is 256 Kbytes

um_all00_384KB.mot file when the user program size is 384 Kbytes

um_all00_512KB.mot file when the user program size is 512 Kbytes

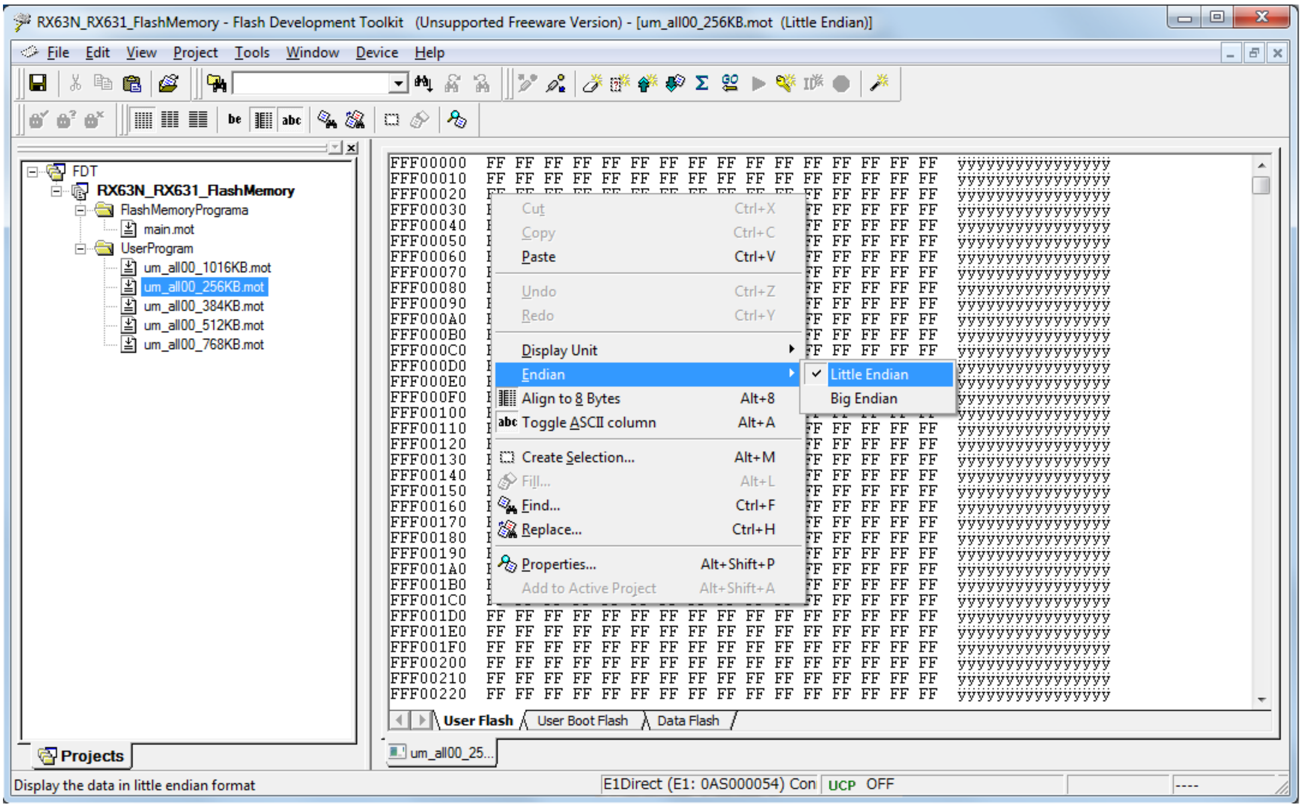
um_all00_768KB.mot file when the user program size is 768 Kbytes

um_all00_1016KB.mot file when the user program size is 1016 Kbytes



(2) Open data files to be merged on the hex editor window and set the endian

In the example, files “main.mot” and “um_all00_256KB.mot” are opened in the hex editor window. Little endian is selected for both files.



(3) Select user program data to be programmed in the target MCU user area to merge

Select the range as follows:

Addresses 0xFFFC 0000 to 0xFFFF FFFF when the user program size is 256 Kbytes

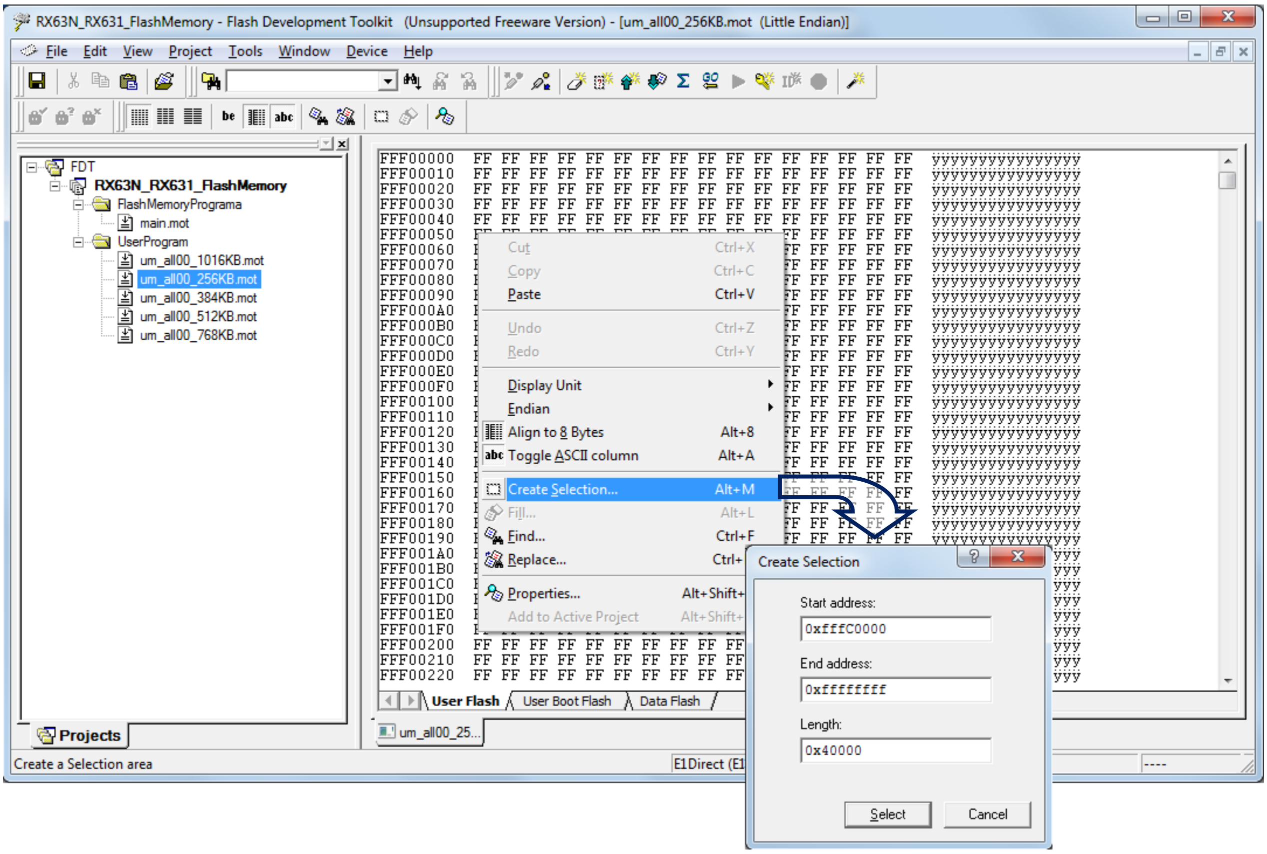
Addresses 0xFFFA 0000 to 0xFFFF FFFF when the user program size is 384 Kbytes

Addresses 0xFFF8 0000 to 0xFFFF FFFF when the user program size is 512 Kbytes

Addresses 0xFFF4 0000 to 0xFFFF FFFF when the user program size is 768 Kbytes

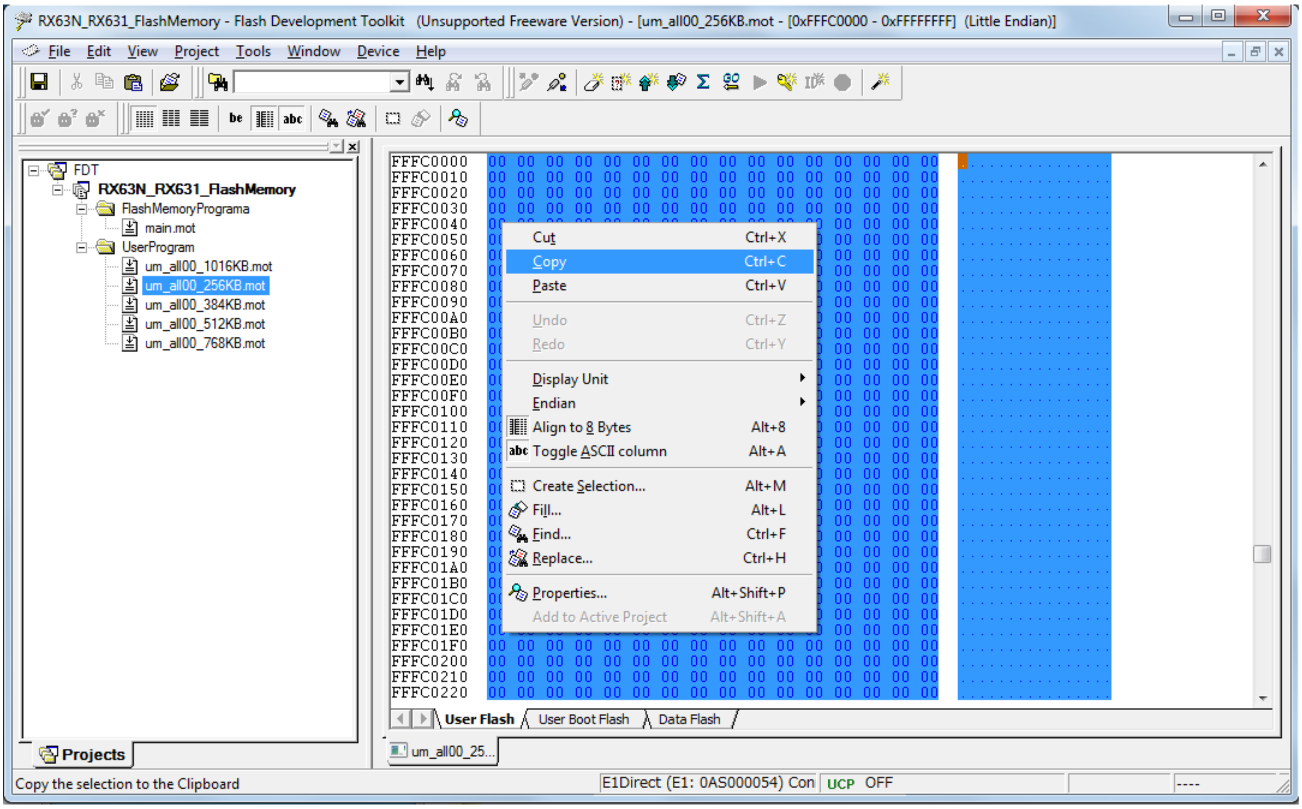
Addresses 0xFFF0 0000 to 0xFFFF FFFF when the user program size is 1016 Kbytes

In the example, addresses 0xFFFC 0000 to 0xFFFF FFFF of the um_all00_256KB.mot file are selected.



(4) Copy the highlighted user program data to the Windows clipboard

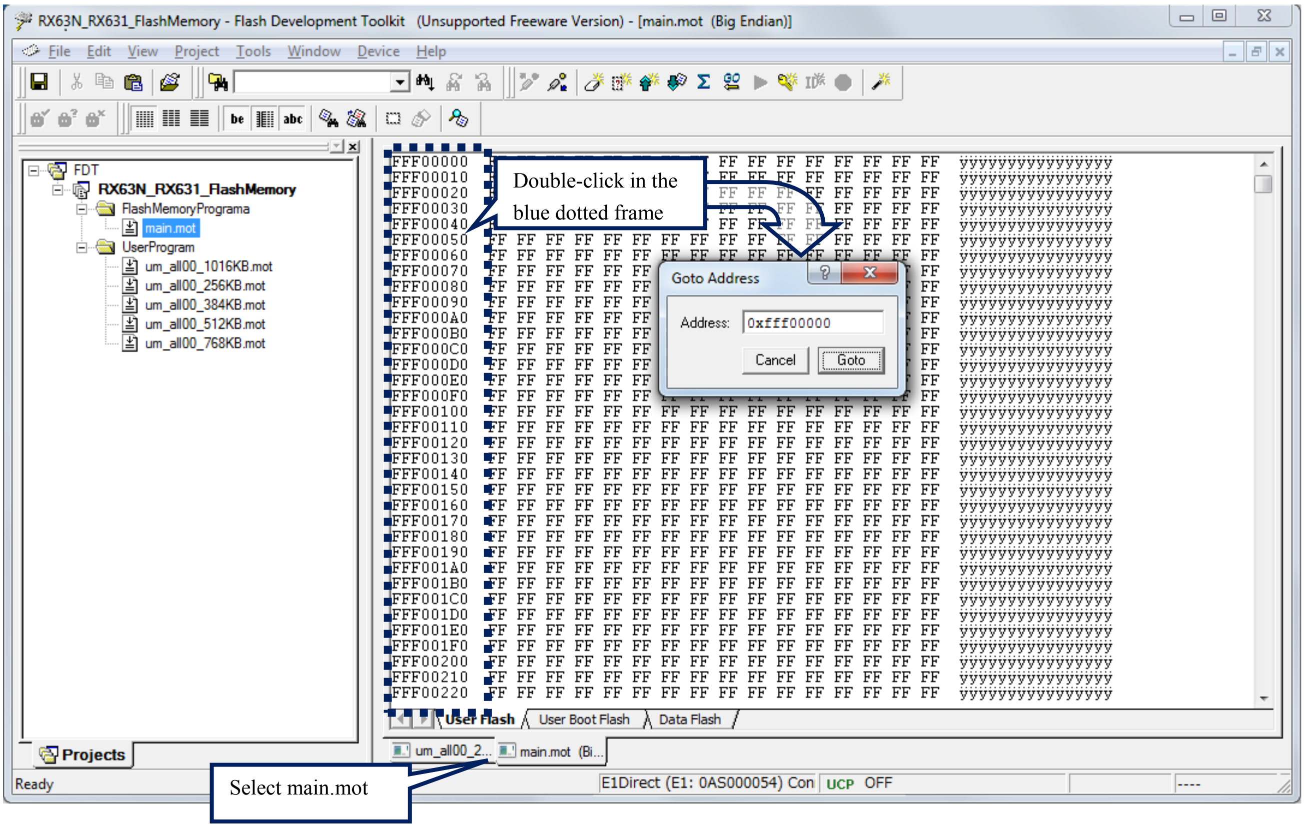
In the example, addresses 0xFFFC 0000 to 0xFFFF FFFF of the um_all00_256KB.mot file are copied to the Windows clipboard.



(5) Merge and create data to be programmed to the RSK+RX63N user area

Select the main.mot file in the hex editor window, and paste the data that was copied to the Windows clipboard in step (4) into addresses 0xFFF0 0000 and higher.

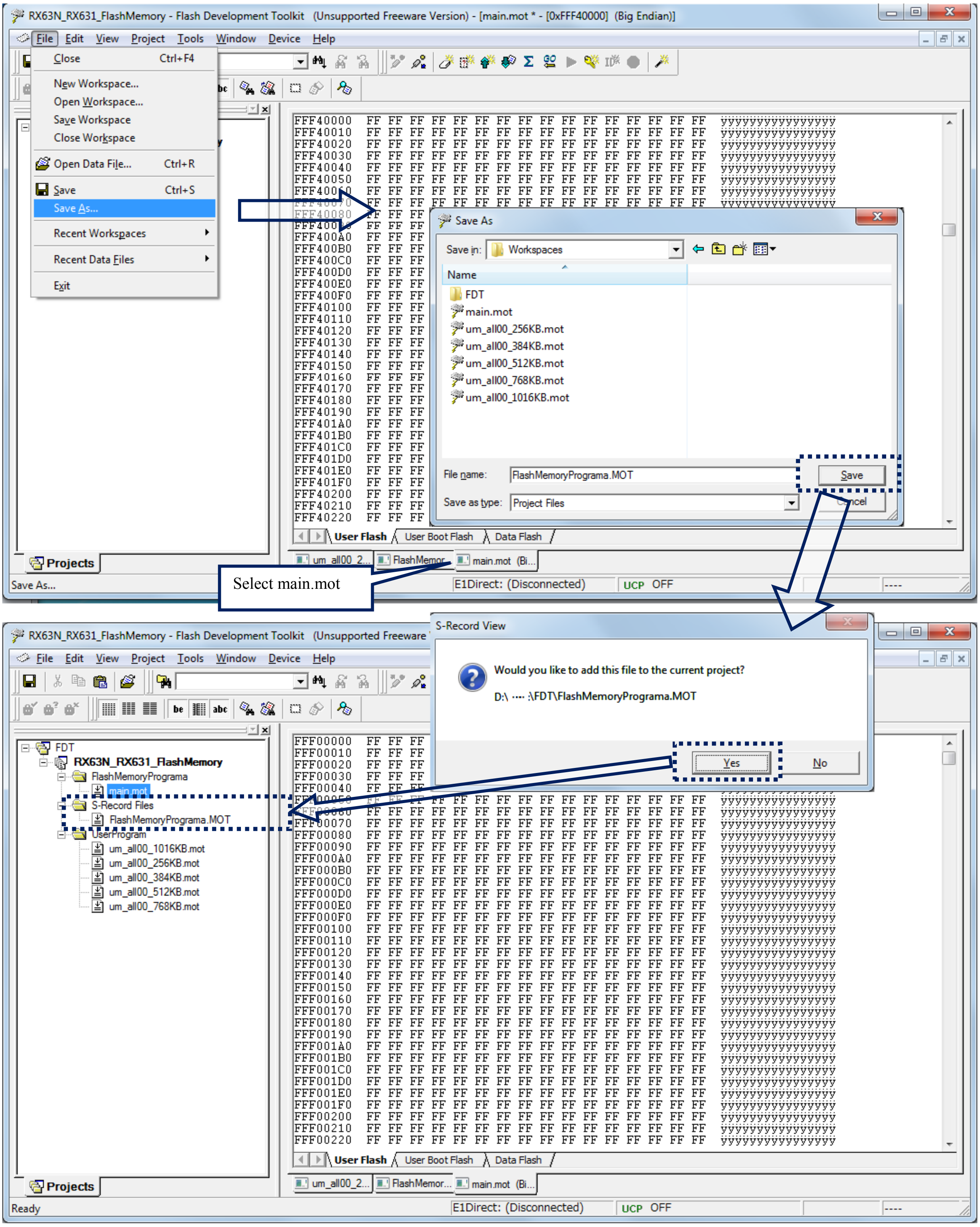
In the example, the start address of paste destination in the main.mot file is set to 0xFFF0 0000. After setting the start address, paste the data from the clipboard.



(6) Save data to be programmed to the RSK+RX63N user area

Select the main.mot file in the hex editor window, name the file to save the data that was created in step (5), and add the file to the project.

In the example, the FlashMemoryPrograma.MOT file is saved in the S-Record Files folder.



(7) Confirm data to be programmed to the RSK+RX63N user area

Confirm the allocation of the merged data in the data file that was created in step (6). Select the data file to be programmed to the RSK+RX63N user area in the workspace window, and confirm the address range of the block used.

Confirm the address range as follows:

Addresses 0xFFF0 0000 to 0xFFF3 FFFF when the user program size is 256 Kbytes

Addresses 0xFFF0 0000 to 0xFFF5 FFFF when the user program size is 384 Kbytes

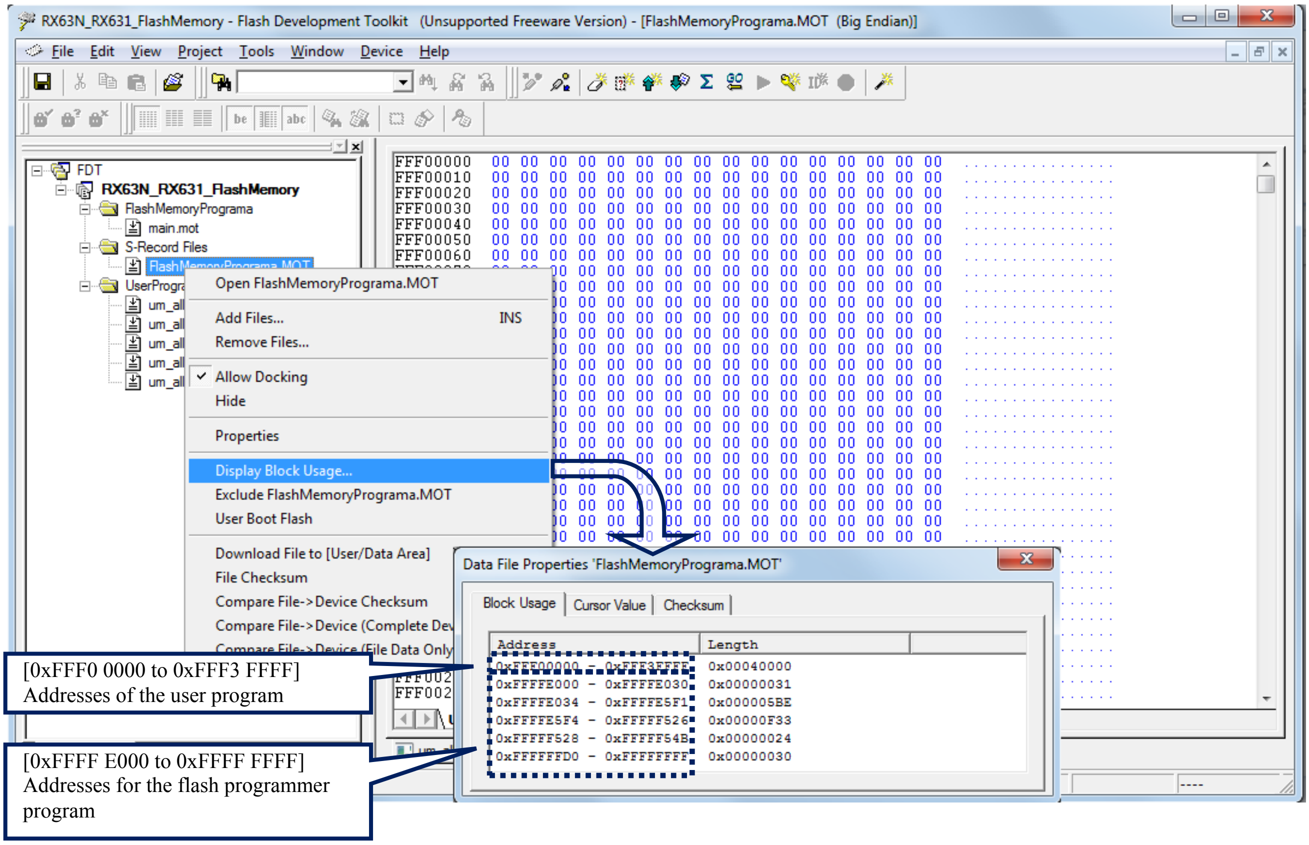
Addresses 0xFFF0 0000 to 0xFFF7 FFFF when the user program size is 512 Kbytes

Addresses 0xFFF0 0000 to 0xFFFB FFFF when the user program size is 768 Kbytes

Addresses 0xFFF0 0000 to 0xFFFF DFFF when the user program size is 1016 Kbytes

Addresses 0xFFFF E000 to 0xFFFF FFFF for the program of the flash programmer

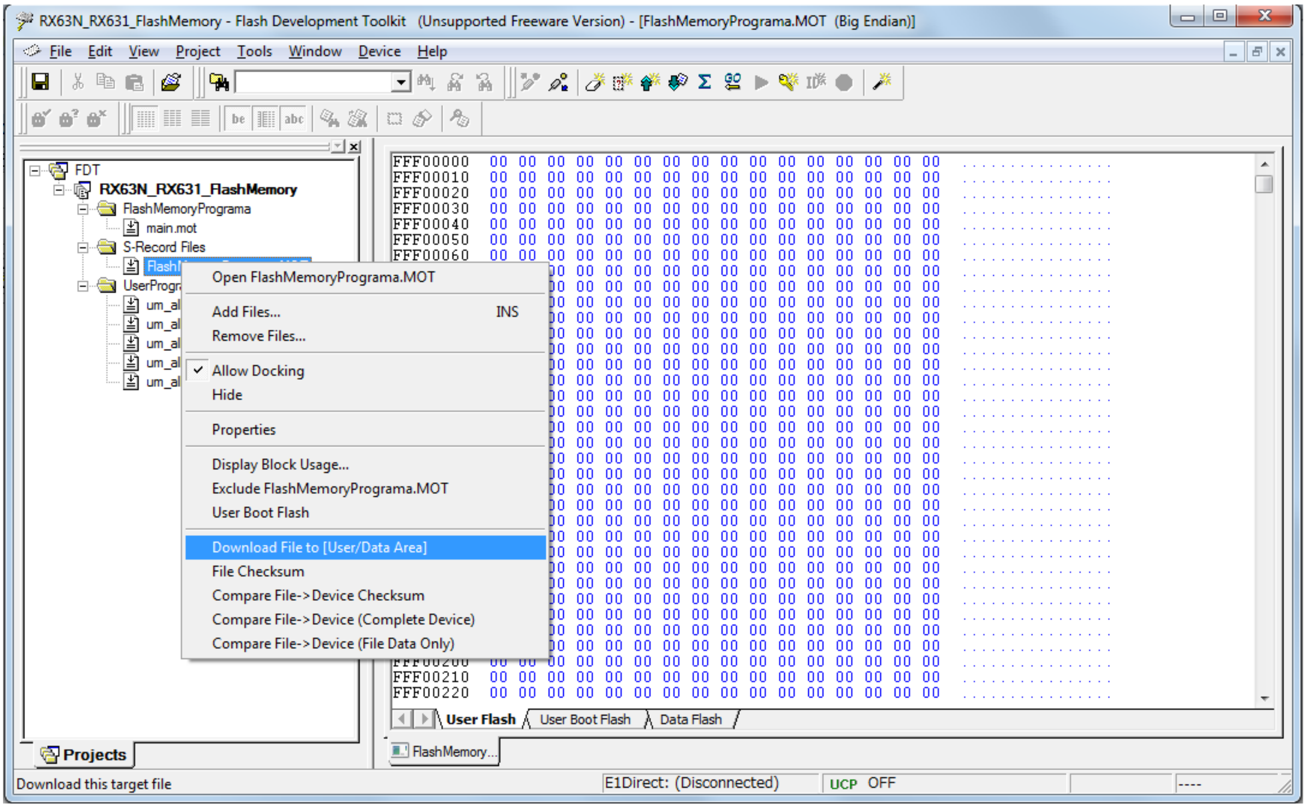
In the example, the address range of the block used is confirmed when the user program size is 256 Kbytes.



