APPLICATION NOTE

RX100 Series

R01AN1938EJ0110
Rev. 1.10
Aug. 20. 2020

RX100 Series Flash Programmer (SCI) Using the Renesas Starter Kit+ for RX63N

Abstract

This document describes a flash programmer for RX100 Series using the Renesas Starter Kit+ for RX63N (hereinafter referred to as RSK+RX63N).

The target for rewriting is the RX100 Series. Boot mode (SCI) is used for rewriting the user area in the RX100 Series.

Products

RX100 Series

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 RX100 Series MCU in boot mode (SCI), the flash programmer rewrites the user area in the RX100 Series 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 operating frequency select command)
1 Mbps (after the program/erase status 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

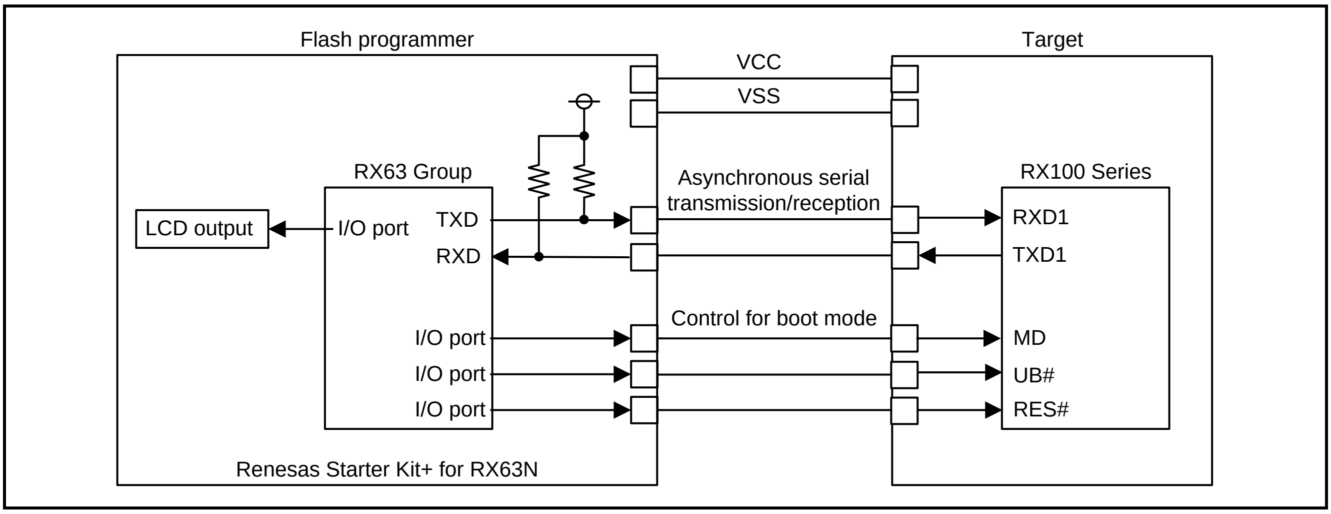
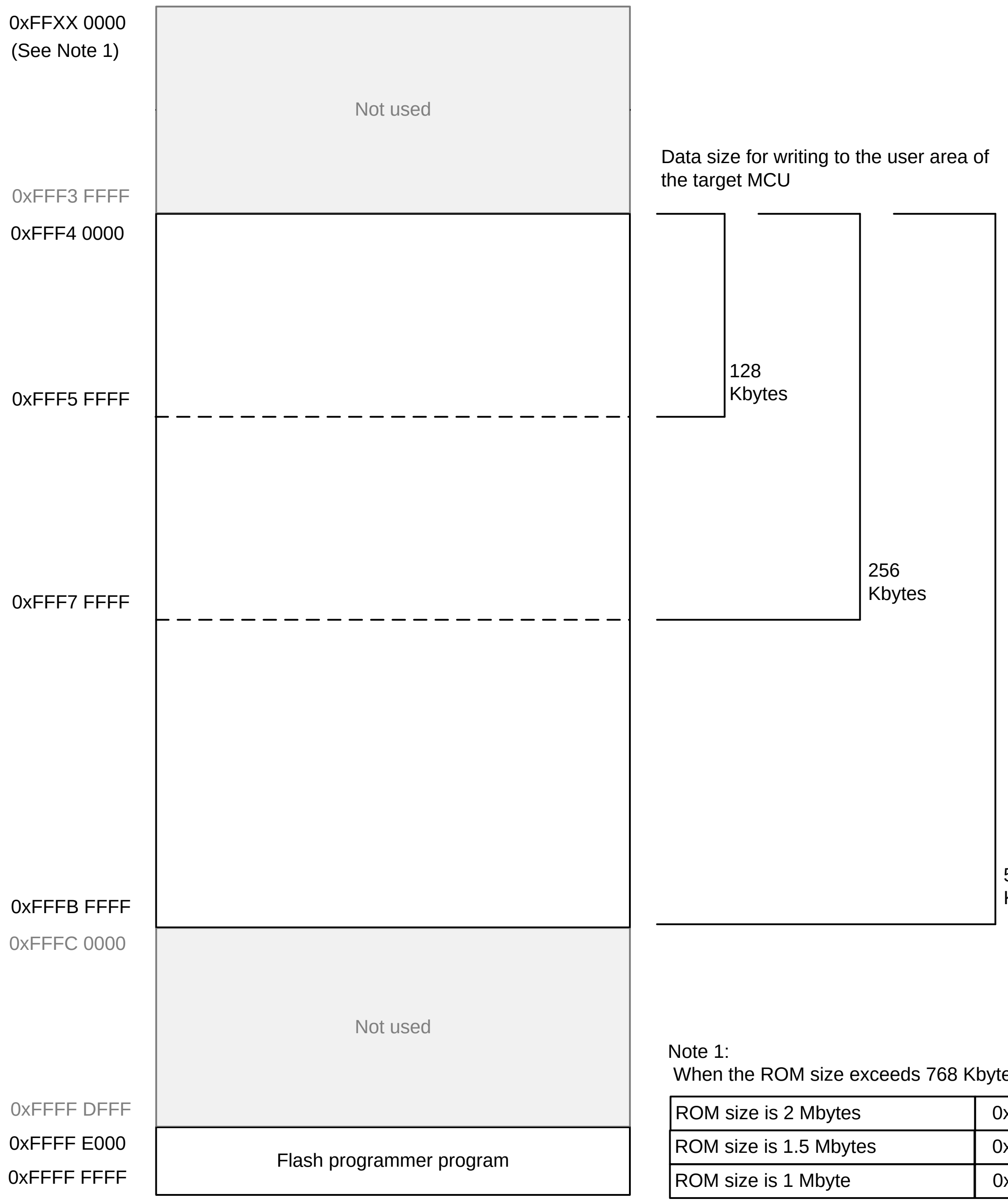


Figure 1.1 Flash Programmer Usage Example

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:
When the ROM size exceeds 768 Kbytes

ROM size is 2 Mbytes	0xFFE0 0000
ROM size is 1.5 Mbytes	0xFFE8 0000
ROM size is 1 Mbyte	0xFFF0 0000

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 e^2 2020-04
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.02.00
	Compile option (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 1.10
Board used	Renesas Starter Kit+ for RX63N (part number: R0K50563NC000BE)

Notes:

If the same version of the toolchain (C compiler) specified in the original project is not in the import destination, the toolchain will not be selected and an error will occur.

Check the selected status of the toolchain on the project configuration dialog.

For the setting method, refer to FAQ 3000404.

FAQ 3000404 :Program ""make"" not found in PATH’ error when attempting to build an imported project (e^2 studio)"

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.

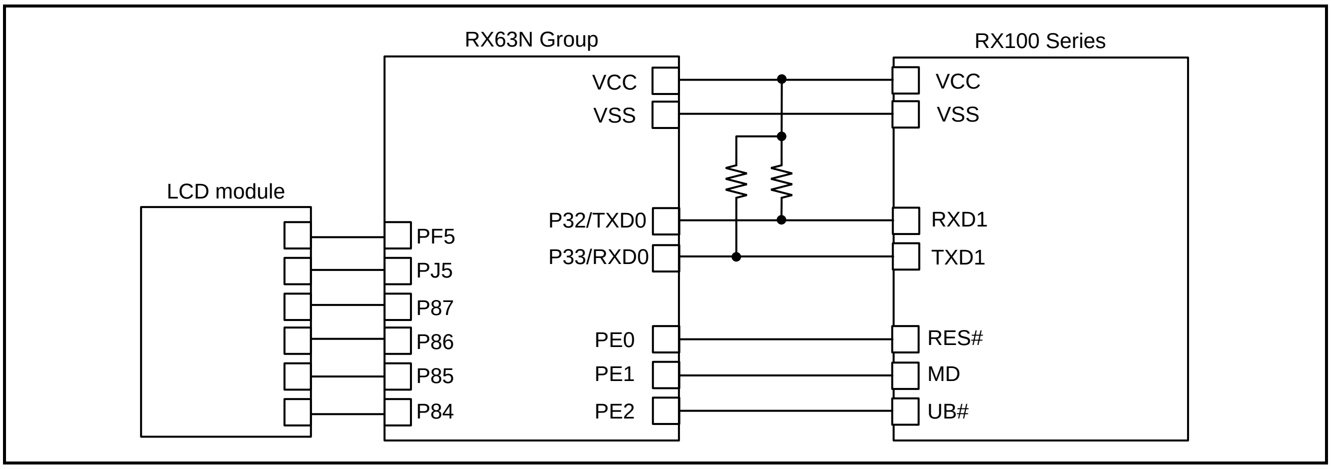


Figure 4.1 Connection Example

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	UB# 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 (R20UT0508EJ1200) for details on using the FDT.

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

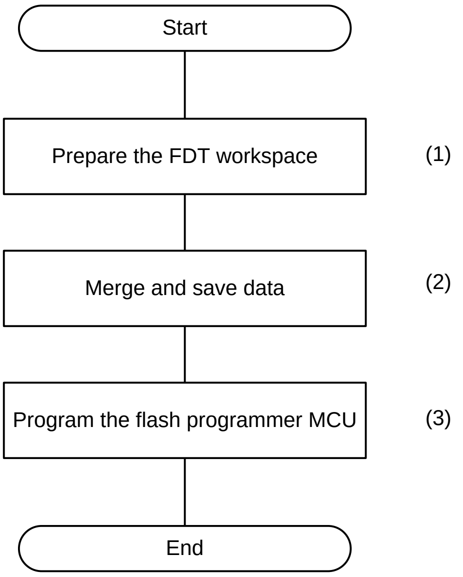


Figure 5.1 Flow of Programming the RSK+RX63N

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

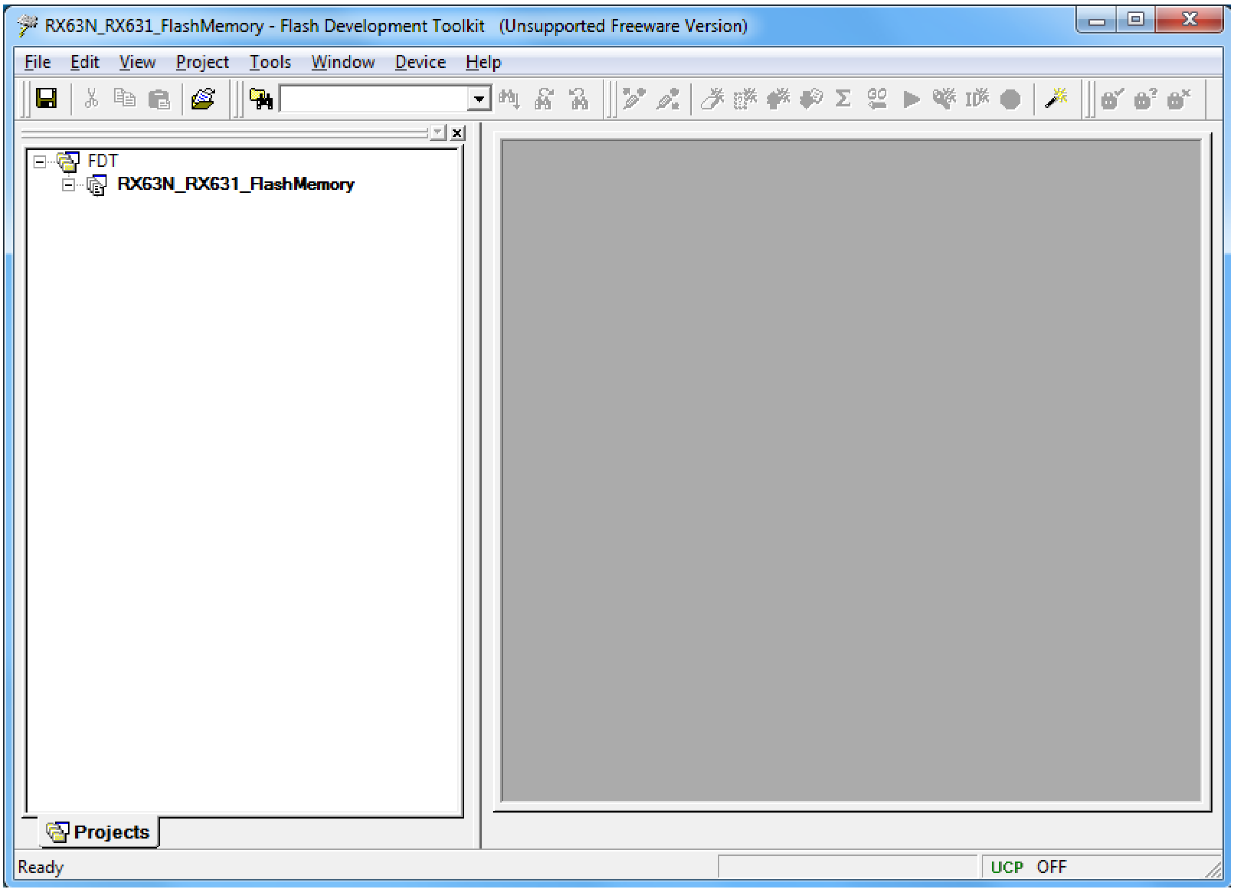
(2) Refer to 5.1.2 Merge and Save Data for details.

(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.

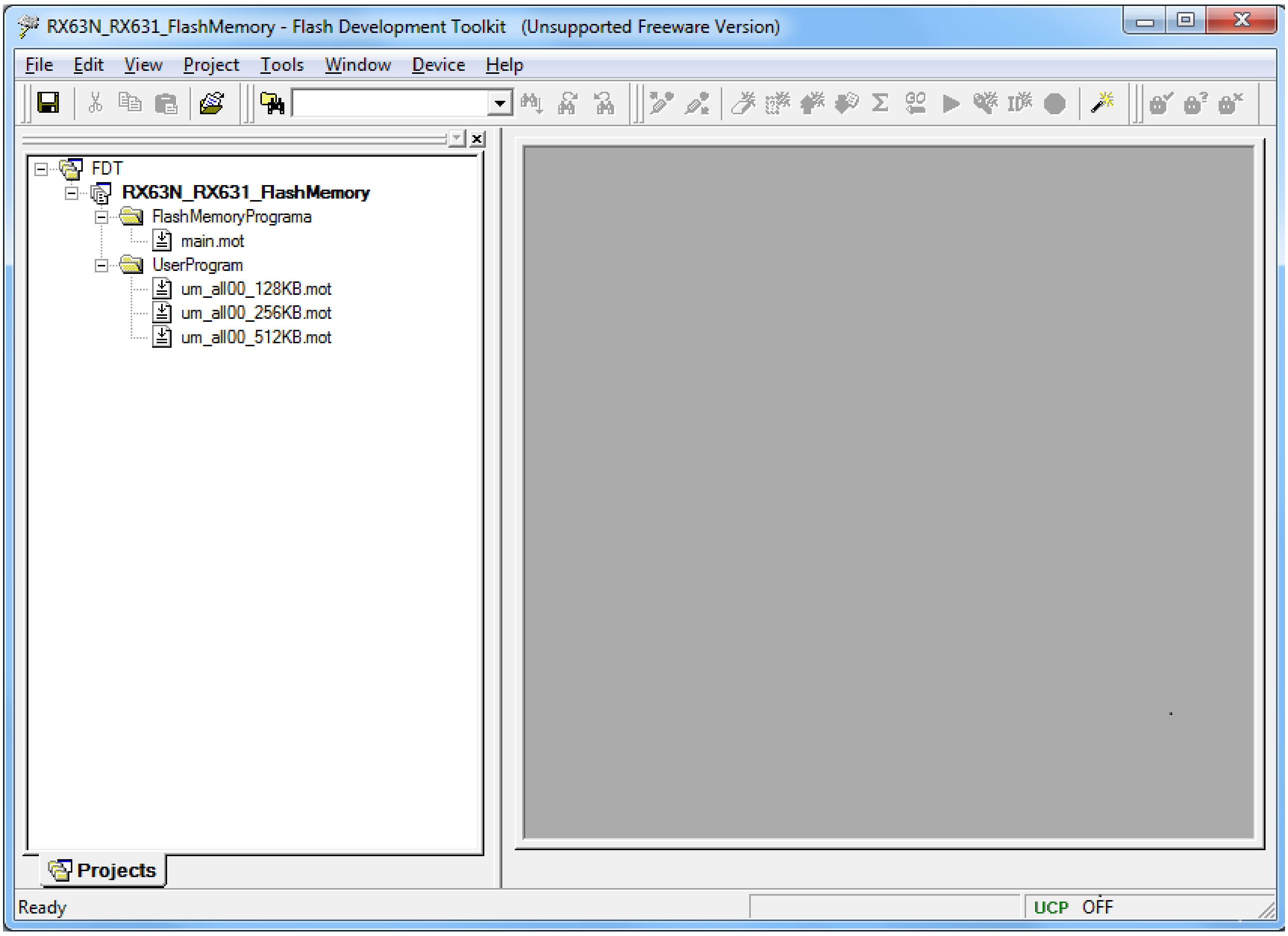
The main.mot file of the flash programmer's program is added to the FlashMemoryPrograma folder.

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_128KB.mot file when the user program size is 128 Kbytes

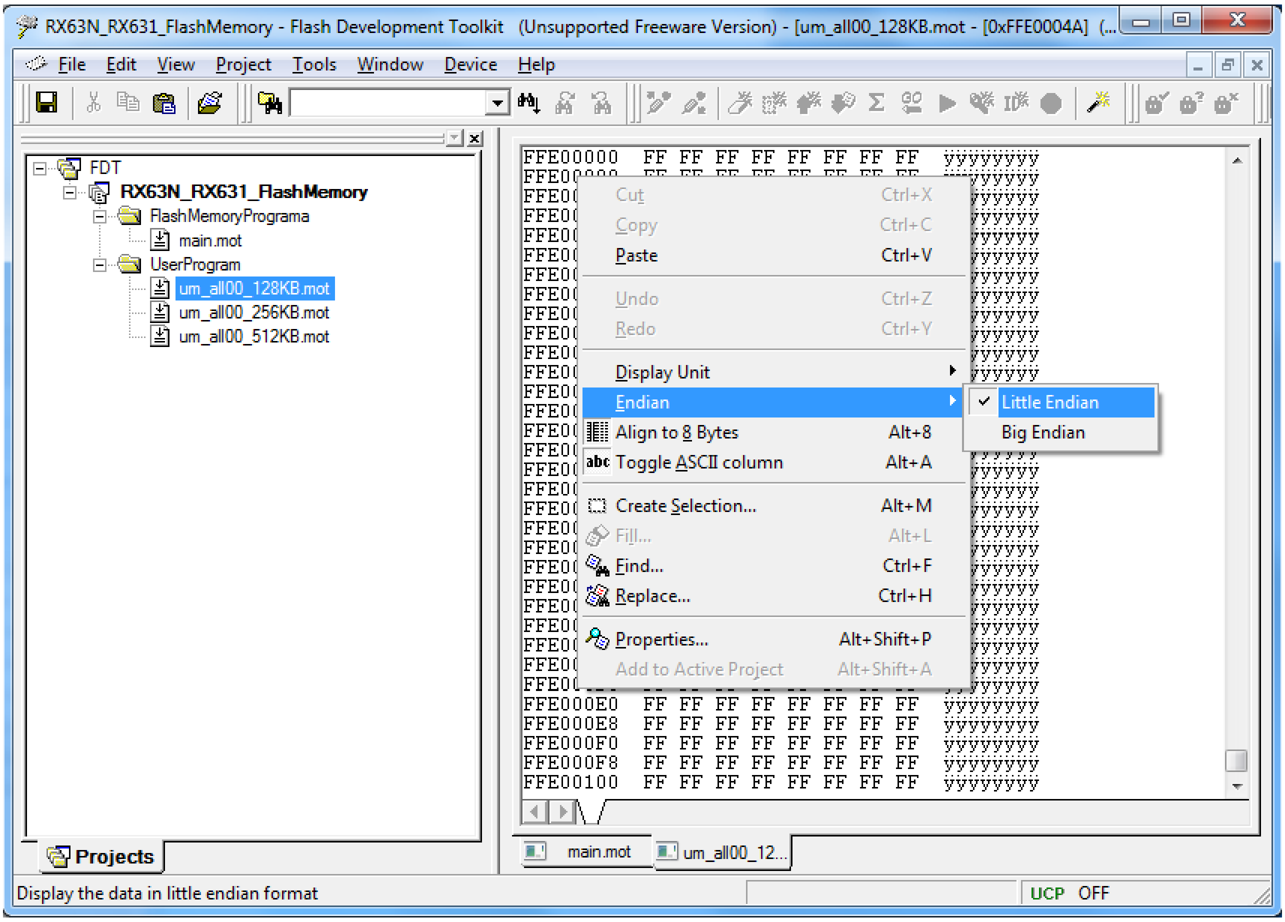
um_all00_256KB.mot file when the user program size is 256 Kbytes

um_all00_512KB.mot file when the user program size is 512 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_128KB.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

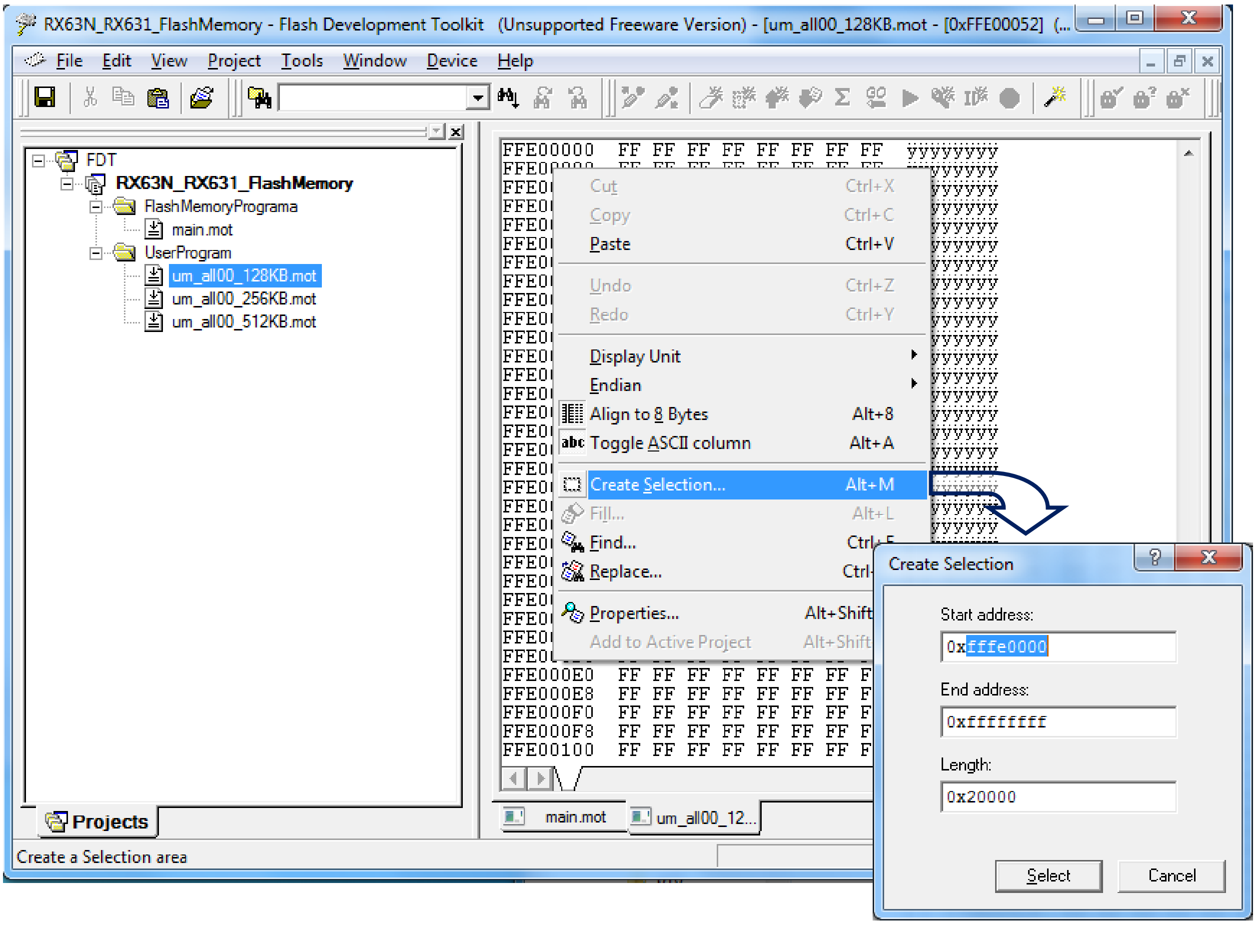
Select the range as follows:

Addresses 0xFFFE 0000 to 0xFFFF FFFF when the user program size is 128 Kbytes

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

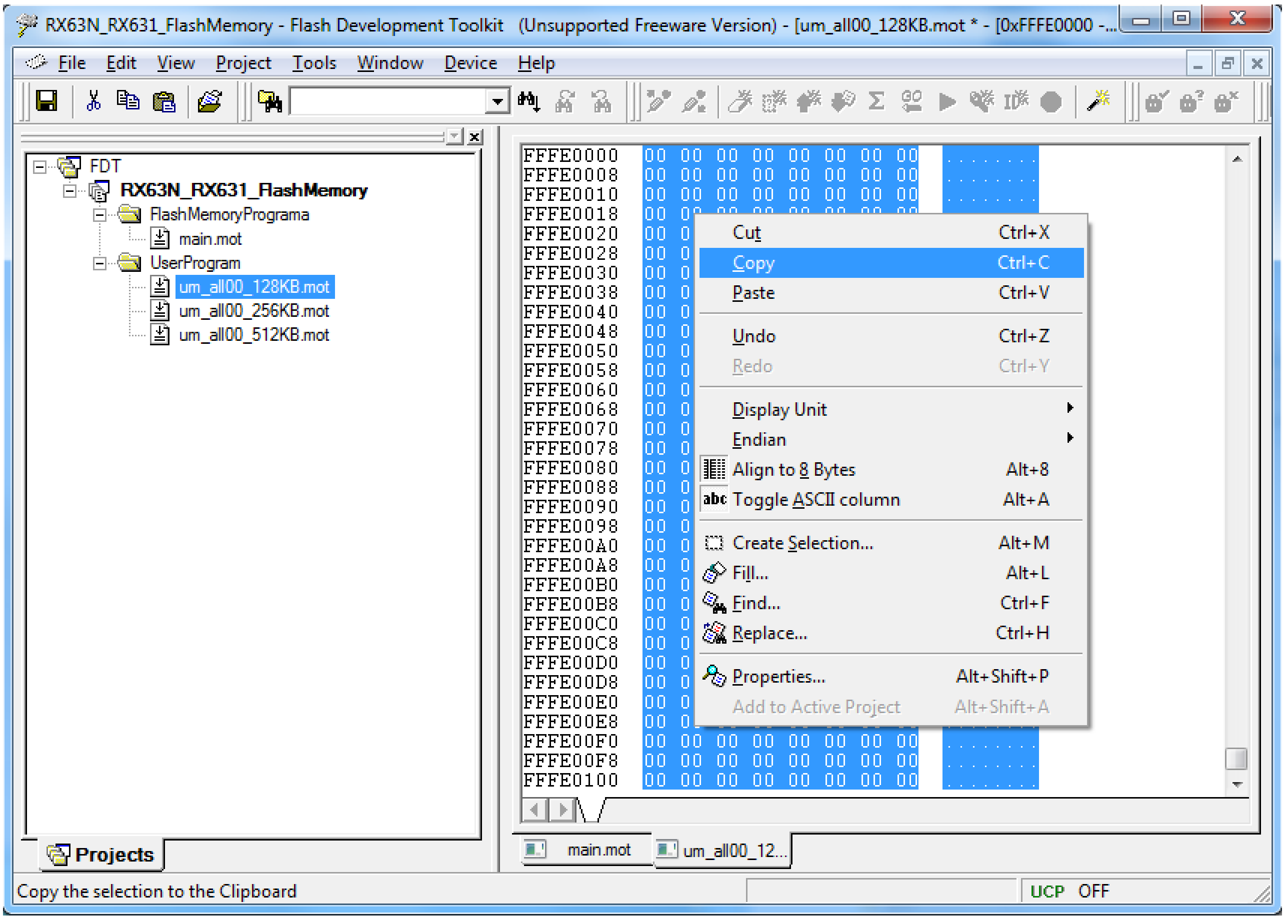
Addresses 0xFFF8 0000 to 0xFFFF FFFF when the user program size is 512 Kbytes

In the example, addresses 0xFFFE 0000 to 0xFFFF FFFF of the um_all00_128KB.mot file are selected.



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

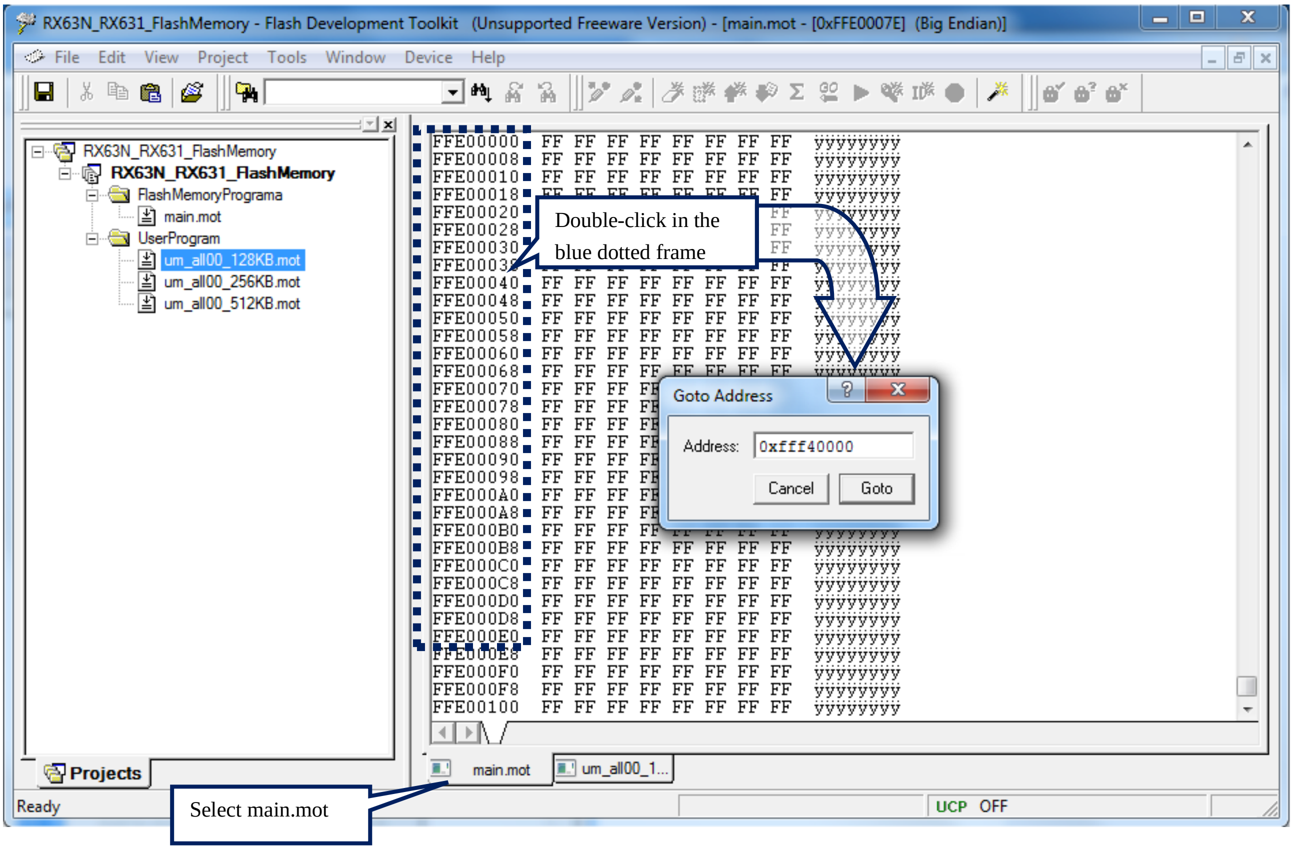
In the example, addresses 0xFFFE 0000 to 0xFFFF FFFF of the um_all00_128KB.mot file are copied to the Windows clipboard.



(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 0xFFF4 0000 and higher.

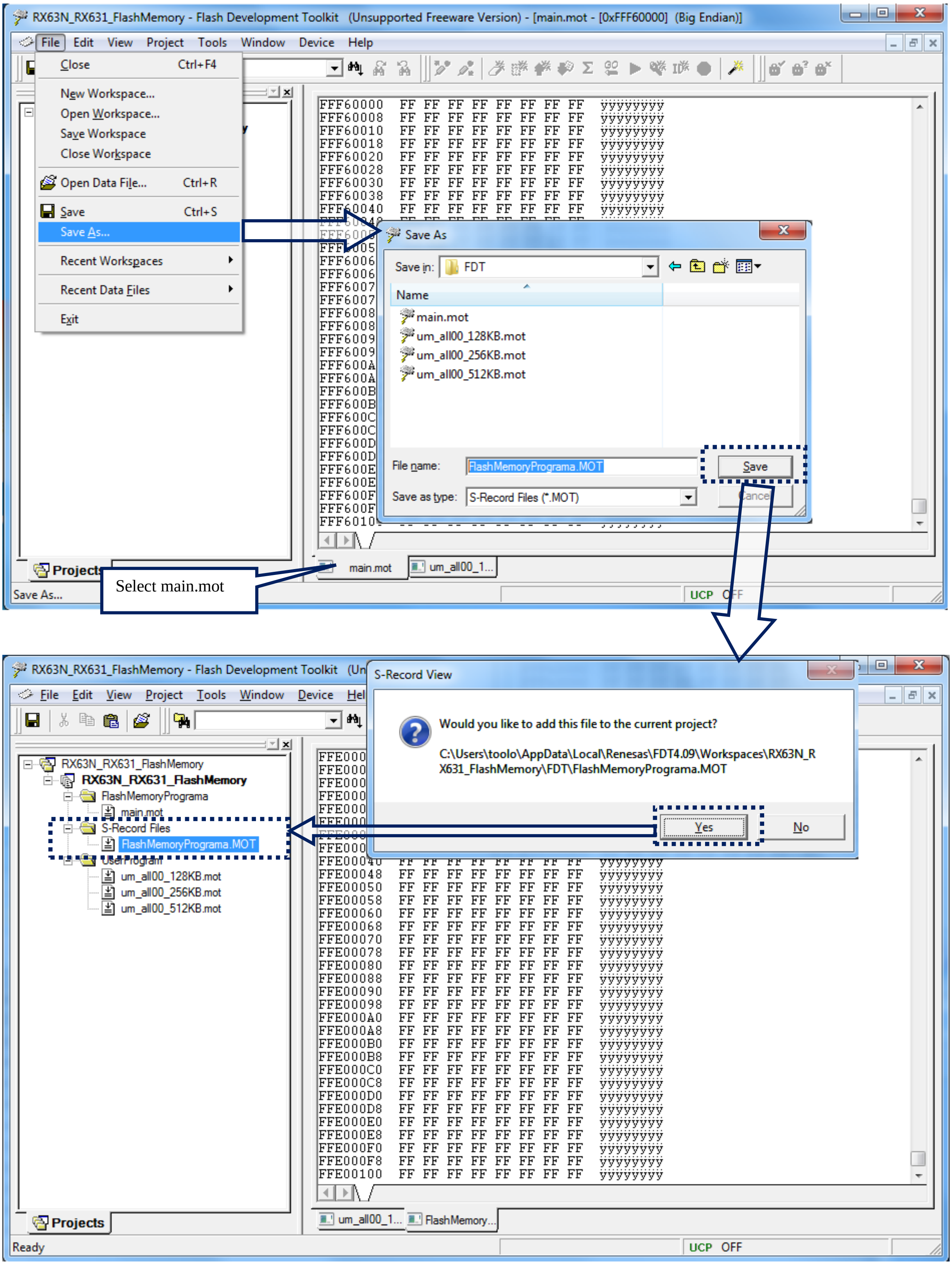
In the example, the start address of paste destination in the main.mot file is set to 0xFFF4 0000. After setting the start address, paste the data from the clipboard.



(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:

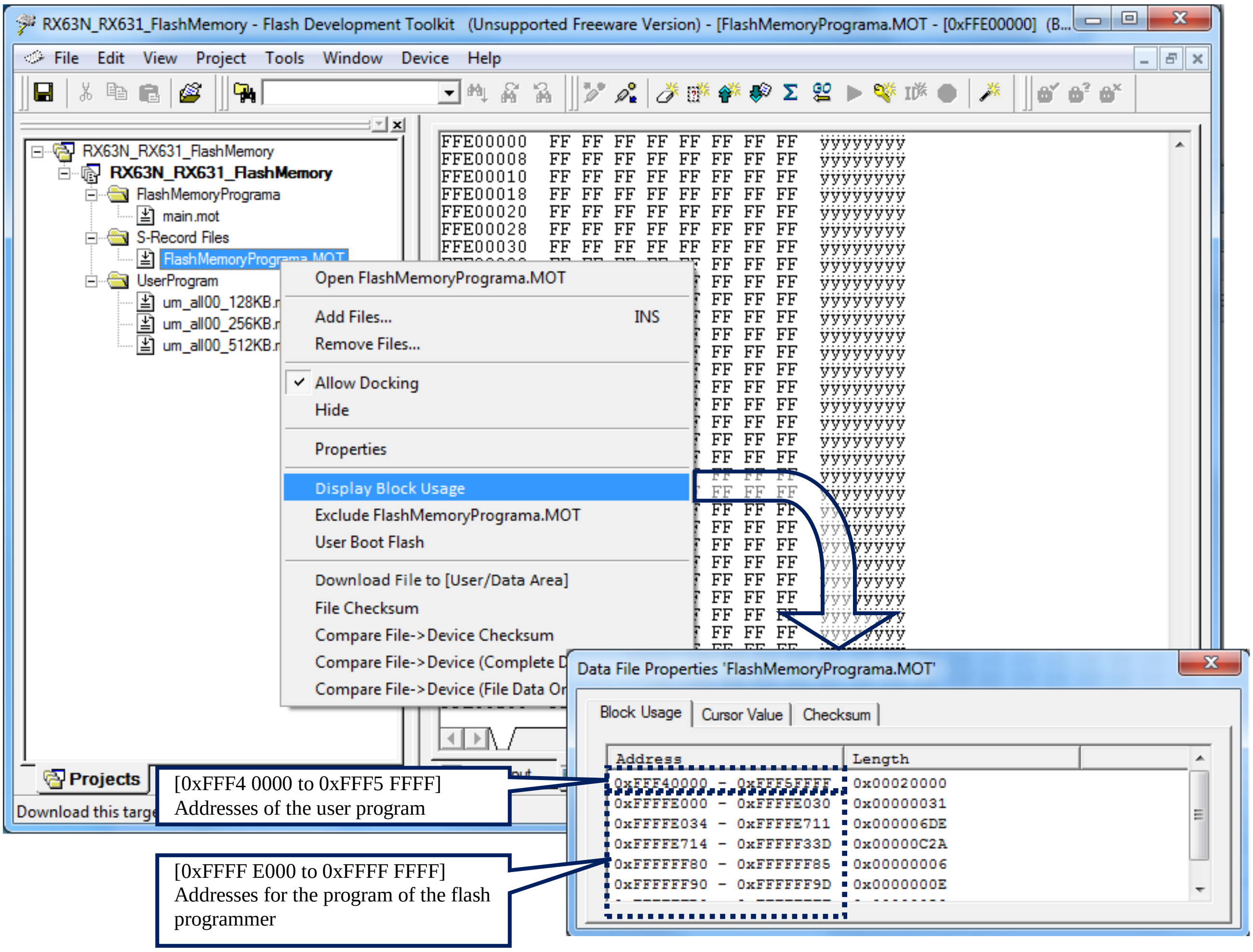
Addresses 0xFFF4 0000 to 0xFFF5 FFFF when the user program size is 128 Kbytes

Addresses 0xFFF4 0000 to 0xFFF7 FFFF when the user program size is 256 Kbytes

Addresses 0xFFF4 0000 to 0xFFFB FFFF when the user program size is 512 Kbytes

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

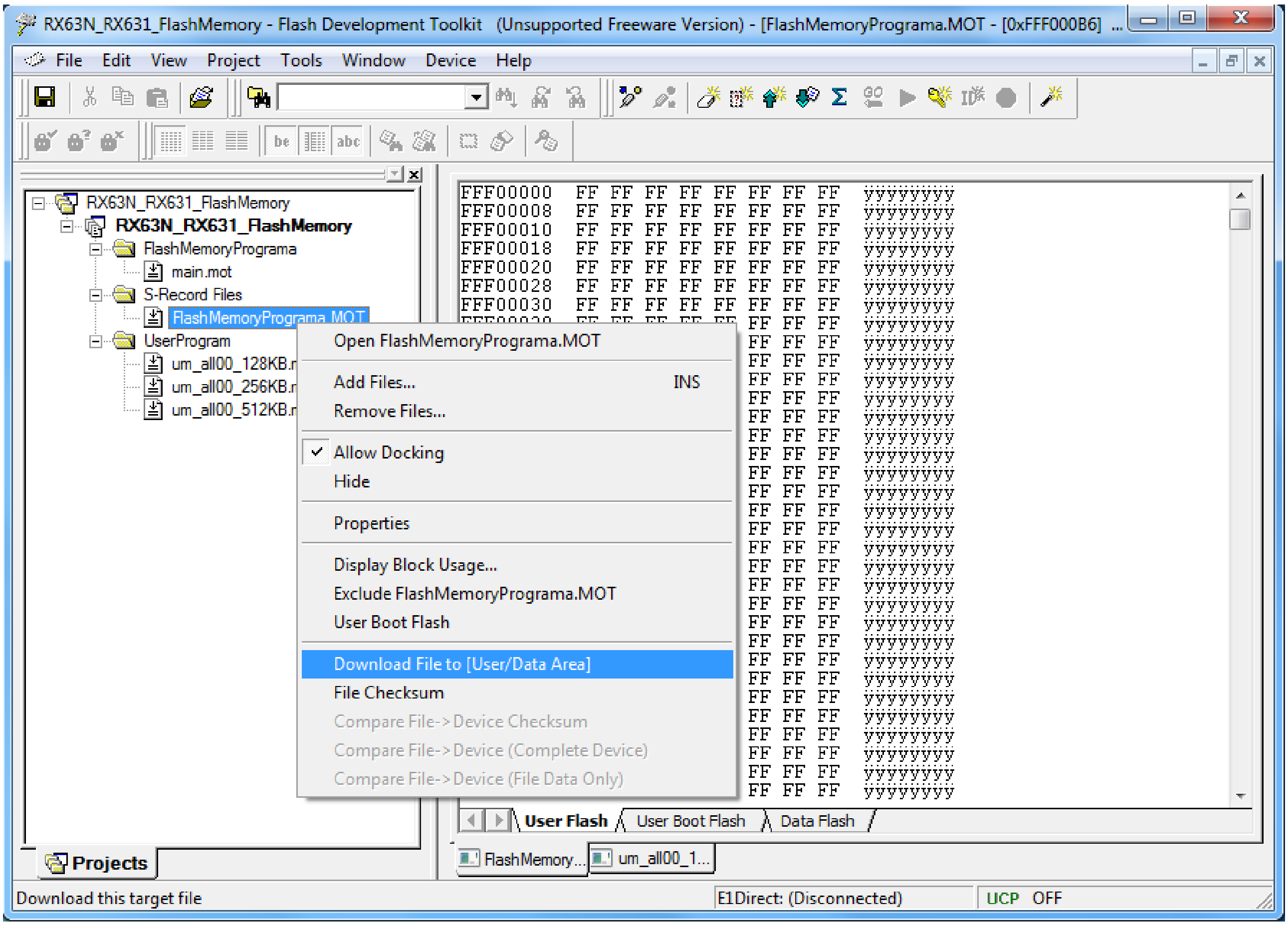
In the example, the address range of the block used is confirmed when the user program size is 128 Kbytes.



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 (SCI) and the bit rate is automatically adjusted to connect to the MCU at 19,200 bps.

After connecting, the supported device inquiry command, device select command, and block information inquiry command are sent to obtain information of the target MCU, and the operating frequency select command is sent to change the bit rate to 1 Mbps.

The program/erase state transition command is sent to check the ID code protection of the target MCU and perform the processing for the boot mode ID code protection.

The target MCU user area is erased and then programmed according to the obtained information of the target MCU. After the user area has been programmed, 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.

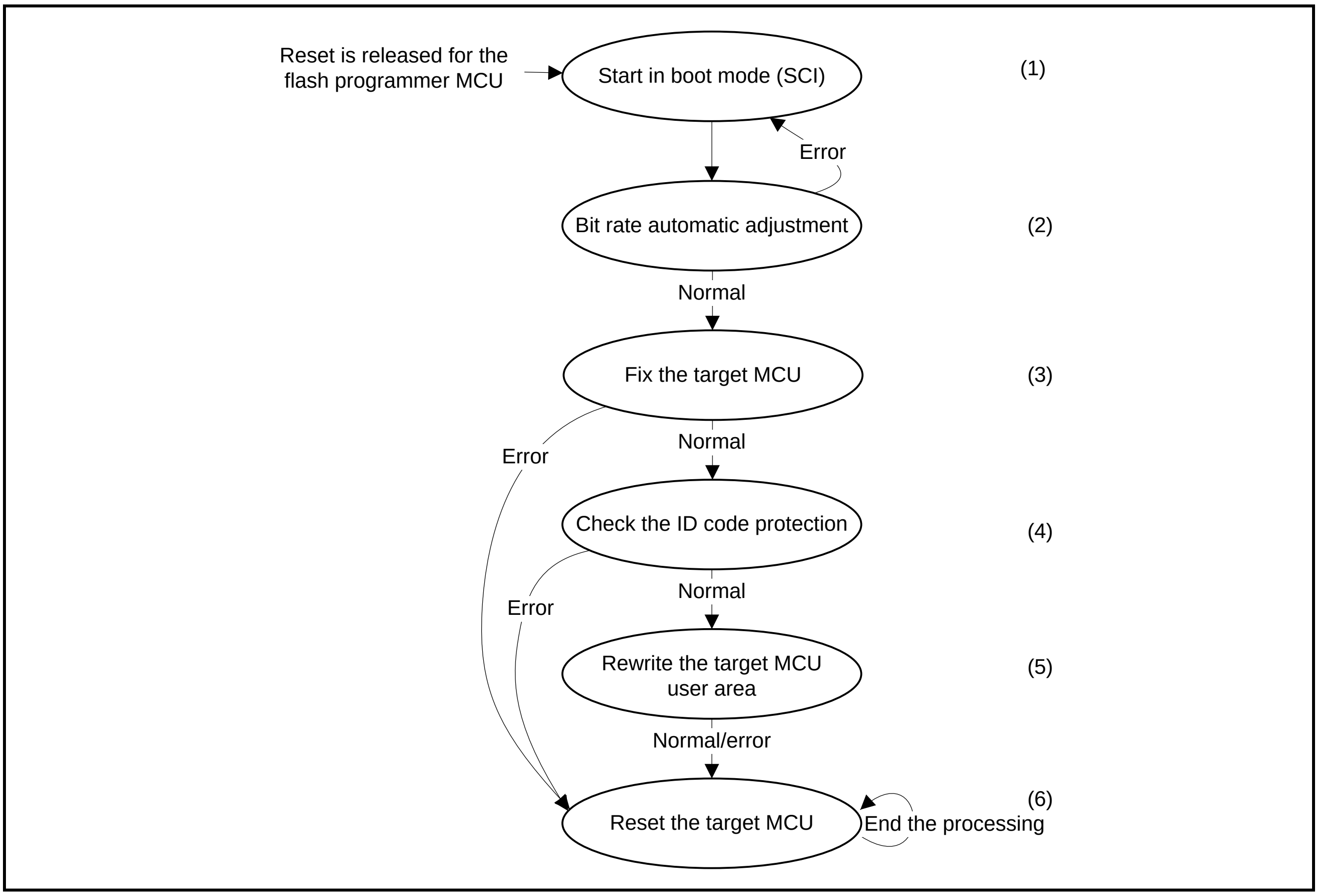


Figure 5.2 Flash Programmer State Transition

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

(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 UB# pin of the target MCU to high.

(4) After waiting 3 ms, the flash programmer sets the RES# pin of the target MCU to high.