5.1.3 Program the RSK+RX63N User Area

Select and download the data file to be programmed to the RSK+RX63N user area.

In the example, the FlashMemoryPrograma.MOT file in the S-Record Files folder is downloaded.



5.2 Operation Overview

The target MCU is started in boot mode and the bit rate is automatically adjusted to connect to the MCU at 19,200 bps.

After connecting, the supported device inquiry command is sent to obtain information of the target MCU, and then the device select command and block information inquiry command are sent. Also, the operating frequency select command is sent to change the bit rate to 1.5 Mbps.

The programming/erasure state transition command is sent to check the ID code protection of the target MCU and the processing for the boot mode ID code protection is performed.

The user program is written in the target MCU according to the information obtained from the target MCU. After programming is completed, the programmed area in the target MCU is read to verify the read data with the programmed data.

Figure 5.2 shows the Flash Programmer State Transition.

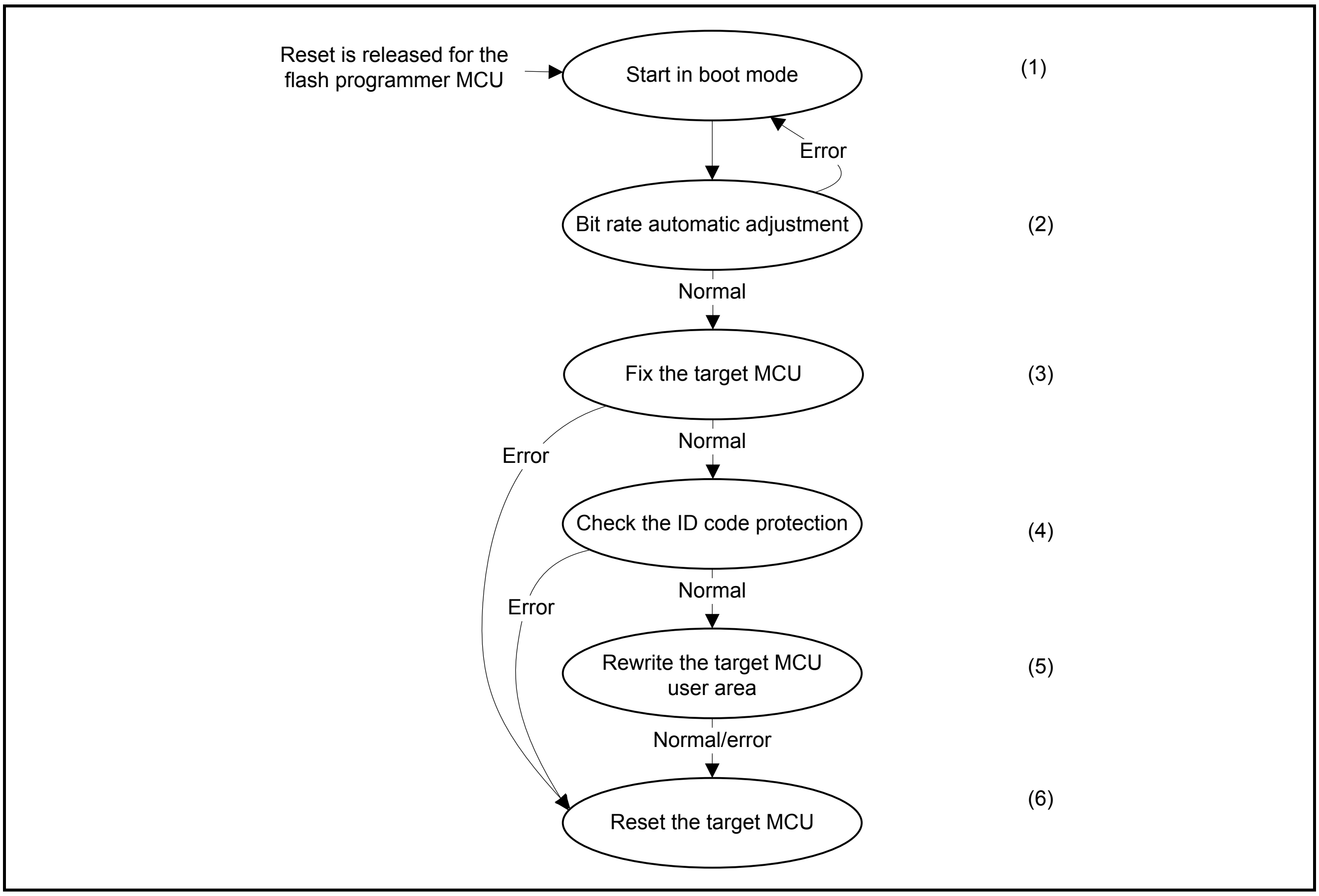


Figure 5.2 Flash Programmer State Transition

(1) Refer to 5.2.1 Start the MCU in Boot Mode for details

(2) Refer to 5.2.2 Bit Rate Automatic Adjustment for details

(3) Refer to 5.2.3 Fix the Target MCU for details

(4) Refer to 5.2.4 Check ID Code Protection for details

(5) Refer to 5.2.5 Rewrite the Target MCU User Area for details

(6) Refer to 5.2.6 Reset the Target MCU for details

5.2.1 Start the MCU in Boot Mode

(1) The flash programmer sets the RES# pin of the target MCU to low.

(2) The flash programmer sets the MD pin of the target MCU to low.

(3) The flash programmer sets the PC7 pin of the target MCU to low.

(4) After waiting 2 ms, the flash programmer sets the RES# pin of the target MCU to high.

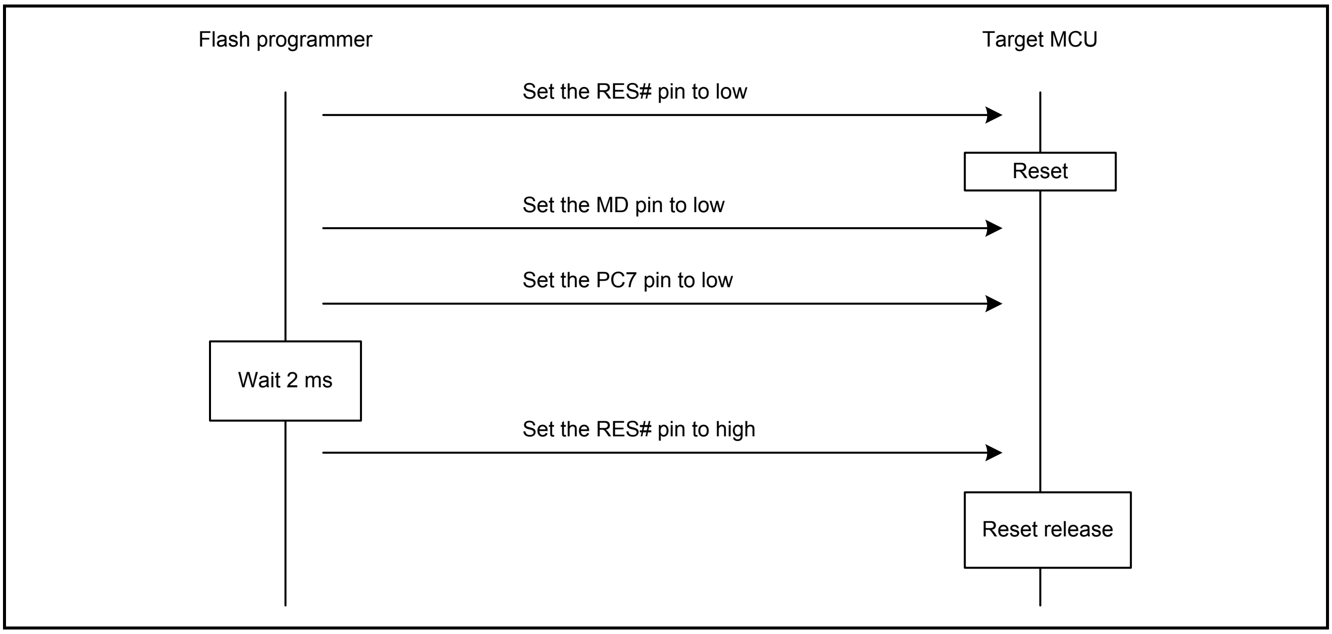


Figure 5.3 Start Procedure in Boot Mode

5.2.2 Bit Rate Automatic Adjustment

The flash programmer starts the target MCU in boot mode, waits 400 ms, and then sends “00h” 30 times to adjust the bit rate to 19,200 bps.

When the flash programmer receives 00h, send 55h to the target MCU. When 00h cannot be received, the flash programmer restarts the target MCU in boot mode and performs bit rate automatic adjustment again.

After sending 55h, the flash programmer completes the bit rate automatic adjustment when it receives E6h. When the flash programmer sends 55h and then receives FFh, it restarts the target MCU in boot mode and performs bit rate automatic adjustment again.

Figure 5.4 shows the Bit Rate Automatic Adjustment Procedure.

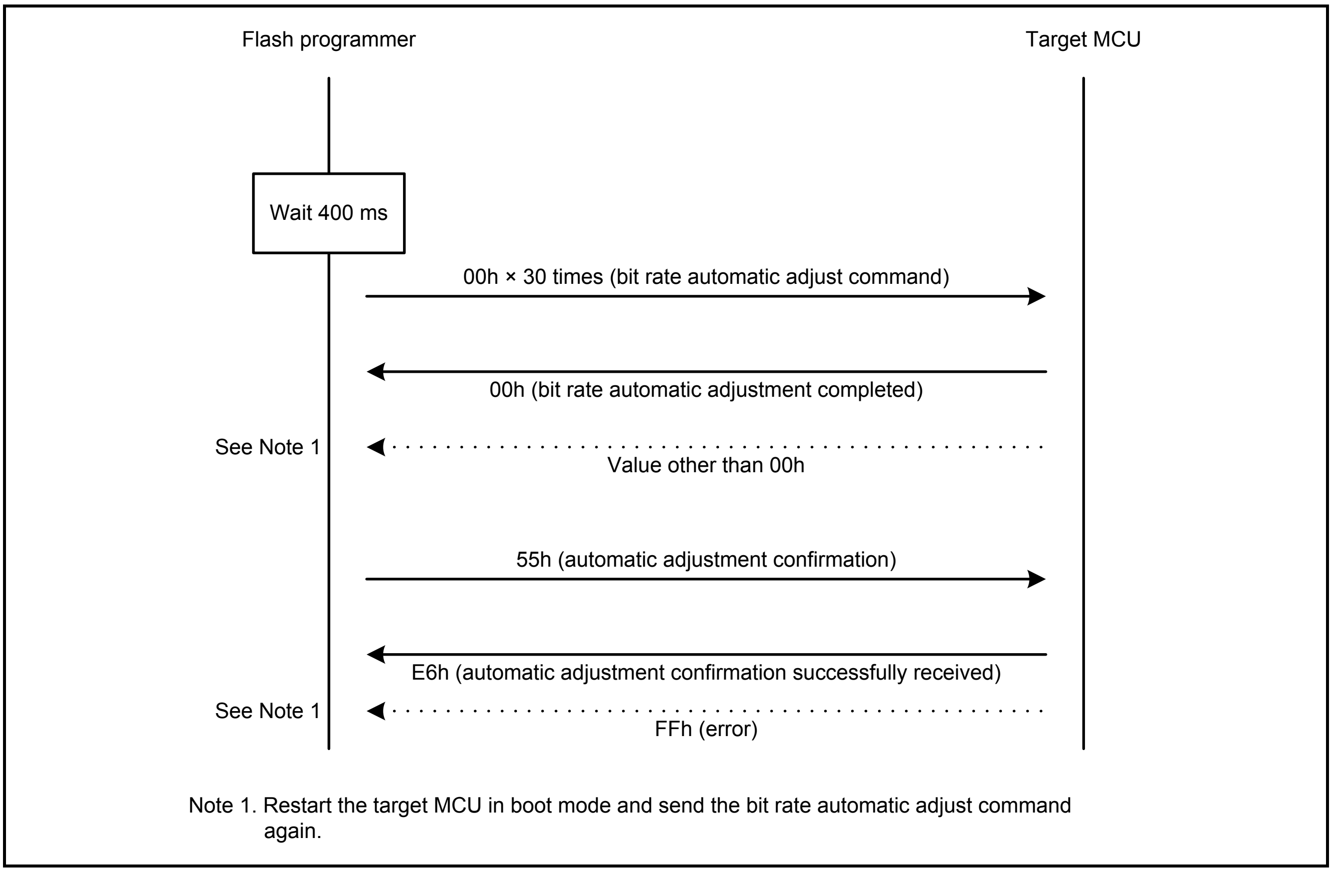


Figure 5.4 Bit Rate Automatic Adjustment Procedure

5.2.3 Fix the Target MCU

To fix the target MCU, the flash programmer performs steps (1) to (4) below.

(1) The flash programmer sends the supported device inquiry command and stores device codes for selecting the endian of data to be programmed in the user area.

When the flash programmer receives a response (data starting with 30h) to the supported device inquiry command, it stores device codes for selecting the endian of data to be programmed in the user area. When the flash programmer receives data other than the response (data starting with 30h), it resets the target MCU to abort.

Figure 5.5 shows the Procedure to Store Device Codes.

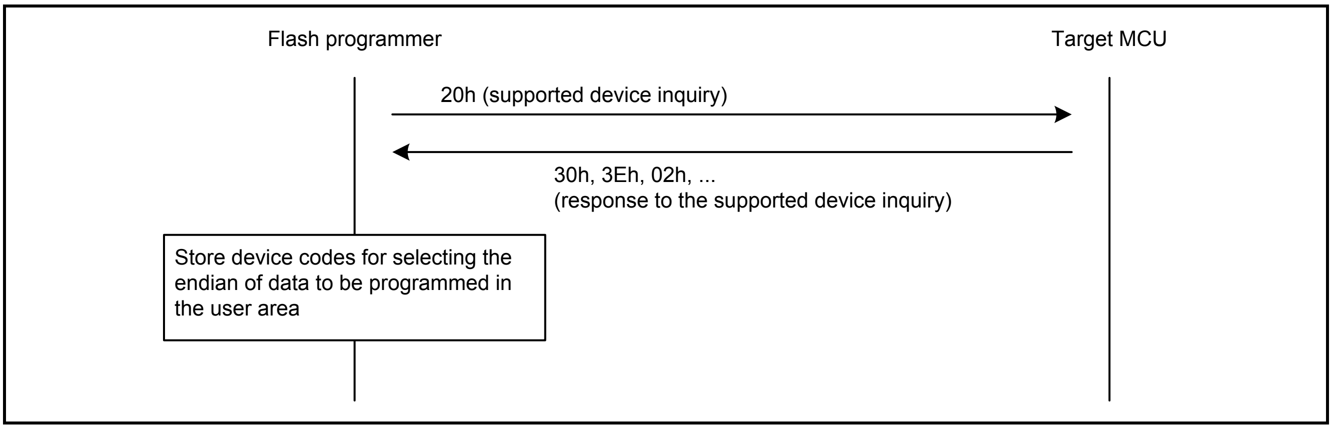


Figure 5.5 Procedure to Store Device Codes

(2) The flash programmer sends the device select command to select the endian of data to be programmed in the user area.

The flash programmer sends the device select command (10h) to select the endian of data to be programmed in the user area. The flash programmer uses the device code corresponding to the endian of the flash programmer in the device codes that were stored by the support device inquiry command.

When the flash programmer receives a response (06h) after sending the device select command, it completes the endian selection. When the flash programmer receives data other than the response (06h) after sending the device select command, it resets the target MCU to abort.

Figure 5.6 shows the Procedure to Select the Endian.

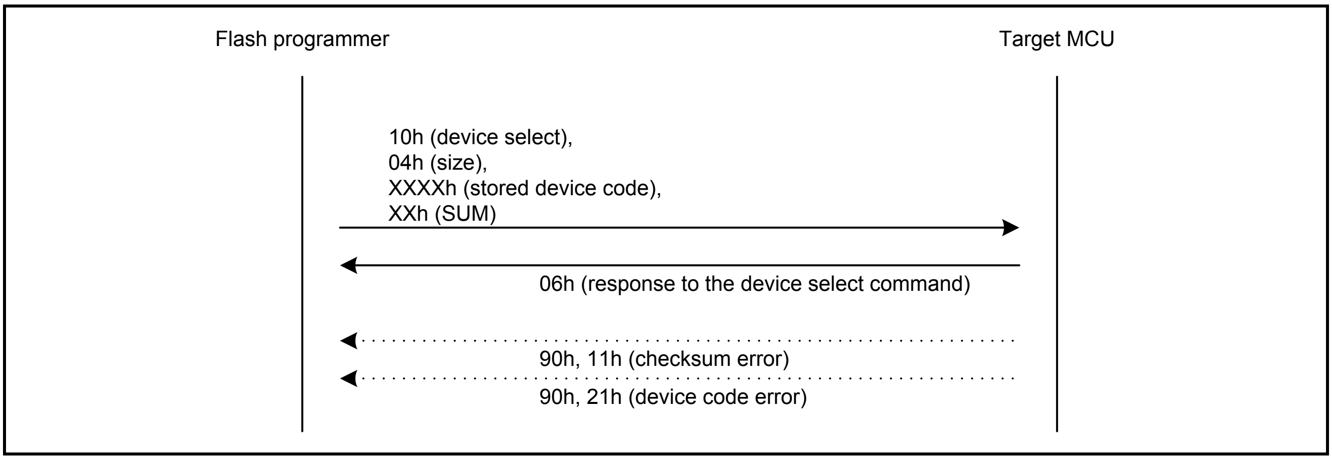


Figure 5.6 Procedure to Select the Endian

(3) The flash programmer sends the clock mode selection command to select the clock mode of the target MCU.

The flash programmer sends the clock mode selection command (11h) to specify the clock mode. When the flash programmer receives data other than 06h (response to the clock mode selection command), it resets the target MCU to abort.

Figure 5.7 shows the Procedure to Select Clock Mode.

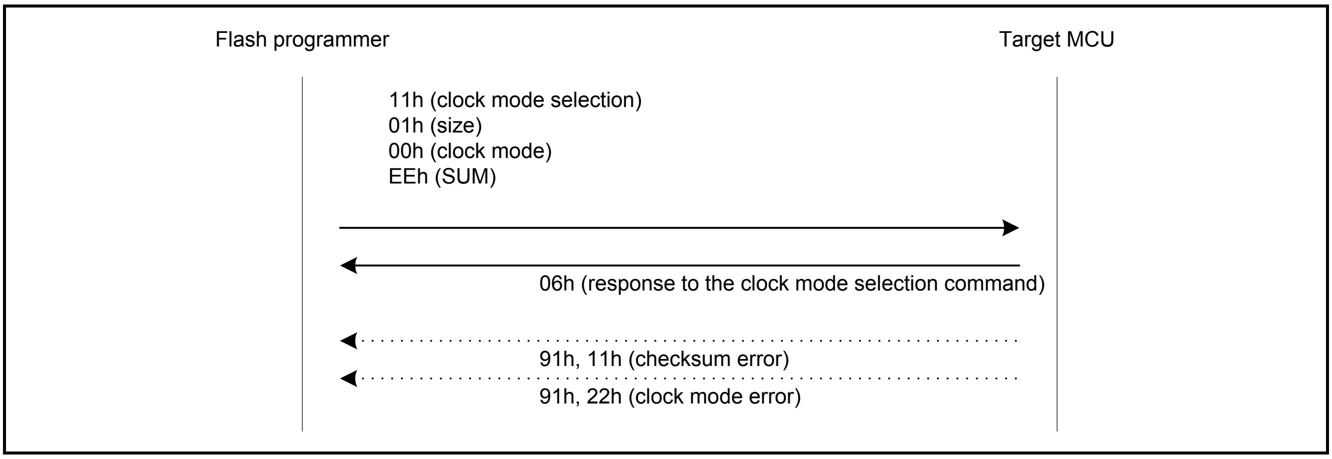


Figure 5.7 Procedure to Select Clock Mode

(4) The flash programmer sends the new bit rate selection command to change the bit rate for communication with the target MCU to 1.5 Mbps.

The flash programmer sends the new bit rate selection command (3Fh) to change the bit rate to 1.5 Mbps. With this command, the input frequency of the target MCU is sent. When the flash programmer receives ACK (06h) after sending the new bit rate selection command, it waits 25 ms at the bit rate for sending the new bit rate selection command and changes the bit rate to 1.5 Mbps. After that, when the flash programmer sends confirmation data (06h) at the changed bit rate and receives a response (06h), it completes changing the bit rate.

When the flash programmer receives data other than 06h (response) after sending the new bit rate selection command or confirmation data (06h), it resets the target MCU to abort.

Figure 5.8 shows the Procedure to Change the Bit Rate.

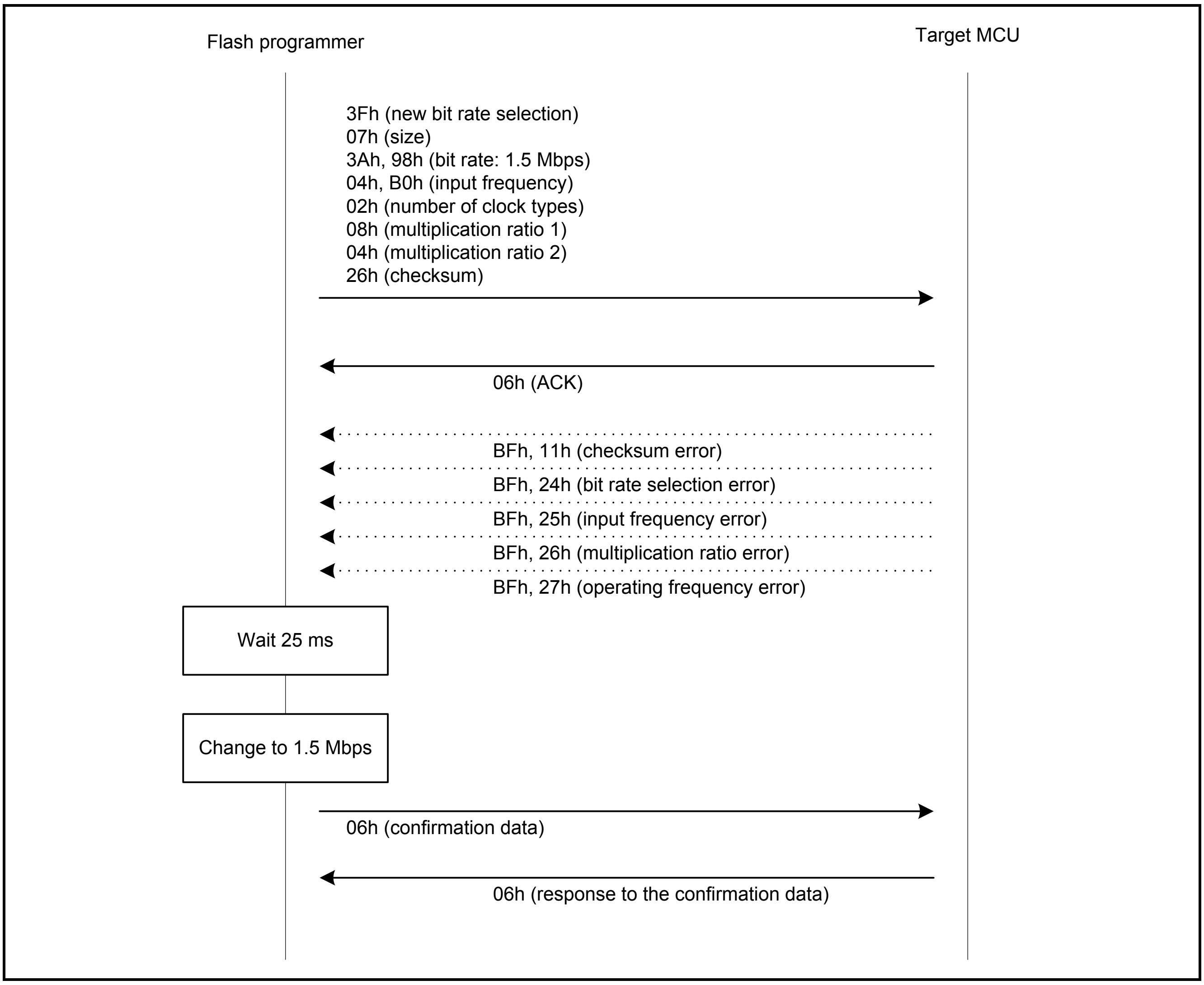


Figure 5.8 Procedure to Change the Bit Rate

5.2.4 Check ID Code Protection

The flash programmer performs steps (1) to (3) below to check the ID code protection.

(1) The flash programmer sends the programming/erasure state transition command to check and store the status of the ID code protection for the target MCU.

The flash programmer sends the programming/erasure state transition command (40h) to check the ID code protection for the target MCU.

After the flash programmer sends the programming/erasure state transition command, it determines the status according to the received response and store the status in the ID code protection status buffer.

Table 5.1 lists the Responses and Values Stored in the ID Code Protection Status Buffer.

Table 5.1 Responses and Values Stored in the ID Code Protection Status Buffer

Response	Values Stored in the ID Code Protection Status Buffer
26h	00h
16h	01h

When the flash programmer receives data other than values listed in Table 5.1 after sending the programming/erasure state transition command, it resets the target MCU to abort.

Figure 5.9 shows the Procedure to Check ID Code Protection by the Programming/Erasure State Transition Command.

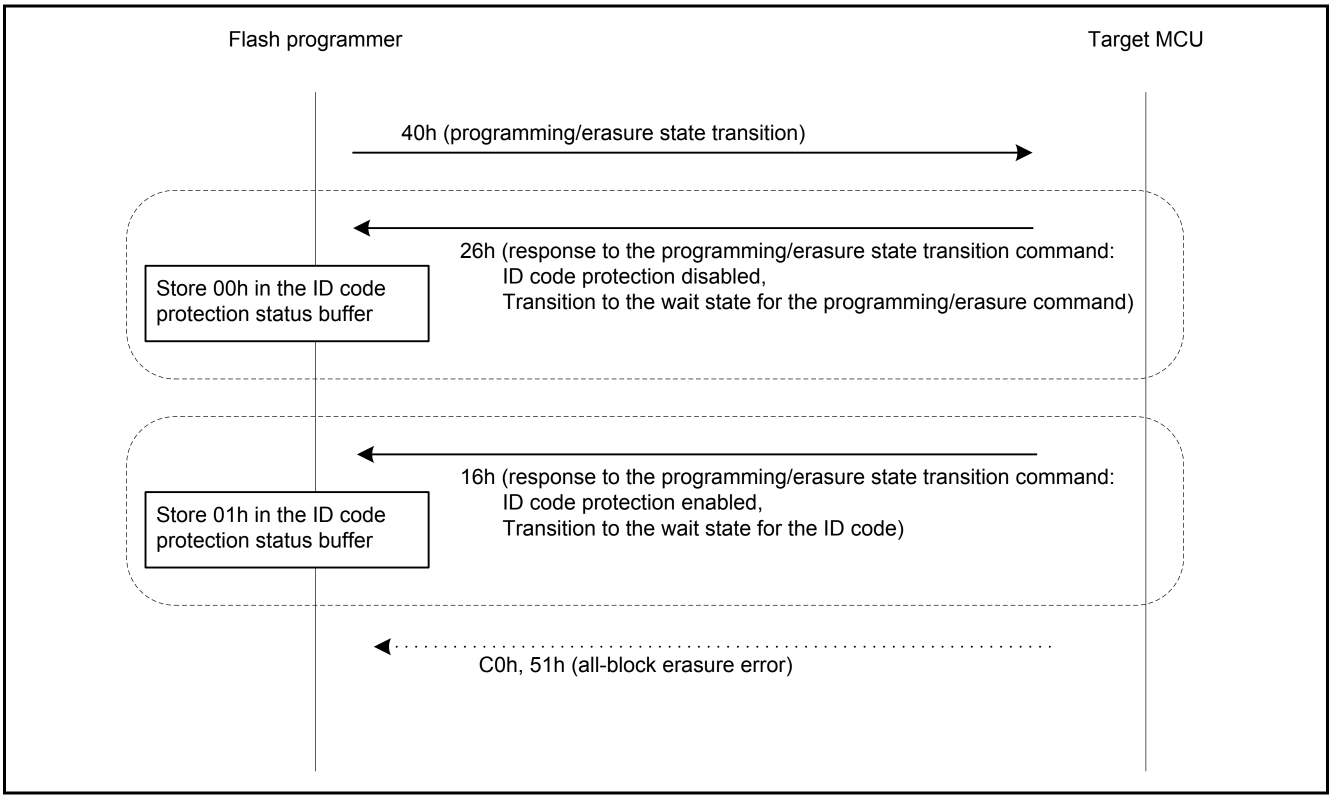


Figure 5.9 Procedure to Check ID Code Protection by the Programming/Erasure State Transition Command

(2) The flash programmer sends the ID code check command to check and store the status of the ID code protection for the target MCU.

The flash programmer performs this step when the value stored in the ID code protection status buffer is 01h.

The flash programmer sends the ID code check command (60h) to determine the state of ID code protection for the target MCU. The control code, and ID code 1 to ID code 15 are set by reading and using data to be programmed in the target MCU user area.

When the flash programmer receives data other than 26h after sending the ID code check command, it resets the target MCU to abort.

Figure 5.10 shows the Procedure to Check ID Code Protection by the ID Code Check Command.

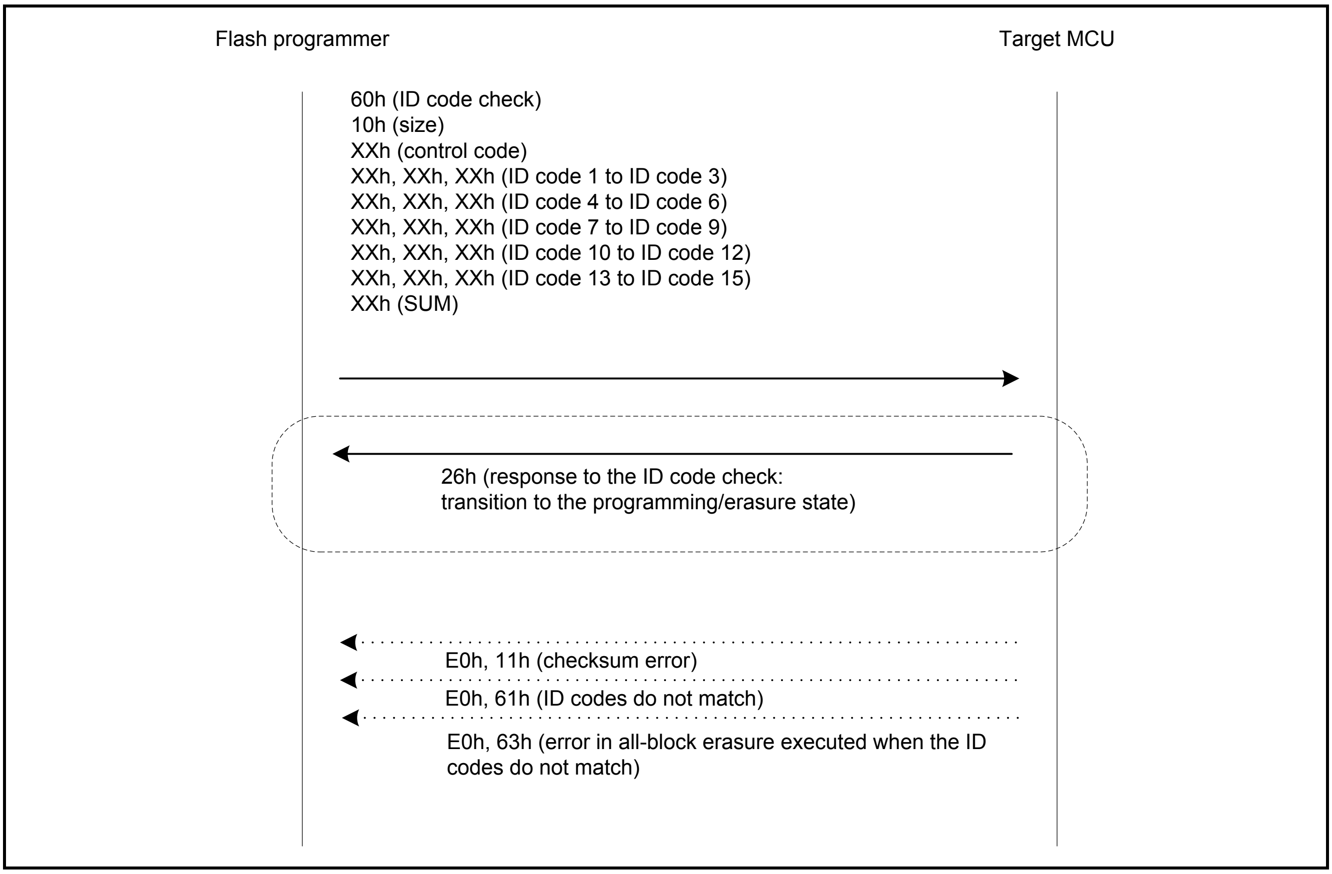


Figure 5.10 Procedure to Check ID Code Protection by the ID Code Check Command

(3) The flash programmer erases the flash memory of the target MCU for writing the user program in the target MCU.

The flash programmer performs this step when the value stored in the ID code protection status buffer is 01h.

The flash programmer sends the erasure selection command (48h). When the flash programmer receives 06h (response to the erasure selection command) after sending the erasure selection command, it completes erasure selection. When the flash programmer receives data other than 06h (response to the erasure selection command) after sending the erasure selection command, it resets the target MCU to abort.

The flash programmer sends a block erase command (58h) as many times as the number of blocks in the target MCU, and then it sends a block erase command (58h 04h FFh A5h) to end erasing blocks. When receiving 06h (response to the block erase command) after sending a block erase command, it completes a block erase operation. When the flash programmer receives data other than 06h (response to the block erase command) after sending the block erase command, it resets the MCU to abort.

Figure 5.11 shows the Procedure to Erase the Flash Memory of the Target MCU.

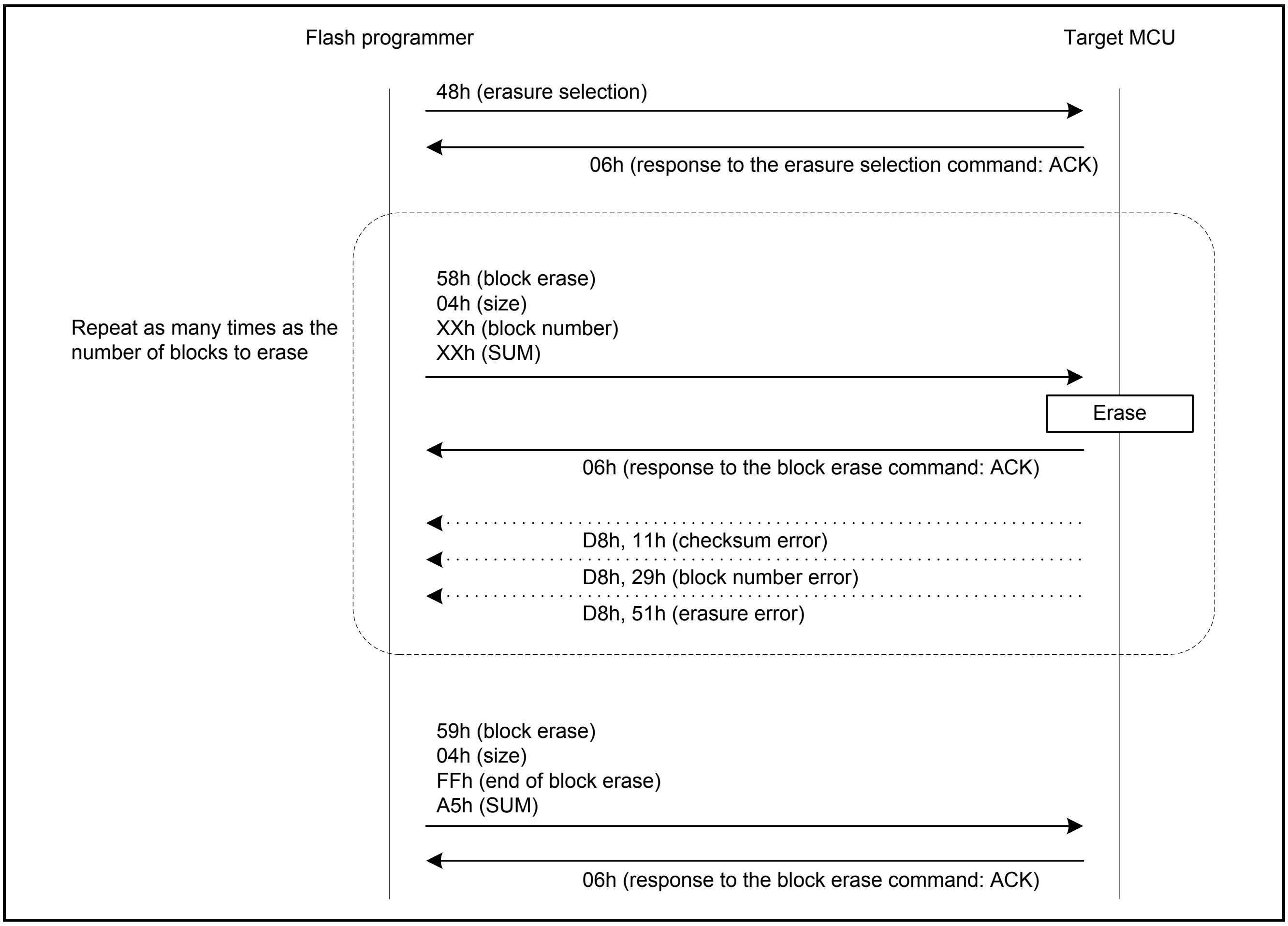


Figure 5.11 Procedure to Erase the Flash Memory of the Target MCU

5.2.5 Rewrite the Target MCU User Area

To rewrite the target MCU user area, the flash programmer performs steps (1) to (2) below.

(1) The flash programmer writes the user program to the target MCU user area.

The flash programmer sends 43h (user/data area programming selection command). After that, when the flash programmer receives 06h (response to the user/data area programming selection command), it completes user/data area programming selection. When it receives data other than 06h, it resets the target MCU to abort.

After completion of selection, the flash programmer sends a program command (50h) for the size of the user program to be programmed in the target MCU setting the 256-byte aligned addresses for program addresses and setting program data in 256 bytes.

The range of program addresses (destination of the target MCU) is as follows:

Addresses from 0xFFFC 0000 to 0xFFFF FFFF when the user program size is 256 Kbytes

Addresses from 0xFFFA 0000 to 0xFFFF FFFF when the user program size is 384 Kbytes

Addresses from 0xFFF8 0000 to 0xFFFF FFFF when the user program size is 512 Kbytes

Addresses from 0xFFF4 0000 to 0xFFFF FFFF when the user program size is 768 Kbytes

Addresses from 0xFFF0 2000 to 0xFFFF FFFF when the user program size is 1016 Kbytes

The range of program data (source data on the RSK+RX63N) is as follows:

Addresses from 0xFFF0 0000 to 0xFFF3 FFFF when the user program size is 256 Kbytes

Addresses from 0xFFF0 0000 to 0xFFF5 FFFF when the user program size is 384 Kbytes

Addresses from 0xFFF0 0000 to 0xFFF7 FFFF when the user program size is 512 Kbytes

Addresses from 0xFFF0 0000 to 0xFFFB FFFF when the user program size is 768 Kbytes

Addresses from 0xFFF0 0000 to 0xFFFF DFFF when the user program size is 1016 Kbytes

After sending the program command for the size of the user program to be programmed in the target MCU user area, the flash programmer sends 50h FFh FFh FFh FFh B4h (program command to end programming). When the flash programmer receives 06h (response to the program command), it completes a program operation. When the flash programmer receives data other than 06h, it resets the MCU to abort.

Figure 5.12 shows the Procedure to Program the User Area.

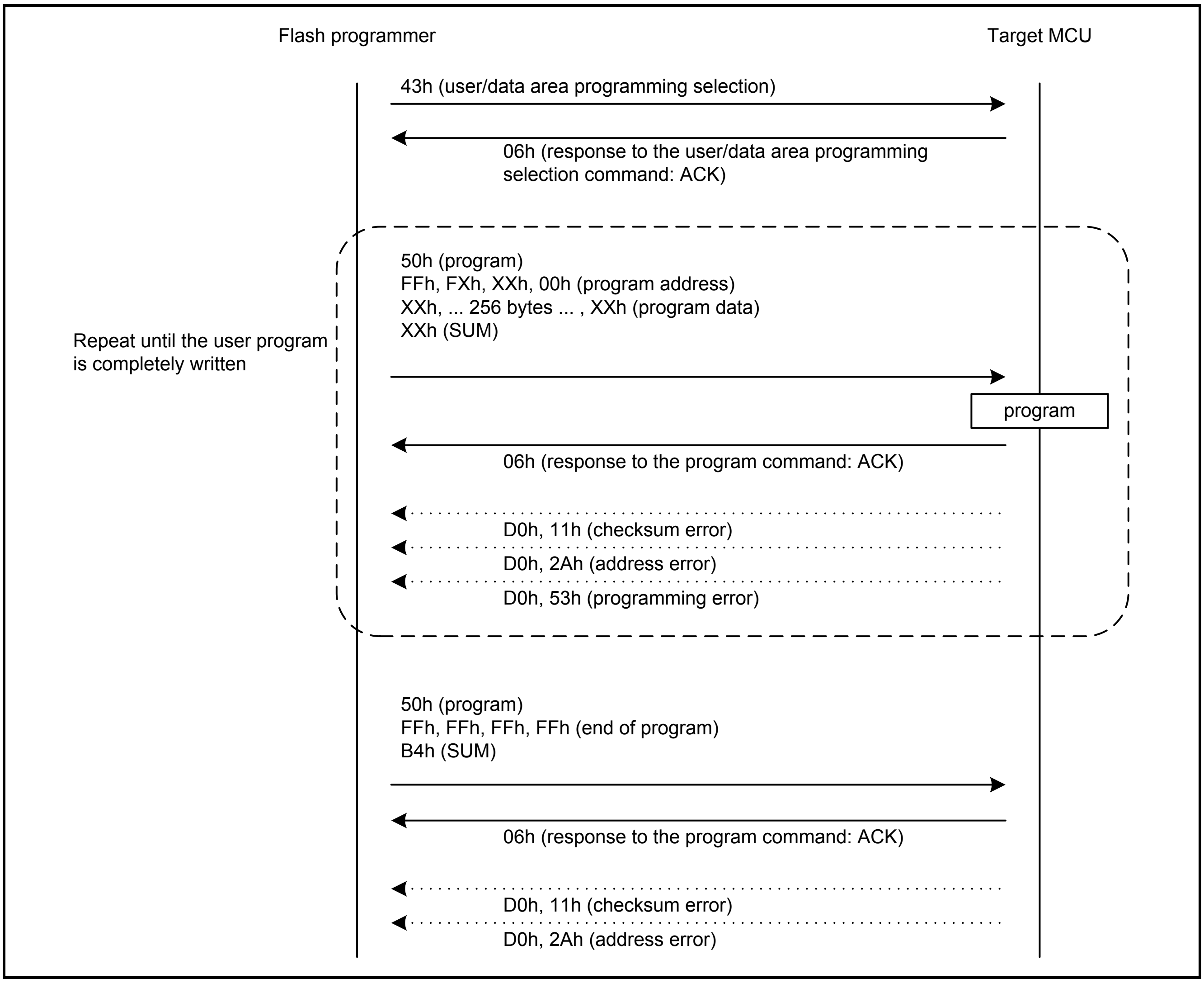


Figure 5.12 Procedure to Program the User Area

(2) The flash programmer confirms that the target MCU has been programmed correctly.

To confirm that the data has been programmed in the target MCU user area successfully, the flash programmer reads the data in the target MCU user area and compares the read data with the written data.

The flash programmer sends 52h (memory read command) for the size of the user program written in the target MCU user area setting 256-byte aligned addresses for the read addresses.

The range of read addresses is as follows:

Addresses from 0xFFFC 0000 to 0xFFFF FFFF when the user program size is 256 Kbytes

Addresses from 0xFFFA 0000 to 0xFFFF FFFF when the user program size is 384 Kbytes

Addresses from 0xFFF8 0000 to 0xFFFF FFFF when the user program size is 512 Kbytes

Addresses from 0xFFF4 0000 to 0xFFFF FFFF when the user program size is 768 Kbytes

Addresses from 0xFFF0 2000 to 0xFFFF FFFF when the user program size is 1016 Kbytes

When the flash programmer receives data starting with 52h (response to the memory read command), it compares the read data with the source data in the RSK+RX63N user area. When the data do not match, or when the flash programmer receives data other than the response (data starting with 52h), it resets the target MCU to abort.

The range of source addresses is as follows:

Addresses from 0xFFF0 0000 to 0xFFF3 FFFF when the user program size is 256 Kbytes

Addresses from 0xFFF0 0000 to 0xFFF5 FFFF when the user program size is 384 Kbytes

Addresses from 0xFFF0 0000 to 0xFFF7 FFFF when the user program size is 512 Kbytes

Addresses from 0xFFF0 0000 to 0xFFFB FFFF when the user program size is 768 Kbytes

Addresses from 0xFFF0 0000 to 0xFFFF DFFF when the user program size is 1016 Kbytes

Figure 5.13 shows the Procedure to Confirm Data in the User Area.