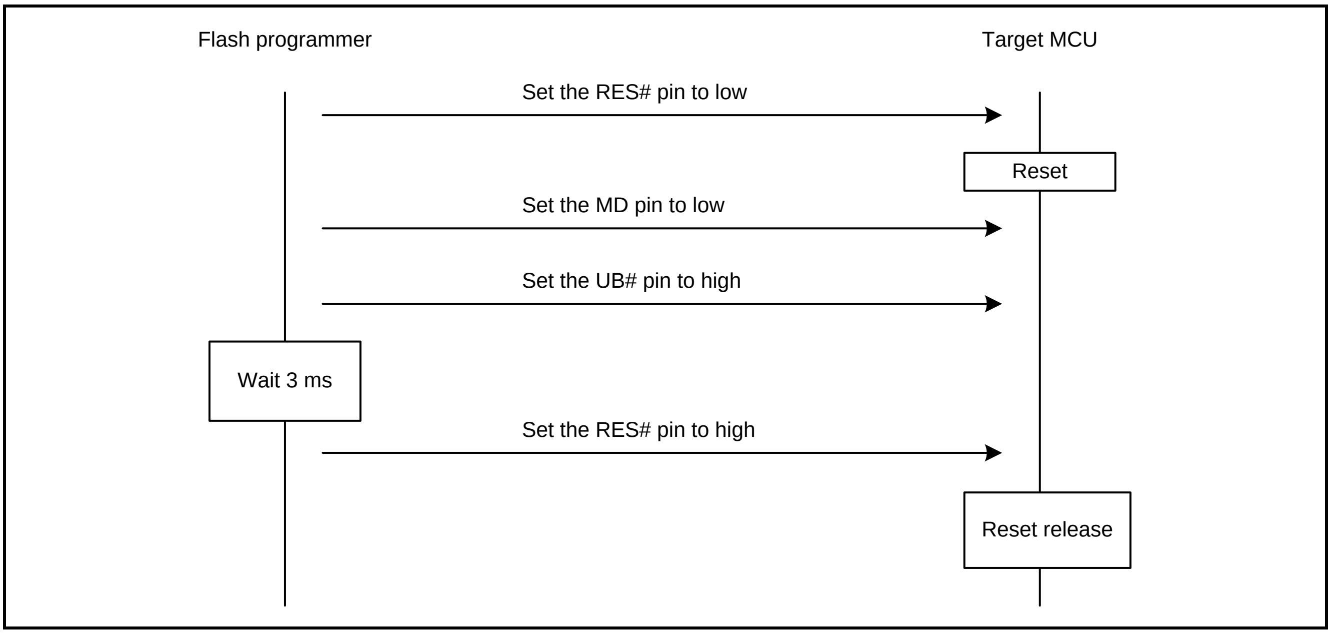


Figure 5.3 Start Procedure in Boot Mode (SCI)

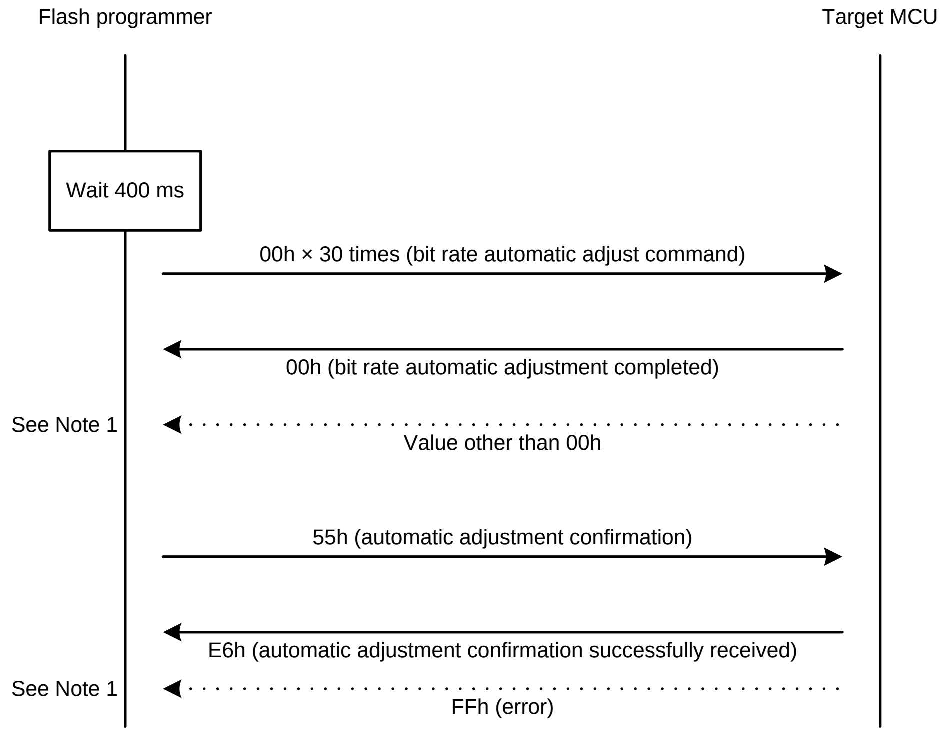
5.2.2 Bit Rate Automatic Adjustment

The flash programmer starts the target MCU in boot mode (SCI), 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 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.



Note 1. Restart the target MCU in boot mode (SCI) and send the bit rate automatic adjust command again.

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 the identification code for selecting the endian of data to be programmed in the user area.

The flash programmer receives a response (data starting with 30h) to the supported device inquiry command to store identification 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 Identification Codes.

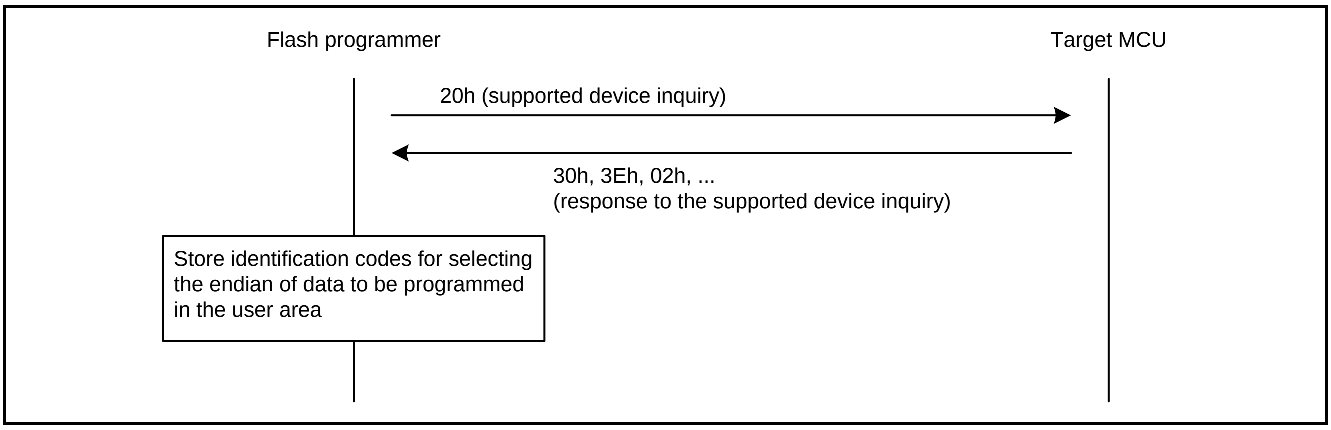


Figure 5.5 Procedure to Store Identification Codes

(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 identification code corresponding to the endian of the flash programmer in the identification codes that were stored by the support devise inquiry command.

After the flash programmer sends the device select command, the endian selection is completed when a response (46h) is received. When the flash programmer receives a response (46h) after sending the device select command, it completes the endian selection. When the flash programmer receives data other than the response (46h) after sending the device select command, it resets the target MCU to abort.

Figure 5.6 shows the Procedure to Select the Endian.

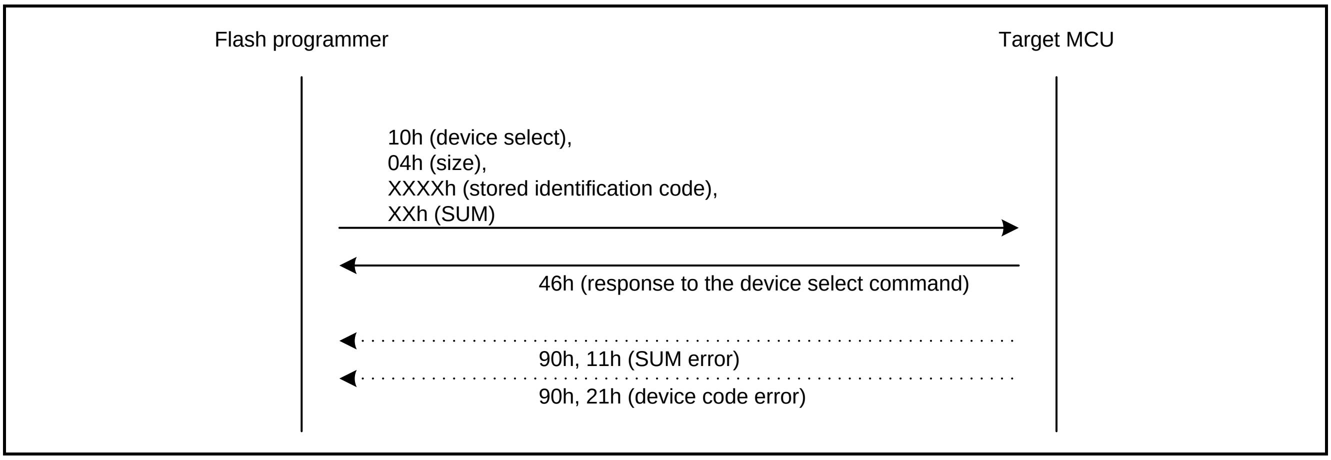


Figure 5.6 Procedure to Select the Endian

(3) The flash programmer sends the block information inquiry command to store the block configuration of the target MCU.

When the flash programmer receives a response (data starting with 36h) to the block information inquiry command, it stores the block configuration of the target MCU in the block information storage buffer. When the flash programmer receives data other than the response (data starting with 36h) to the block information inquiry command, it resets the target MCU to abort.

Figure 5.7 shows the Procedure to Store the Block Information.

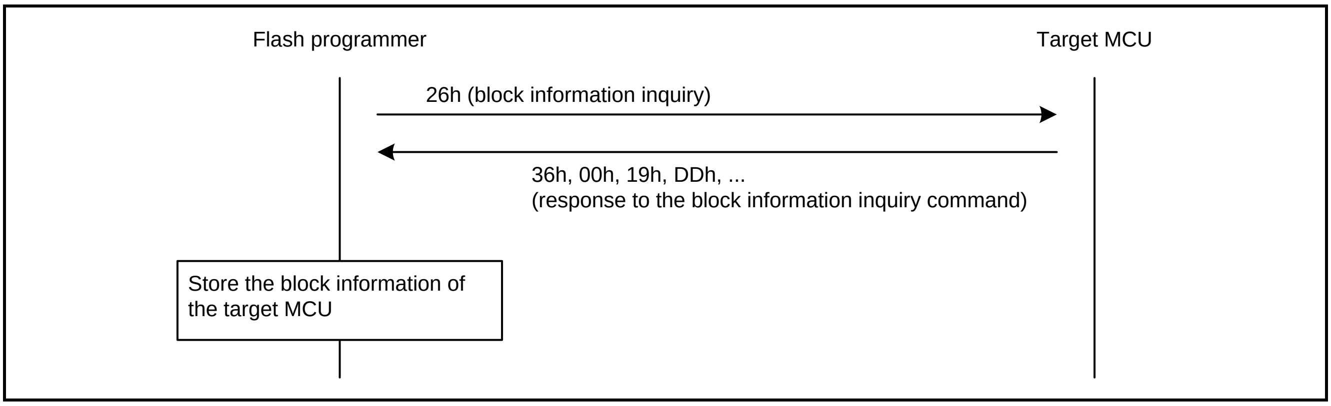


Figure 5.7 Procedure to Store the Block Information

(4) The flash programmer sends the operating frequency select command to change the bit rate with the target MCU to 1 Mbps.

The flash programmer sends the operating frequency select command (3Fh) to change the bit rate to 1 Mbps. When the flash programmer receives ACK (06h) after sending the operating frequency select command, it waits the 1-bit period at the bit rate for sending the operating frequency select command and then changes the bit rate to 1 Mbps. After that, the flash programmer sends a communication confirmation data (06h) at the changed bit rate. When the flash programmer receives 06h (response to the confirmation data), it completes changing the bit rate.

When the flash programmer receives data other than the response (06h) after sending the operating frequency select command, or when it receives data other than 06h (response to the confirmation data) after sending the communication confirmation data (06h), it resets the target MCU to abort.

Figure 5.8 shows the Procedure to Change the Bit Rate.

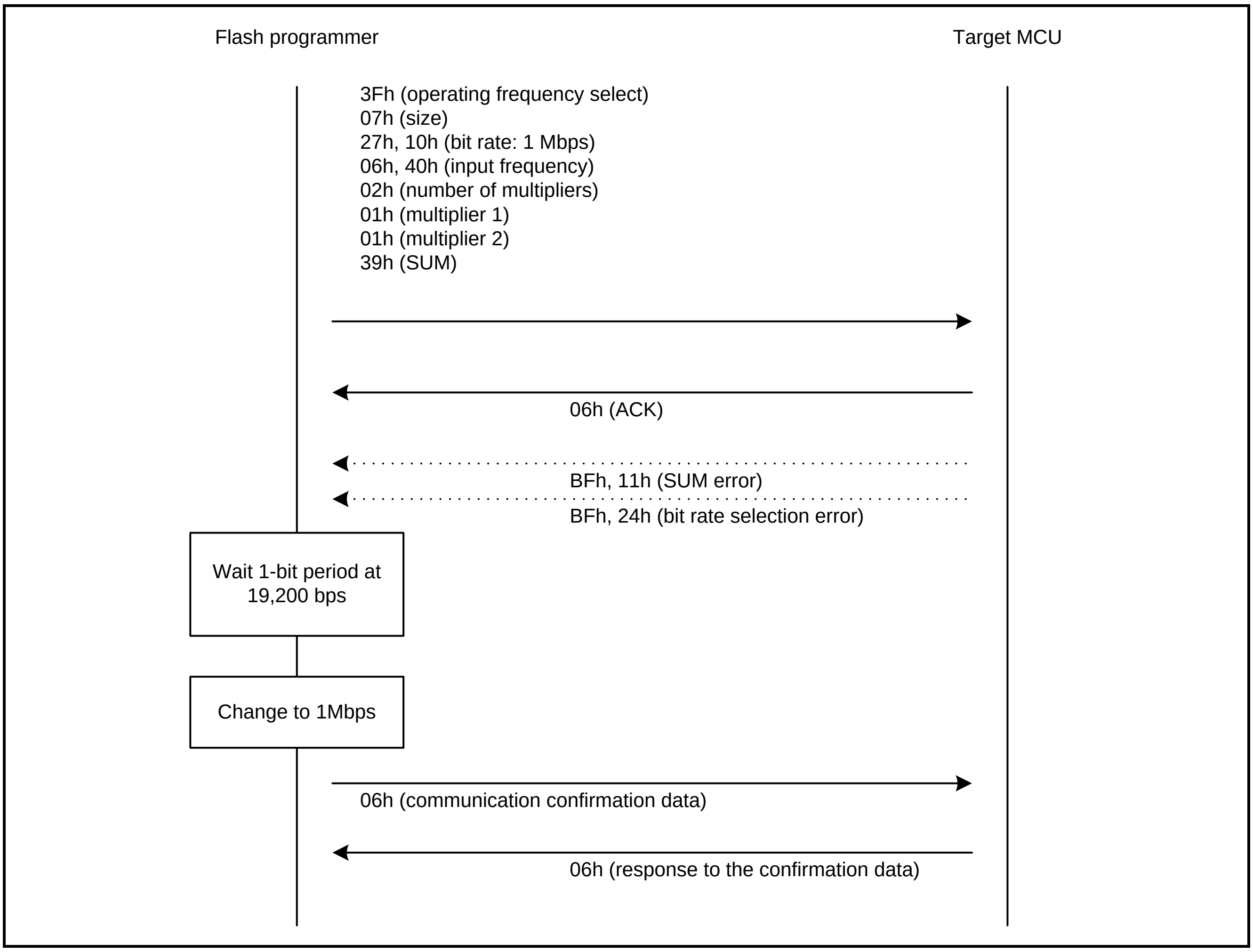


Figure 5.8 Procedure to Change the Bit Rate

5.2.4 Check ID Code Protection

The flash programmer performs steps (1) and (2) below to check the ID code protection.

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

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

After the flash programmer sends the program/erase 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
06h	00h
16h	01h
56h	02h

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

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

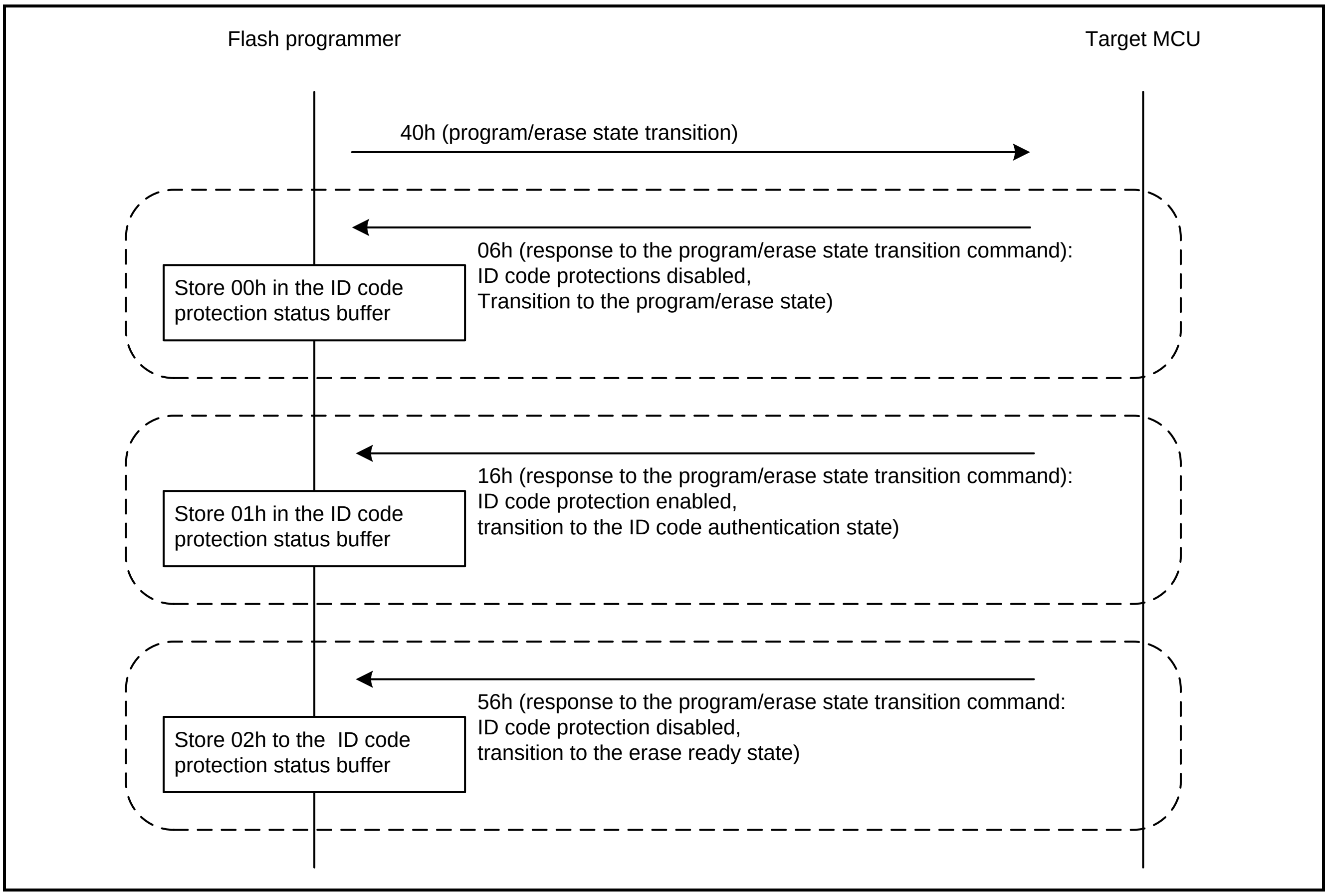


Figure 5.9 Procedure to Check ID Code Protection by the Program/Erase State Transition Command

(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.

The flash programmer sends the ID code check command (60h) to check the response received after sending the ID code check command and store the corresponding value in the ID code protection status buffer.

Table 5.2 lists the Responses and Values Stored in the ID Code Protection Status Buffer.

Table 5.2 Responses and Values Stored in the ID Code Protection Status Buffer

Response	Values Stored in the ID Code Protection Status Buffer
06h	00h
56h	02h

The flash programmer receives data other than values listed in Table 5.2 after sending the program/erase state transition command, reset the target MCU to abort.

Figure 5.4 shows the Bit Rate Automatic Adjustment Procedure.

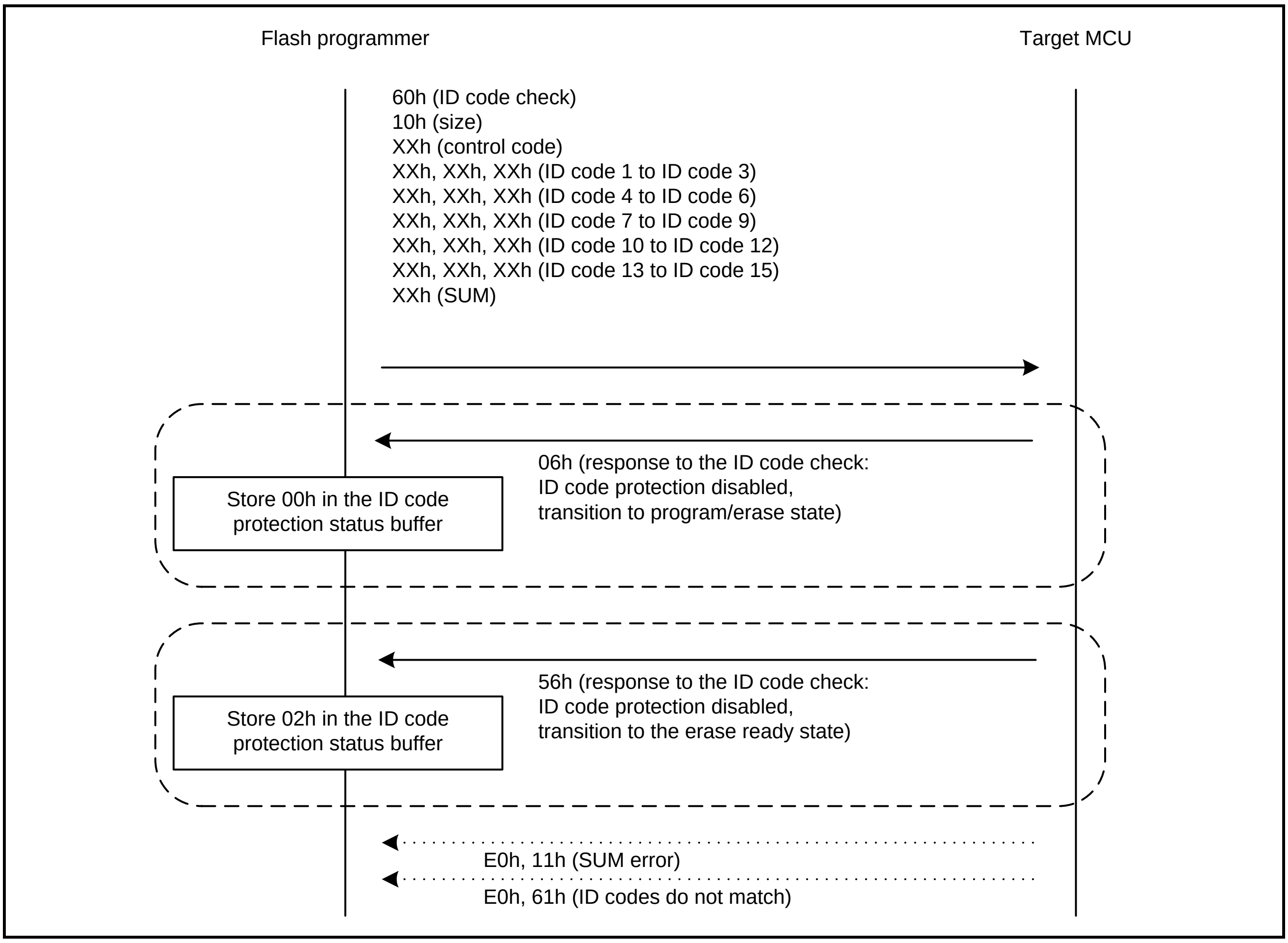


Figure 5.10 Procedure to Check ID Code Protection by the ID Code Check Command

5.2.5 Rewrite the Target MCU User Area

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

(1) The flash programmer erases the flash memory of the target MCU for rewriting the user program in the target MCU user area.

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

The flash programmer sends 59h (block erase command) as many times as the number of blocks to erase. The number of blocks to erase is calculated from the value stored in the ID code protection status buffer as follows:

When the stored value is 02h, the sum of values in block information storage buffer 3 and block information storage buffer 6

When the stored value is a value other than 02h, the value in block information storage buffer 3

After sending the block erase command as many times as the number of blocks to erase, the flash programmer sends 59h 04h FFh FFh FFh FFh A7h (block erase command to end block erase). When the flash programmer receives 06h (response to the block erase command) after sending the block erase command, it completes the block erase operation. When the flash programmer receives data other than 06h (response to the block erase command), it resets the target MCU to abort.