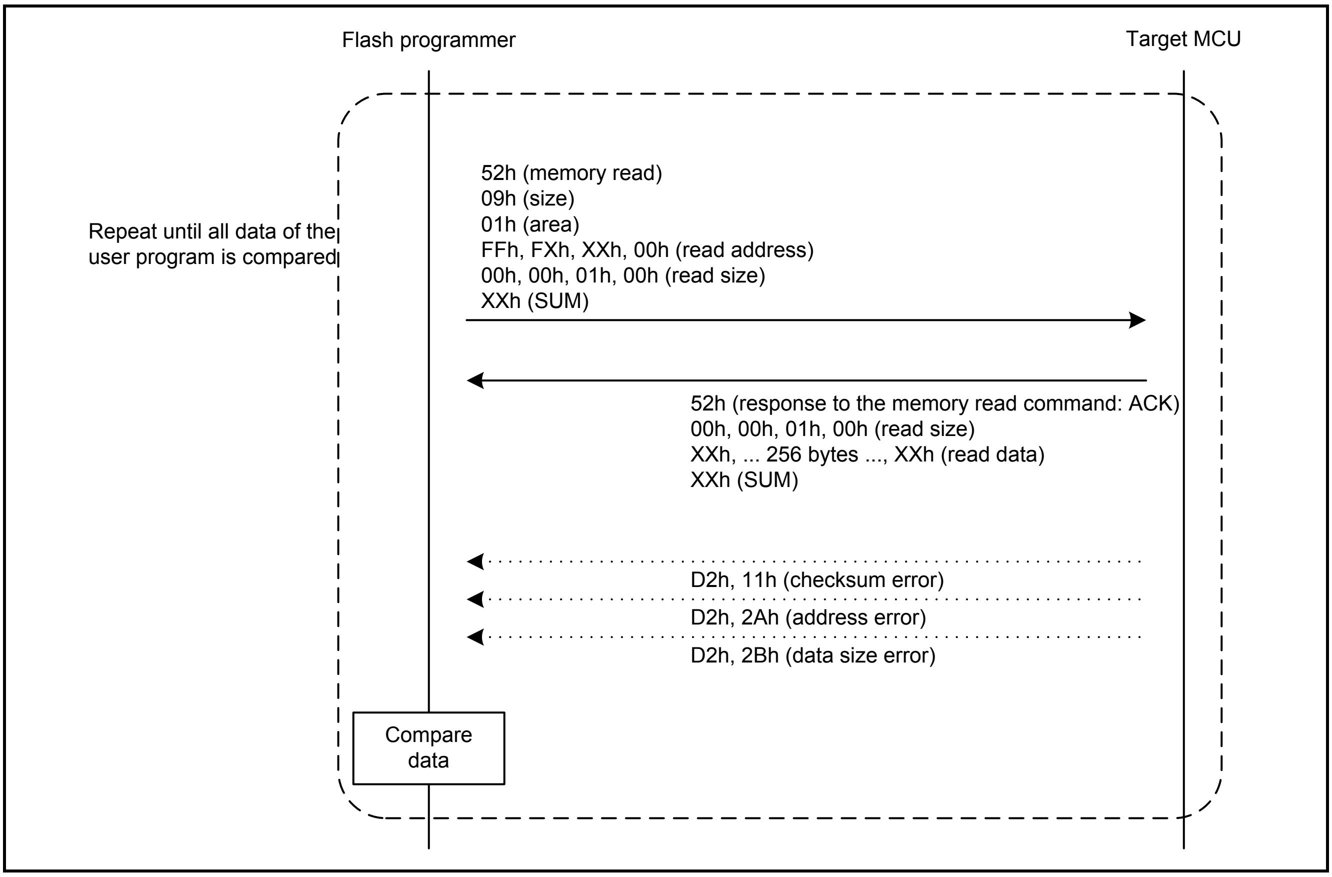


Figure 5.13 Procedure to Confirm Data in the User Area

5.2.6 Reset the Target MCU

(1) The flash programmer drives the MD pin of the target MCU high.

(2) The flash programmer drives the RES# pin of the target MCU low.

(3) The flash programmer waits 2 ms and then drives the RES# pin of the target MCU high.

(4) The flash programmer goes into an infinite loop.

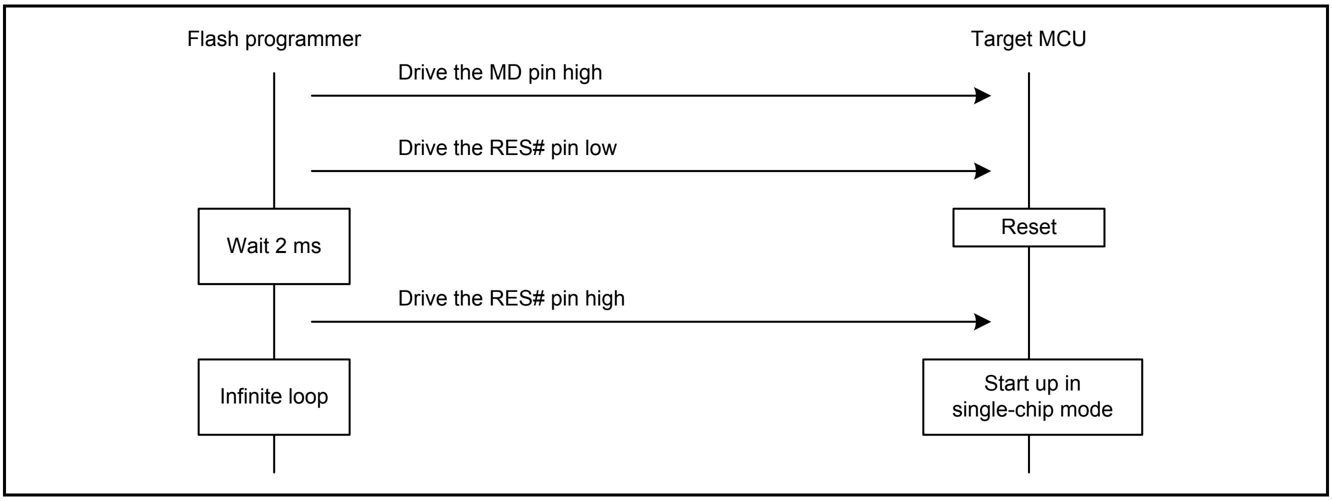


Figure 5.14 Procedure to Reset the Target MCU

5.3 File Composition

Table 5.2 lists the Files Used in the Sample Code. Table 5.3 lists the Standard Include Files. Table 5.4 and Table 5.5 list the Functions and Setting Values in the Reference Application Note. Files generated by the integrated development environment are not included in these tables.

Table 5.2 Files Used in the Sample Code

File Name	Outline
main.c	Main processing, processing to send a command, processing to receive a response
cmt_wait.c	Wait processing with the CMT
cmt_wait.h	Header file for cmt_wait.c

Table 5.3 Standard Include Files

File Name	Description
stdint.h	Defines macros declaring the integer type having the specified widths
stdbool.h	Defines macros for the Boolean type and value
machine.h	Defines formats of intrinsic functions for the RX Family
string.h	Library for comparing strings, copying, etc.

Table 5.4 Functions and Setting Values in the Reference Application Note (RX63N Group, RX631 Group Initial Setting)

File Name	Function	Setting Value
r_init_stop_module.c	R_INIT_StopModule()	-
r_init_stop_module.h	-	The DMAC/DTC or EXDMAC is set to stop.
r_init_non_existent_port.c	R_INIT_NonExistentPort()	-
r_init_non_existent_port.h	-	The 176-pin package is selected.
r_init_clock.c	R_INIT_Clock()	-
r_init_clock.h	-	The PLL is selected as the system clock. The sub-clock is not used.

Table 5.5 Functions and Setting Values in the Reference Application Note (RX63N Renesas Starter Kit Sample Code for Hi-performance Embedded Workshop)

File Name	Function	Setting Value
lcd.c	Init_LCD() Display_LCD()	-
lcd.h	-	-
rskrx63ndef.h	-	-

5.4 Option-Setting Memory

Table 5.6 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.6 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	After a reset, the IWDT stops. After a reset, the WDT stops.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	After a reset, voltage monitoring 0 reset is disabled. After a reset, the HOCO oscillation is disabled.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.5 Constants

Table 5.7 to Table 5.14 list the Constants Used in the Sample Code.

Table 5.7 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
ROMVOL_256KB	(256 * 1024)	Selected when the user area size of the target MCU is 256 Kbytes.
ROMVOL_384KB	(384 * 1024)	Selected when the user area size of the target MCU is 384 Kbytes.
ROMVOL_512KB	(512 * 1024)	Selected when the user area size of the target MCU is 512 Kbytes.
ROMVOL_768KB	(768 * 1024)	Selected when the user area size of the target MCU is 768 Kbytes.
ROMVOL_1MB	(1024 * 1024)	Selected when the user area size of the target MCU is 1 Mbytes.
TARGET_ROMVOL	ROMVOL_256KB	User area size of the target MCU (256 Kbytes selected)
TARGET_DATA_ADD	0xFFF00000	Start address for storing data programmed in the target MCU user area
FLASH_PRGRMA_SIZE	(8 * 1024)	Size of the flash programmer program (8 Kbytes)
READING_HEAD_ADD	WRITING_HEAD_ADD	Start address for reading the target MCU (same as the start address for programming)
MDES_ADD	0xFFFFFF80	MDES Determine Address
WRITING_TIME	(TARGET_ROMVOL / 256)	Number of times the target MCU is programmed (in 256 byte units)
READING_TIME	WRITING_TIME	Number of times the target MCU is read (same as the number of times the target MCU is programmed)
RES_BUF_SIZE	(262)	Size of the received data storage buffer
OK	(0)	True value
NG	(1)	False value
ERRLOOP_ON	(1)	Selected when error processing (infinite loop) is performed if an error is detected during reception.
ERRLOOP_OFF	(0)	Selected when error processing (infinite loop) is not performed if an error is detected during reception.
INTERVAL_ON	(1)	Selected when an interval is set during transmission.
INTERVAL_OFF	(0)	Selected when no interval is set during transmission.
RES_ACK_NORMAL	(0x06)	Normal ACK is received.
RES_ID_DISABLED	(0x26)	ACK for disabling ID code protection is received.
RES_ID_ENABLED	(0x16)	ACK for enabling ID code protection is received.
ARRAY_SIZE_OF(a)	(sizeof(a) / sizeof(a[0]))	Macro function obtaining the number of bytes for data sending commands
WT_BASE_US	(1000000L)	Operand for calculating wait time in 1 µs units
WT_BASE_MS	(1000L)	Operand for calculating wait time in 1 ms units
WT_CMT_CLOCK	(48L * WT_BASE_US)	CMT count source frequency (PCLKB: 48 MHz)
WT_CMT_DIVIDE	(512L)	CMT count source division ratio
WAIT_1MS	((1. * (WT_CMT_CLOCK/ WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (1 ms)
WAIT_2MS	((2. * (WT_CMT_CLOCK/ WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (2 ms)
WAIT_25MS	((25. * (WT_CMT_CLOCK/ WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (25 ms)
WAIT_100MS	((100. * (WT_CMT_CLOCK / WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (100 ms)
WAIT_400MS	((400. * (WT_CMT_CLOCK / WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (400 ms)

Table 5.8 Constants Used in the Sample Code (ROMVOL_256KB is Selected as TARGET_ROMVOL)

Constant Name	Setting Value	Contents
TARGET_ID1_ADD	0xFFF3FFA0	Reference address for the control code, and ID code 1 to ID code 3 programmed to the target MCU
TARGET_ID2_ADD	0xFFF3FFA4	Reference address for ID code 4 to ID code 7 programmed to the target MCU
TARGET_ID3_ADD	0xFFF3FFA8	Reference address for ID code 8 to ID code 11 programmed to the target MCU
TARGET_ID4_ADD	0xFFF3FFAC	Reference address for ID code 12 to ID code 15 programmed to the target MCU
WRITING_HEAD_ADD	0xFFFC0000	Start address for programming the target MCU
MAX_BLK_NUMBER	0x15	Maximum block number of the target MCU

Table 5.9 Constants Used in the Sample Code (ROMVOL_384KB is Selected as TARGET_ROMVOL)

Constant Name	Setting Value	Contents
TARGET_ID1_ADD	0xFFF5FFA0	Reference address for the control code, and ID code 1 to ID code 3 programmed to the target MCU
TARGET_ID2_ADD	0xFFF5FFA4	Reference address for ID code 4 to ID code 7 programmed to the target MCU
TARGET_ID3_ADD	0xFFF5FFA8	Reference address for ID code 8 to ID code 11 programmed to the target MCU
TARGET_ID4_ADD	0xFFF5FFAC	Reference address for ID code 12 to ID code 15 programmed to the target MCU
WRITING_HEAD_ADD	0xFFFA0000	Start address for programming the target MCU
MAX_BLK_NUMBER	0x1D	Maximum block number of the target MCU

Table 5.10 Constants Used in the Sample Code (ROMVOL_512KB is Selected as TARGET_ROMVOL)

Constant Name	Setting Value	Contents
TARGET_ID1_ADD	0xFFF7FFA0	Reference address for the control code, and ID code 1 to ID code 3 programmed to the target MCU
TARGET_ID2_ADD	0xFFF7FFA4	Reference address for ID code 4 to ID code 7 programmed to the target MCU
TARGET_ID3_ADD	0xFFF7FFA8	Reference address for ID code 8 to ID code 11 programmed to the target MCU
TARGET_ID4_ADD	0xFFF7FFAC	Reference address for ID code 12 to ID code 15 programmed to the target MCU
WRITING_HEAD_ADD	0xFFF80000	Start address for programming the target MCU
MAX_BLK_NUMBER	0x25	Maximum block number of the target MCU

Table 5.11 Constants Used in the Sample Code (ROMVOL_768KB is Selected as TARGET_ROMVOL)

Constant Name	Setting Value	Contents
TARGET_ID1_ADD	0xFFFBFFA0	Reference address for the control code, and ID code 1 to ID code 3 programmed to the target MCU
TARGET_ID2_ADD	0xFFFBFFA4	Reference address for ID code 4 to ID code 7 programmed to the target MCU
TARGET_ID3_ADD	0xFFFBFFA8	Reference address for ID code 8 to ID code 11 programmed to the target MCU
TARGET_ID4_ADD	0xFFFBFFAC	Reference address for ID code 12 to ID code 15 programmed to the target MCU
WRITING_HEAD_ADD	0xFFF40000	Start address for programming the target MCU
MAX_BLK_NUMBER	0x2D	Maximum block number of the target MCU

Table 5.12 Constants Used in the Sample Code (ROMVOL_1MB is Selected as TARGET_ROMVOL)

Constant Name	Setting Value	Contents
TARGET_ID1_ADD	0xFFFFDFA0	Reference address for the control code, and ID code 1 to ID code 3 programmed to the target MCU
TARGET_ID2_ADD	0xFFFFDFA4	Reference address for ID code 4 to ID code 7 programmed to the target MCU
TARGET_ID3_ADD	0xFFFFDFA8	Reference address for ID code 8 to ID code 11 programmed to the target MCU
TARGET_ID4_ADD	0xFFFFDFAC	Reference address for ID code 12 to ID code 15 programmed to the target MCU
WRITING_HEAD_ADD	0xFFF02000	Start address for programming the target MCU
MAX_BLK_NUMBER	0x35	Maximum block number of the target MCU

Table 5.13 Constants Used in the Sample Code (Definition Used for Entering Boot Mode)

Constant Name	Setting Value	Contents
BTMD_PMR	(PORTE.PMR.BYTE)	Output pin is assigned to pins PC7, MD, and RES# of the target MCU (port mode register).
BTMD_PODR	(PORTE.PODR.BYTE)	Output pin is assigned to pins PC7, MD, and RES# of the target MCU (port output data register).
BTMD_PDR	(PORTE.PDR.BYTE)	Output pin is assigned to pins PC7, MD, and RES# of the target MCU (port direction register).
UB_PIN	(PORTE.PODR.BIT.B2)	Output is assigned to the PC7 pin of the target MCU.
MD_PIN	(PORTE.PODR.BIT.B1)	Output is assigned to the MD pin of the target MCU.
RES_PIN	(PORTE.PODR.BIT.B0)	Output is assigned to the RES# pin of the target MCU.
BTMD_PDR_INIT	(0x07)	Initial value of the output from pins PC7, MD, and RES# of the target MCU
BTMD_PODR_INIT	(0x00)	Initial value of high level output from the PC7 pin of the target MCU

Table 5.14 Constants Used in the Sample Code (Definition for Asynchronous Serial Communication)

Constant Name	Setting Value	Contents
SCIn	SCI0	SCI channel: SCI0
MSTP_SCIn	MSTP(SCI0)	SCI0 module stop bit
IR_SCIn_RXIn	IR(SCI0,RXI0)	SCI0.RXI0 interrupt status flag
IR_SCIn_TXIn	IR(SCI0,TXI0)	SCI0.TXI0 interrupt status flag
RXDn_PDR	(PORT3.PDR.BIT.B3)	SCI0.RXI0 pin direction control bit
RXDn_PMR	(PORT3.PMR.BIT.B3)	SCI0.RXI0 pin mode control bit
RXDnPFS	P33PFS	SCI0.RXI0 pin function control register
RXDnPFS_SELECT	(0x0B)	RXD0 pin function select bit setting value
TXDn_PODR	(PORT3.PODR.BIT.B2)	SCI0.TXI0 pin output data store bit
TXDn_PDR	(PORT3.PDR.BIT.B2)	SCI0.TXI0 pin direction control bit
TXDn_PMR	(PORT3.PMR.BIT.B2)	SCI0.TXI0 pin mode control bit
TXDnPFS	P32PFS	SCI0.TXI0 pin function control register
TXDnPFS_SELECT	(0x0B)	TXD0 pin function select bit setting value
SSR_ERROR_FLAGS	(0x38)	Bit pattern of error flags in the SCI.SSR register
BRR_SET(bps)	(WT_CMT_CLOCK/(32*(0.5)*(bps))-1+0.5)	Macro function to calculate the SCI.BRR register setting value

5.6 Structure/Union List

Figure 5.15 shows the Structure/Union Used in the Sample Code.

```
typedef struct{
    uint32_t   TrnSize;      /* expected value of the transmit size of command */
    uint32_t   RecSize;      /* expected value of the receive size of response */
    uint8_t    ACKRes;       /* ACK value of response */
    uint8_t    *Command;     /* boot command sequence data pointer */
} boot_cmd_t;
```

Figure 5.15 Structure/Union Used in the Sample Code

5.7 Variables

Table 5.15 lists the Global Variable, and Table 5.16 lists the static Variables.

Table 5.15 Global Variable

Type	Variable Name	Contents	Function
volatile uint8_t	CMT_InterruptFlag	Wait time enable flag	CMT_WaitSet CMT_Wait Excep_CMT0_CMI0 ReceiveResponse

Table 5.16 static Variables

Type	Variable Name	Contents	Function
uint8_t	ResponseBuffer[RES_BUF_SIZE]	Receive data storage buffer	main ReceiveResponse
uint8_t	TransferMode	Transmit mode flag	main TransferCommand
uint8_t	ReceiveMode	Receive mode flag	main ReceiveResponse
uint8_t	IDProtectMode	ID code protection status buffer	main
uint32_t	BufferIndex	Index of the receive data storage buffer	ReceiveResponse
uint32_t	DeviceCode	Device code storage buffer	main
uint8_t	CMD_BitRateAdjustment_1st[]	Bit rate automatic adjust command data	-
uint8_t	CMD_BitRateAdjustment_2nd[]	Bit rate automatic adjustment confirm command data	-
uint8_t	CMD_EnquiryDevice[]	Supported device inquiry command data	-
uint8_t	CMD_SelectDevice[]	Device selection command data	-
uint8_t	CMD_SelectClockMode[]	Clock mode selection command data	-
uint8_t	CMD_OperatingFreqSel_1st[]	New bit rate selection command data	-
uint8_t	CMD_OperatingFreqSel_2nd[]	New bit rate selection confirm command data	-
uint8_t	CMD_PEstatusTransition[]	Programming/erasure state transition command data	-
uint8_t	CMD_IDCodeCheck[]	ID code check command data	-
uint8_t	CMD_EraseSelection[]	Erasure selection command data	-
uint8_t	CMD_BlockErase[]	Block erase command data	-
uint8_t	CMD_ ProgramSelection[]	User/data area program selection command data	-
uint8_t	CMD_Program[]	Program command data	-
uint8_t	CMD_ProgramTermination[]	Program end command data	-
uint8_t	CMD_MemoryRead[]	Memory read command data	-
boot_cmd_t	BitRateAdjustment_1st	Bit rate automatic adjust command structure	main
boot_cmd_t	BitRateAdjustment_2nd	Bit rate automatic adjustment confirm command structure	main
boot_cmd_t	EnquiryDevice	Supported device inquiry command structure	main
boot_cmd_t	SelectDevice	Device selection command structure	main
boot_cmd_t	SelectClockMode	Clock mode selection command structure	main
boot_cmd_t	OperatingFreqSel_1st	New bit rate selection command structure	main
boot_cmd_t	OperatingFreqSel_2nd	New bit rate selection confirm command structure	main
boot_cmd_t	PEstatusTransition	programming/erasure state transition command structure	main
boot_cmd_t	IDCodeCheck	ID code check command structure	main
boot_cmd_t	EraseSelection	Erasure selection command structure	main
boot_cmd_t	BlockErase	Block erase command structure	main
boot_cmd_t	ProgramSelection	User/data area programming selection command structure	main
boot_cmd_t	Program	Programming command structure	main
boot_cmd_t	ProgramTermination	Programming end command structure	main
boot_cmd_t	MemoryRead	Memory read command structure	main

5.8 Functions

Table 5.17 lists the Functions.

Table 5.17 Functions

Function Name	Outline
main	Main processing and communication protocol control
peripheral_init	Initialization of the peripheral functions
CMT_WaitInit	Initialization of the timer for wait time with the CMT
CMT_WaitSet	Wait time setting with the CMT
CMT_Wait	Wait time processing with the CMT
Excep_CMT0_CMI0	Interrupt handling for CMI0 in CMT0
SCI_Init	Initialization of the SCI
SCI_change	Processing to change the SCI bit rate
CalcSumData	Processing to calculate the SUM data
BootModeEntry	Processing to start the target MCU in boot mode
BootModeRelease	Processing to reset the target MCU
TransferCommand	Processing to send a command
ReceiveResponse	Processing to receive a response
U4memcpy	Copying unsigned 4-byte data

5.9 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	lcd.h, cmt_wait.h
Declaration	void main(void)
Description	After initialization, starts the target MCU in boot mode and rewrite the user area.
Arguments	None
Return Value	None

peripheral_init	
Outline	Initialization of the peripheral functions
Header	lcd.h, cmt_wait.h
Declaration	void peripheral_init(void)
Description	Initializes the peripheral functions used.
Arguments	None
Return Value	None

CMT_WaitInit	
Outline	Initialization of the timer for wait time with the CMT
Header	cmt_wait.h
Declaration	void CMT_WaitInit(void)
Description	Initializes the timer for wait time (CMT0).
Arguments	None
Return Value	None

CMT_WaitSet	
Outline	Wait time setting with the CMT
Header	cmt_wait.h
Declaration	void CMT_WaitSet(uint16_t cnt)
Description	Sets the CMCOR register to the time (μs) specified in the argument and starts incrementing the CMCNT register.
Arguments	uint16_t cnt: Wait time
Return Value	None
Remarks	Minimum wait time: 1 ÷ (PCLKB[MHz] ÷ 512) ≈ 10.67 μs

CMT_Wait	
Outline	Wait time processing with the CMT
Header	cmt_wait.h
Declaration	void CMT_Wait(uint16_t cnt)
Description	Waits the time (μs) specified in the argument.
Arguments	uint16_t cnt: Wait time
Return Value	None
Remarks	Minimum wait time: 1 ÷ (PCLKB[MHz] ÷ 512) ≈ 10.67 μs

Excep_CMT0_CMI0	
Outline	Interrupt handling for CMI0 in CMT0
Header	cmt_wait.h
Declaration	void Excep_CMT0_CMI0(void)
Description	Interrupt handling for compare match between CMT0.CMCNT and CMT0.CMCOR
Arguments	None
Return Value	None

SCI_Init	
Outline	Initialization of the SCI
Header	None
Declaration	void SCI_Init(void)
Description	Initializes the SCI.
Arguments	None
Return Value	None

SCI_change	
Outline	Processing to change the SCI bit rate
Header	None
Declaration	void SCI_change(void)
Description	Changes the SCI bit rate from 19,200 bps to 1.5 Mbps.
Arguments	None
Return Value	None

CalcSumData	
Outline	Processing to calculate the SUM data
Header	None
Declaration	uint8_t CalcSumData(uint8_t *pData, uint32_t Length)
Description	Calculates the SUM data in the boot communication protocol.
Arguments	uint8_t *pData: Data address for SUM uint32_t Length: Amount of data for SUM
Return Value	SUM data

BootModeEntry	
Outline	Processing to start the target MCU in boot mode
Header	None
Declaration	void BootModeEntry(void)
Description	Controls pins MD, PC7, and RES# to start the target MCU in boot mode.
Arguments	None
Return Value	None

BootModeRelease	
Outline	Processing to reset the target MCU
Header	None
Declaration	void BootModeRelease(uint8_t mode)
Description	Resets the target MCU.
Arguments	uint8_t mode: Select the output pattern for the second line of the debug LCD
Return Value	None

TransferCommand	
Outline	Processing to send a command
Header	None
Declaration	void TransferCommand(boot_cmd_t *pCmd)
Description	Sends command data of the command structure specified in the argument.
Arguments	boot_cmd_t *pCmd: Address of the command structure to be sent
Return Value	None
Remarks	Call CMT_Wait(WAIT_1MS) if the TransferMode variable is INTERVAL_ON.

ReceiveResponse	
Outline	Processing to receive a response
Header	None
Declaration	uint8_t ReceiveResponse(boot_cmd_t *pCmd)
Description	Receives a response for the number of bytes of the expected response size in the command structure.
Arguments	boot_cmd_t *pCmd: Address of the command structure to be received
Return Value	OK: Reception completed successfully NG: Timeout (5 seconds) or error response received
Remarks	When the ReceiveMode variable is ERRLOOP_ON and the return value is NG, call the BootModeRelease(NG) function and do not return from the ReceiveResponse function.

U4memcpy	
Outline	Copying unsigned 4-byte data
Header	None
Declaration	void *U4memcpy(void *pS1, const void *pS2)
Description	Copies 4 bytes of data in the source memory area to the destination memory area. If the data arrangement is little endian, reverses bytes of the unsigned 4-byte data in the destination.
Arguments	void *pS1: Address of the destination memory area const void *pS2: Address of the source memory area
Return Value	pS1 value

5.10 Flowcharts

5.10.1 Main Processing and Communication Protocol Control

Figure 5.16 to Figure 5.28 show the Main Processing and Communication Protocol Control.