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

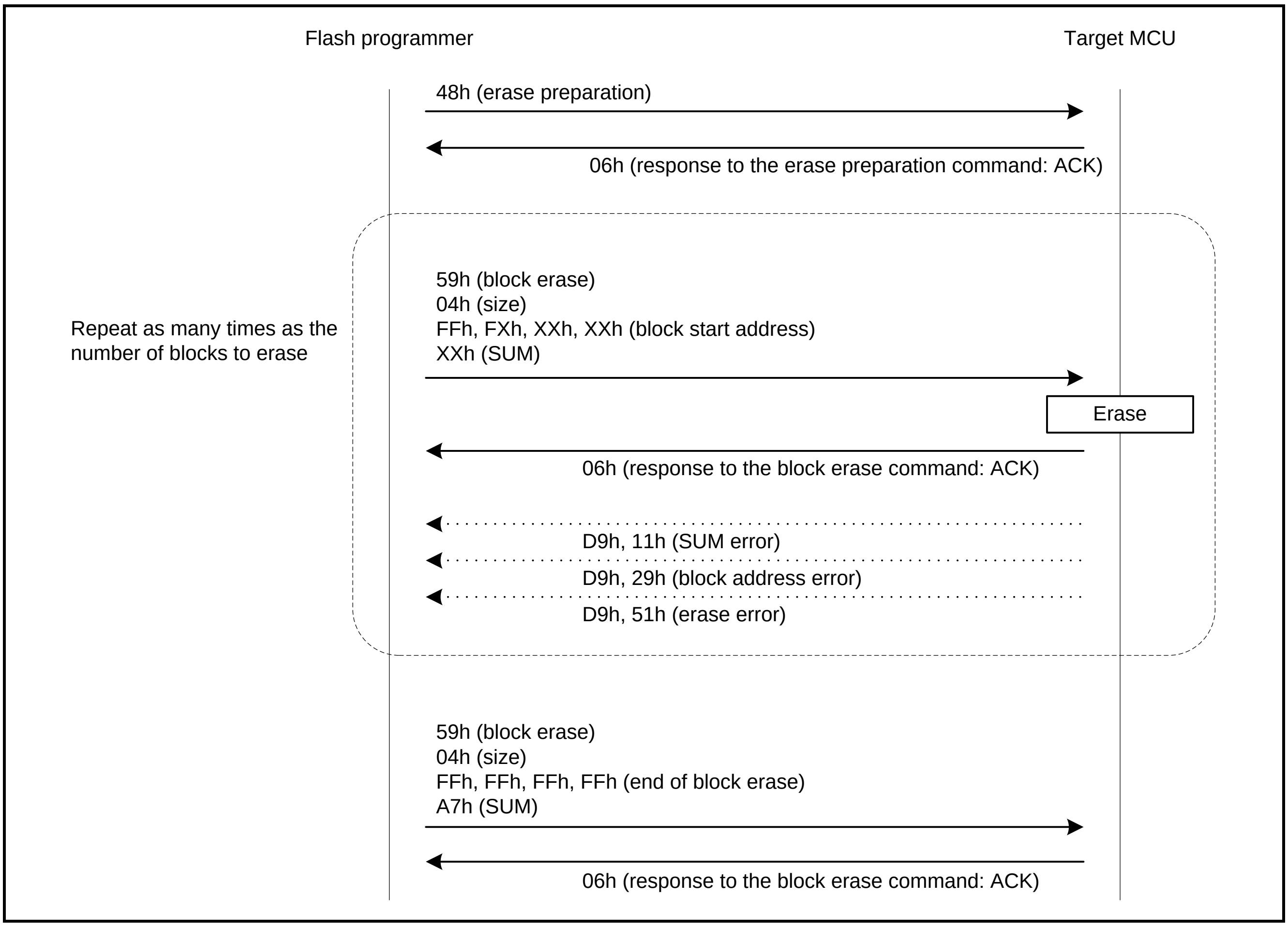


Figure 5.11 Procedure to Erase the Flash Memory of the Target MCU

(2) The flash programmer confirms that the target MCU has erased the flash memory successfully.

The flash programmer sends the boot mode status inquiry command (4Fh).

When the flash programmer receives a response (data starting with 5Fh) after sending the boot mode status inquiry command, the flash programmer confirms that the target MCU has erased the flash memory successfully.

When the flash programmer receives data other than the response (data starting with 5Fh), it resets the target MCU to abort.

Figure 5.12 shows the Procedure to Complete Preparation for Programming the Target MCU.

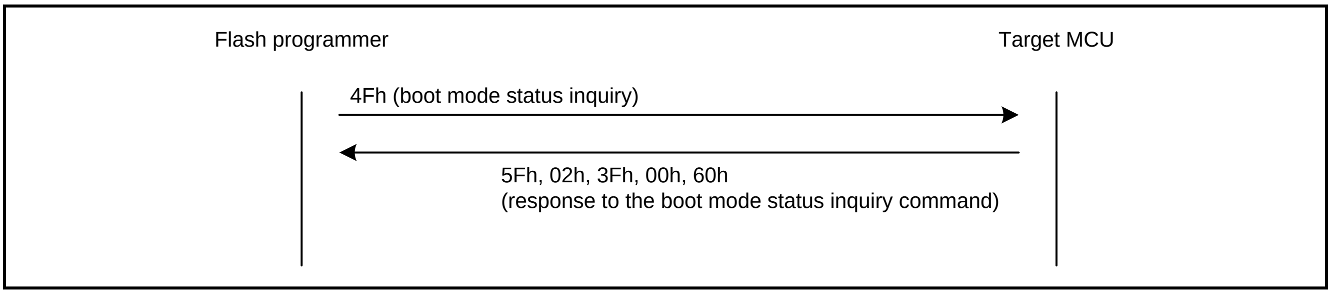


Figure 5.12 Procedure to Complete Preparation for Programming the Target MCU

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

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

After completion of preparation, the flash programmer sends 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 0xFFFE 0000 to 0xFFFF FFFF when the user program size is 128 Kbytes

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

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

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

Addresses from 0xFFF4 0000 to 0xFFF5 FFFF when the user program size is 128 Kbytes

Addresses from 0xFFF4 0000 to 0xFFF7 FFFF when the user program size is 256 Kbytes

Addresses from 0xFFF4 0000 to 0xFFFB FFFF when the user program size is 512 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 the program operation. When the flash programmer receives data other than 06h, it resets the MCU to abort.

Figure 5.13 shows the Procedure to Program the User Area.

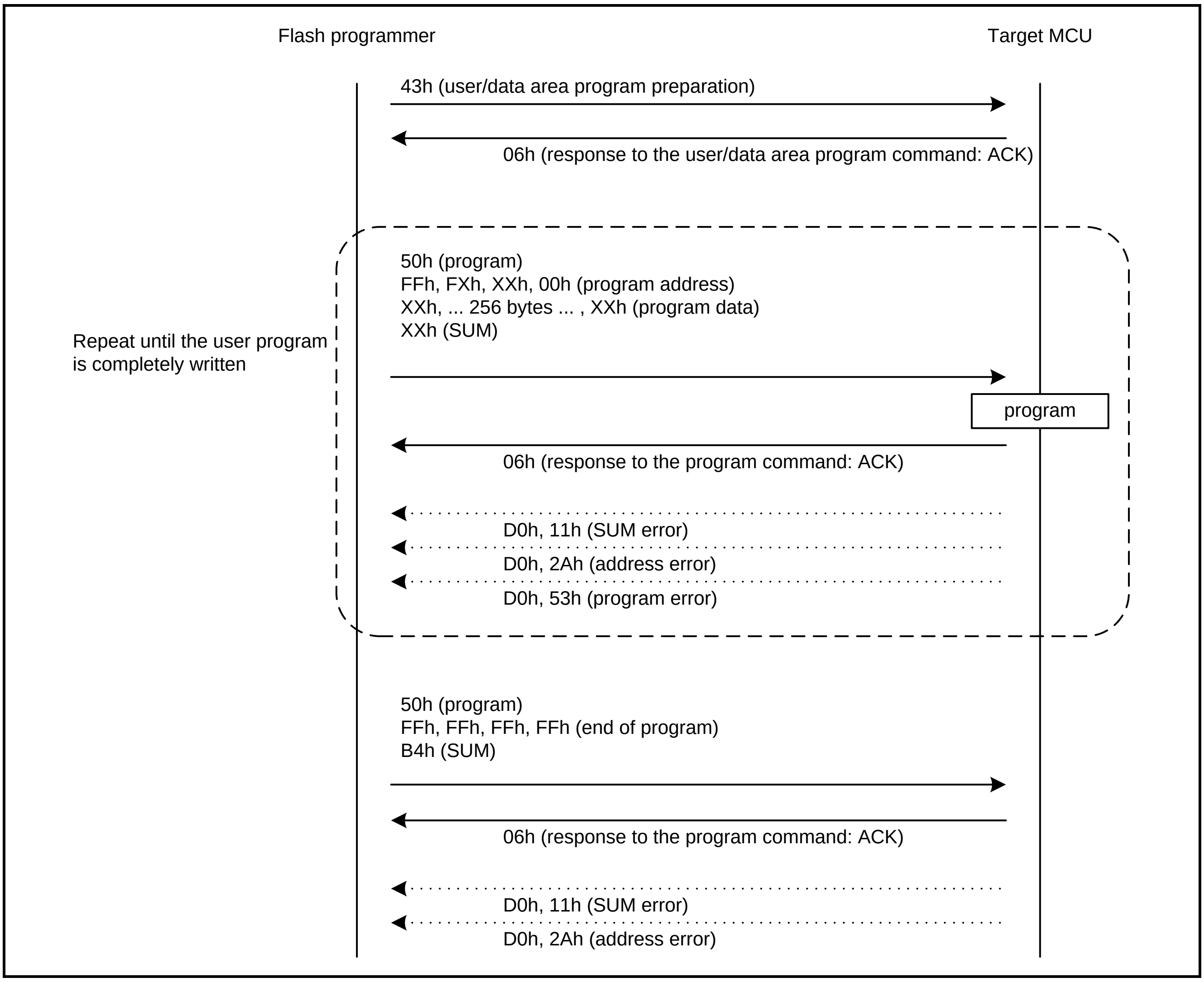


Figure 5.13 Procedure to Program the User Area

(4) The flash programmer confirms that the target MCU has been programed 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 0xFFFE 0000 to 0xFFFF FFFF when the user program size is 128 Kbytes

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

Addresses from 0xFFF8 0000 to 0xFFFF FFFF when the user program size is 512 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 0xFFF4 0000 to 0xFFF5 FFFF when the user program size is 128 Kbytes

Addresses from 0xFFF4 0000 to 0xFFF7 FFFF when the user program size is 256 Kbytes

Addresses from 0xFFF4 0000to 0xFFFB FFFF when the user program size is 512 Kbytes

Figure 5.14 shows the Procedure to Confirm Data in the User Area.

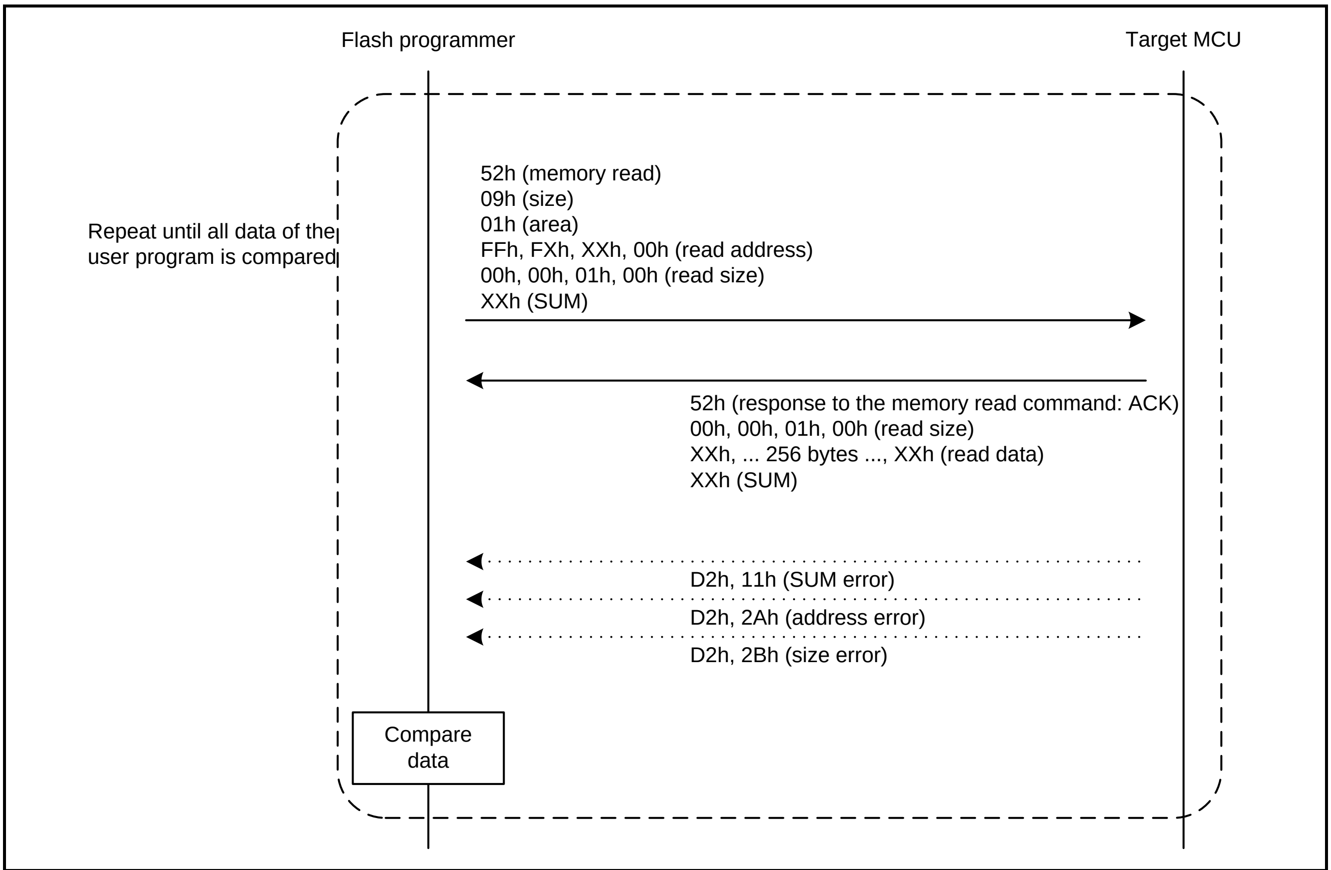


Figure 5.14 Procedure to Confirm Data in the User Area

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 3 ms and then drives the RES# pin of the target MCU high.

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

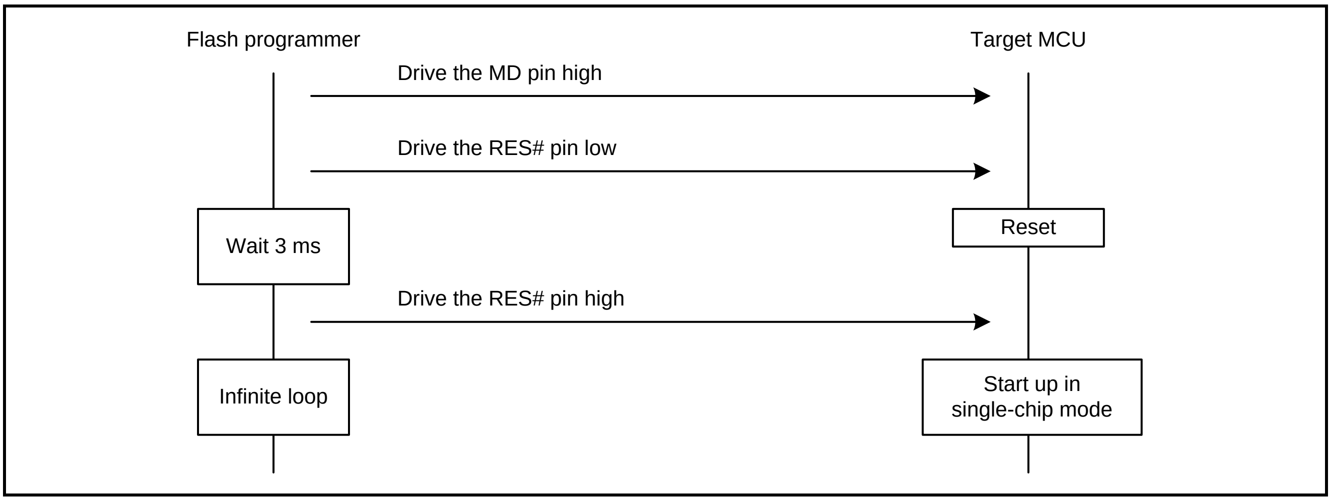


Figure 5.15 Procedure to Reset the Target MCU

5.3 File Composition

Table 5.3 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.3 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 5.4 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 5.5 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 5.6 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 5.7 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.7 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 5.8 to Table 5.13 list the Constants Used in the Sample Code.

Table 5.8 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
ROMVOL_128KB	(128 * 1024)	Selected when the user area size of the target MCU is 128 Kbytes
ROMVOL_256KB	(256 * 1024)	Selected when the user area size of the target MCU is 256 Kbytes
ROMVOL_512KB	(512 * 1024)	Selected when the user area size of the target MCU is 512 Kbytes
TARGET_ROMVOL	ROMVOL_128KB	User area size of the target MCU (128 Kbytes selected)
TARGET_DATA_ADD	0xFFF40000	Start address for storing data programmed in the target MCU user area
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_ACK_ID	(0x16)	ACK for enabling ID code protection is received.
RES_ACK_BERS_EXSPC	(0x46)	ACK for block erase extended specification is received.
RES_ACK_MERSMD	(0x56)	ACK for erase ready operation 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_52US	((52. * (WT_CMT_CLOCK / WT_CMT_DIVIDE)) / WT_BASE_US +0.5)	Wait time with the CMT (52 μs)
WAIT_1MS	((1. * (WT_CMT_CLOCK / WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (1 ms)
WAIT_3MS	((3. * (WT_CMT_CLOCK / WT_CMT_DIVIDE)) / WT_BASE_MS +0.5)	Wait time with the CMT (3 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 5.9 Constants Used in the Sample Code (ROMVOL_128KB 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	0xFFFE0000	Start address for programming the target MCU

Table 5.10 Constants Used in the Sample Code (ROMVOL_256KB 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	0xFFFC0000	Start address for programming the target MCU

Table 5.11 Constants Used in the Sample Code (ROMVOL_512KB 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	0xFFF80000	Start address for programming the target MCU

Table 5.12 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 UB#, MD, and RES# of the target MCU (port mode register).
BTMD_PODR	(PORTE.PODR.BYTE)	Output pin is assigned to pins UB#, MD, and RES# of the target MCU (port output data register).
BTMD_PDR	(PORTE.PDR.BYTE)	Output pin is assigned to pins UB#, MD, and RES# of the target MCU (port direction register).
UB_PIN	(PORTE.PODR.BIT.B2)	Output is assigned to the UB# 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 UB#, MD, and RES# of the target MCU
BTMD_PODR_INIT	(0x04)	Initial value of high level output from the UB# pin of the target MCU

Table 5.13 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 5.16 shows the Structure/Union Used in the Sample Code.

```
typedef struct BOOT_CMD_s
{
    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 5.16 Structure/Union Used in the Sample Code

5.7 Variables

Table 5.14 lists the Global Variable, and Table 5.15 lists the static Variables.

Table 5.14 Global Variable

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

Table 5.15 static Variables

Type	Variable Name	Contents	Function Used
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
uint32_t	BlockInfoData[6]	Block information 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 select command data	-
uint8_t	CMD_BlockInfo[]	Block information inquiry command data	-
uint8_t	CMD_OperatingFreqSel_1st[]	Operating frequency select command data	-
uint8_t	CMD_OperatingFreqSel_2nd[]	Operating frequency selection confirm command data	-
uint8_t	CMD_PEstatusTransition[]	Program/erase state transition command data	-
uint8_t	CMD_IDCodeCheck[]	ID code check command data	-
uint8_t	CMD_ErasePreparation[]	Erase prepare command data	-
uint8_t	CMD_BlockErase[]	Block erase (extended specification) command data	-
uint8_t	CMD_BootModeStatusInquiry[]	Boot mode state inquiry command data	-
uint8_t	CMD_ProgramPreparation[]	User/data area program preparation command data	-
uint8_t	CMD_Program[]	Program command data	-
uint8_t	CMD_ProgramTermination[]	Program end command data	-

Type	Variable Name	Contents	Function Used
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 select command structure	main
BOOT_CMD_t	BlockInfo	Block information inquiry command structure	main
BOOT_CMD_t	OperatingFreqSel_1st	Operating frequency select command structure	main
BOOT_CMD_t	OperatingFreqSel_2nd	Operating frequency selection confirm command structure	main
BOOT_CMD_t	PEstatusTransition	Program/erase state transition command structure	main
BOOT_CMD_t	IDCodeCheck	ID code check command structure	main
BOOT_CMD_t	ErasePreparation	Erase prepare command structure	main
BOOT_CMD_t	BlockErase	Block erase (extended specification) command structure	main
BOOT_CMD_t	BootModeStatusInquiry	Boot mode state inquiry command structure	main
BOOT_CMD_t	ProgramPreparation	User/data area program preparation command structure	main
BOOT_CMD_t	Program	Program command structure	main
BOOT_CMD_t	ProgramTermination	Program end command structure	main
BOOT_CMD_t	MemoryRead	Memory read command structure	main

5.8 Functions

Table 5.16 lists the Functions.

Table 5.16 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, start the target MCU in boot mode (SCI) 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	Initialize 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	Initialize 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	Set the CMCOR register to the time (μs) specified in the argument and start incrementing the CMCNT register.