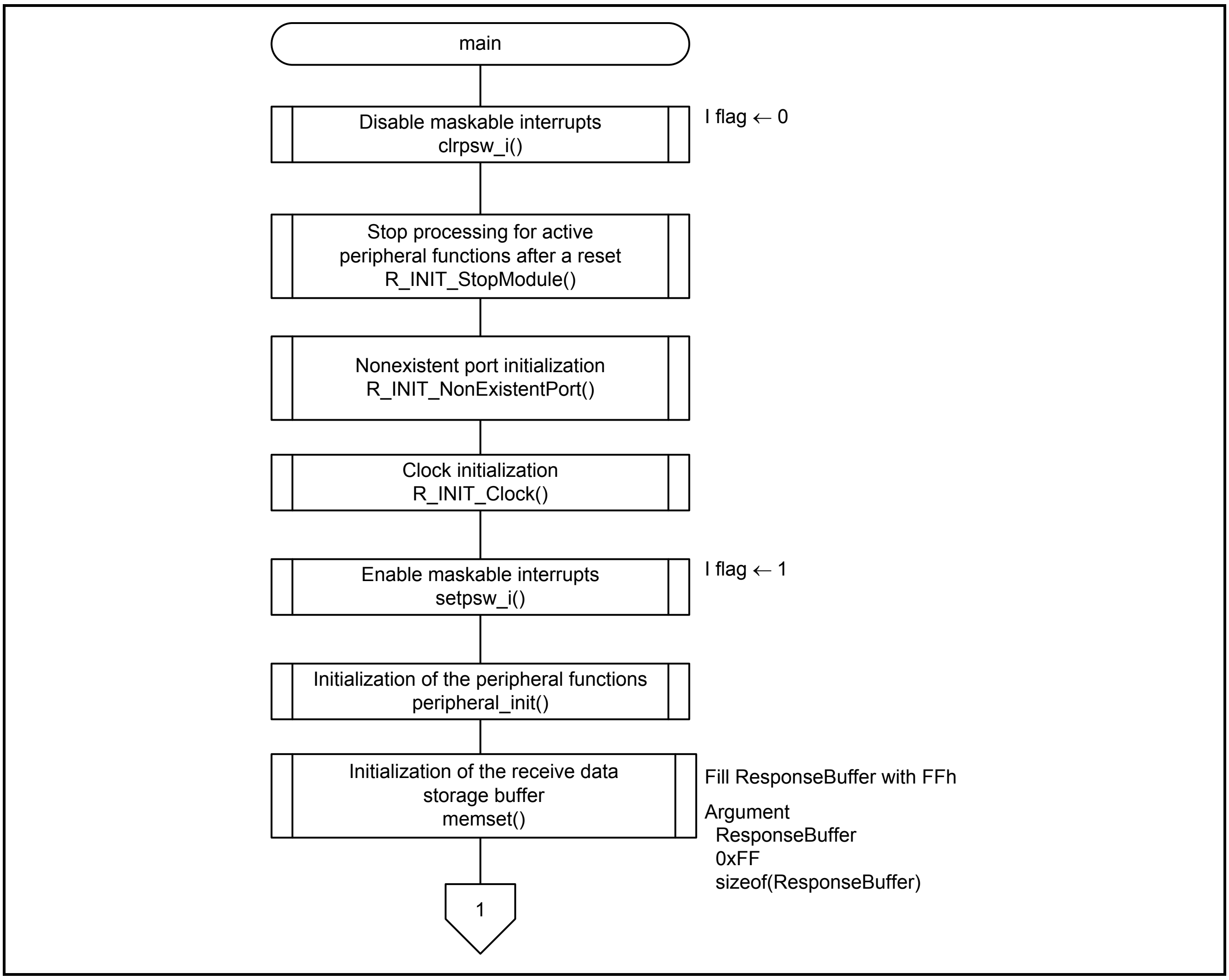


Figure 5.16 Main Processing and Communication Protocol Control (1/13)

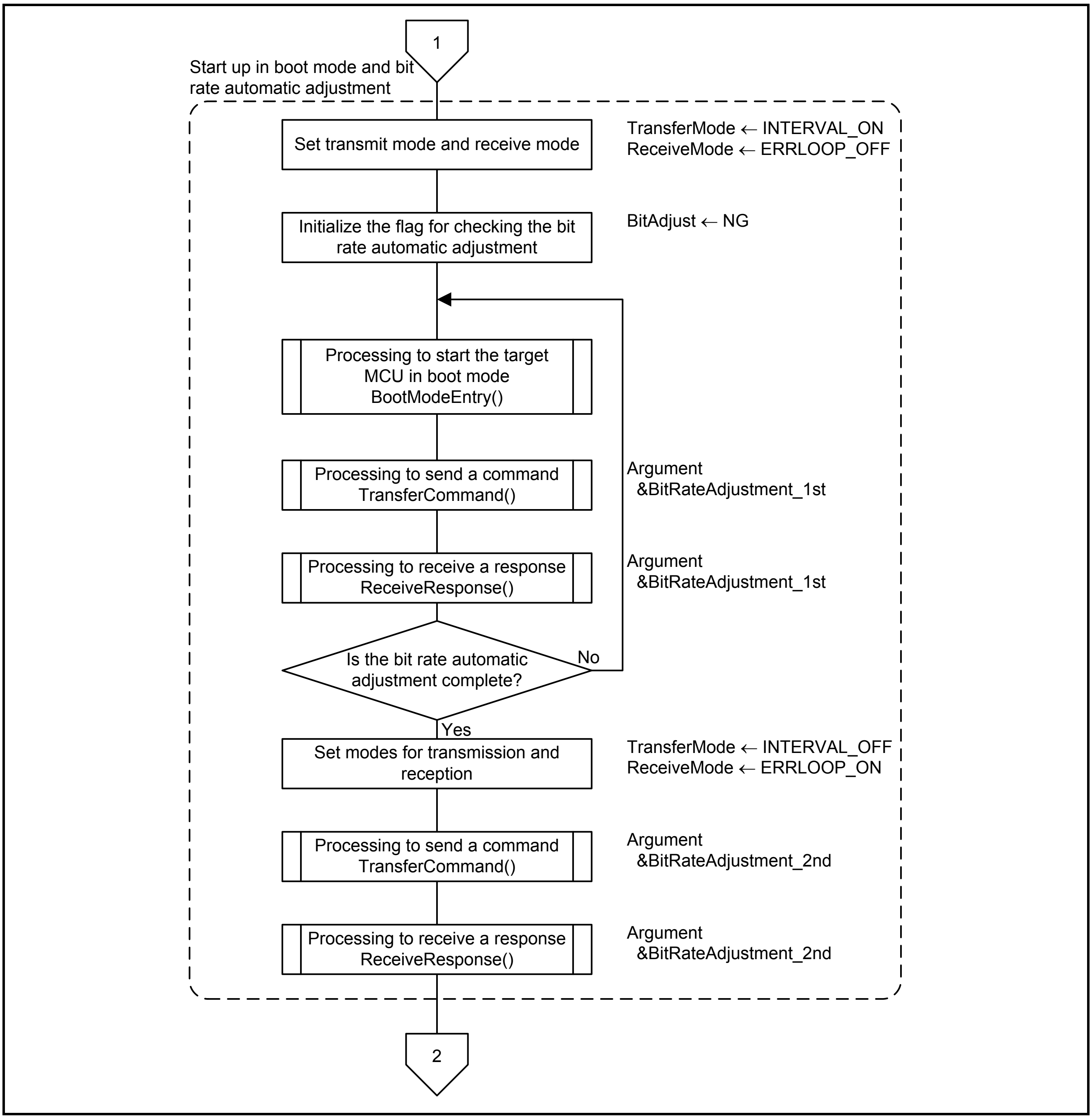


Figure 5.17 Main Processing and Communication Protocol Control (2/13)

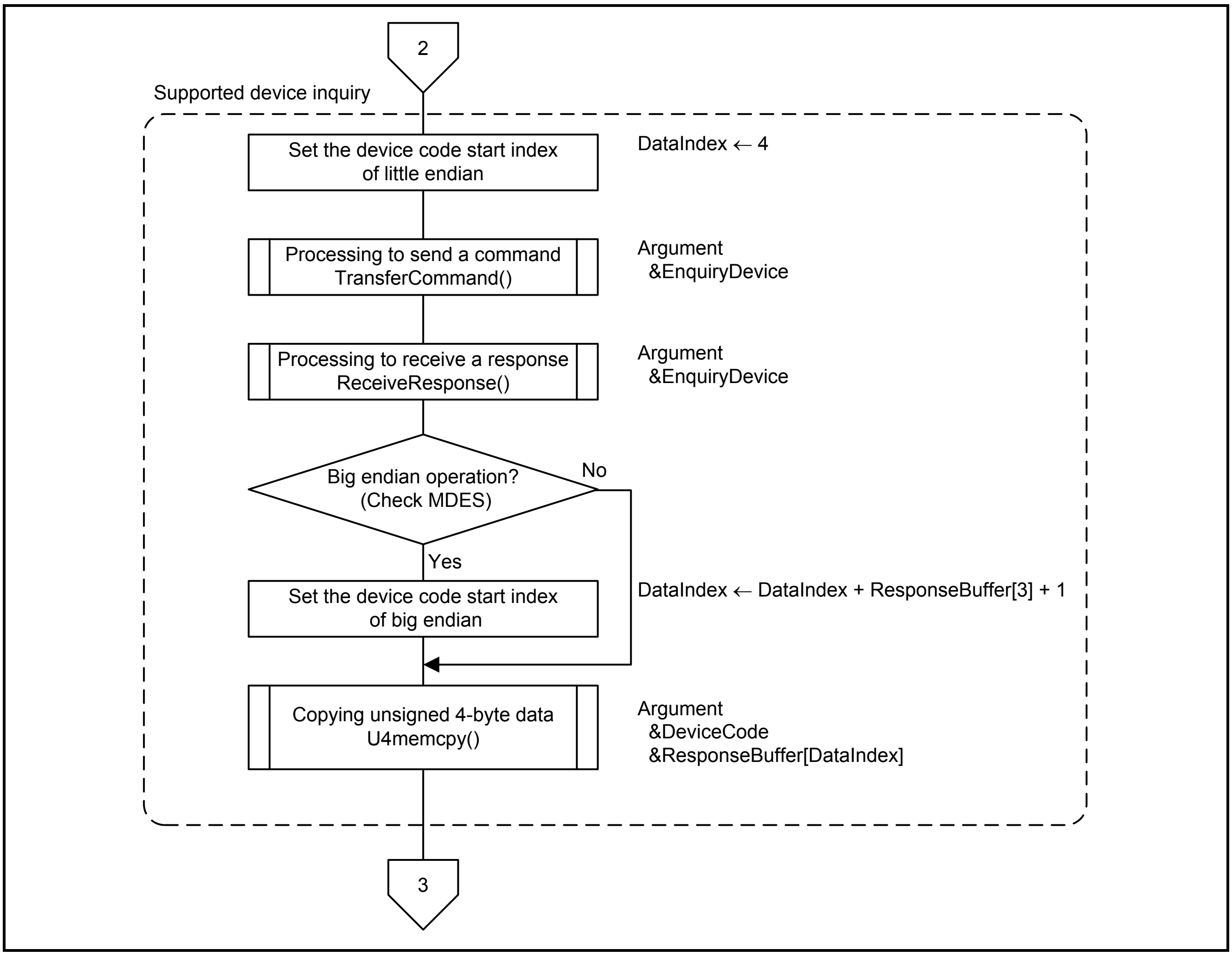


Figure 5.18 Main Processing and Communication Protocol Control (3/13)

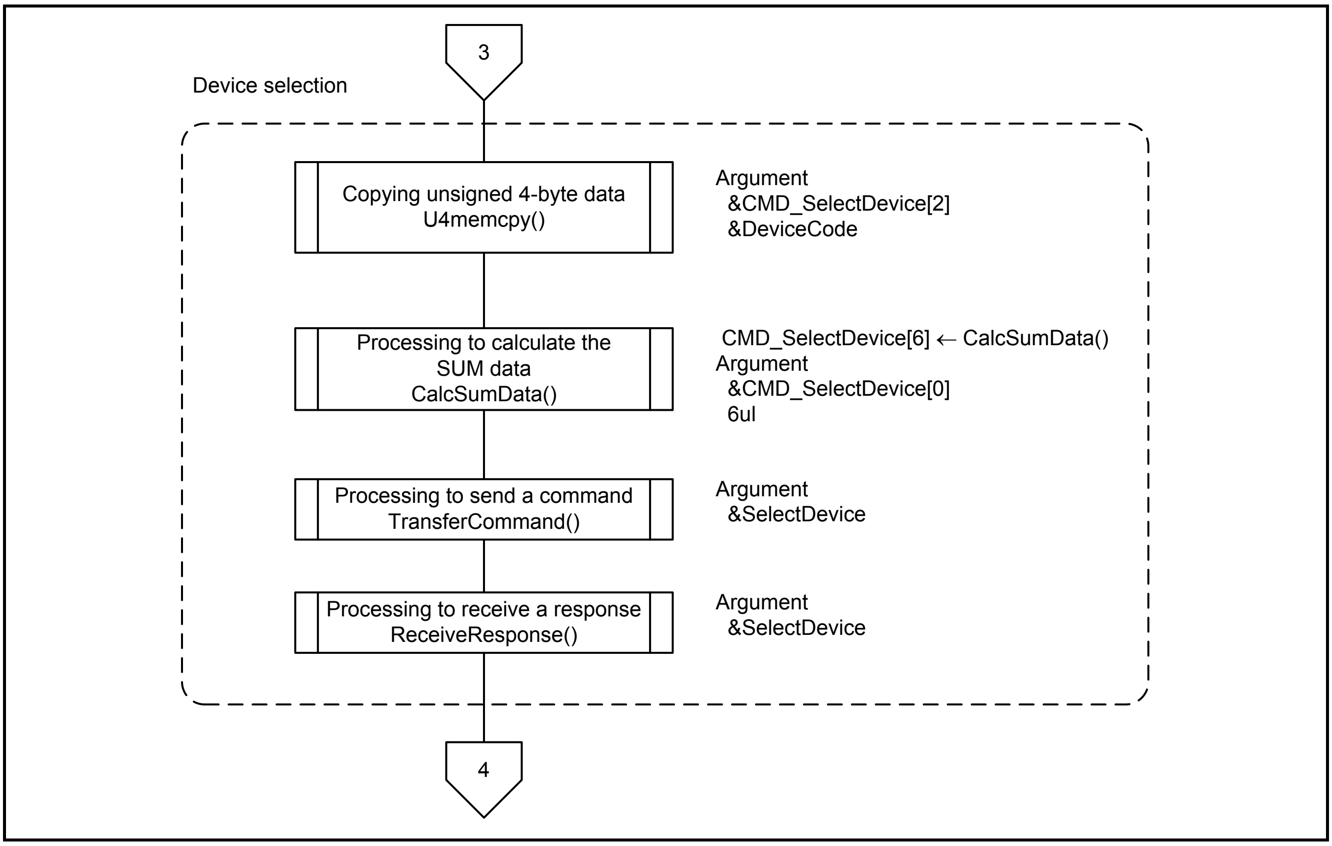


Figure 5.19 Main Processing and Communication Protocol Control (4/13)

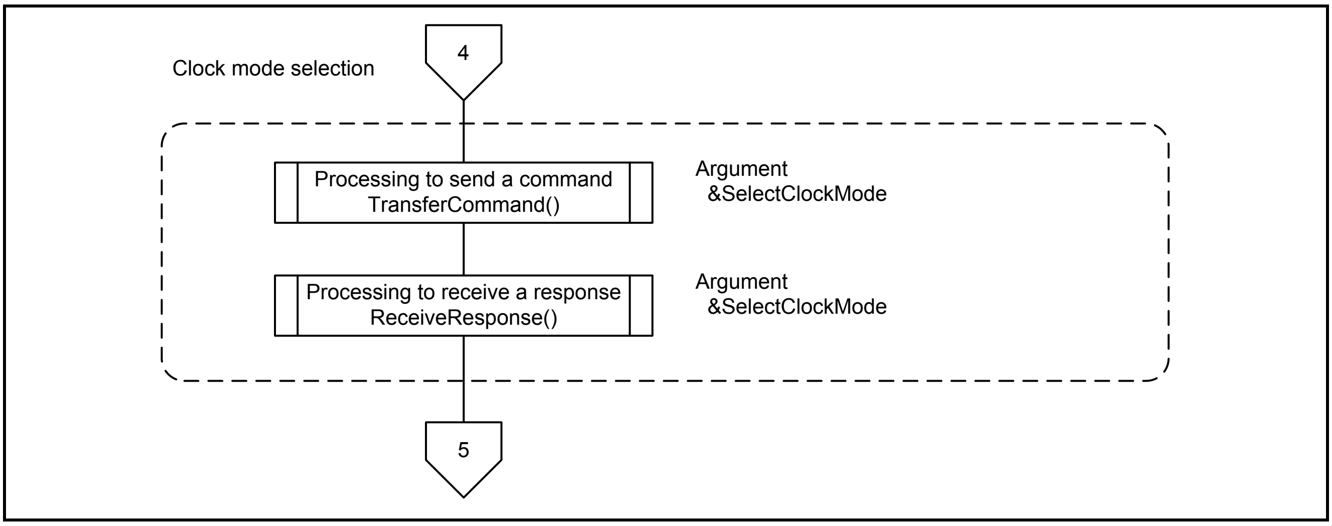


Figure 5.20 Main Processing and Communication Protocol Control (5/13)

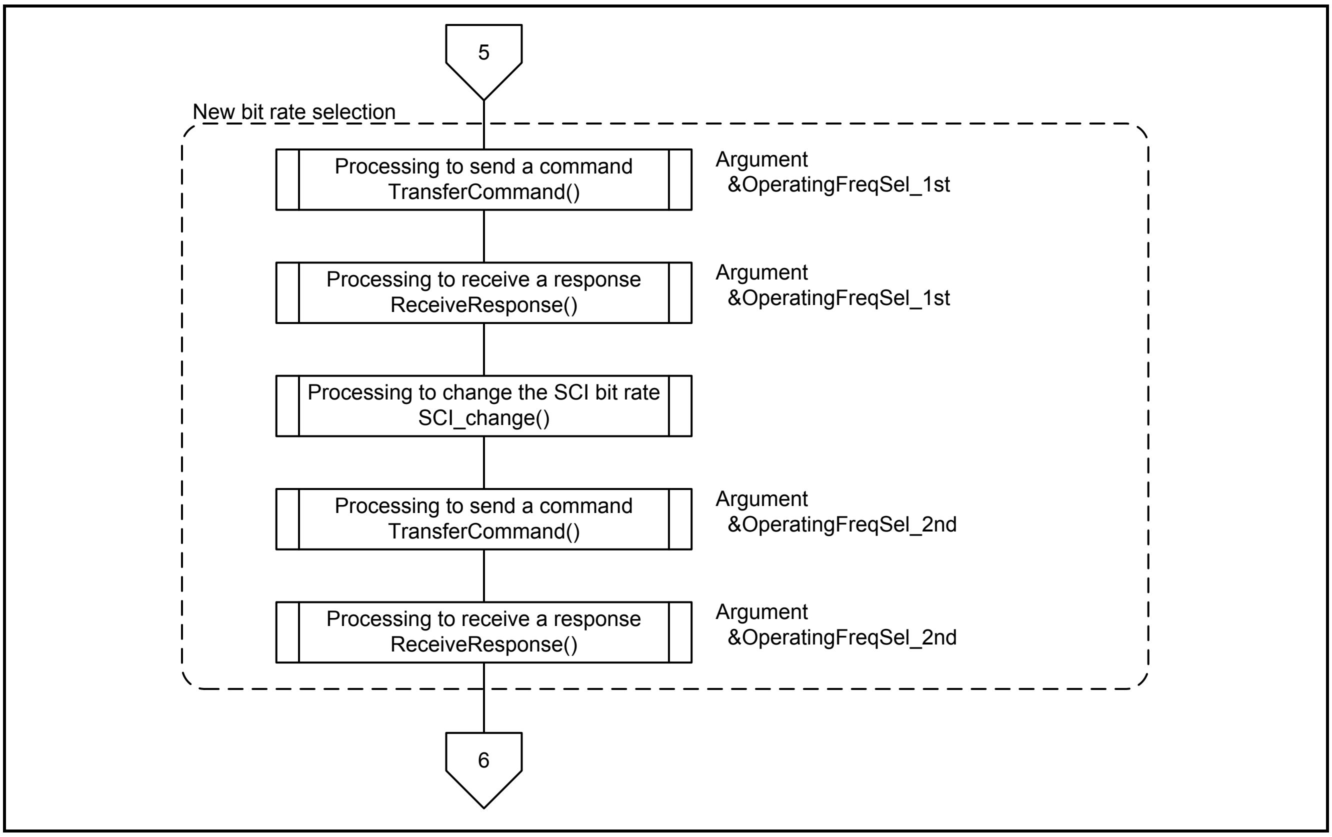


Figure 5.21 Main Processing and Communication Protocol Control (6/13)

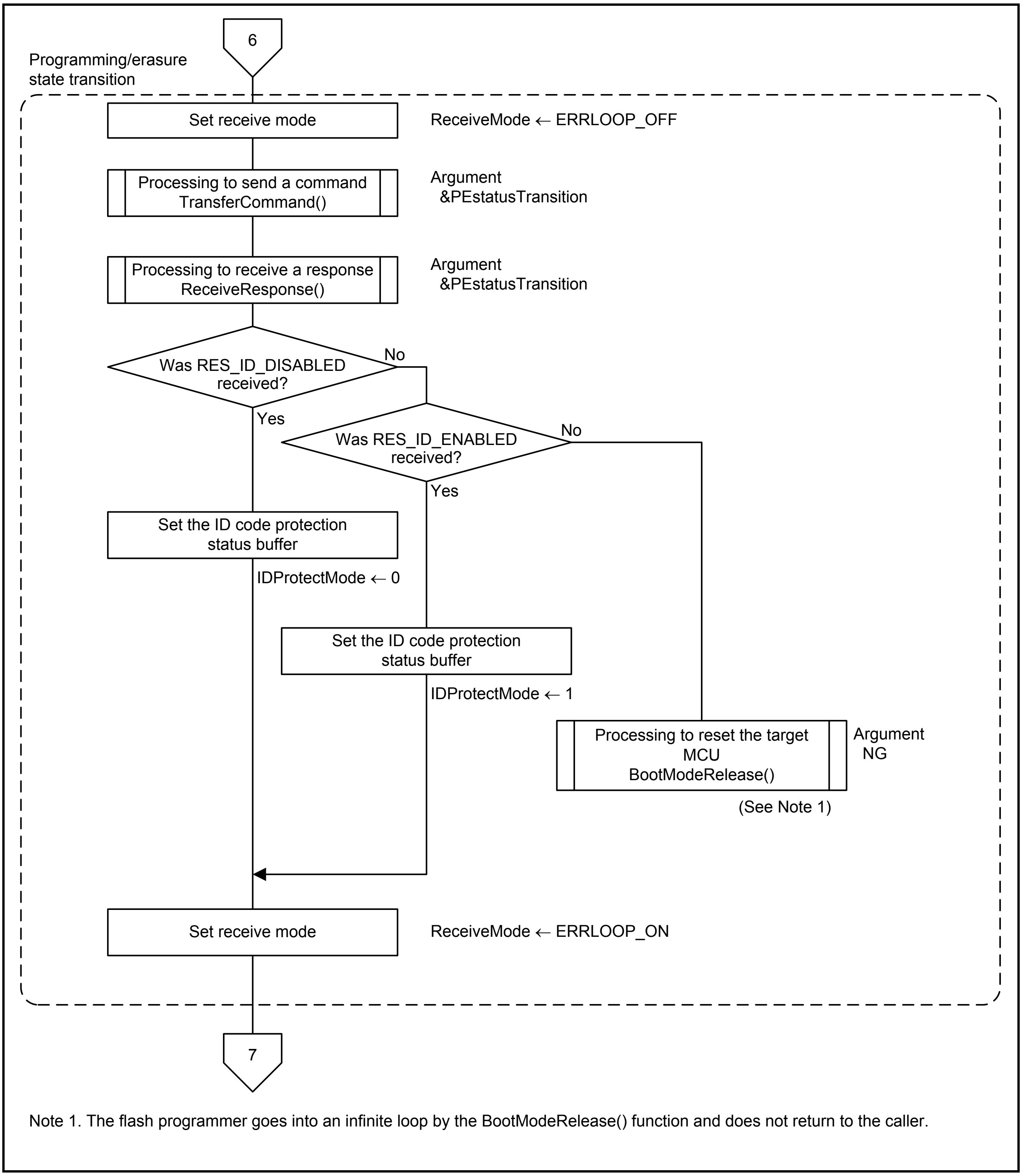


Figure 5.22 Main Processing and Communication Protocol Control (7/13)