Arguments	uint16_t cnt: Wait time
Return Value	None
Remarks	The 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	Wait the time (μs) specified in the argument	
Arguments	uint16_t cnt	Wait time
Return Value	None	
Remarks	The minimum wait time is 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	Initialize 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	Change the SCI bit rate from 19,200 bps to 1 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	Calculate 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	Control pins MD, UB#, and RES# to start the target MCU in boot mode (SCI).	
Arguments	None	
Return Value	None	

BootModeRelease		
Outline	Processing to reset the target MCU	
Header	None	
Declaration	void BootModeRelease(uint8_t mode)	
Description	Reset 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	Send 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	Receive 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 (1 second) 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	Copy 4 bytes of data in the source memory area to the destination memory area. If the data arrangement is little endian, reverse 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 5.17 to Figure 5.30 show the Main Processing and Communication Protocol Control.

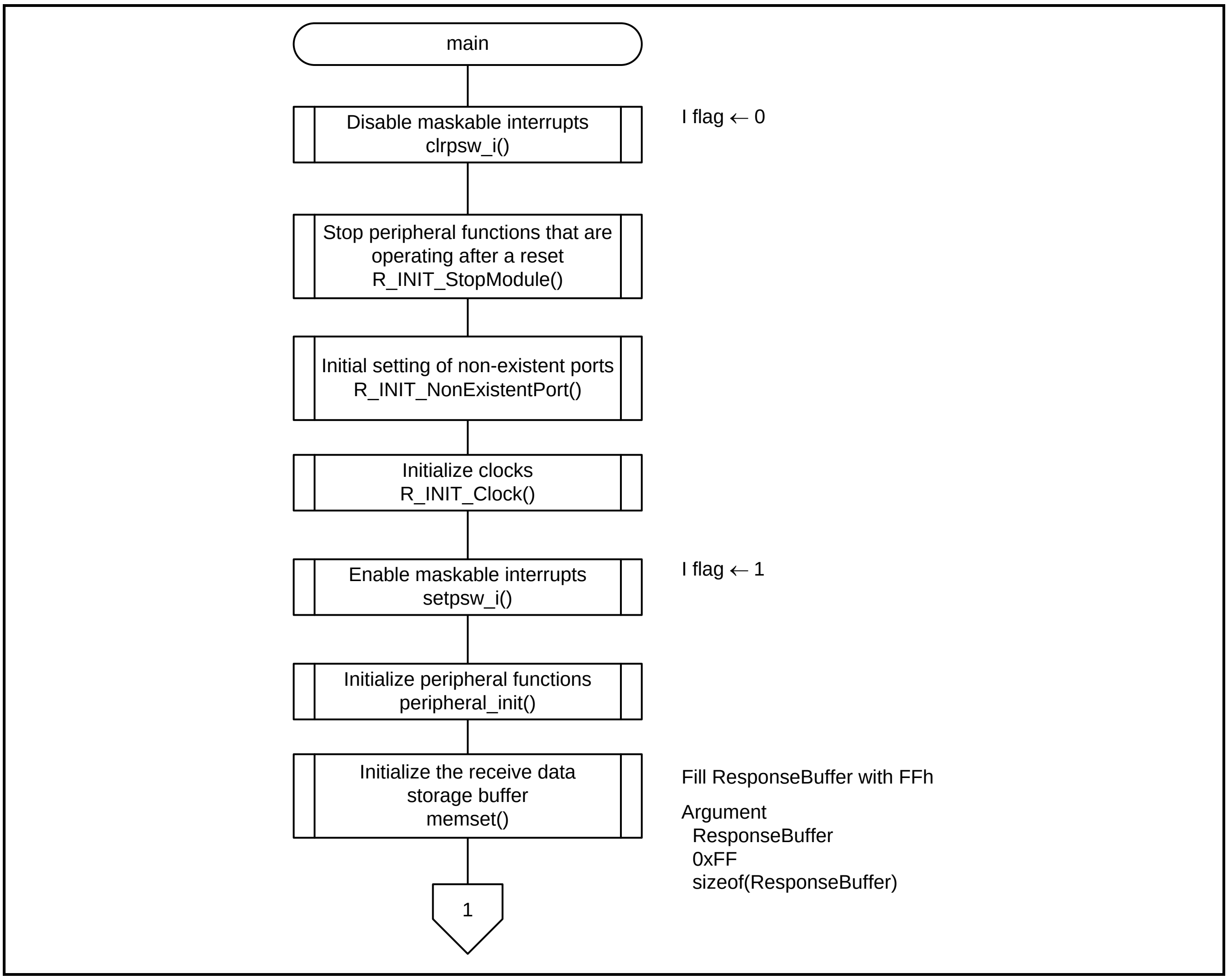


Figure 5.17 Main Processing and Communication Protocol Control

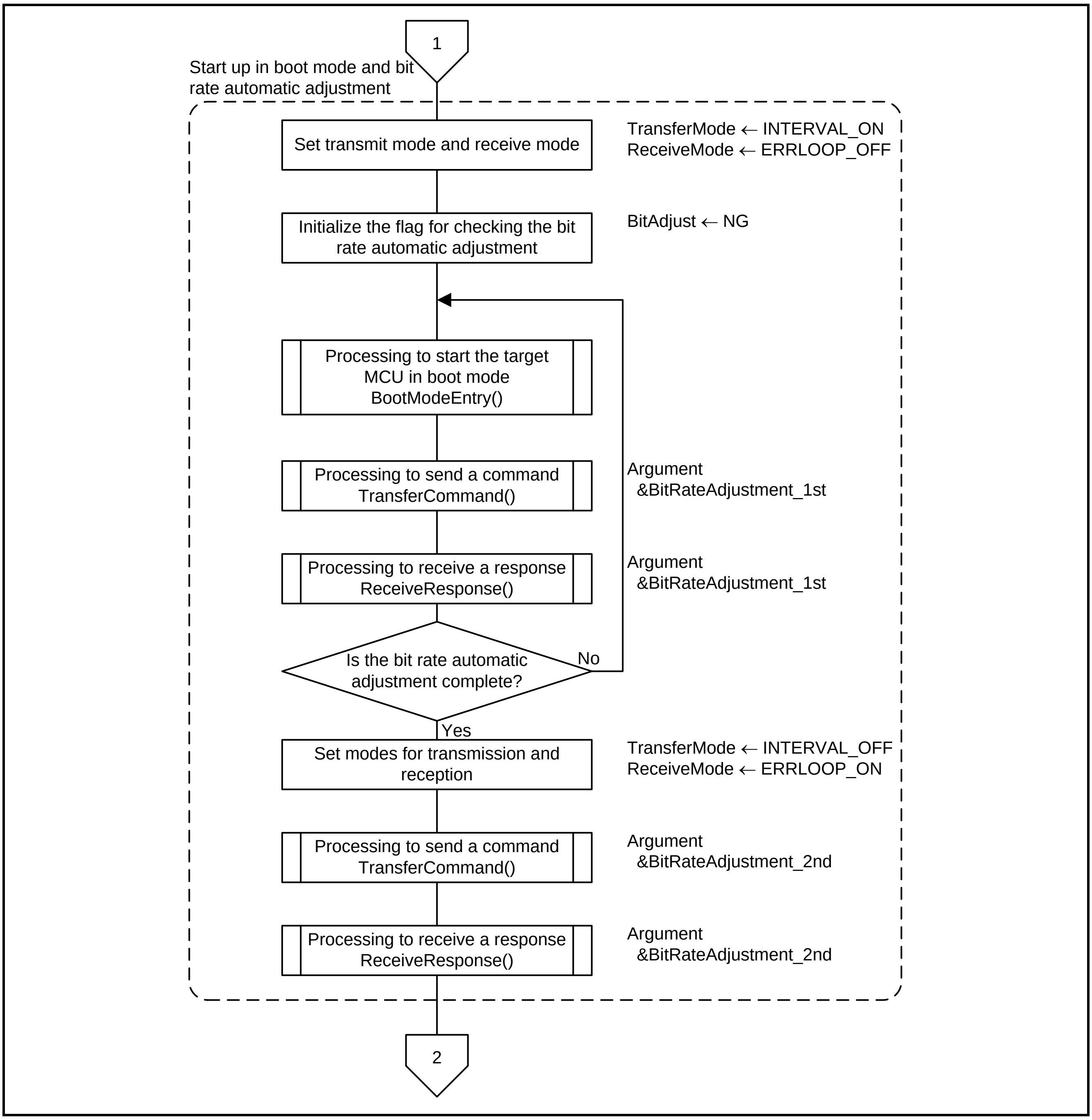


Figure 5.18 Main Processing and Communication Protocol Control

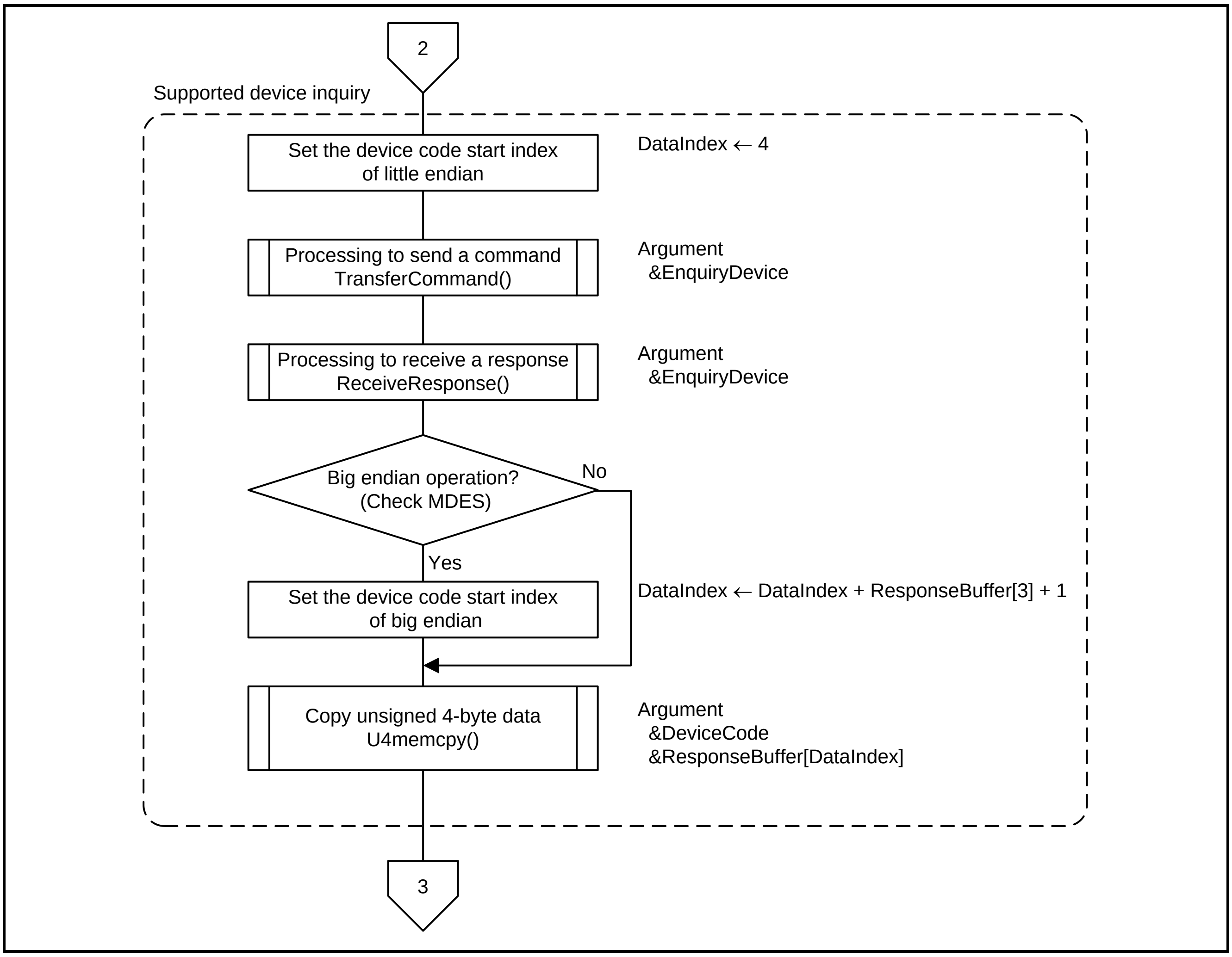


Figure 5.19 Main Processing and Communication Protocol Control

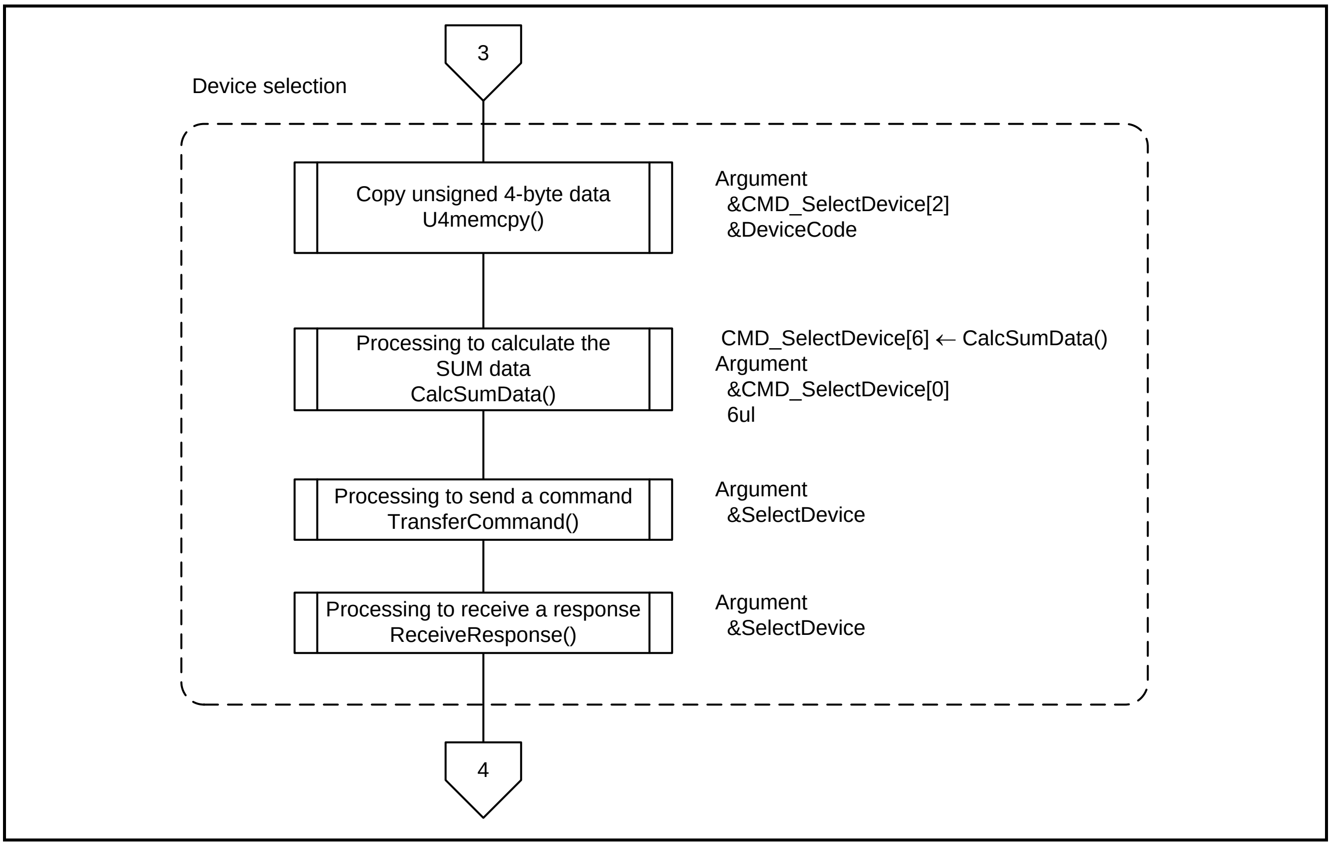


Figure 5.20 Main Processing and Communication Protocol Control

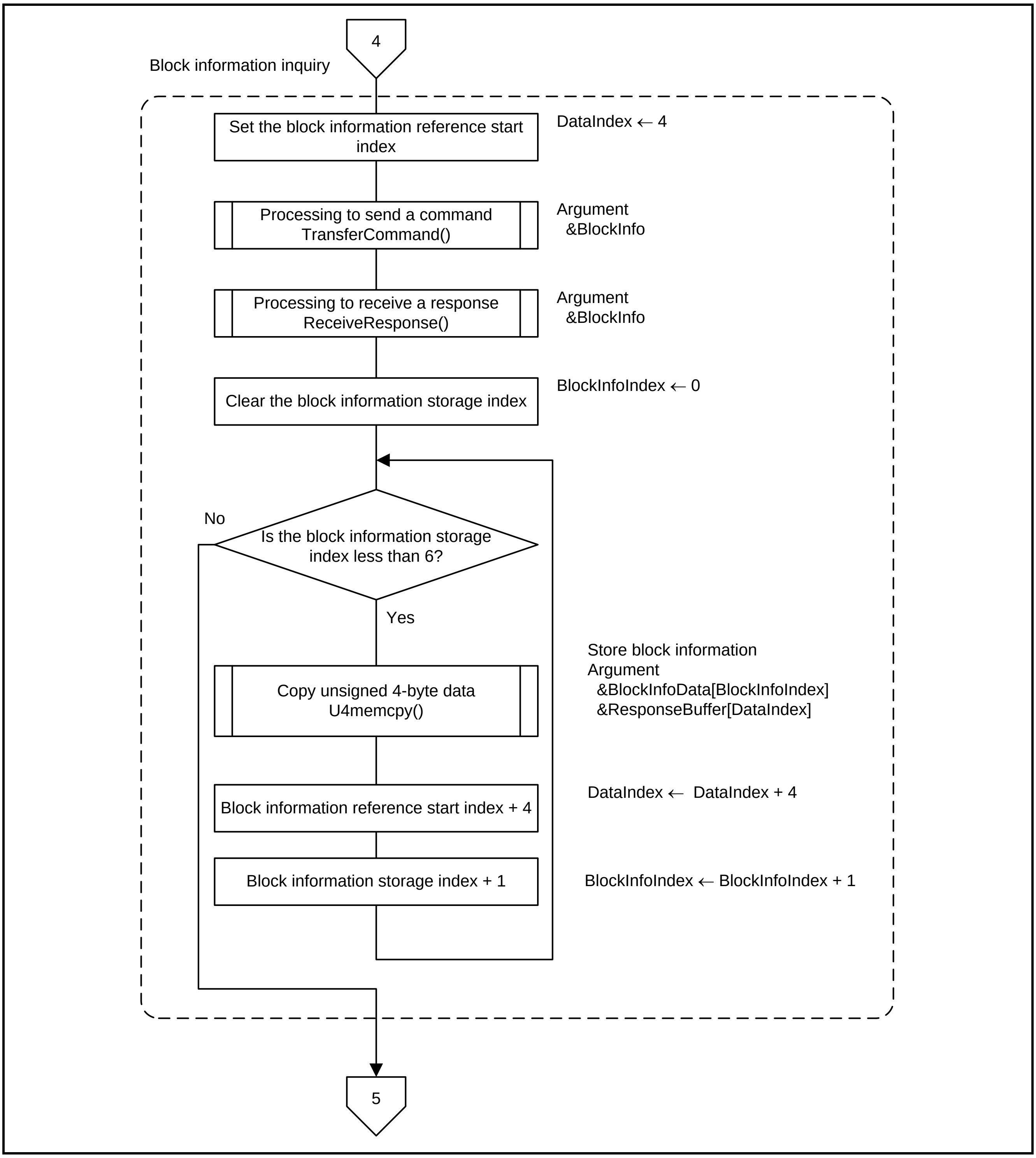


Figure 5.21 Main Processing and Communication Protocol Control

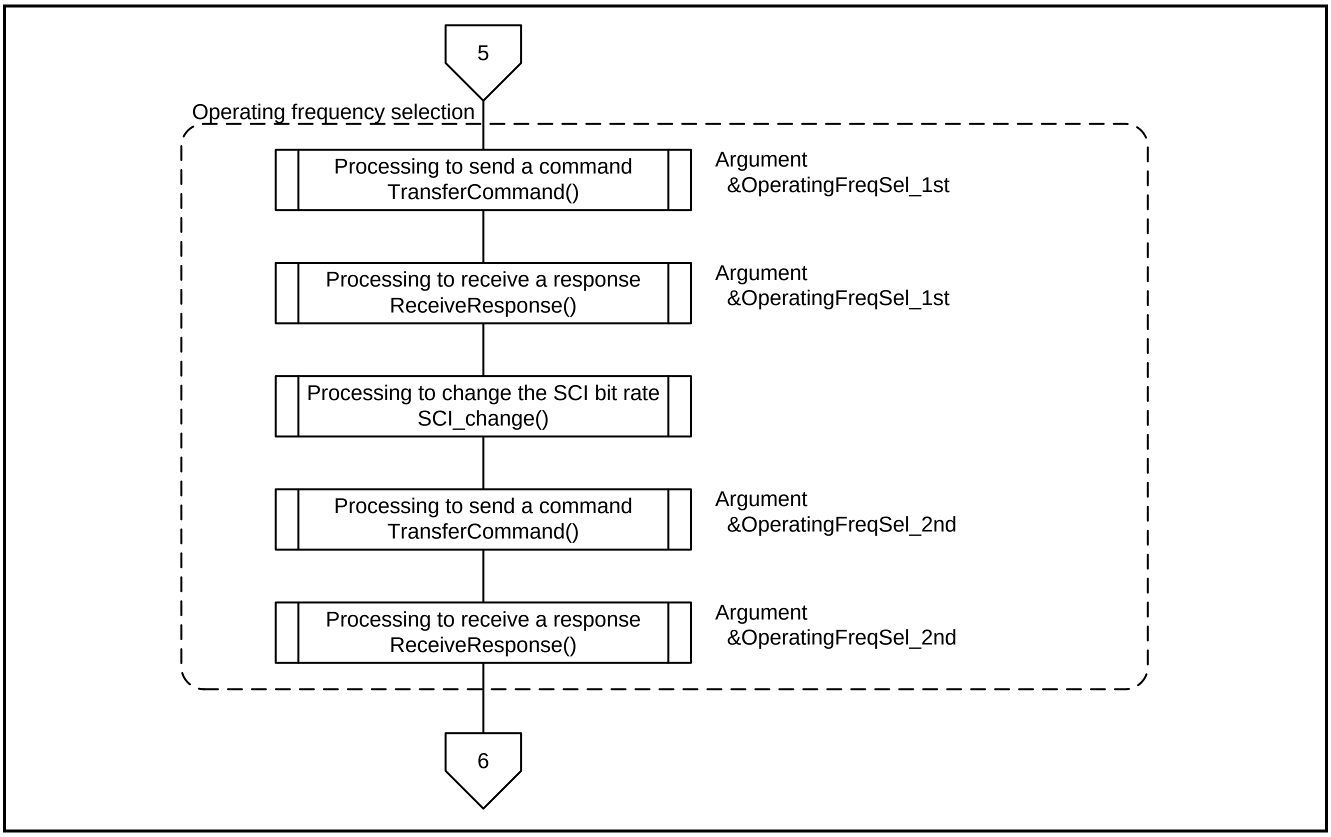


Figure 5.22 Main Processing and Communication Protocol Control

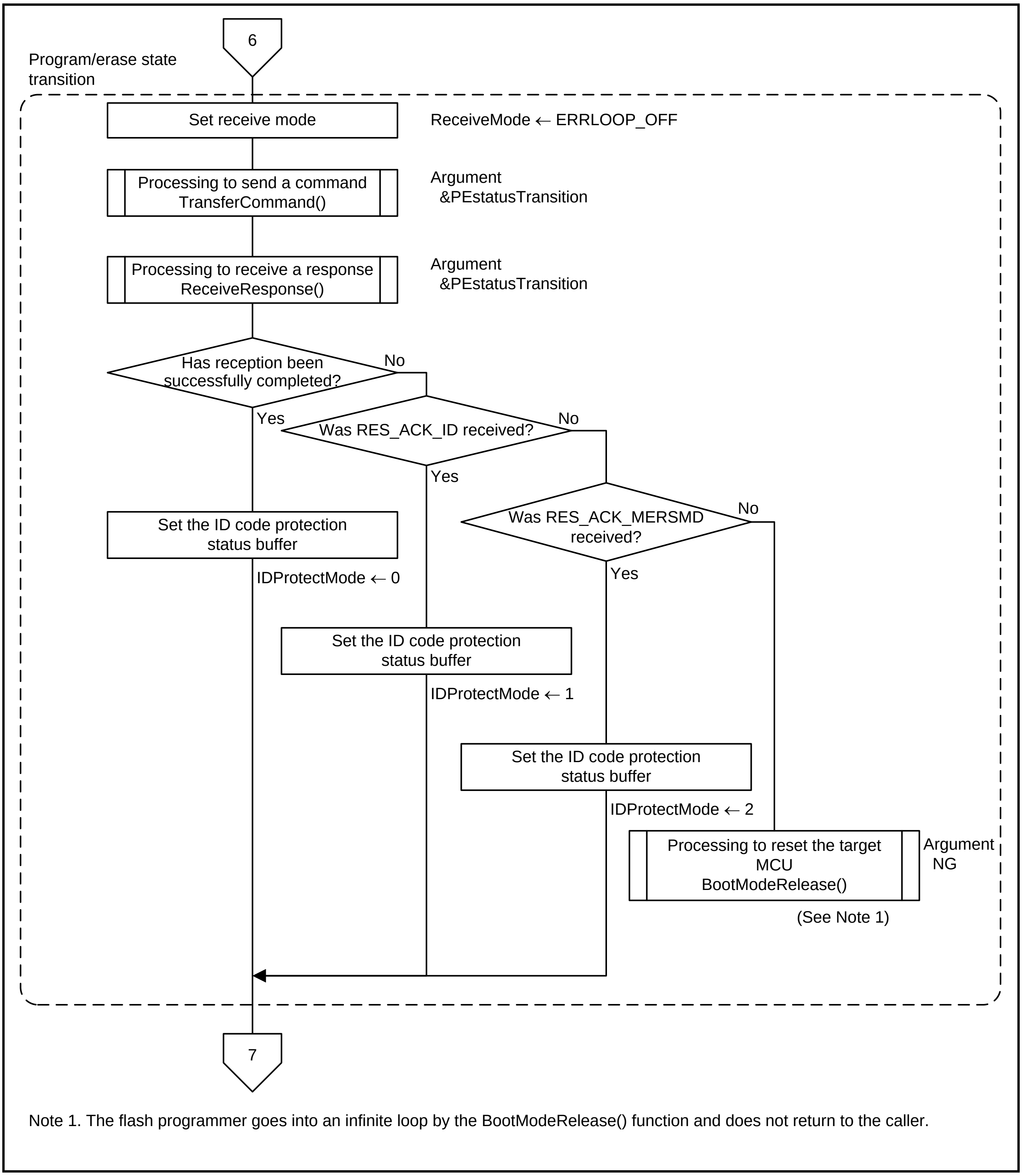
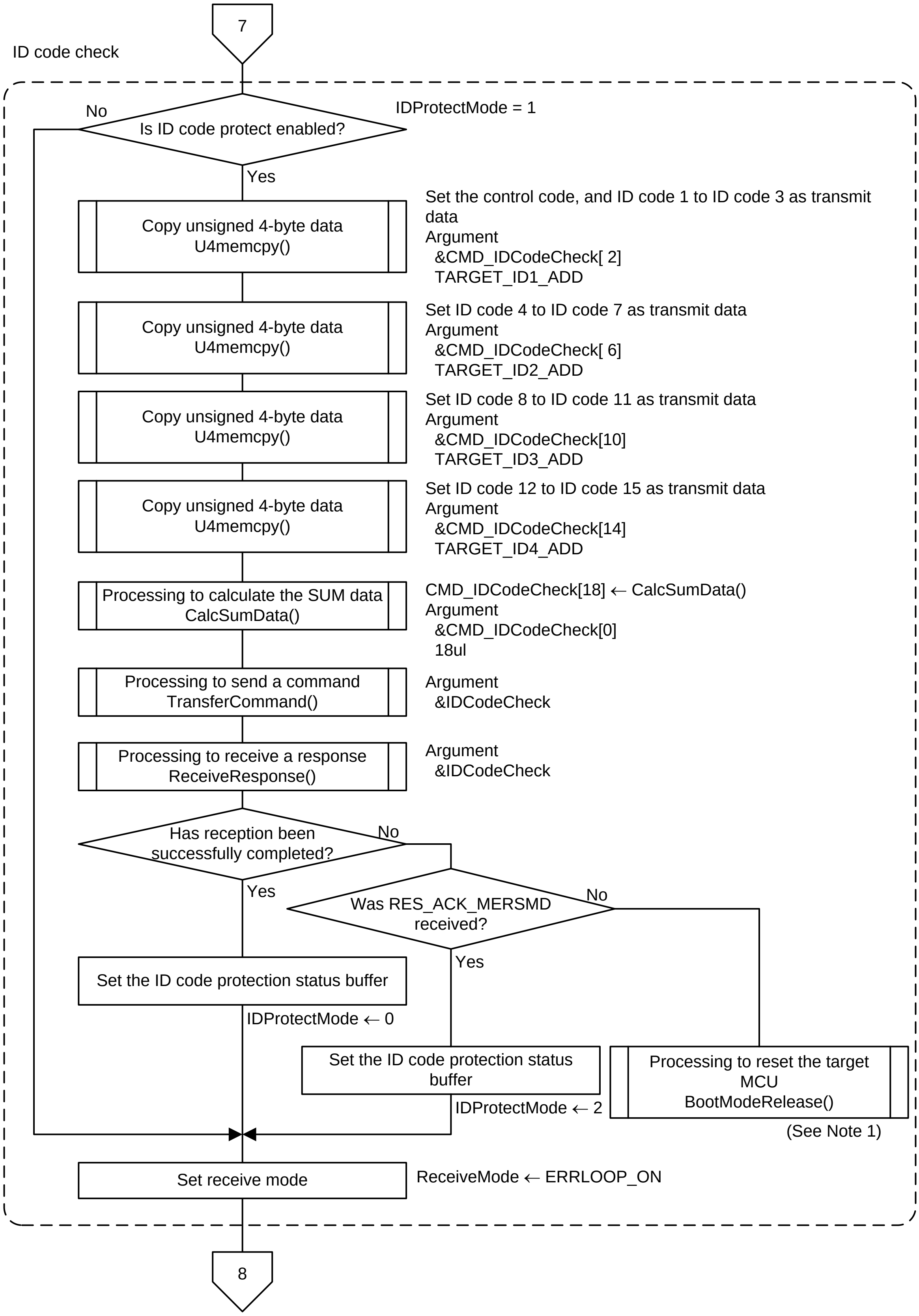


Figure 5.23 Main Processing and Communication Protocol Control



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

Figure 5.24 Main Processing and Communication Protocol Control

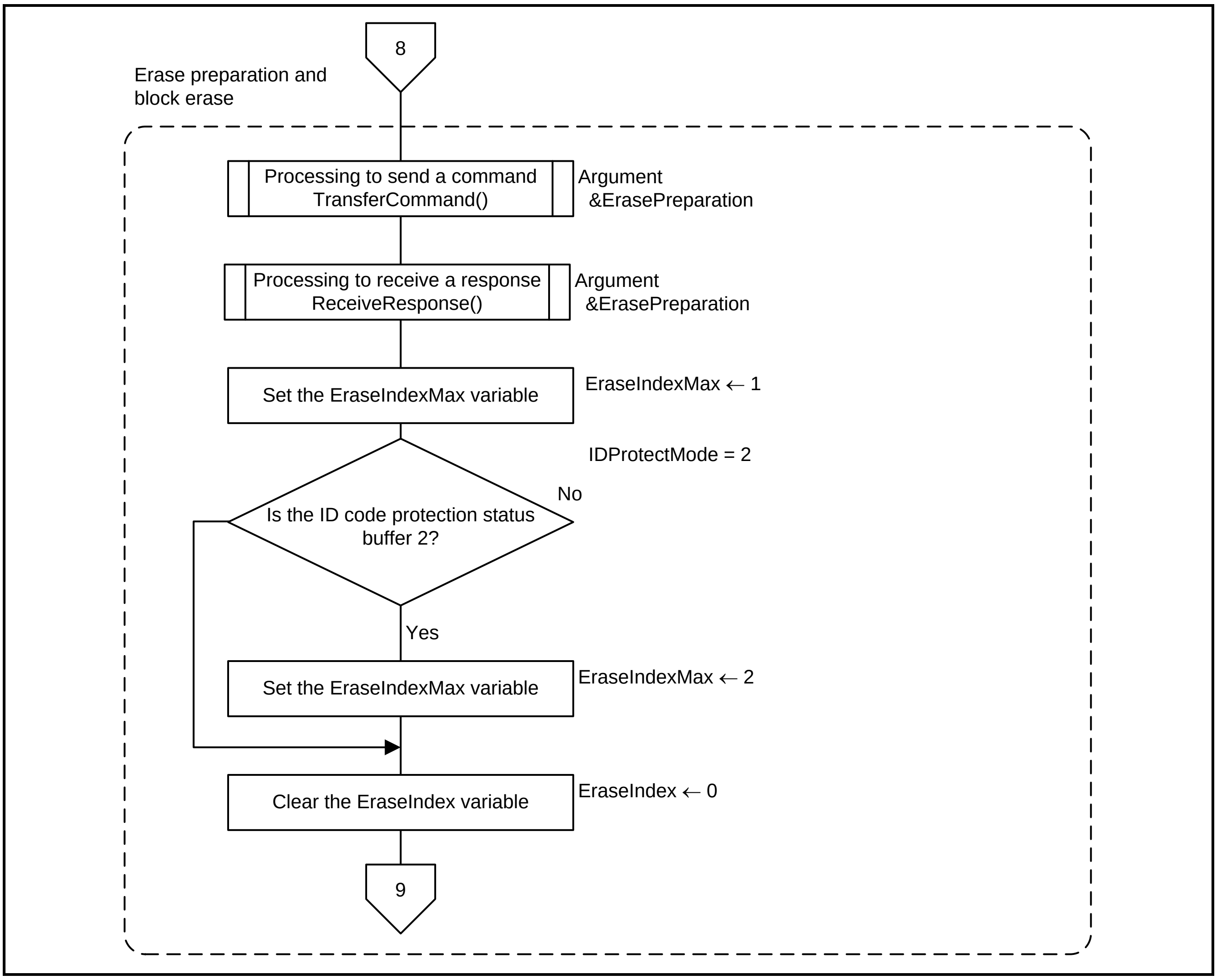


Figure 5.25 Main Processing and Communication Protocol Control

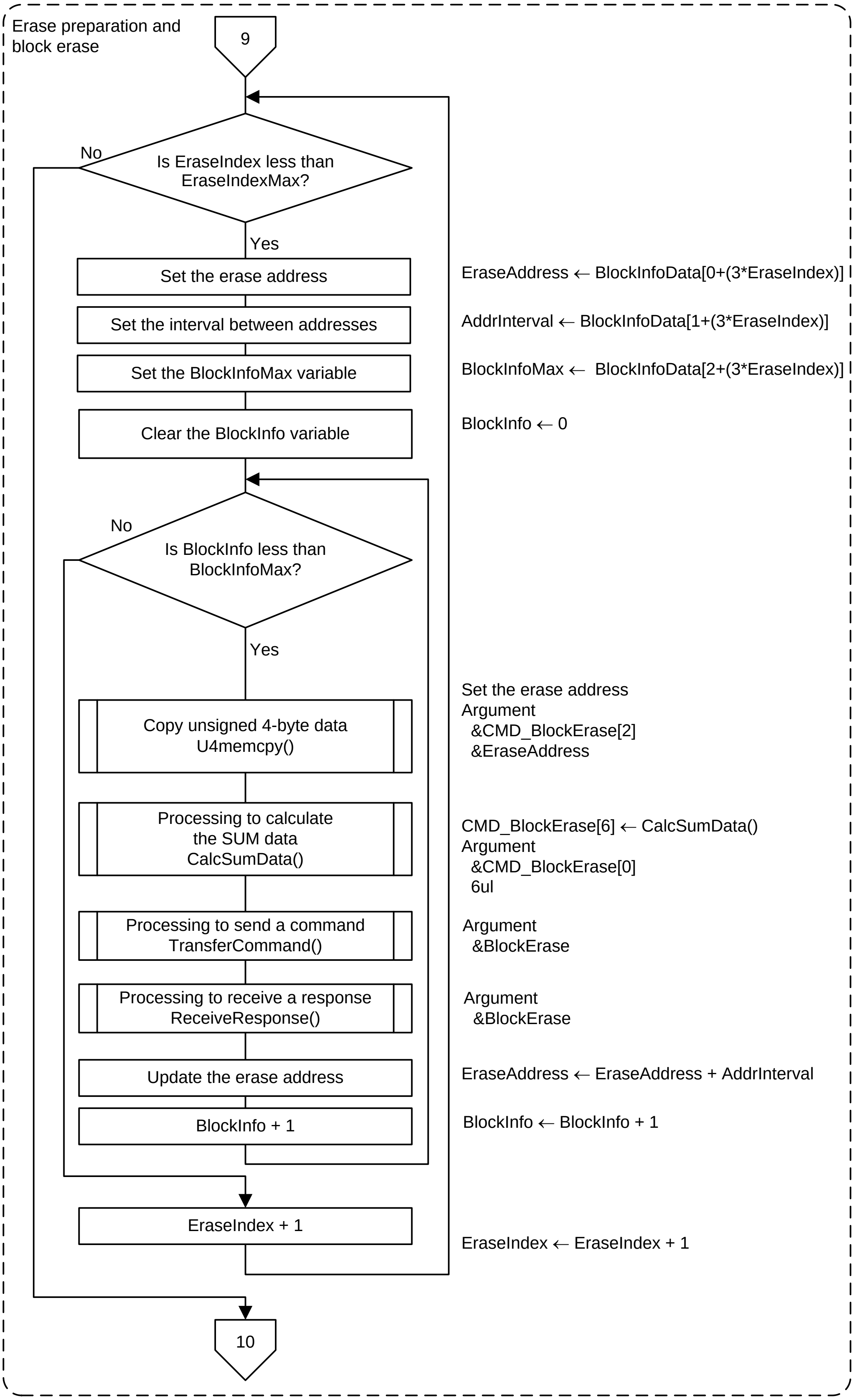


Figure 5.26 Main Processing and Communication Protocol Control

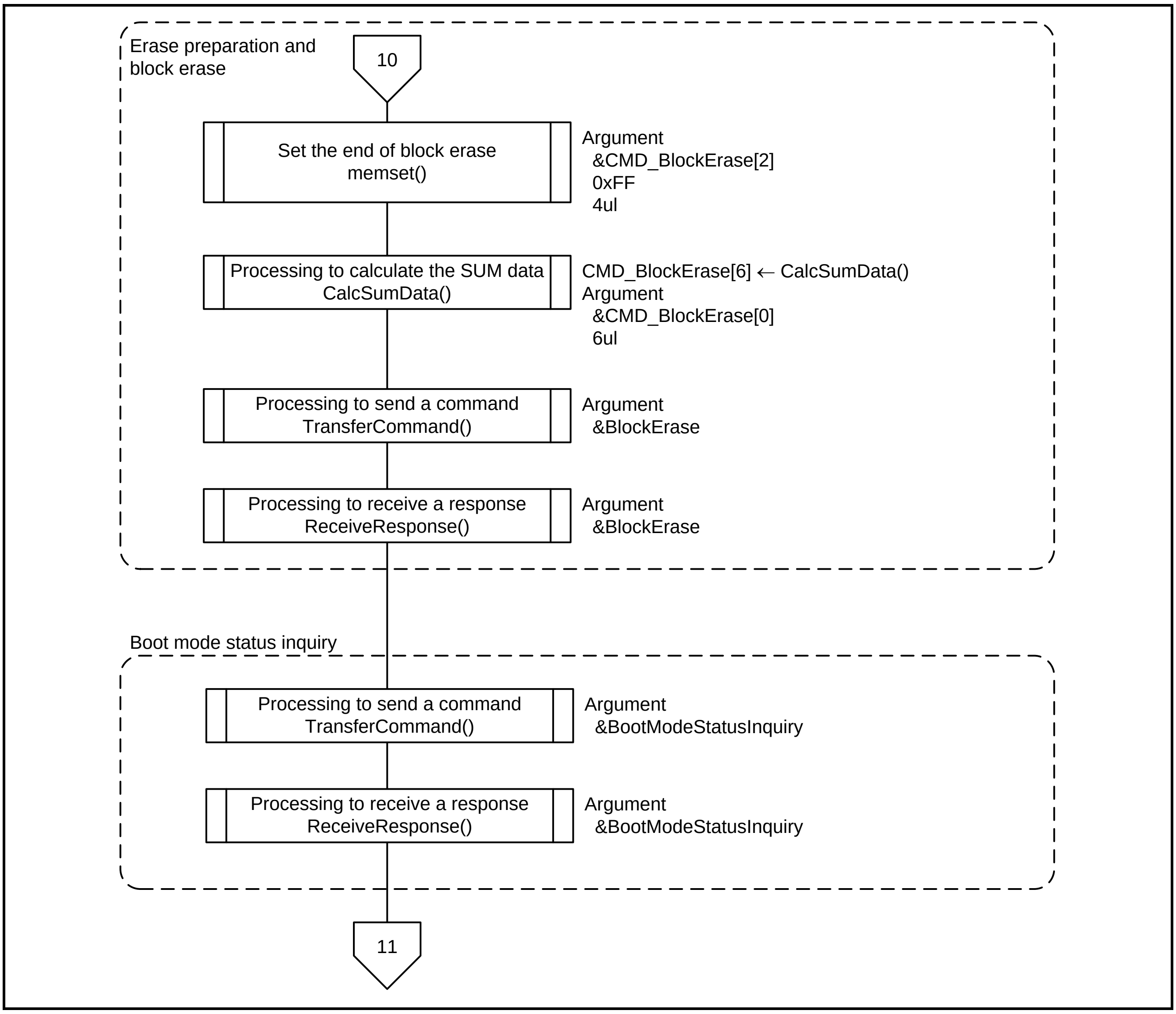


Figure 5.27 Main Processing and Communication Protocol Control

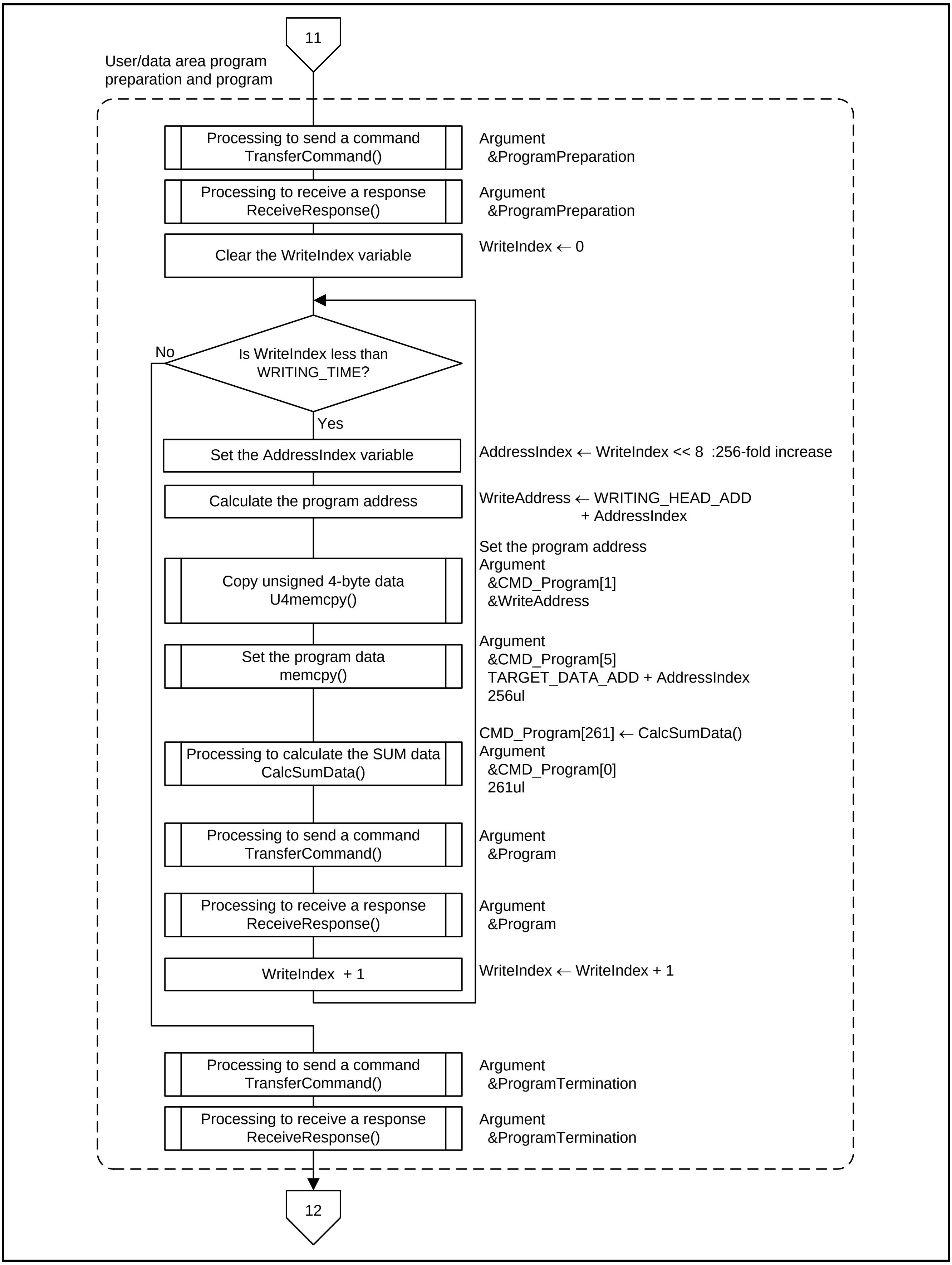


Figure 5.28 Main Processing and Communication Protocol Control