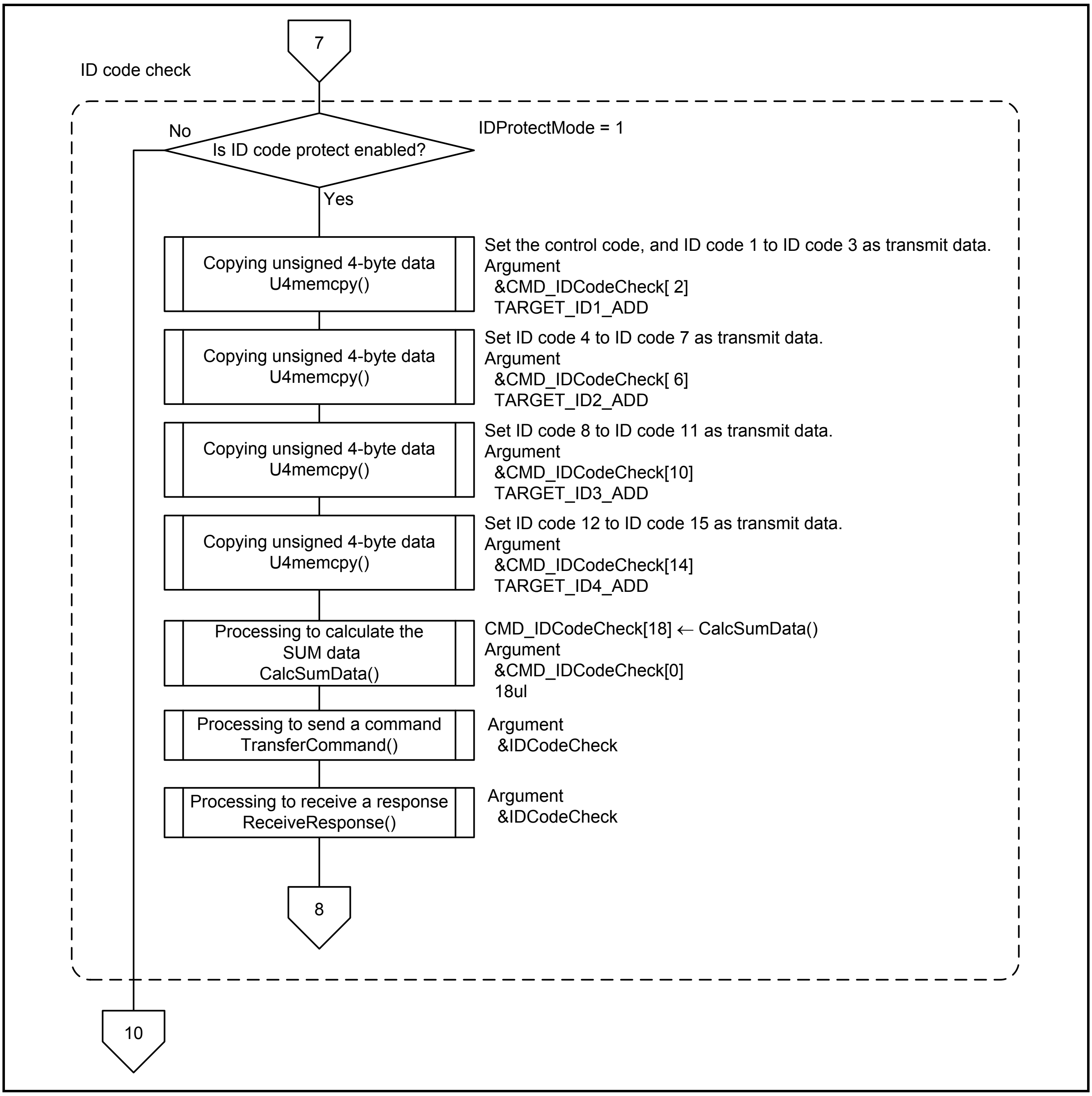


Figure 5.23 Main Processing and Communication Protocol Control (8/13)

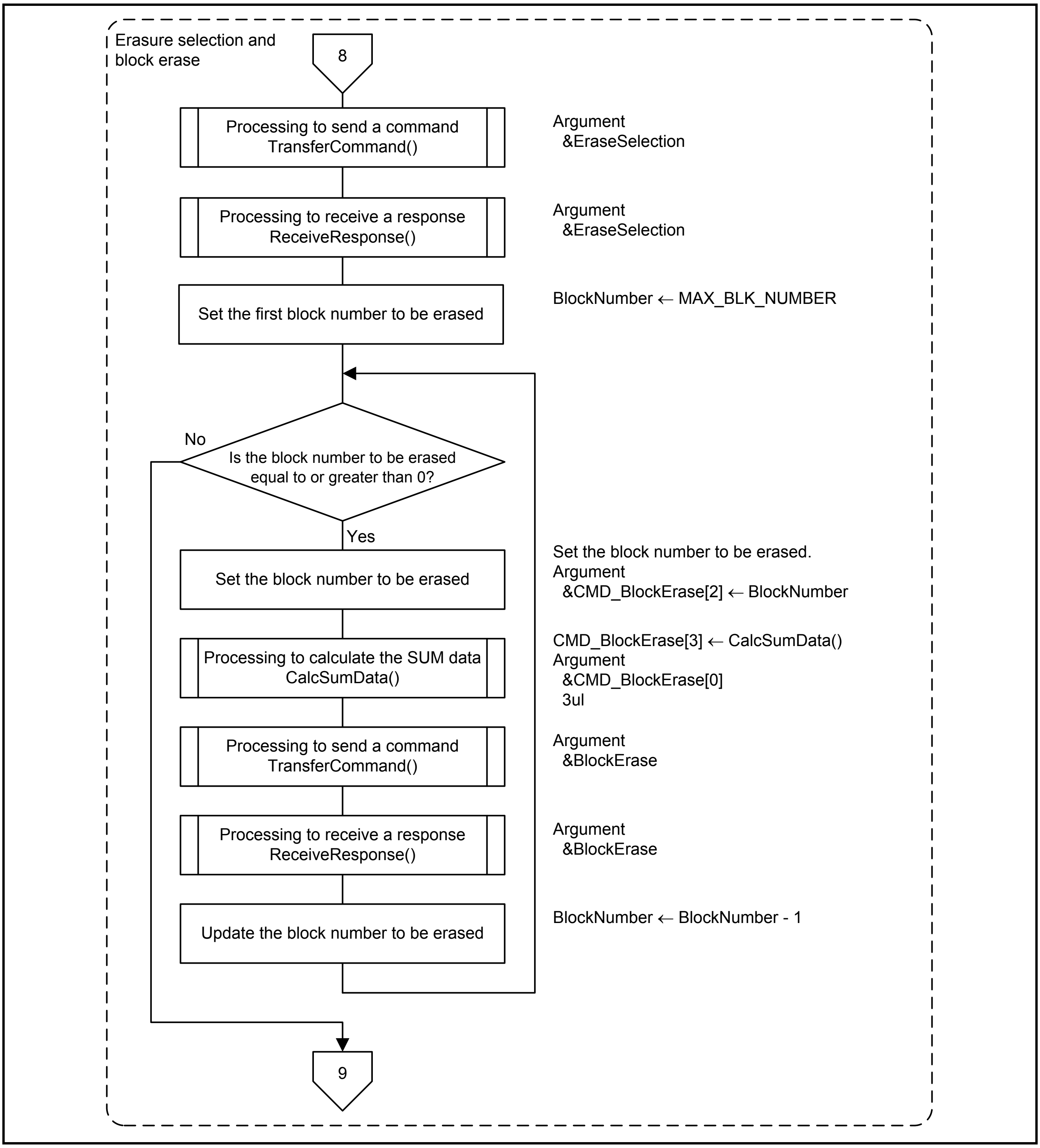


Figure 5.24 Main Processing and Communication Protocol Control (9/13)

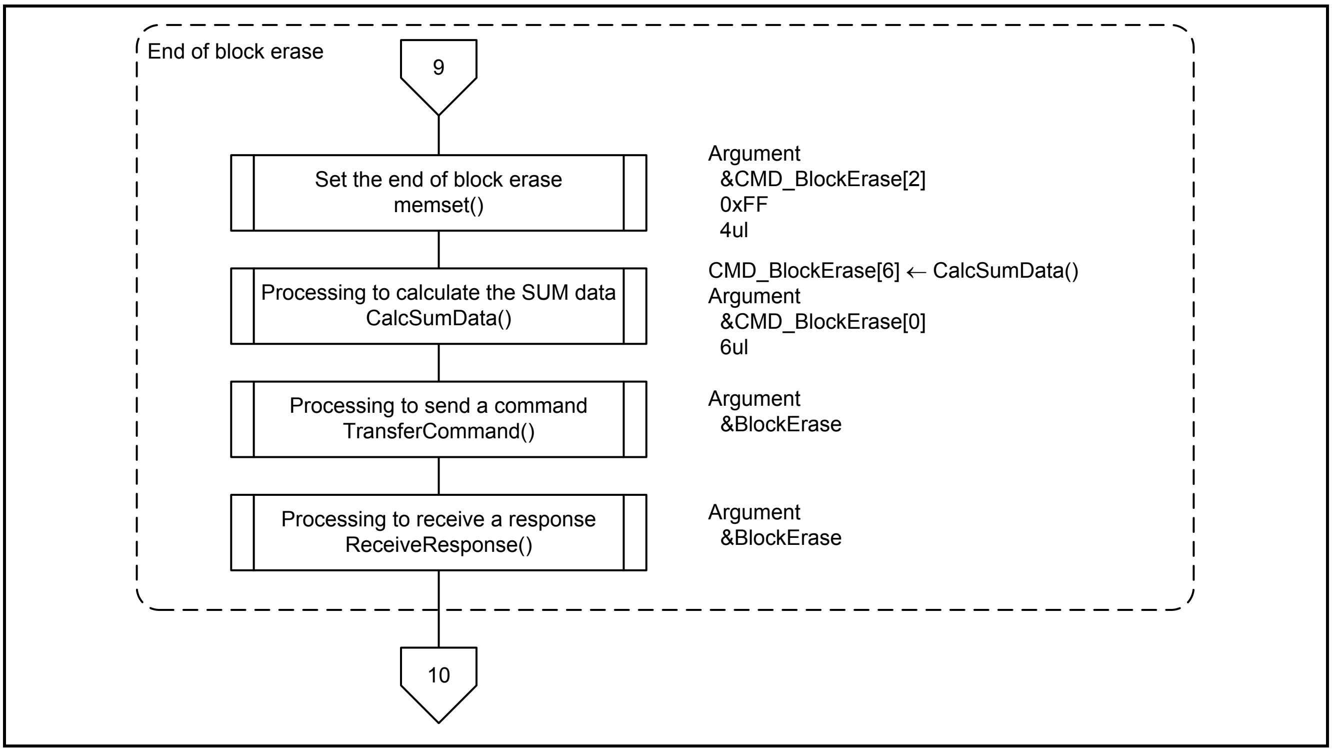


Figure 5.25 Main Processing and Communication Protocol Control (10/13)

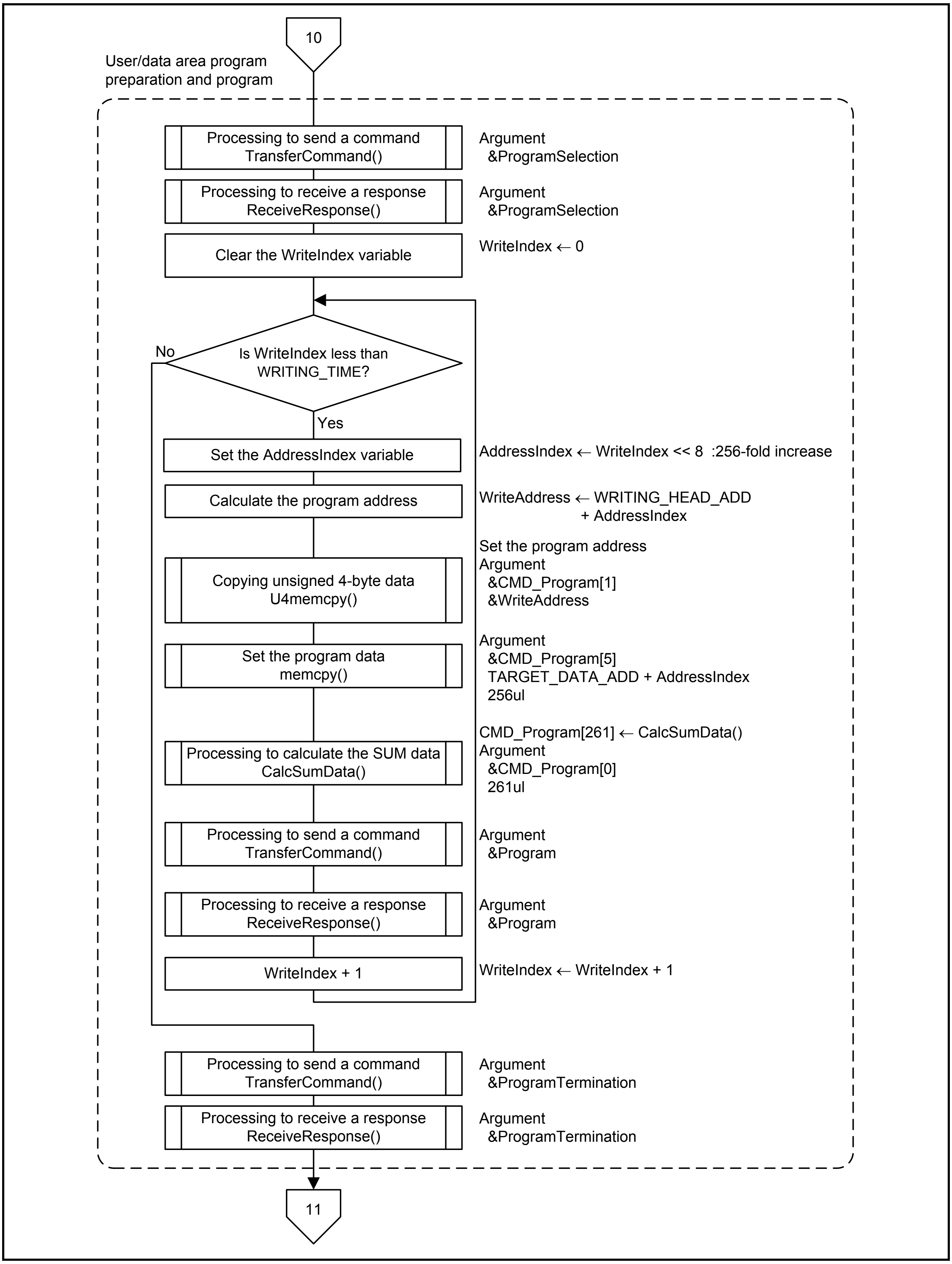
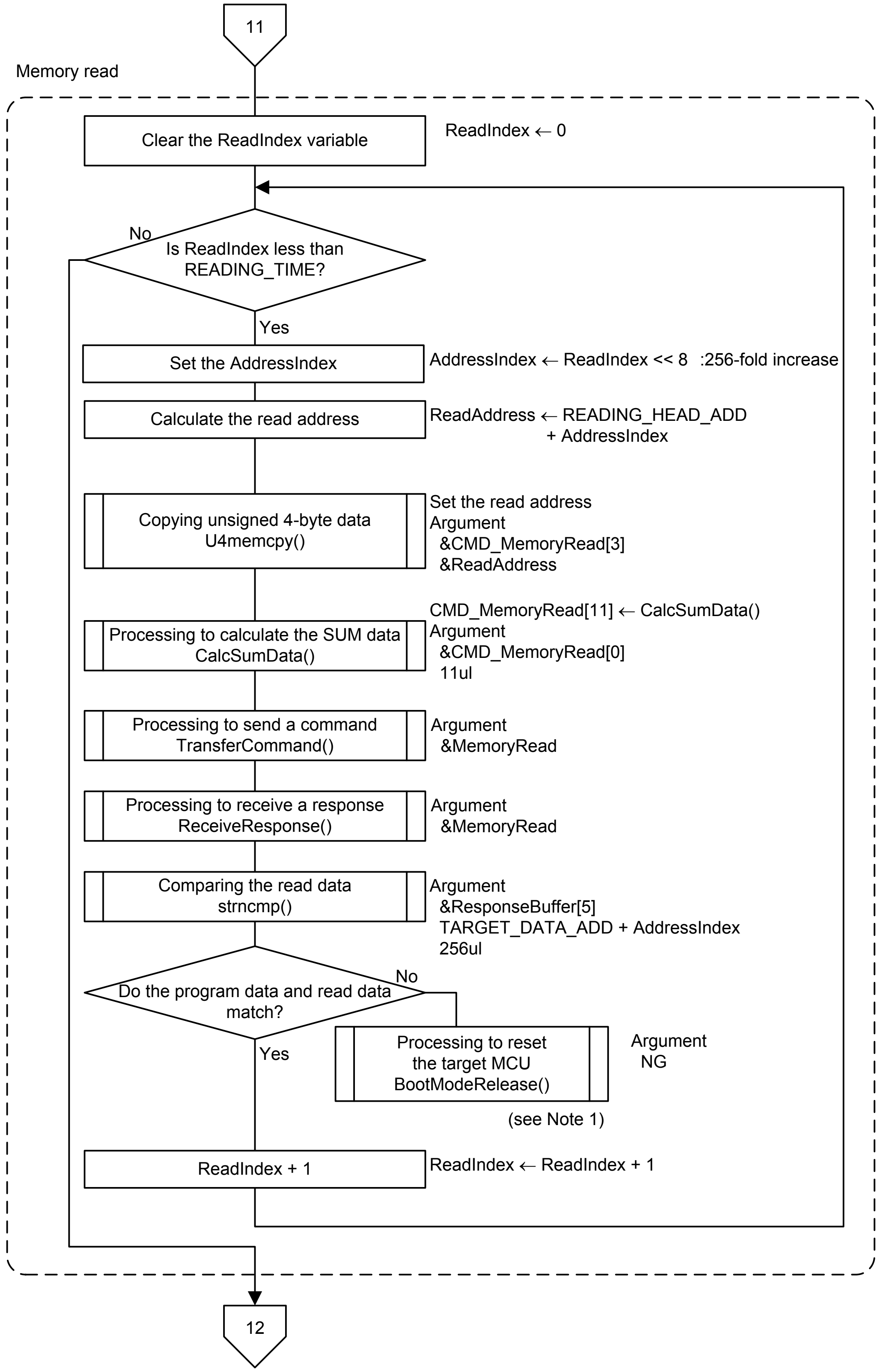
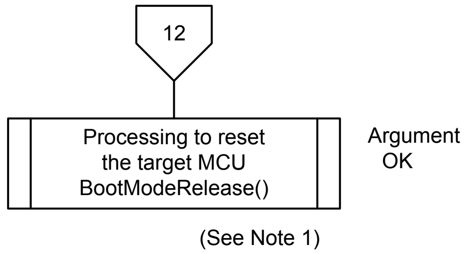


Figure 5.26 Main Processing and Communication Protocol Control (11/13)



Note 1. The flash programmer goes into an infinite loop by the BootModeRelease() function and does not return to the caller.

Figure 5.27 Main Processing and Communication Protocol Control (12/13)



Note 1. The flash programmer goes into an infinite loop by the BootModeRelease() function and does not return to the caller.

Figure 5.28 Main Processing and Communication Protocol Control (13/13)

5.10.2 Initialization of the Peripheral Functions

Figure 5.29 shows the Initialization of the Peripheral Functions.

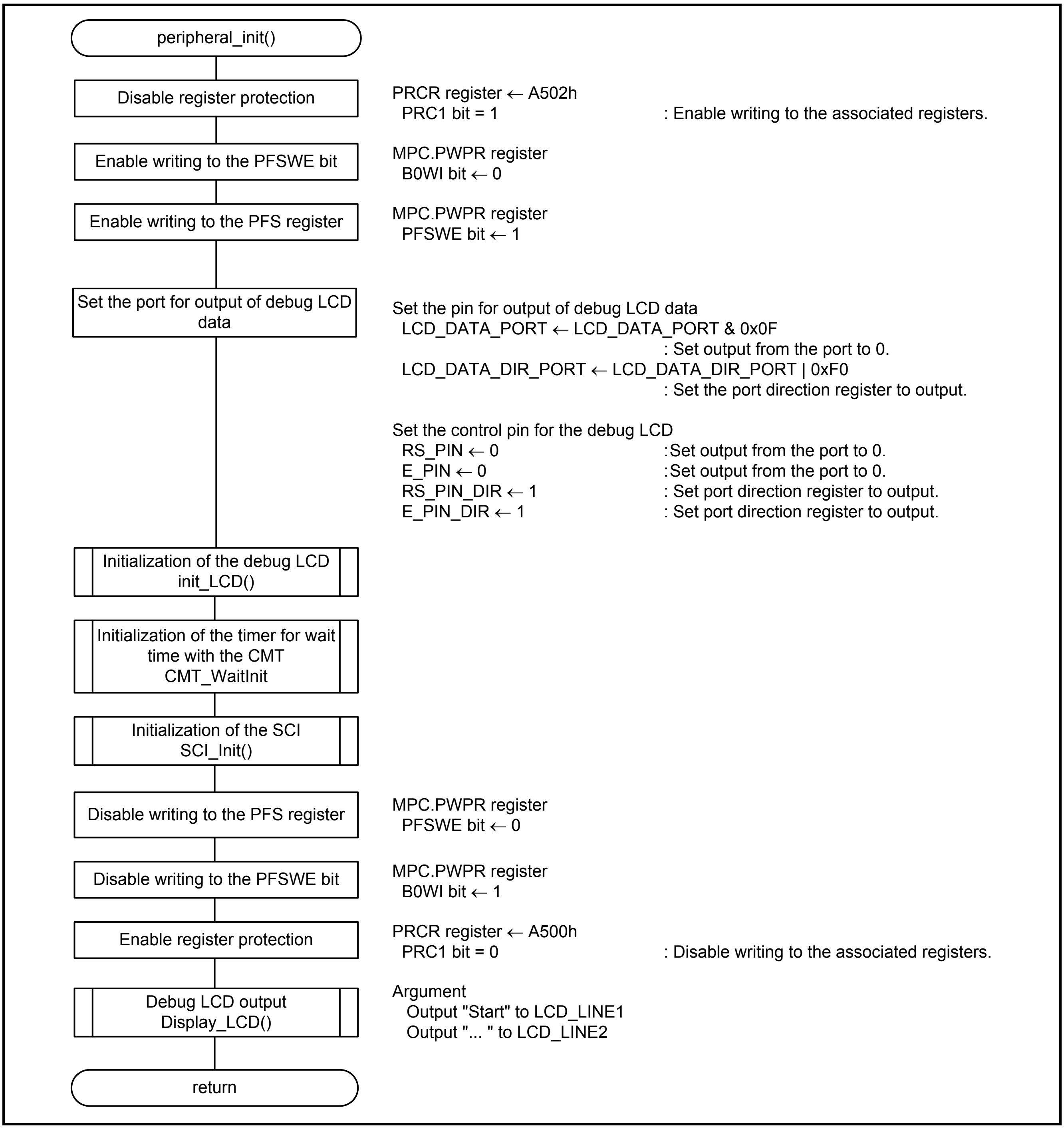


Figure 5.29 Initialization of the Peripheral Functions

5.10.3 Initialization of the Timer for Wait Time With the CMT

Figure 5.30 shows the Initialization of the Timer for Wait Time with the CMT.

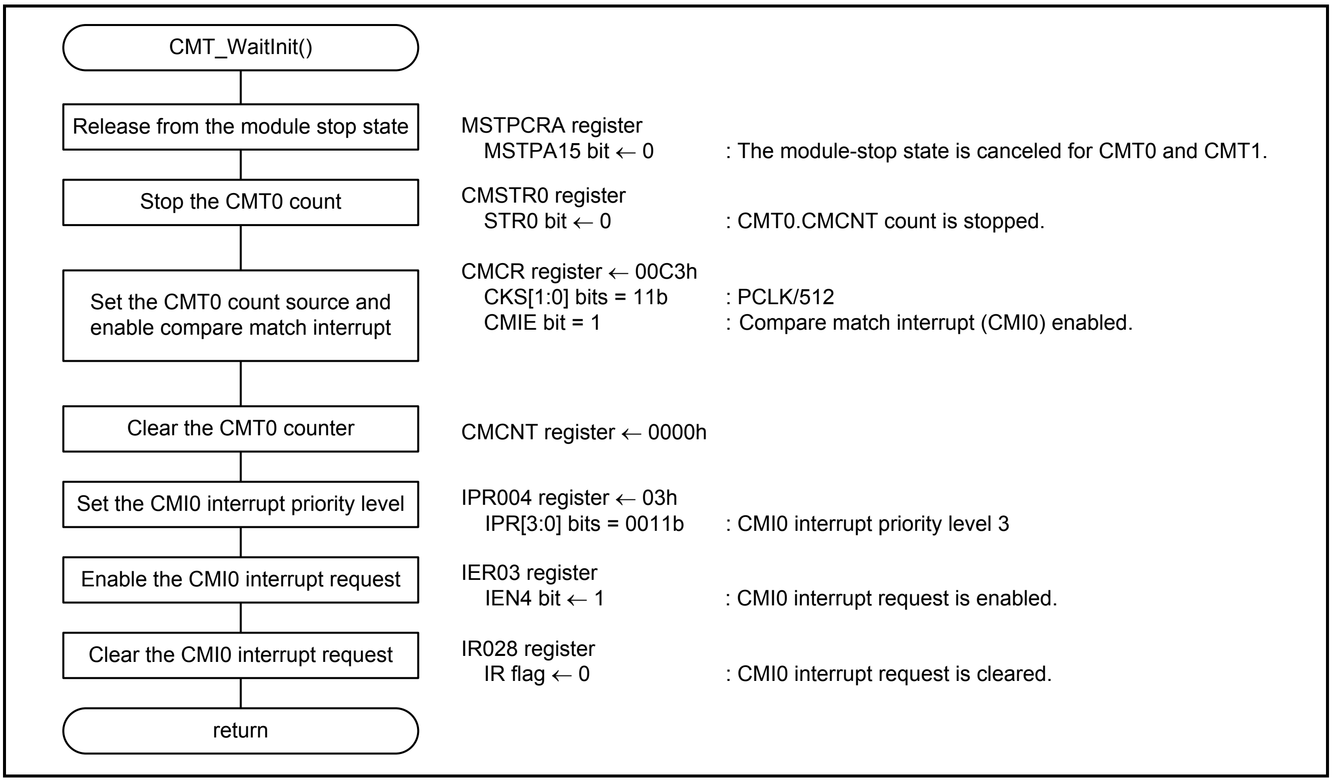


Figure 5.30 Initialization of the Timer for Wait Time with the CMT

5.10.4 Setting Wait Time With the CMT

Figure 5.31 shows Setting Wait Time With the CMT.

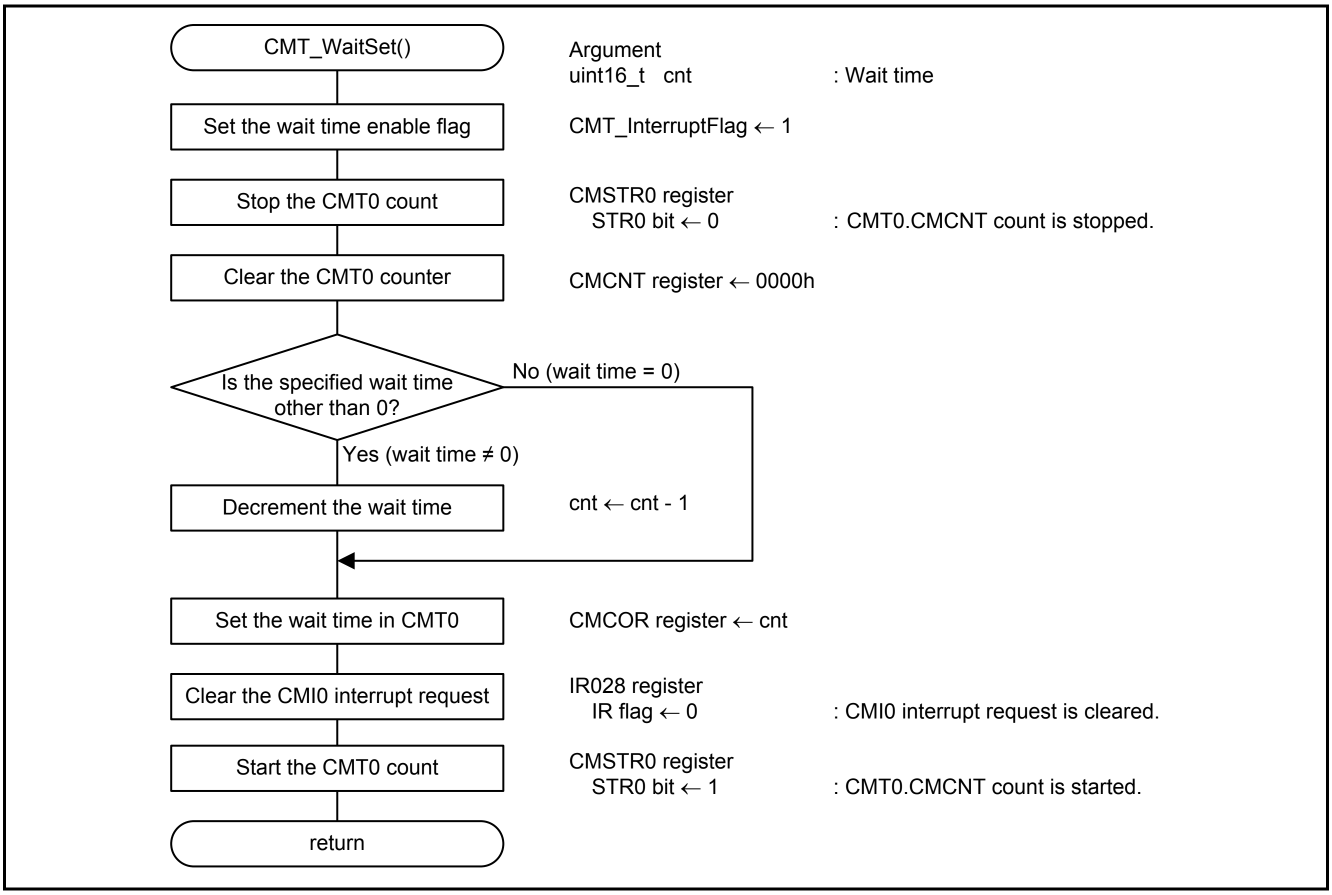


Figure 5.31 Setting Wait Time With the CMT

5.10.5 Wait Processing With the CMT

Figure 5.32 shows Wait Processing With the CMT.

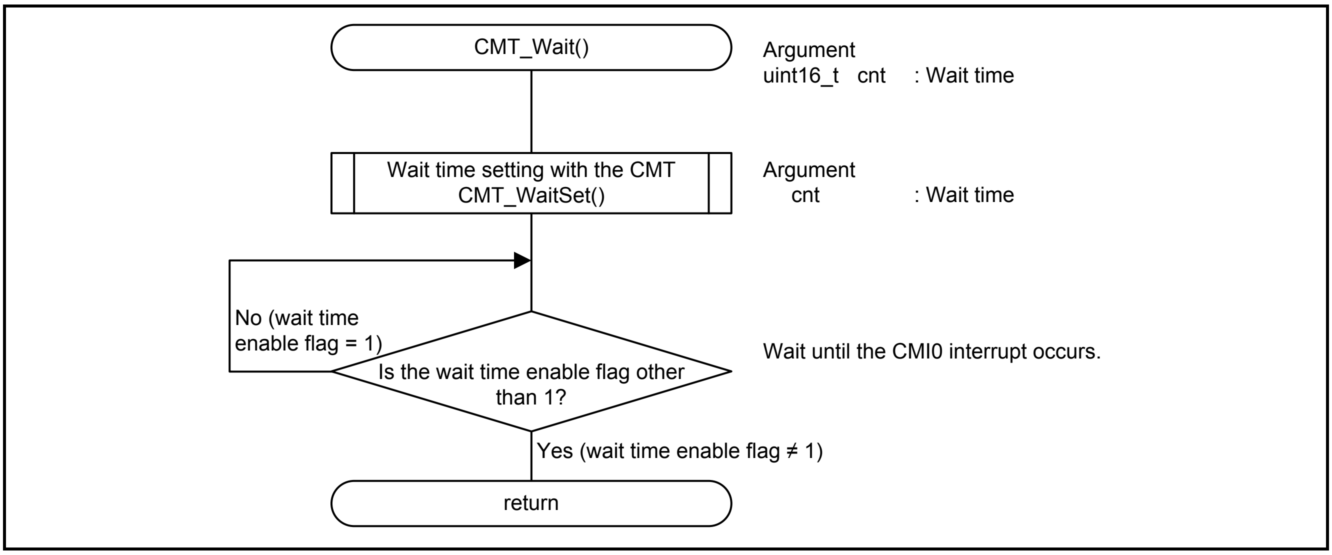


Figure 5.32 Wait Processing With the CMT

5.10.6 Interrupt Handling for CMI0 in CMT0

Figure 5.33 shows Interrupt Handling for CMI0 in CMT0.

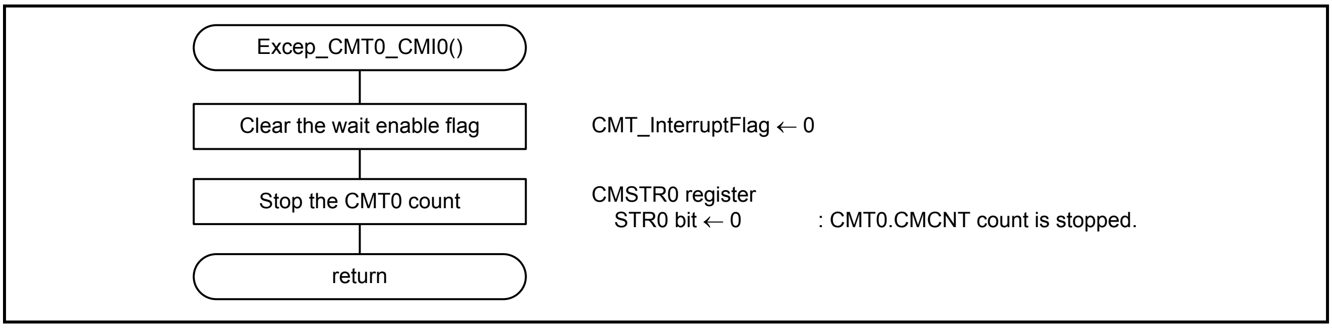


Figure 5.33 Interrupt Handling for CMI0 in CMT0

5.10.7 Initialization of the SCI

Figure 5.34 shows SCI Initialization.

Process	Setting	Description
SCI_Init		
Set port mode	TXDn_PMR ← 0 RXDn_PMR ← 0	:TXDn: Used as a general I/O port. :RXDn: Used as a general I/O port.
Set port output data	TXDn_PODR ← 1	:TXDn: High level
Set the port direction	TXDn_PDR ← 0 RXDn_PDR ← 0	:TXDn: Output :RXDn: Input
Release from SCIn module stop	MSTP_SCIn ← 0	:SCIn is released from the module stopped state.
Disable transmission/reception and the SCIn interrupt request	SCIn.SCR register ← 00h CKE[1:0] bits = 00b TEIE bit = 0 RE bit = 0 TE bit = 0 RIE bit = 0 TIE bit = 0	 : On-chip baud rate generator : TEI interrupt request is disabled. : Serial reception is disabled. : Serial transmission is disabled. : RXI and ERI interrupt requests are disabled. : TXI interrupt request is disabled.
Select the pin function	MPC.TXDnPFS register ← TXDnPFS_SELECT MPC.RXDnPFS register ← RXDnPFS_SELECT	:TXDn :RXDn
Set the format for transmission and reception	SCIn.SMR register ← 00h CKS[1:0] bits = 00b MP bit = 0 STOP bit = 0 PE bit = 0 CHR bit = 0 CM bit = 0 SCIn.SCMR register ← F2h SMIF bit = 0 SINV bit = 0 SDIR bit = 0 SCIn.SEMR register ← 10h ABCS bit = 1 NFEN bit = 0	 :PCLK clock :Multi-processor communications function is disabled. :1 stop bit :No parity :Selects 8 bits as the data length. :Asynchronous mode :Serial communications interface mode :TDR contents are transmitted as they are. Receive data is stored as it is in RDR. :Transfer with LSB first : Selects 8 base clock cycles for 1-bit period. : Noise cancellation function for the RXDn input signal is disabled.
Set the bit rate	SCIn.BRR register ← BRR_SET(19200)	: (48 MHz/(32 × 2^{-1} × 19200 bps)) - 1
Clear the SCIn interrupt request	IR_SCIn_RXIn ← 0 IR_SCIn_TXIn ← 0	:SCIn.RXIn interrupt not requested. :SCIn.TXIn interrupt not requested.
Enable transmission/reception and SCIn interrupt request	SCIn.SCR register ← F0h RE bit = 1 TE bit = 1 RIE bit = 1 TIE bit = 1	 : Serial reception is enabled. : Serial transmission is enabled. : RXI and ERI interrupt requests are enabled. : A TXI interrupt request is enabled.
Set port mode	TXDn_PMR ← 1 RXDn_PMR ← 1	:TXDn: Used for peripheral functions. :RXDn: Used for peripheral functions.
return		

Figure 5.34 SCI Initialization

5.10.8 Processing to Change the SCI Bit Rate

Figure 5.35 shows Processing to Change the SCI Bit Rate.

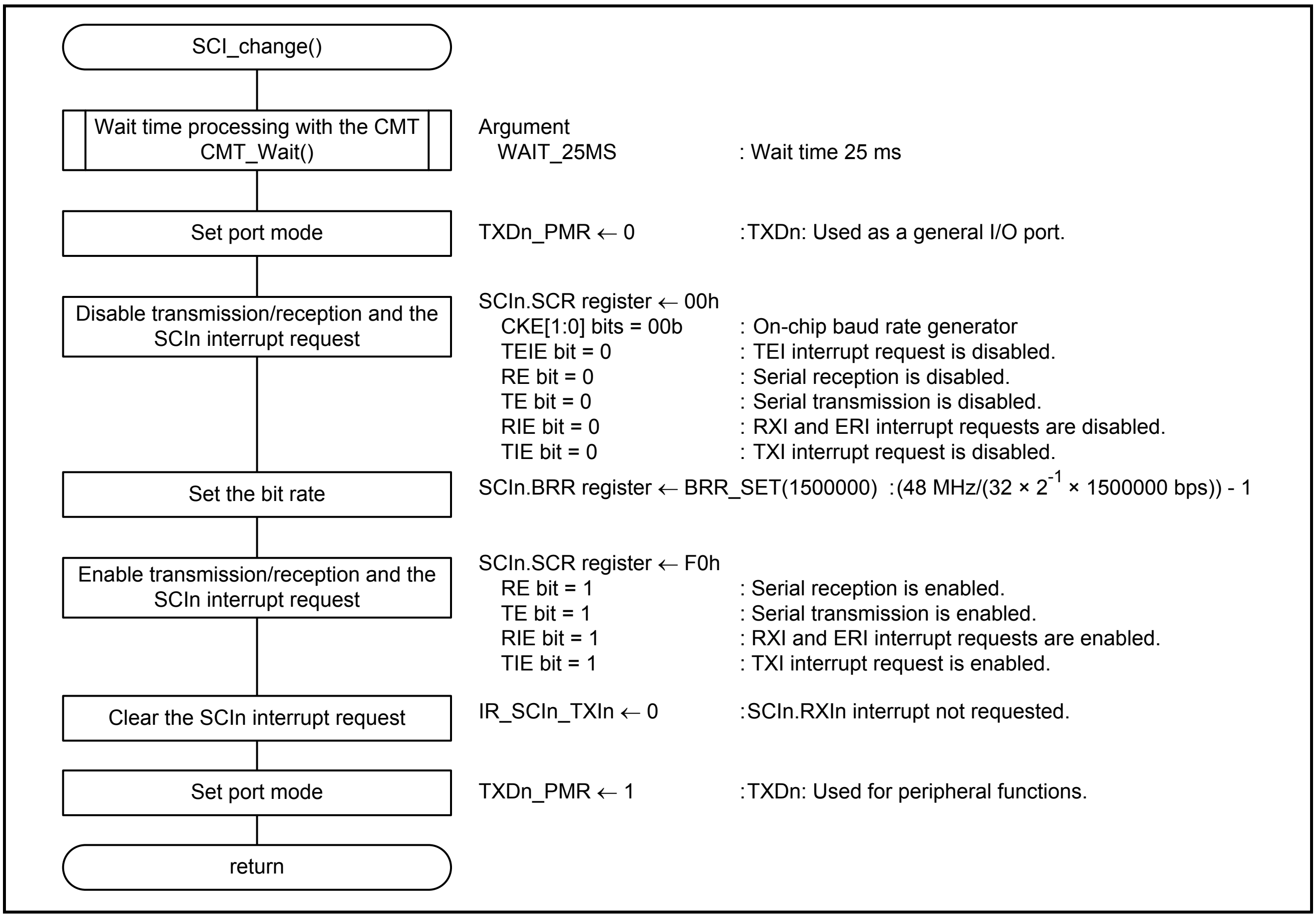


Figure 5.35 Processing to Change the SCI Bit Rate

5.10.9 Processing to Calculate the SUM Data

Figure 5.36 shows Processing to Calculate the SUM Data.

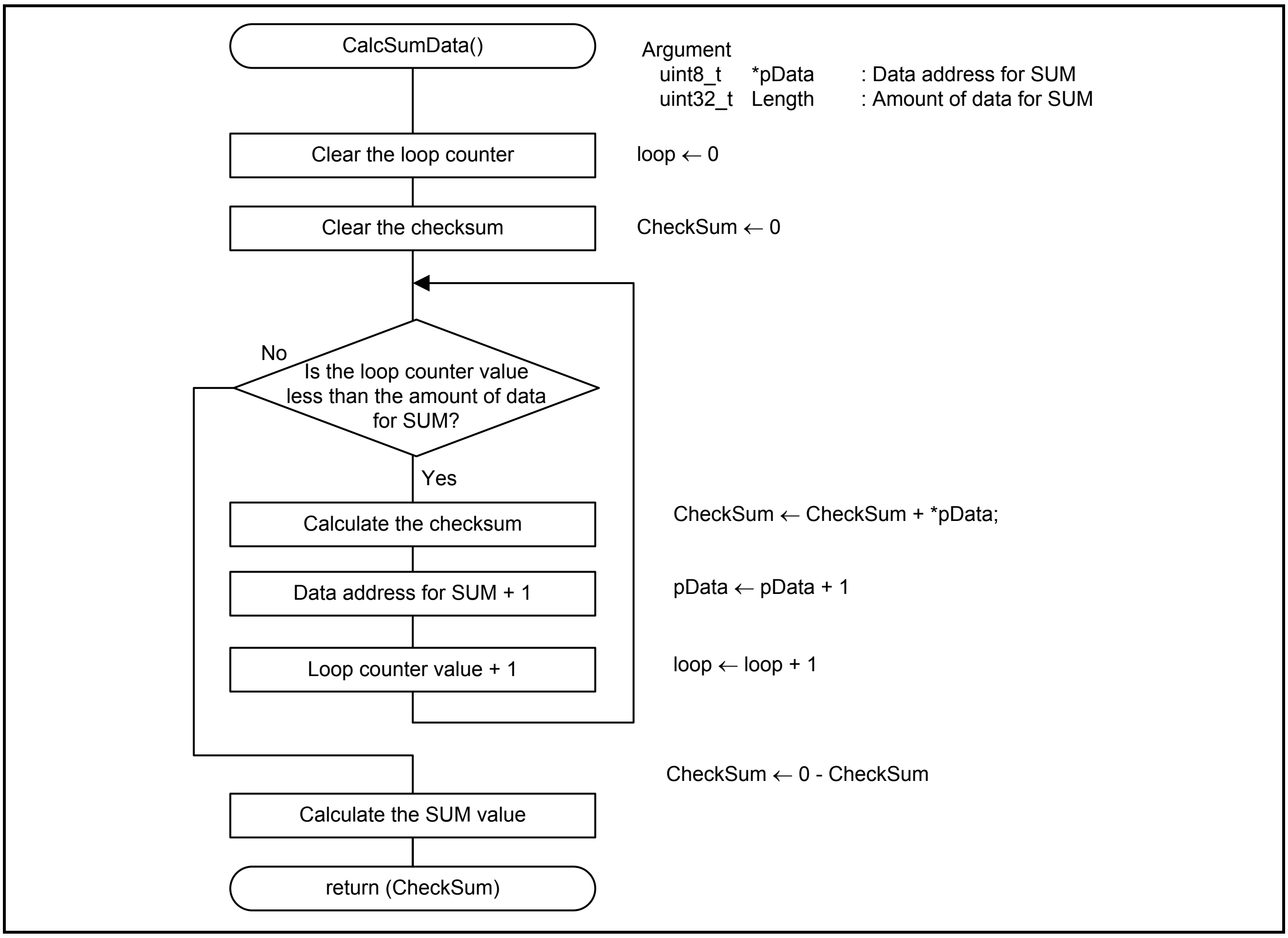


Figure 5.36 Processing to Calculate the SUM Data

5.10.10 Processing to Start the Target MCU in Boot Mode

Figure 5.37 shows Processing to Start the Target MCU in Boot Mode.

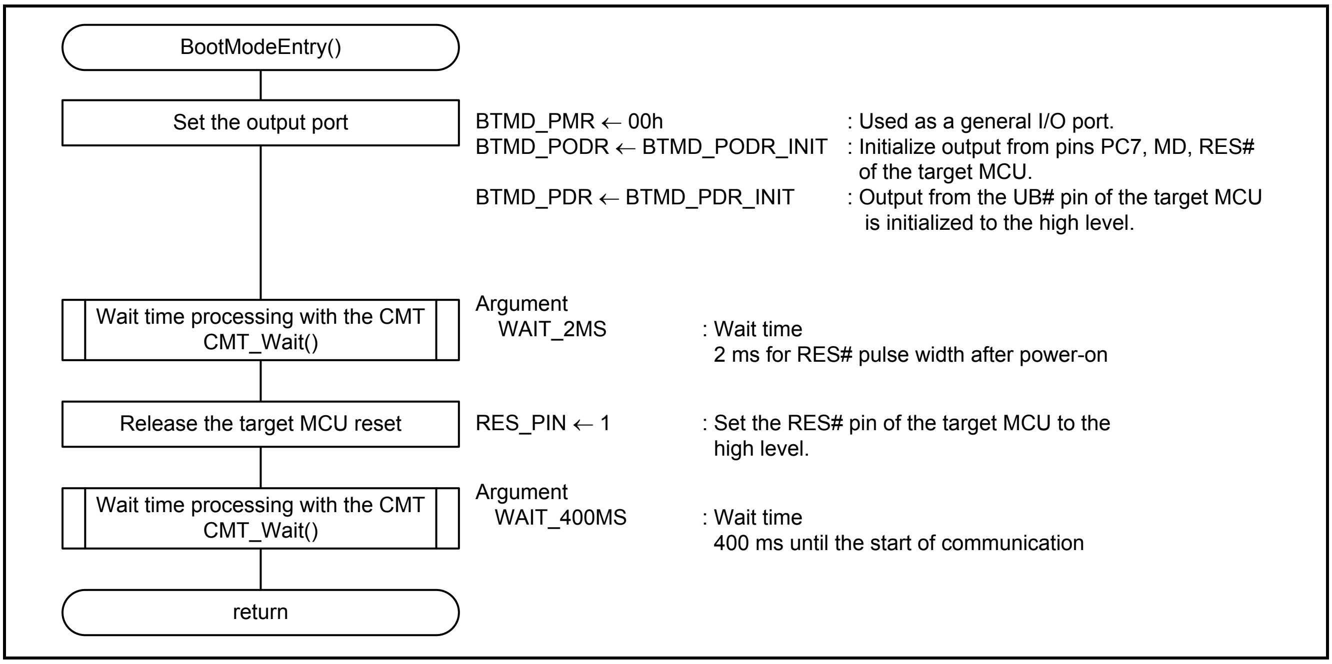


Figure 5.37 Processing to Start the Target MCU in Boot Mode

5.10.11 Processing to Reset the Target MCU

Figure 5.38 shows Processing to Reset the Target MCU.