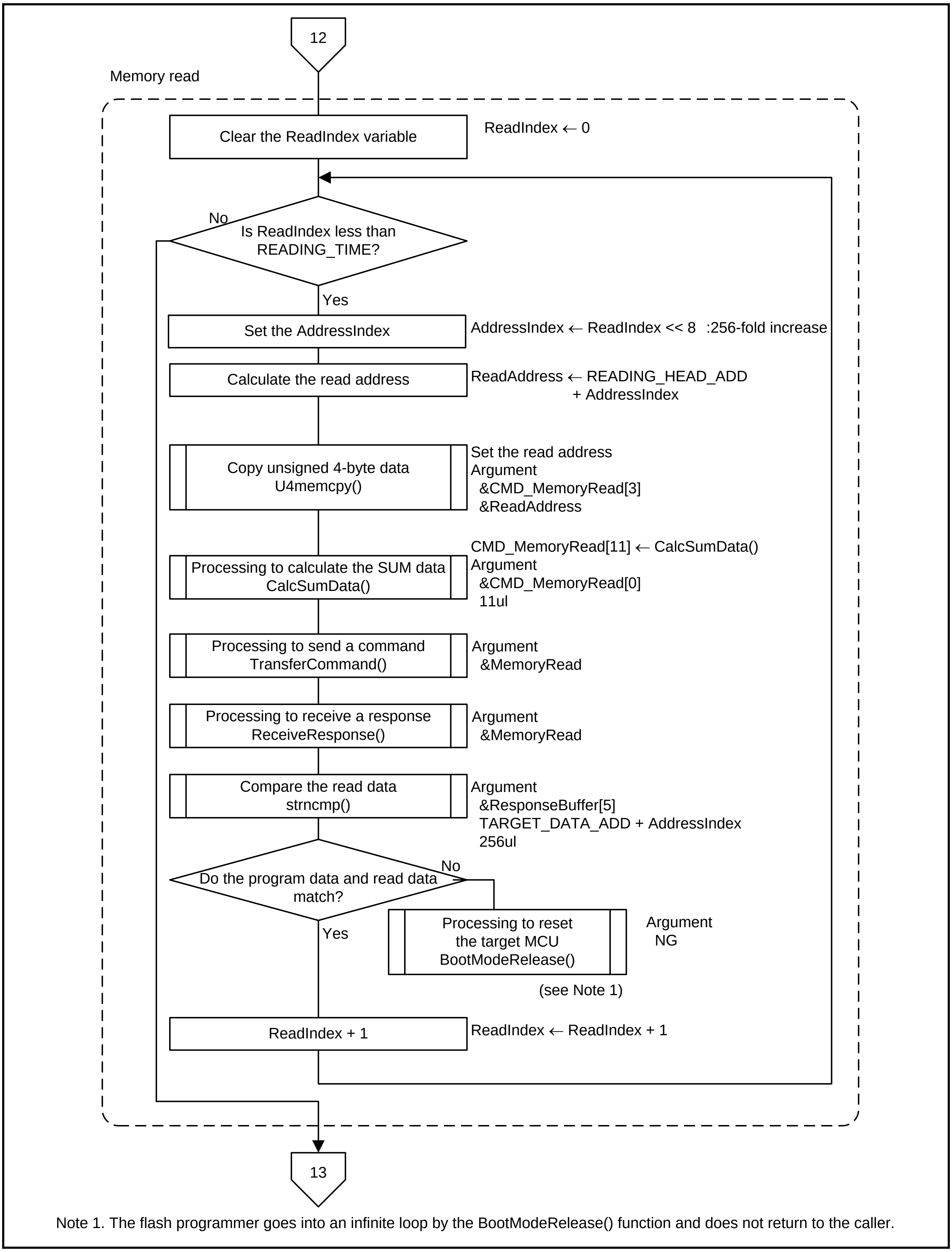
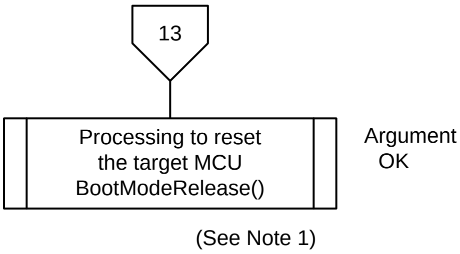


Figure 5.29 Main Processing and Communication Protocol Control



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

Figure 5.30 Main Processing and Communication Protocol Control

5.10.2 Initialization of the Peripheral Functions

Figure 5.31 shows the Initialization of the Peripheral Function.

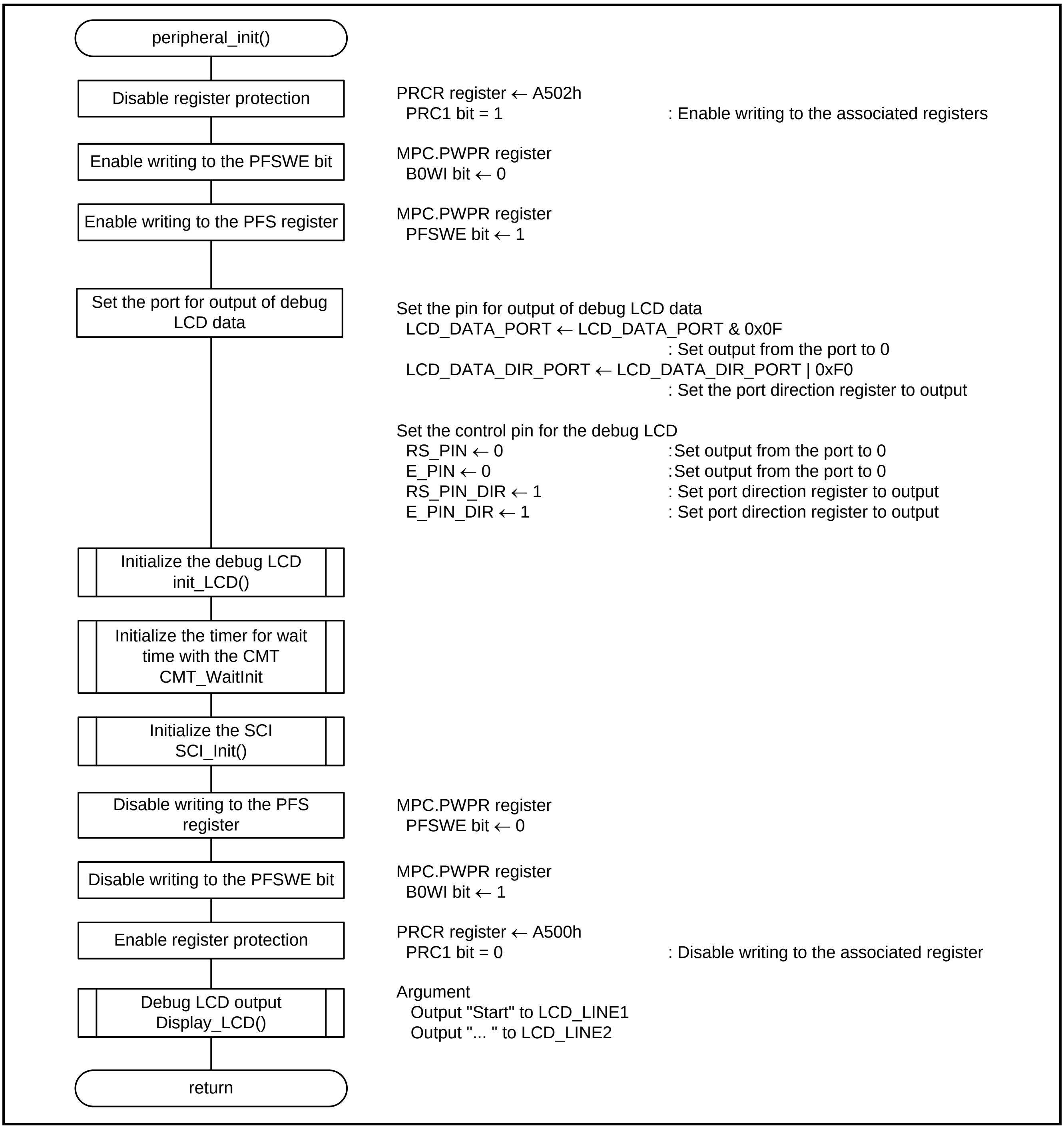


Figure 5.31 Initialization of the Peripheral Functions

5.10.3 Initialization of the Timer for Wait Time with the CMT

Figure 5.32 shows the Initialization of the Timer for Wait Time with the CMT.

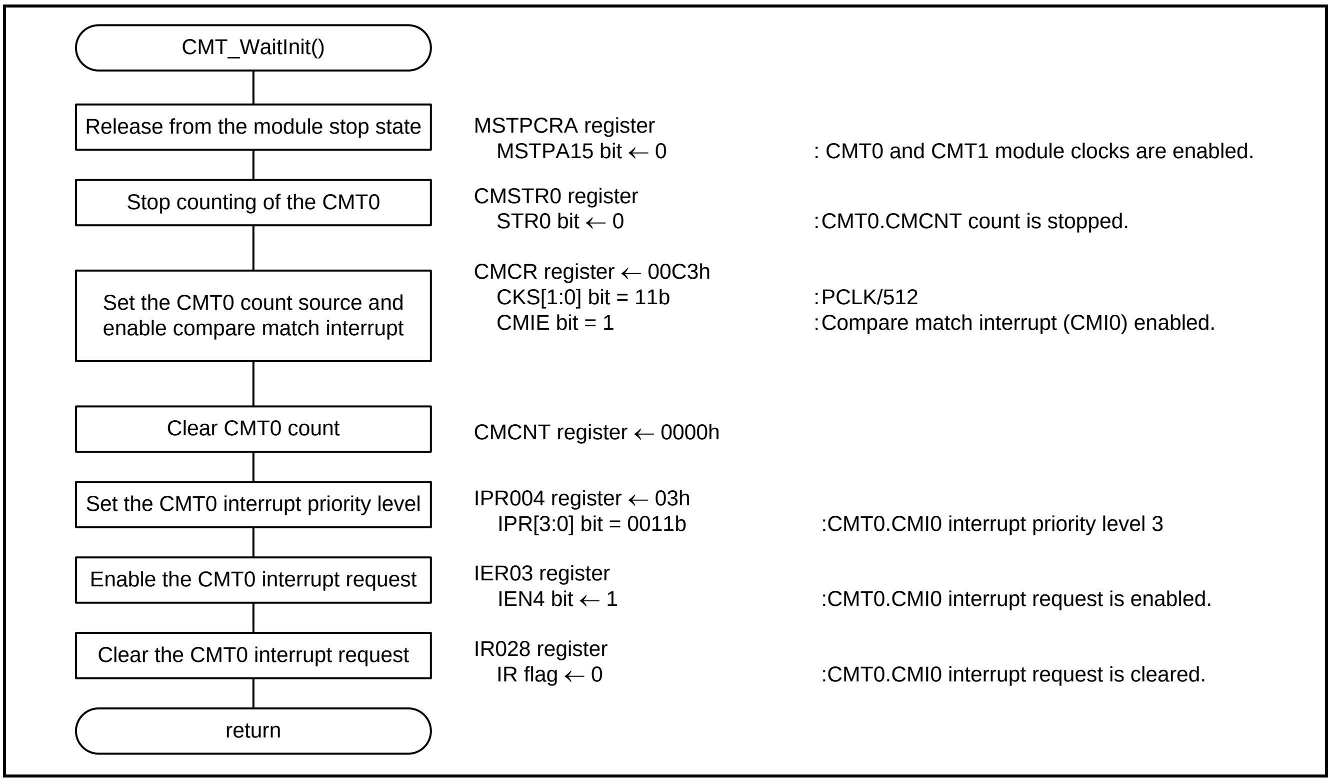


Figure 5.32 Initialization of the Timer for Wait Time with the CMT

5.10.4 Setting Wait Time with the CMT

Figure 5.33 shows Setting Wait Time with the CMT.

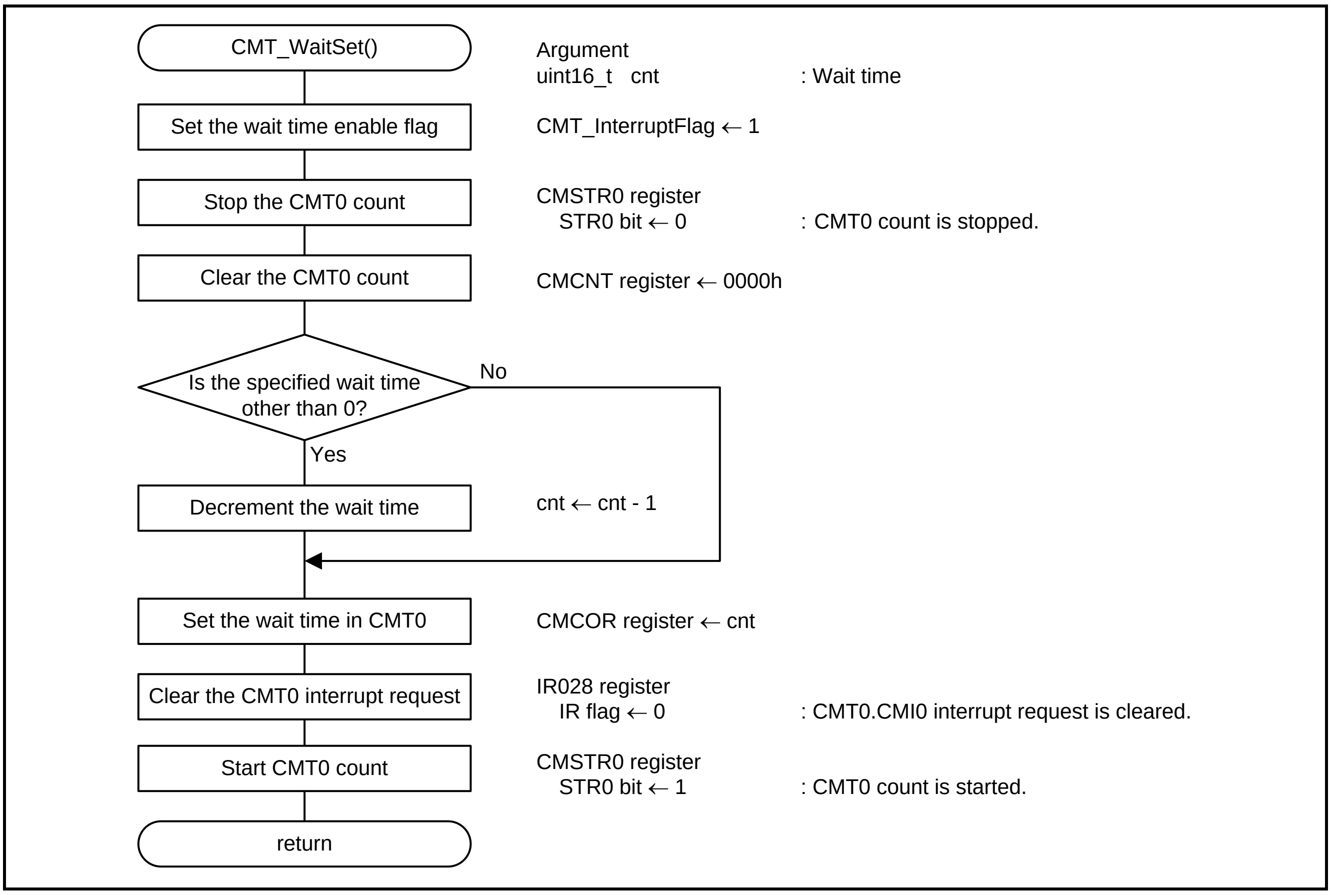


Figure 5.33 Setting Wait Time with the CMT

5.10.5 Wait Processing with the CMT

Figure 5.34 shows Wait Processing with the CMT.

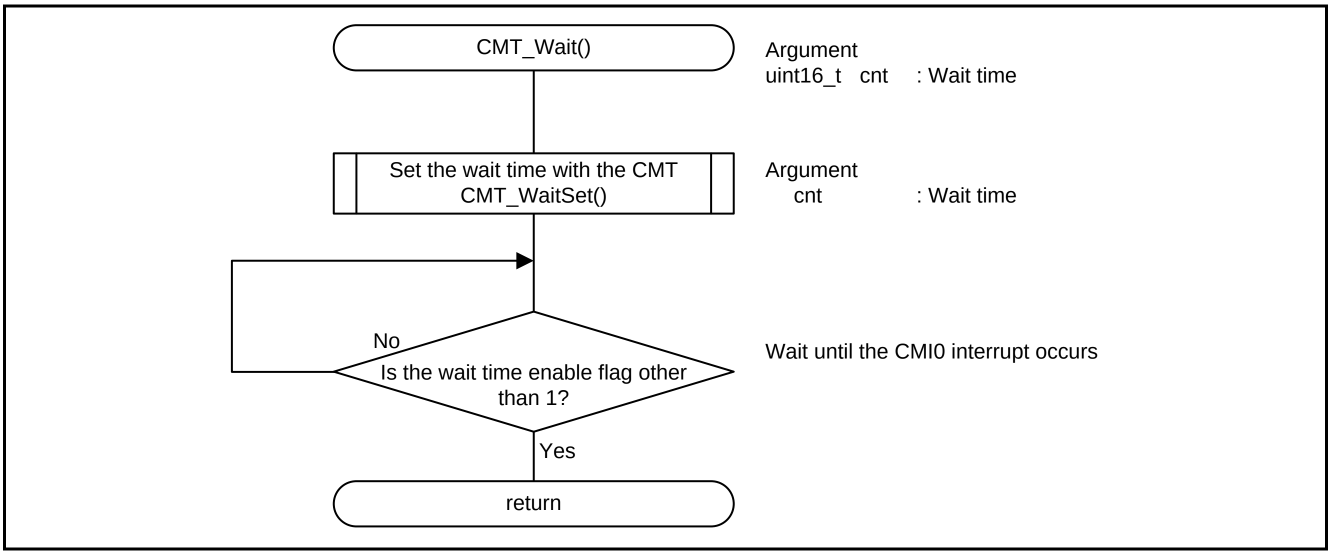


Figure 5.34 Wait Processing with the CMT

5.10.6 Interrupt Handling for CMI0 in CMT0

Figure 5.35 shows Interrupt Handling for CMI0 in CMT0.

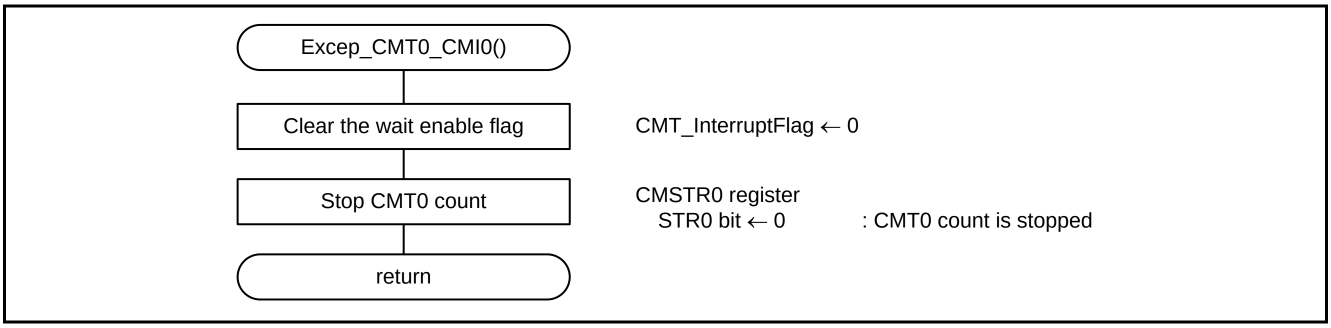


Figure 5.35 Interrupt Handling for CMI0 in CMT0

5.10.7 Initialization of the SCI

Figure 5.36 shows SCI Initialization.

SCI_Init		
Set port mode	TXDn_PMR ← 0	:TXDn: Used as a general I/O port
	RXDn_PMR ← 0	: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	:TXDn: Output
	RXDn_PDR ← 0	: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] = 00b	: On-chip baud rate generator
	TEIE = 0	: TEI interrupt request is disabled
	RE = 0	: Serial reception is disabled
	TE = 0	: Serial transmission is disabled
	RIE = 0	: RXI and ERI interrupt requests are disabled
	TIE = 0	: TXI interrupt request is disabled
Select the pin function	MPC.TXDnPFS register ← TXDnPFS_SELECT	:TXDn
	MPC.RXDnPFS register ← RXDnPFS_SELECT	:RXDn
Set the format for transmission and reception	SCIn.SMR register ← 00h	
	CKS[1:0] = 00b	:PCLK clock
	MP = 0	:Multi-processor communications function is disabled
	STOP = 0	:1 stop bit
	PE = 0	:No parity
	CHR = 0	:Selects 8 bits as the data length
	CM = 0	:Asynchronous mode
	SCIn.SCMR register ← F2h	
	SMIF = 0	:Serial communications interface mode
	SINV = 0	:TDR contents are transmitted as they are. Receive data is stored as it is in RDR
	SDIR = 0	:Transfer with LSB first
	SCIn.SEMR register ← 10h	
	ABCS = 1	: Selects 8 base clock cycles for 1-bit period
	NFEN = 0	: 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	:SCIn.RXIn interrupt not requested
	IR_SCIn_TXIn ← 0	:SCIn.TXIn interrupt not requested
Enable transmission/reception and SCIn interrupt request	SCIn.SCR register ← F0h	
	RE = 1	: Serial reception is enabled
	TE = 1	: Serial transmission is enabled
	RIE = 1	: RXI and ERI interrupt requests are enabled
	TIE = 1	: A TXI interrupt request is enabled
Set port mode	TXDn_PMR ← 1	:TXDn: Used for peripheral functions
	RXDn_PMR ← 1	:RXDn: Used for peripheral functions
return		

Figure 5.36 SCI Initialization

5.10.8 Processing to Change the SCI Bit Rate

Figure 5.37 shows Processing to Change the SCI Bit Rate.

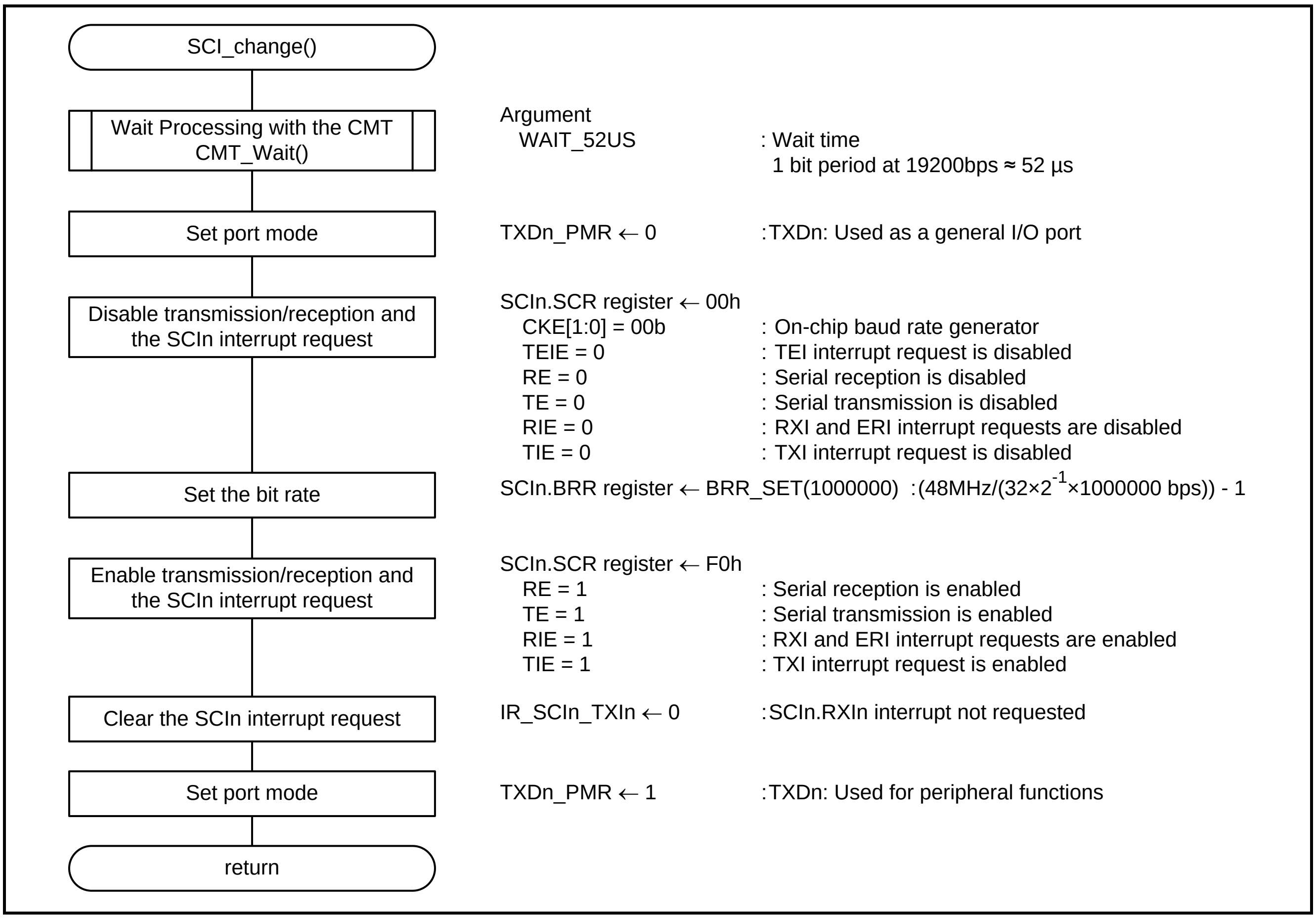


Figure 5.37 Processing to Change the SCI Bit Rate

5.10.9 Processing to Calculate the SUM Data

Figure 5.37 shows Processing to Calculate the SUM Data.

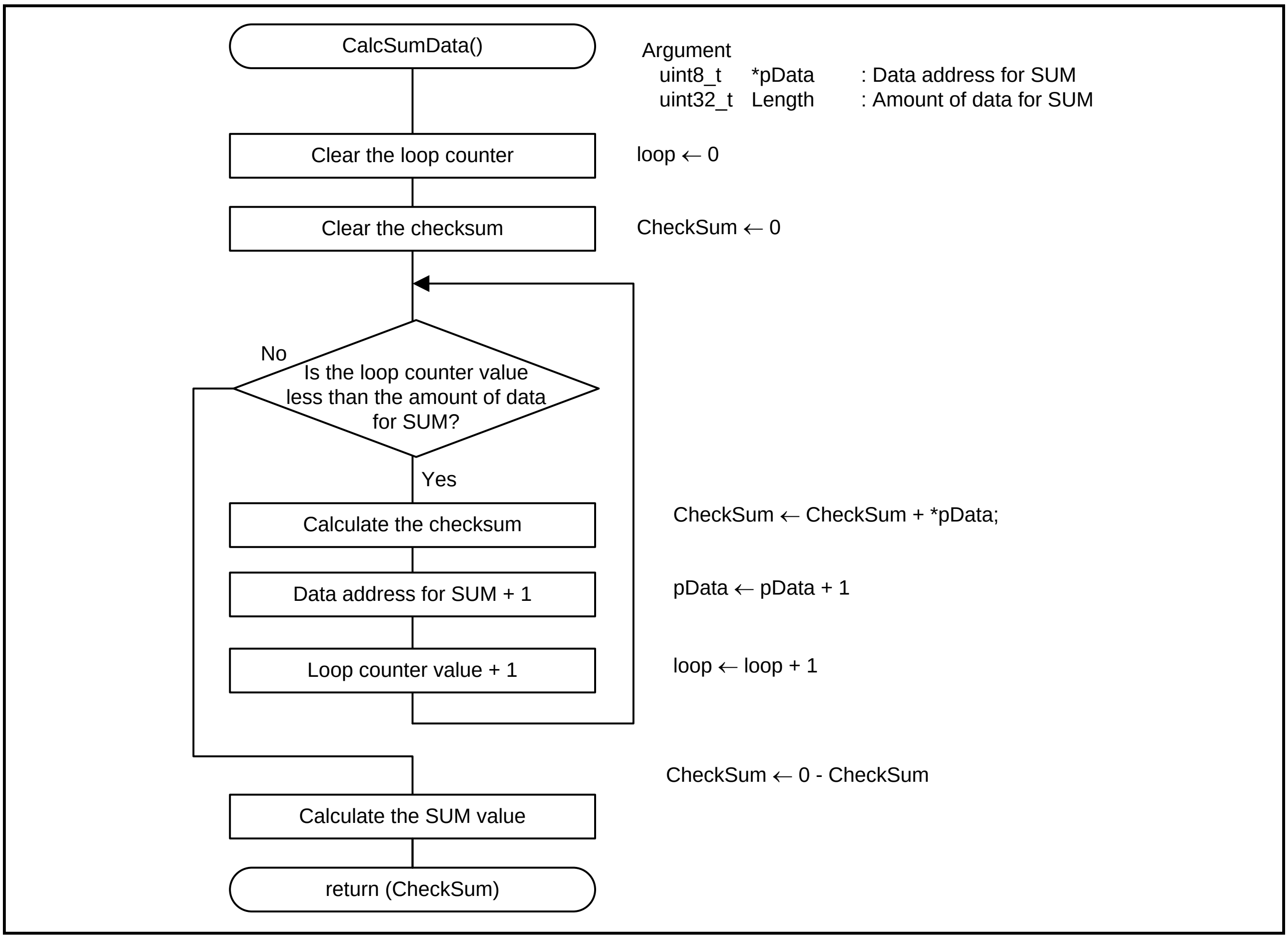


Figure 5.38 Processing to Calculate the SUM Data

5.10.10 Processing to Start the Target MCU in Boot Mode

Figure 5.39 shows Processing to Start the Target MCU in Boot Mode.

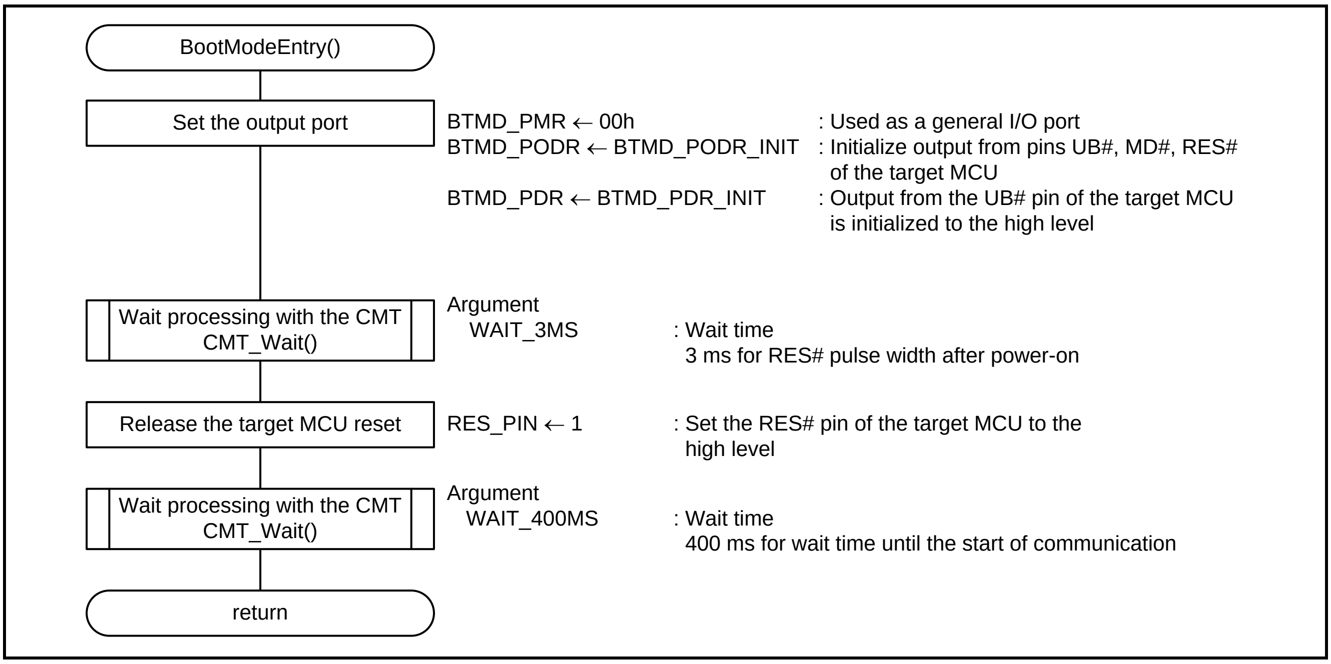


Figure 5.39 Processing to Start the Target MCU in Boot Mode

5.10.11 Processing to Reset the Target MCU

Figure 5.40 shows Processing to Reset the Target MCU.

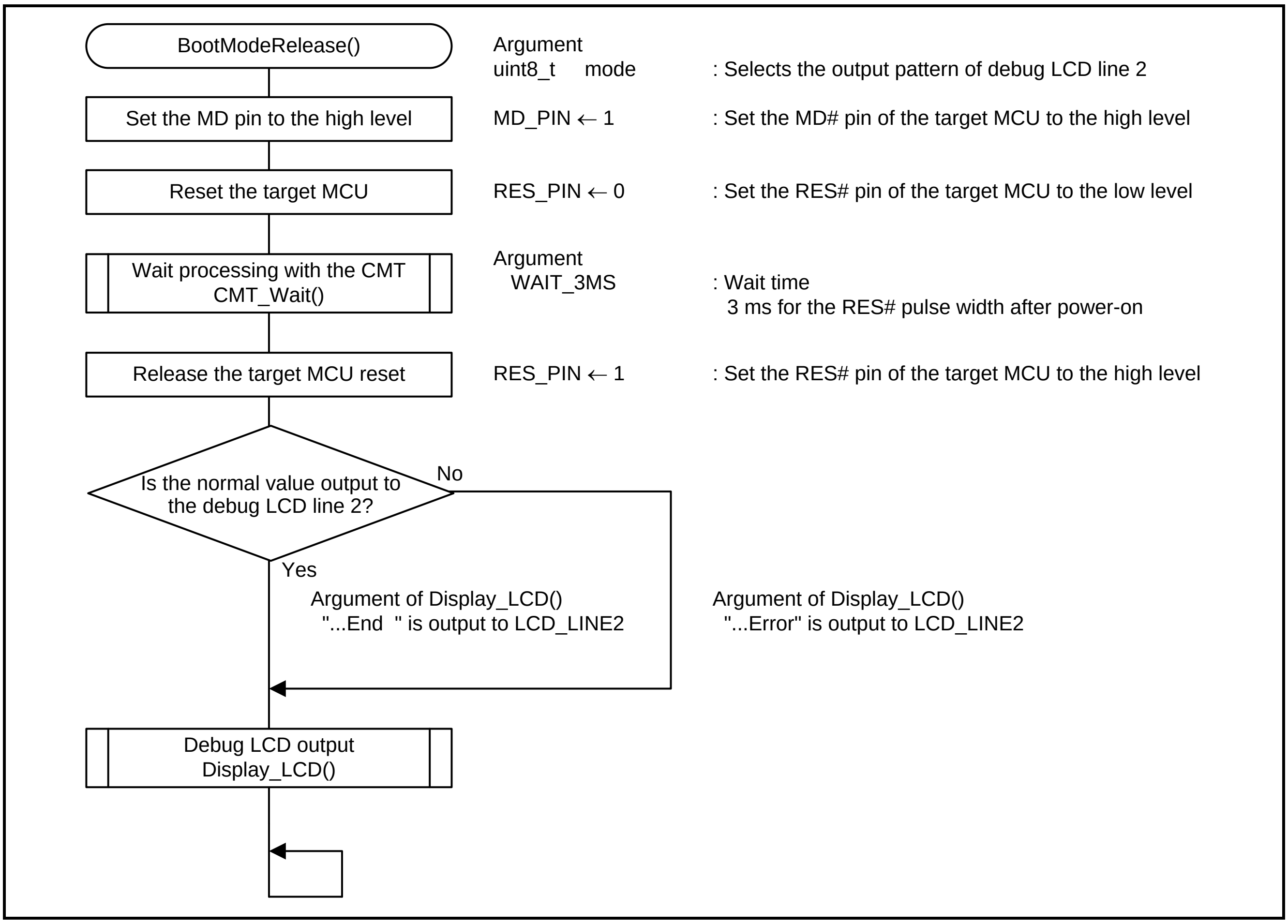


Figure 5.40 Processing to Reset the Target MCU

5.10.12 Processing to Send a Command

Figure 5.41 shows Processing to Send a Command.

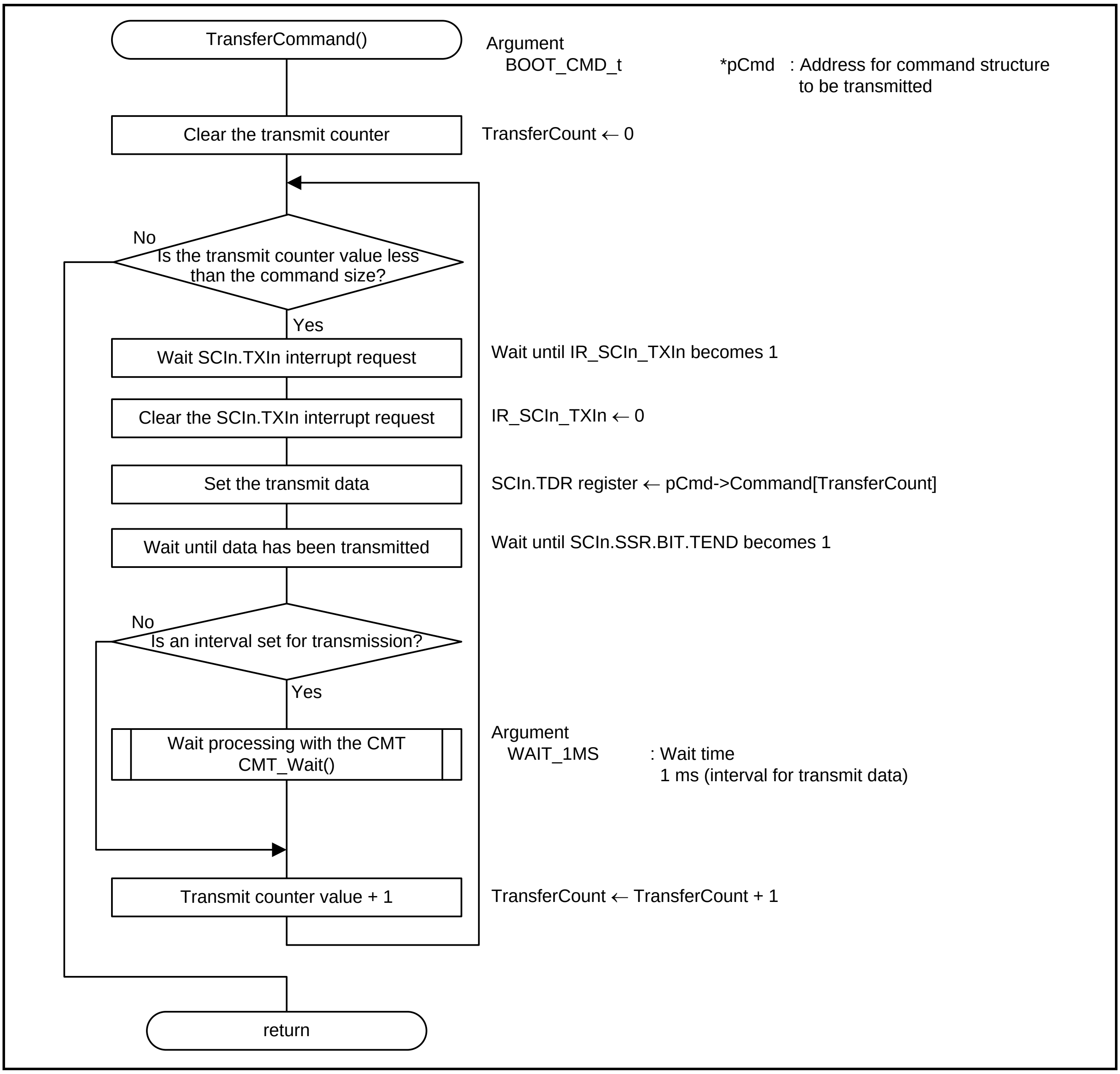


Figure 5.41 Processing to Send a Command

5.10.13 Processing to Receive a Response

Figure 5.42 to Figure 5.45 show Processing to Receive a Response.

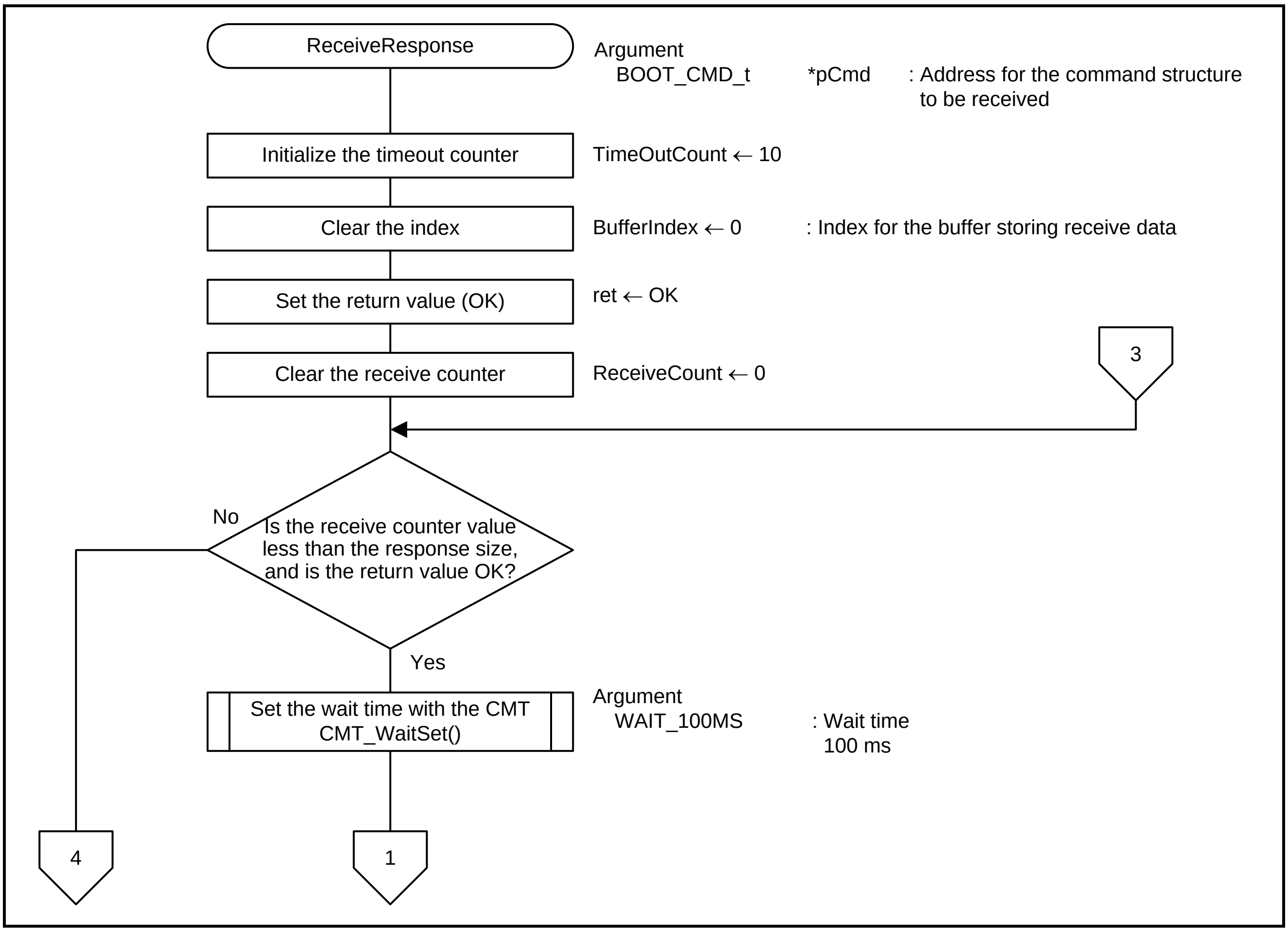
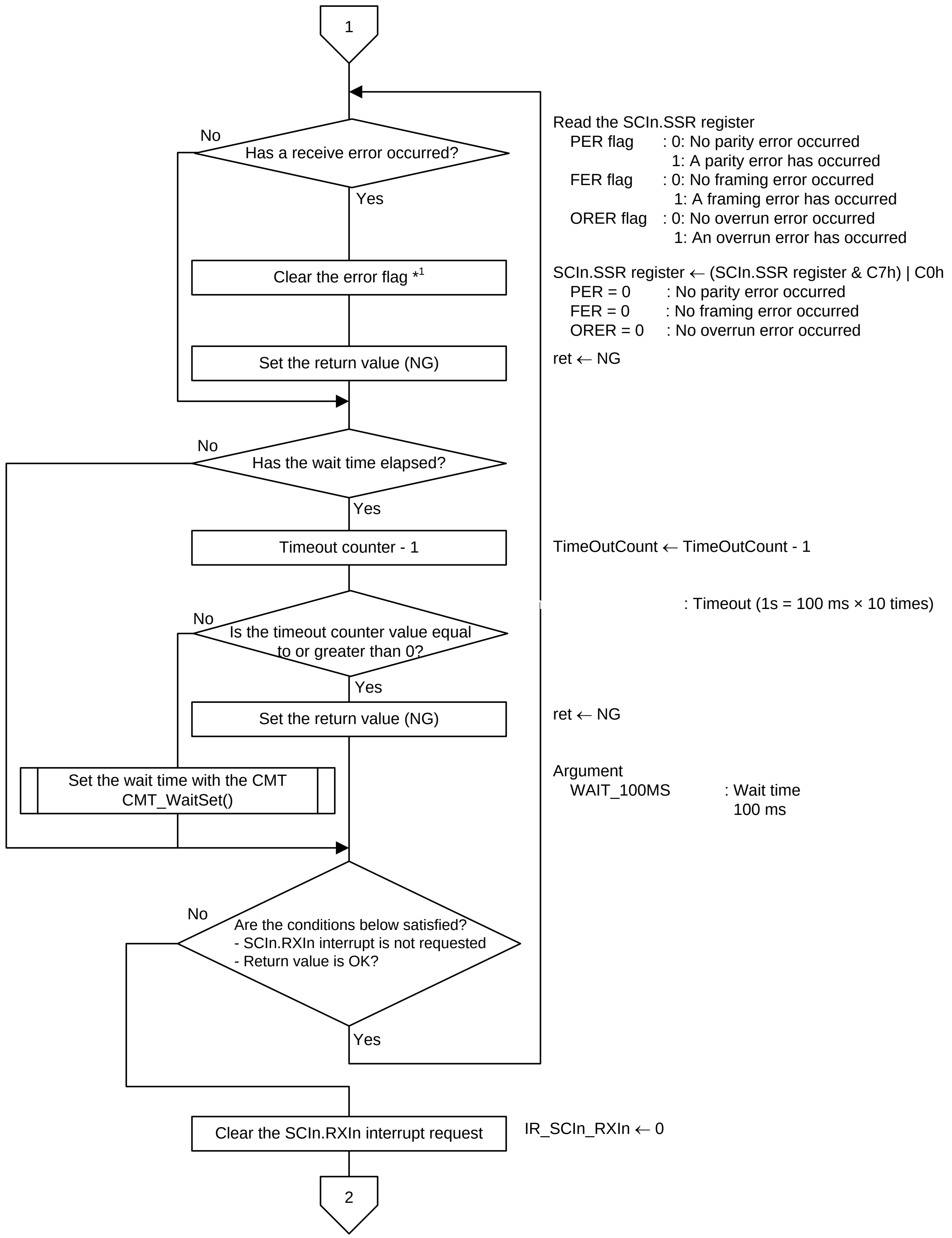


Figure 5.42 Processing to Receive a Response



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 5.43 Processing to Receive a Command

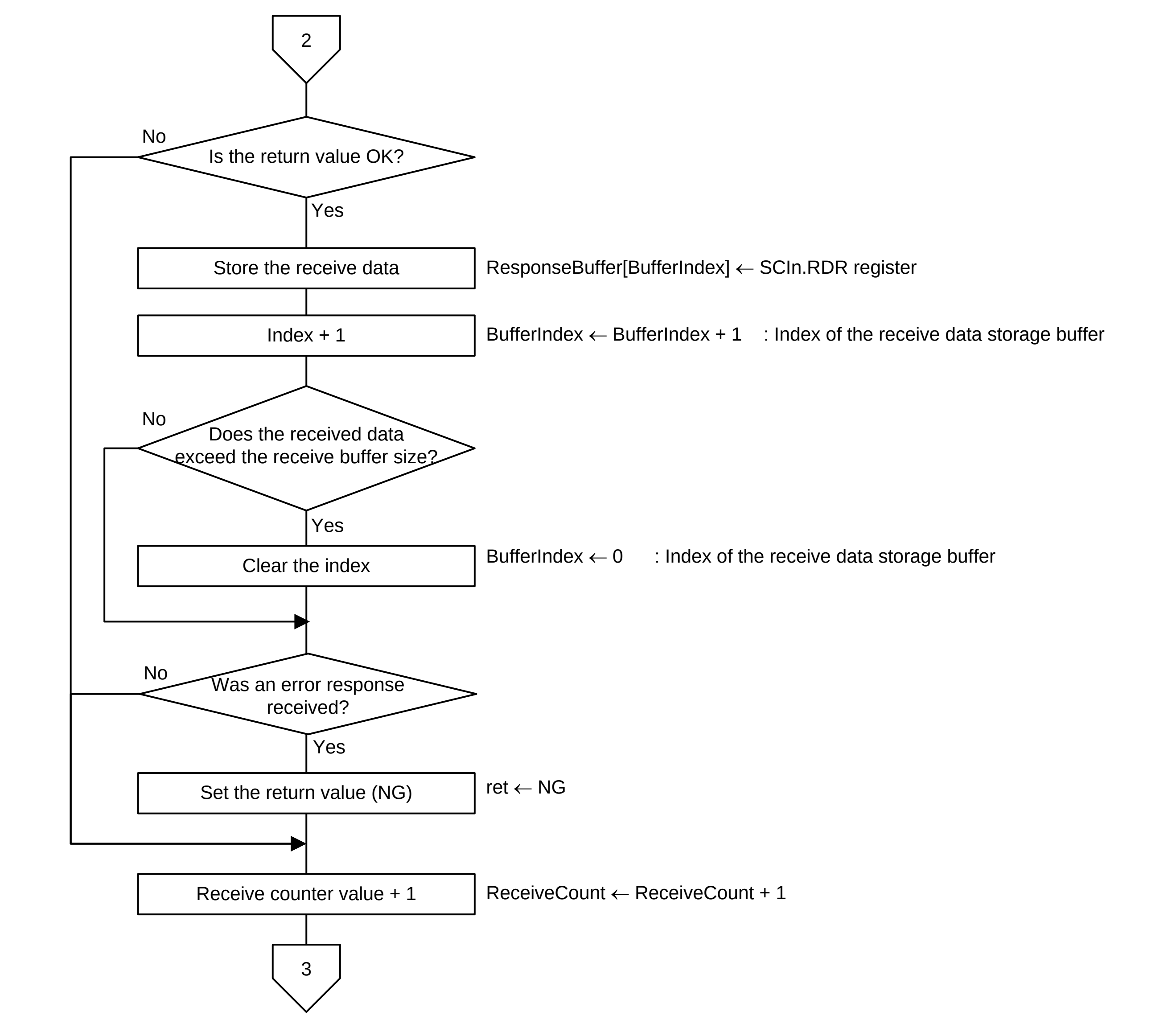
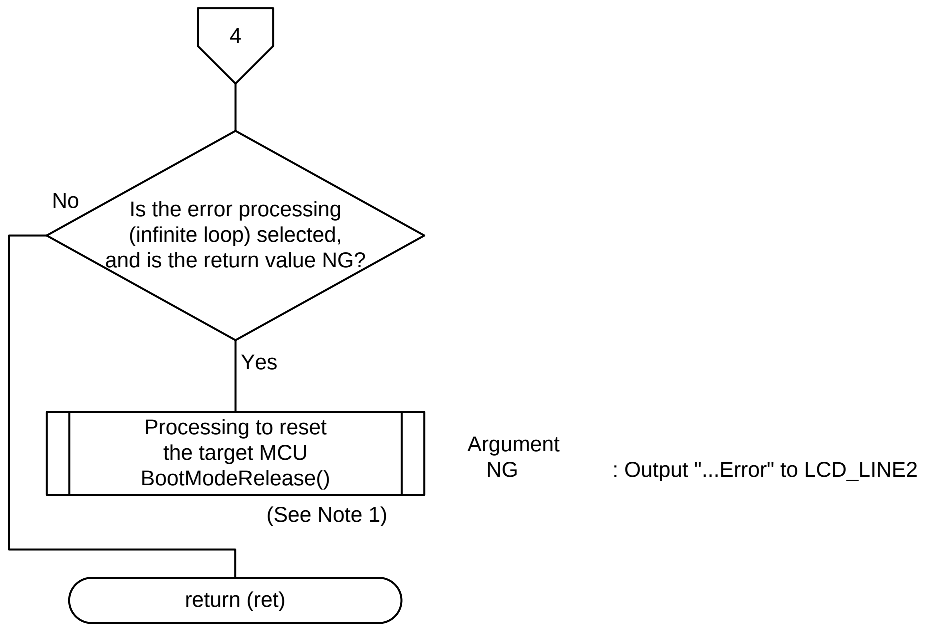


Figure 5.44 Processing to Receive a Command



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

Figure 5.45 Processing to Receive a Command

5.10.14 Copying Unsigned 4-Byte Data

Figure 5.46 show Copying Unsigned 4-Byte Data.

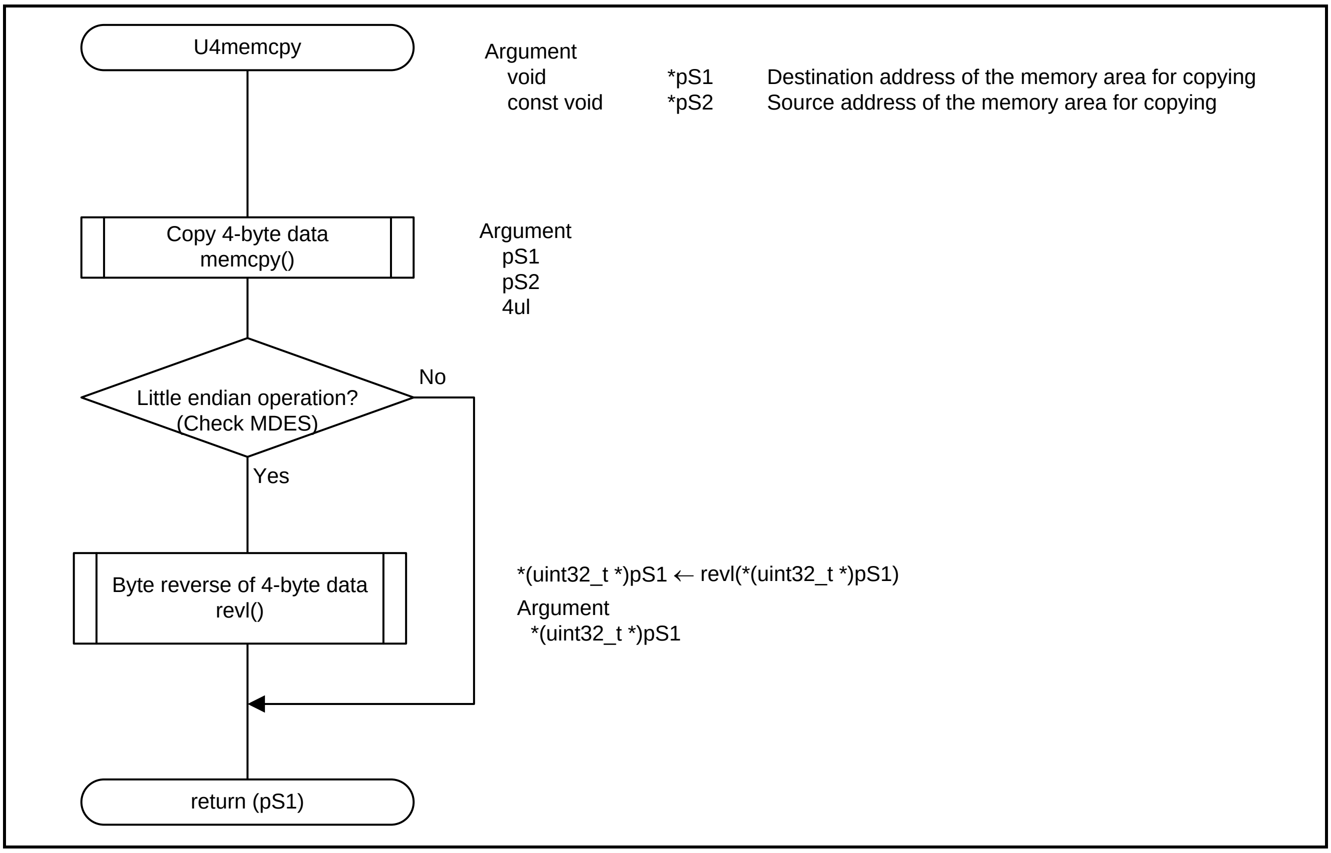


Figure 5.46 Copying Unsigned 4-Byte Data

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User’s Manual: Hardware

RX110 Group User's Manual: Hardware Rev.1.00 (R01UH0421EJ)
RX111 Group User’s Manual: Hardware Rev.1.10 (R01UH0365EJ)
RX63N Group, RX631 Group User's Manual: Hardware Rev.1.80 (R01UH0041EJ)
The latest versions 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 (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	RX100 Series Application Note RX100 Series Flash Programmer (SCI) Using the Renesas Starter Kit+ for RX63N

Rev.	Date	Description	
		Page	Summary
1.00	June 02, 2014	—	First edition issued
1.10	Aug, 20, 2020	—	Update the toolchain version

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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.