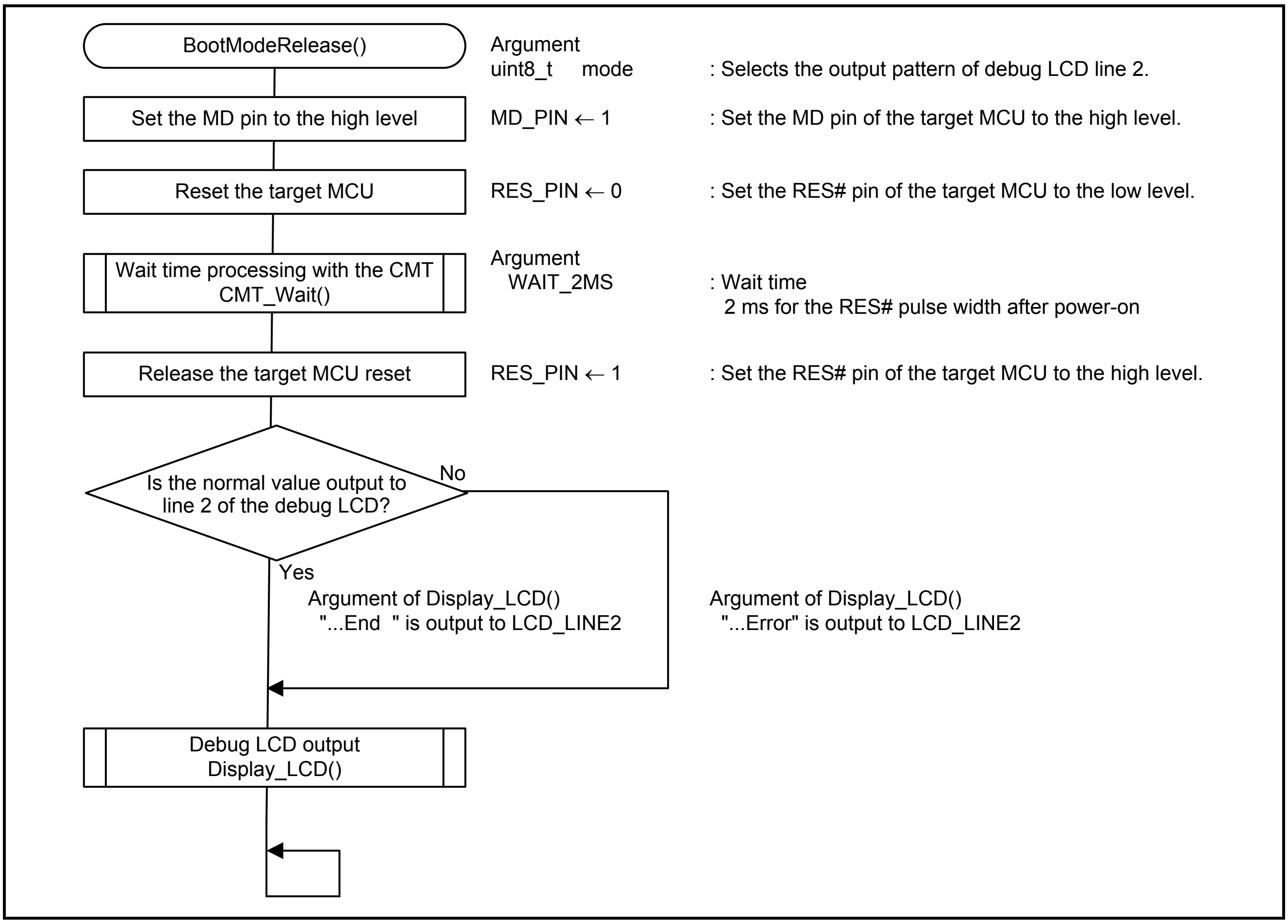


Figure 5.38 Processing to Reset the Target MCU

5.10.12 Processing to Send a Command

Figure 5.39 shows Processing to Send a Command.

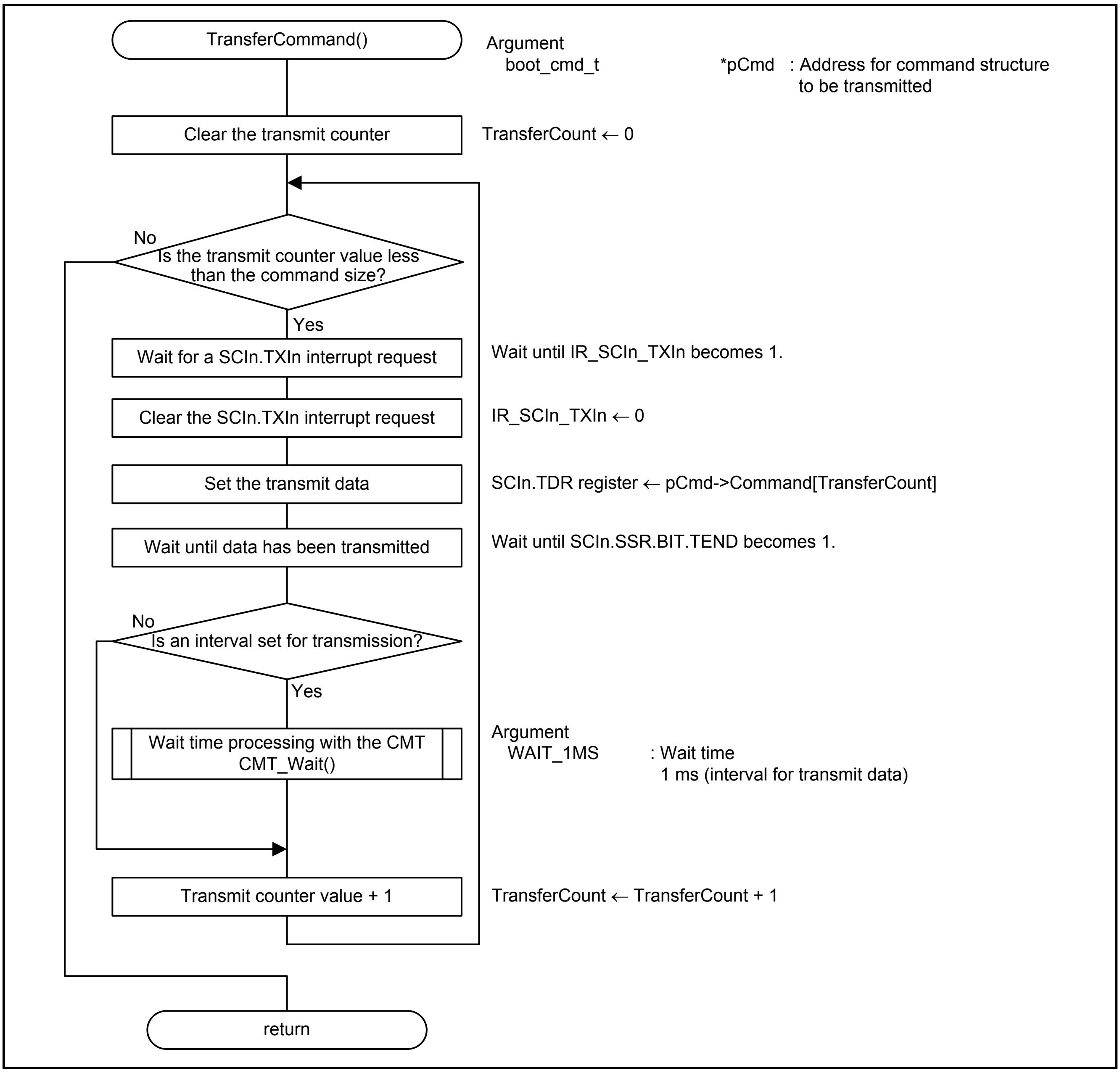


Figure 5.39 Processing to Send a Command

5.10.13 Processing to Receive a Response

Figure 5.40 to Figure 5.43 show Processing to Receive a Response.

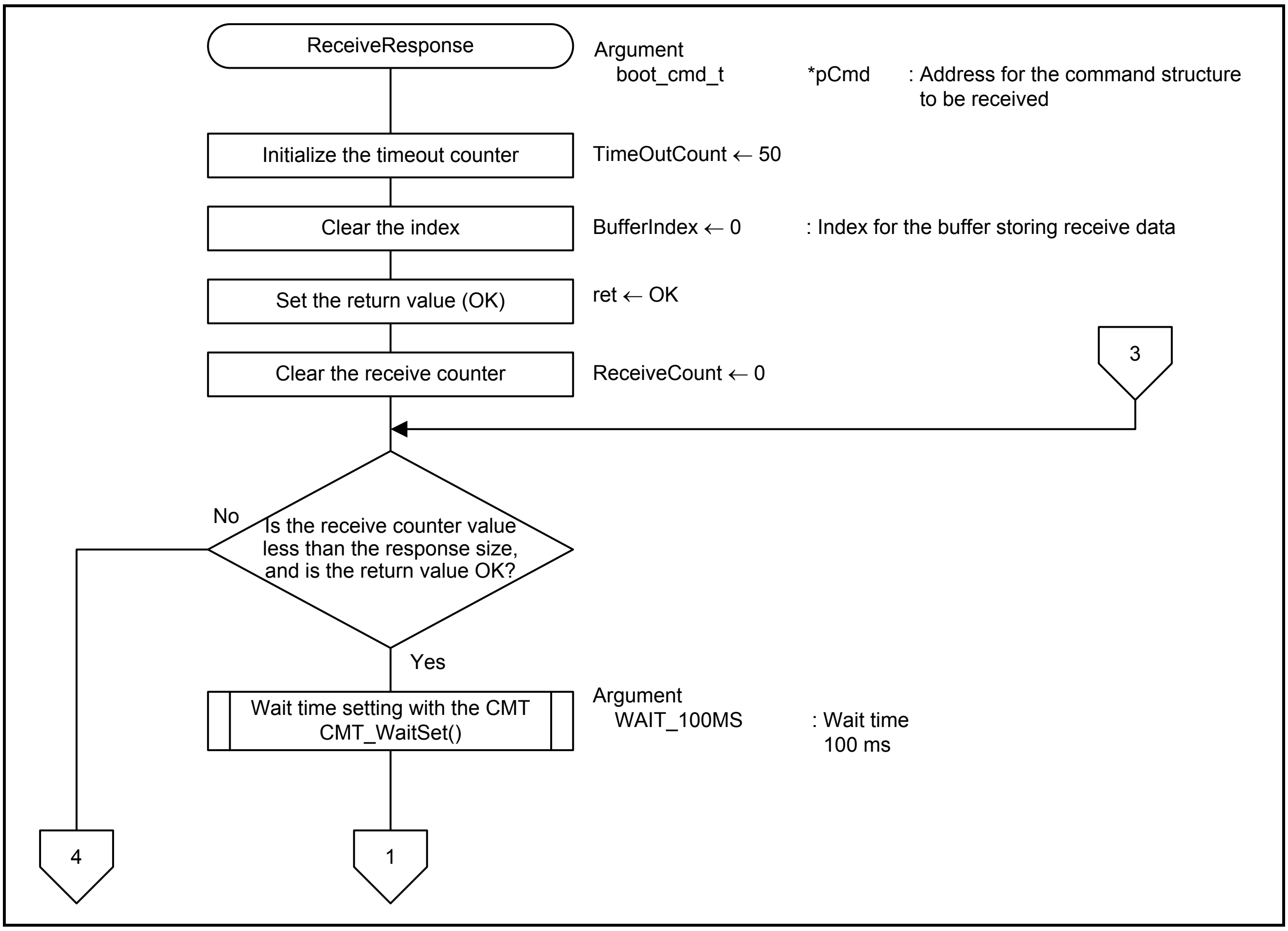
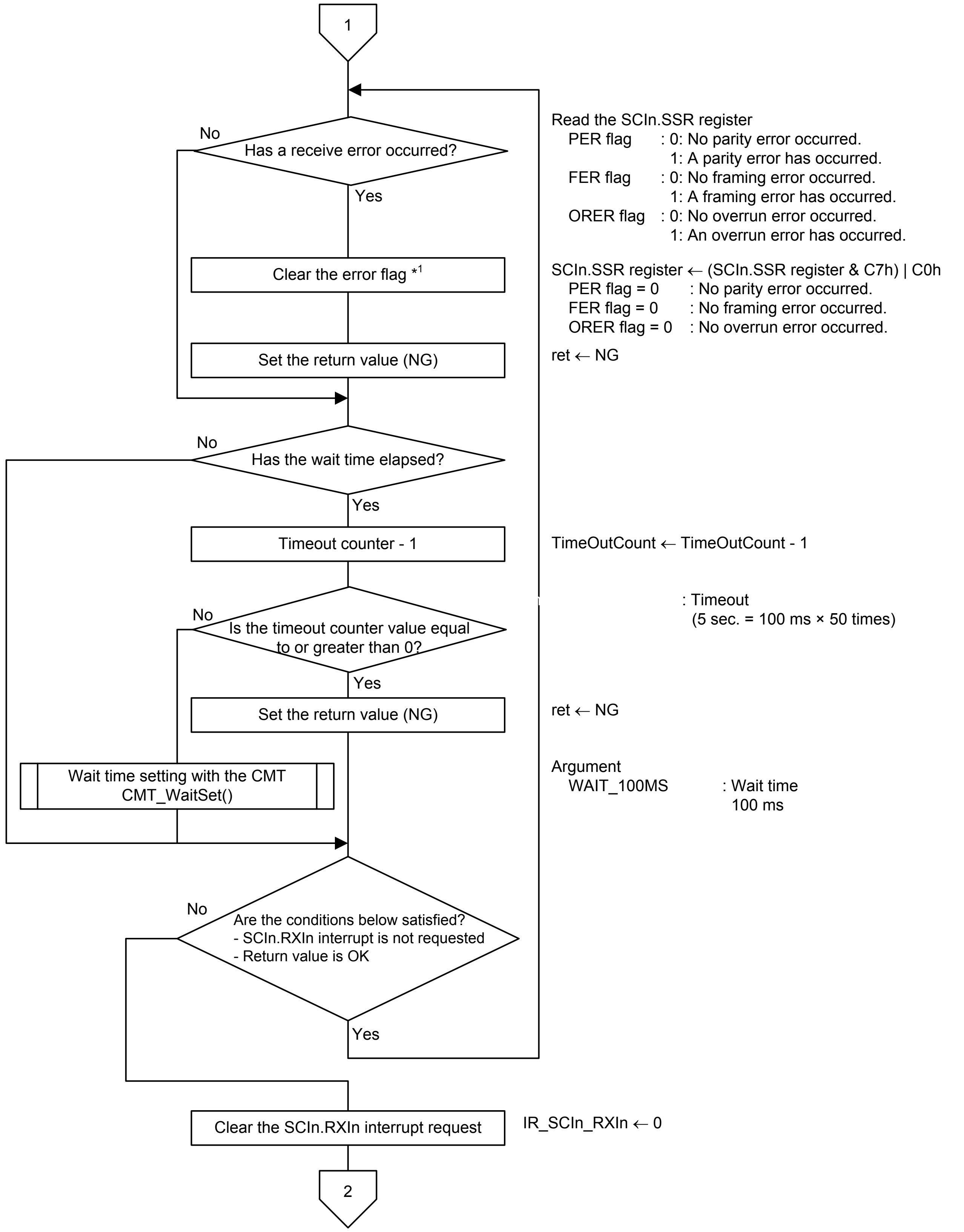


Figure 5.40 Processing to Receive a Response



Note 1. After writing to bits RE and RIE and flags PER and FER and ORER, confirm that the written values can be read.

Figure 5.41 Processing to Receive a Response

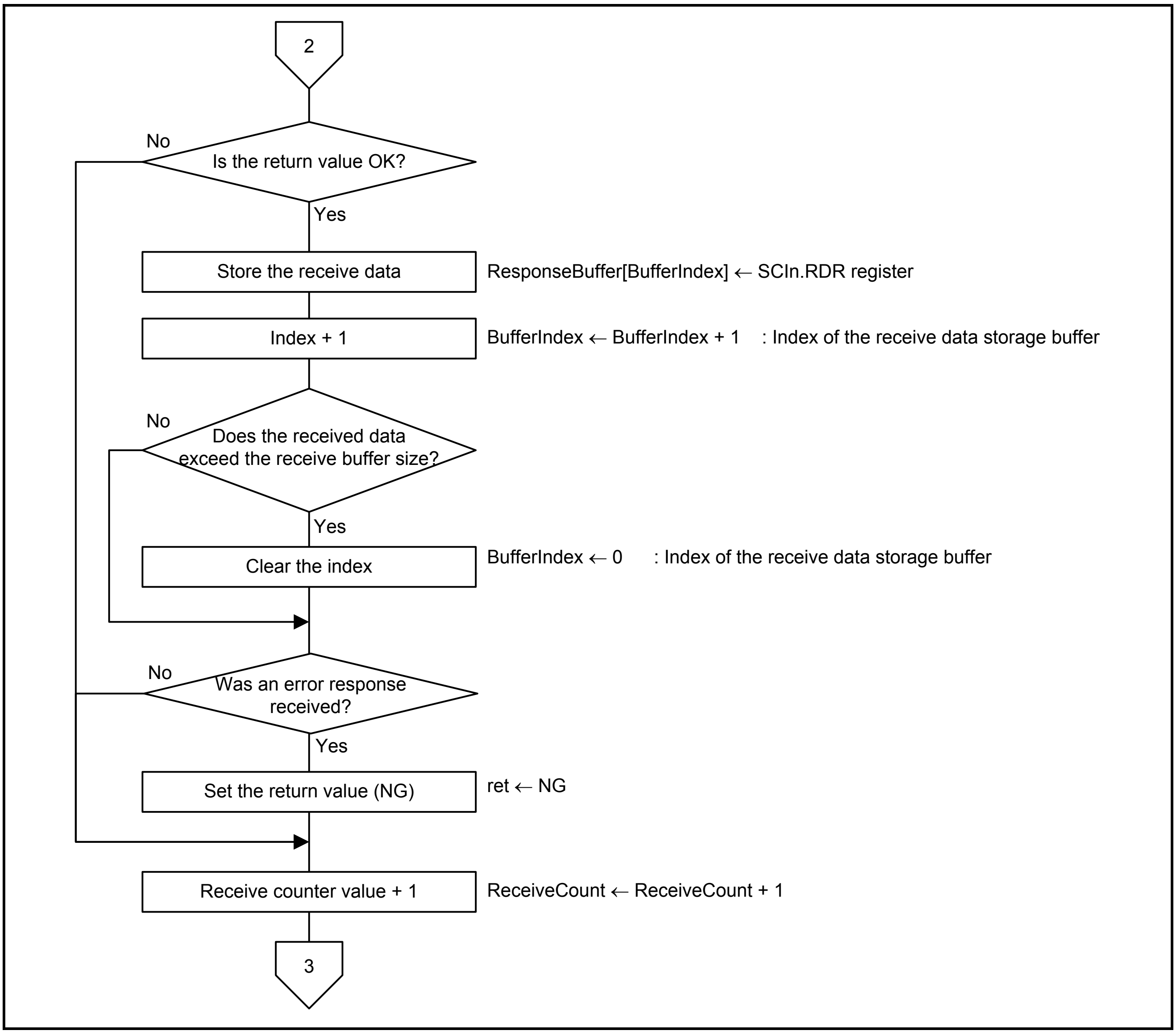
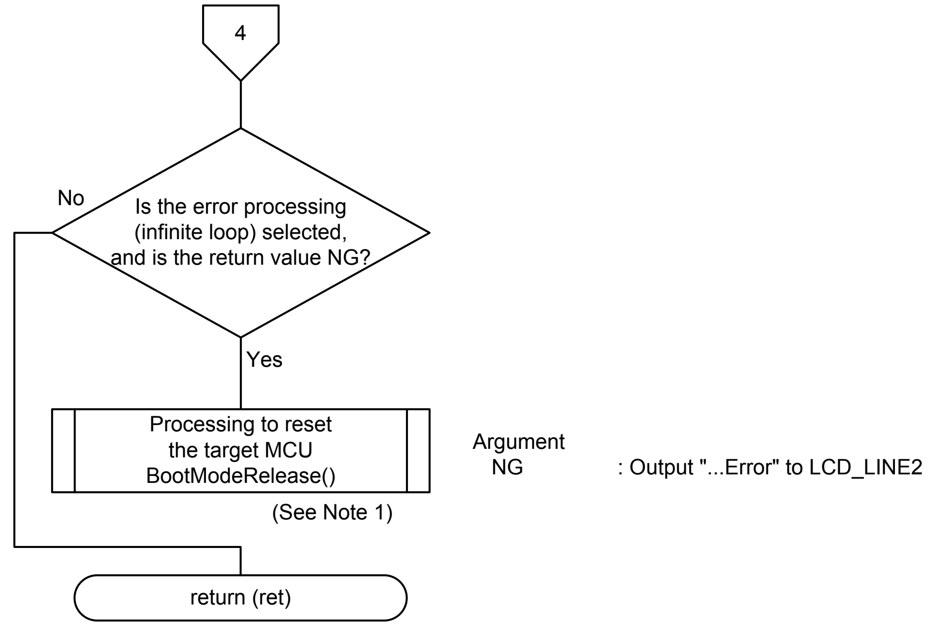


Figure 5.42 Processing to Receive a Response



Note 1. The flash programmer goes into an infinite loop by the BootModeRelease() function and does not return to the caller.

Figure 5.43 Processing to Receive a Response

5.10.14 Copying Unsigned 4-Byte Data

Figure 5.44 show Copying Unsigned 4-Byte Data.

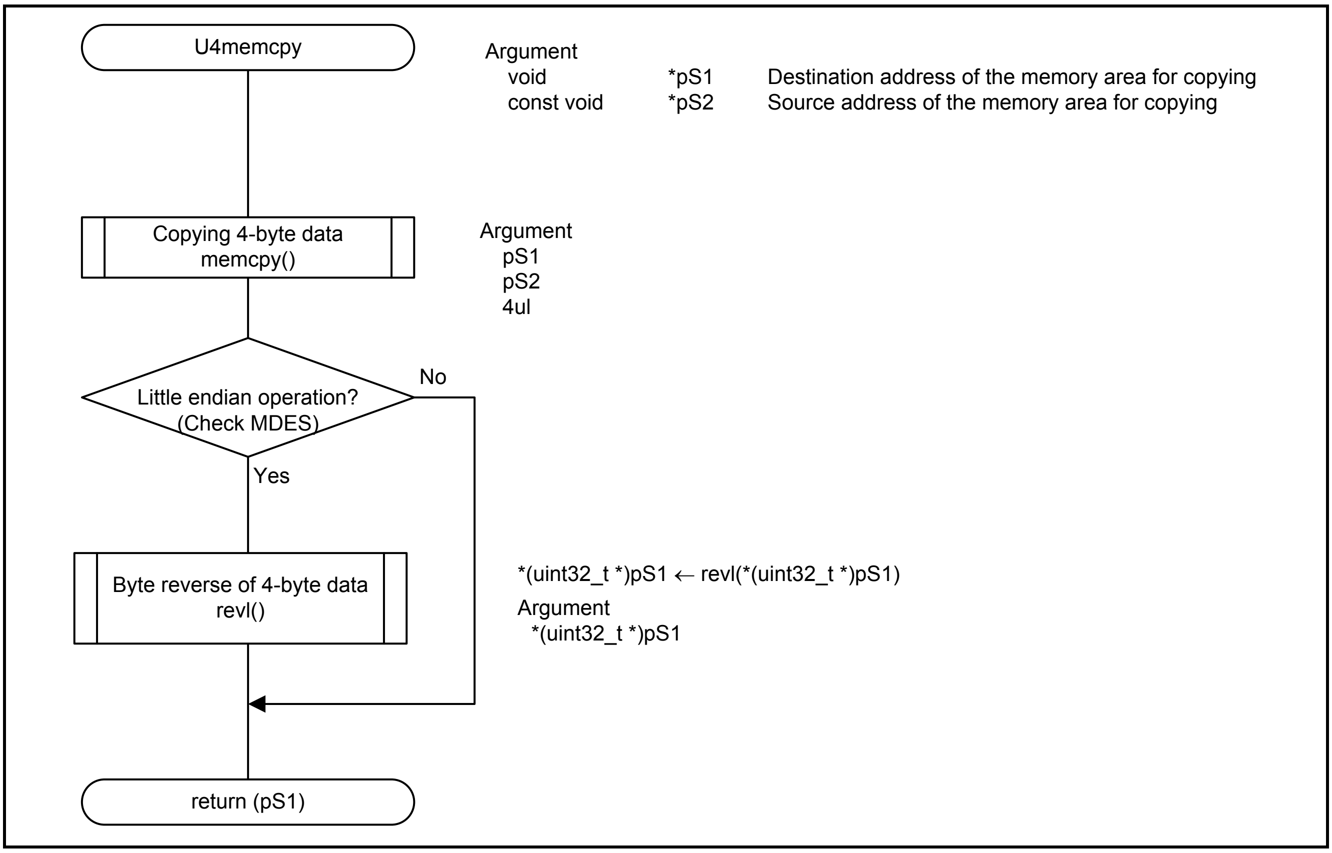


Figure 5.44 Copying Unsigned 4-Byte Data

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware

RX63N Group, RX631 Group User's Manual: Hardware Rev.1.80 (R01UH0041EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website

http://www.renesas.com

Inquiries

http://www.renesas.com/contact/

REVISION HISTORY	RX63N Group Application Note RX63N Group Flash Programmer (Boot Mode) Using the Renesas Starter Kit+ for RX63N

Rev.	**Date**	**Description**	
		Page	**Summary**
1.00	Sep. 10, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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 an MPU or MCU 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.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0