RX62N Group, RX621 Group

On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

R01AN0184EJ0102
Rev.1.02
Feb 29, 2012

Introduction

This application note describes programming and erasing the flash memory for code storage (user MAT) of a RX62N and RX621 Group MCU by using the target erasure block number, programming data size, and programming data transmitted by clock synchronous serial communication from another RX62N and RX621 Group MCU, as described in "RX62N and RX621 Group: On-chip Flash Memory Reprogramming in the User Boot Mode (Master)" (R01AN0185EJ).

For the procedures for sending the erase block number, programming data size, and programming data through clock synchronous communication, refer to "On-chip Flash Memory Reprogramming in the User Boot Mode (Master) for the RX62N and RX621 groups" (R01AN0185EJ).

Target Device

RX62N group and RX621 group

This program is also available for the other RX families that have the similar I/O registers (peripheral device control registers) as the RX62N and RX621 groups. Note, however, that parts of functionalities have been modified or enhanced. Check these changes in the relevant manuals. Extensive evaluation tests should be conducted when using this application note.

Contents

1. Specification

- This application note exemplifies the procedures for programming and erasing user MATs in user boot mode using the R5F562N8BDBG of the RX62N group.
- The slave receives the erase block number, programming data size, and programming data from the master through clock synchronous communication and carries out the process of programming or erasing the given user MAT.
- The clock synchronous communication between the master and slave is accomplished using the SCI channel 2 (SCI2) module.
- The major clock synchronous communication specifications are: 2.4 Mbps bit rate, 8 data bits, and LSB first. The transfer clock is transmitted from the master device.
- In this application note, the slave erases the specified erase block (EB30: 16K bytes) and programs the received 8K bytes (256 bytes × 32) of programming data into the erase block EB30 starting at its start address.
- The slave and master use a handshake to control their communications. The slave uses an I/O port (P01) to send out an Assert (low) in the busy state and a Negate (high) when the busy state is reset. The master receives the output from the slave via an external interrupt pin (IRQ0-A) and starts the next transmission sequence when a rising edge is input.
- When the user MAT erasing/programming process is completed normally, the slave notifies the normal termination using the four LEDs connected to its I/O ports. If an error occurs during communication with the master or during programming erasing processing, the slave also notifies the error with these LEDs.

Figure 1 shows the major specifications relevant to this application note.

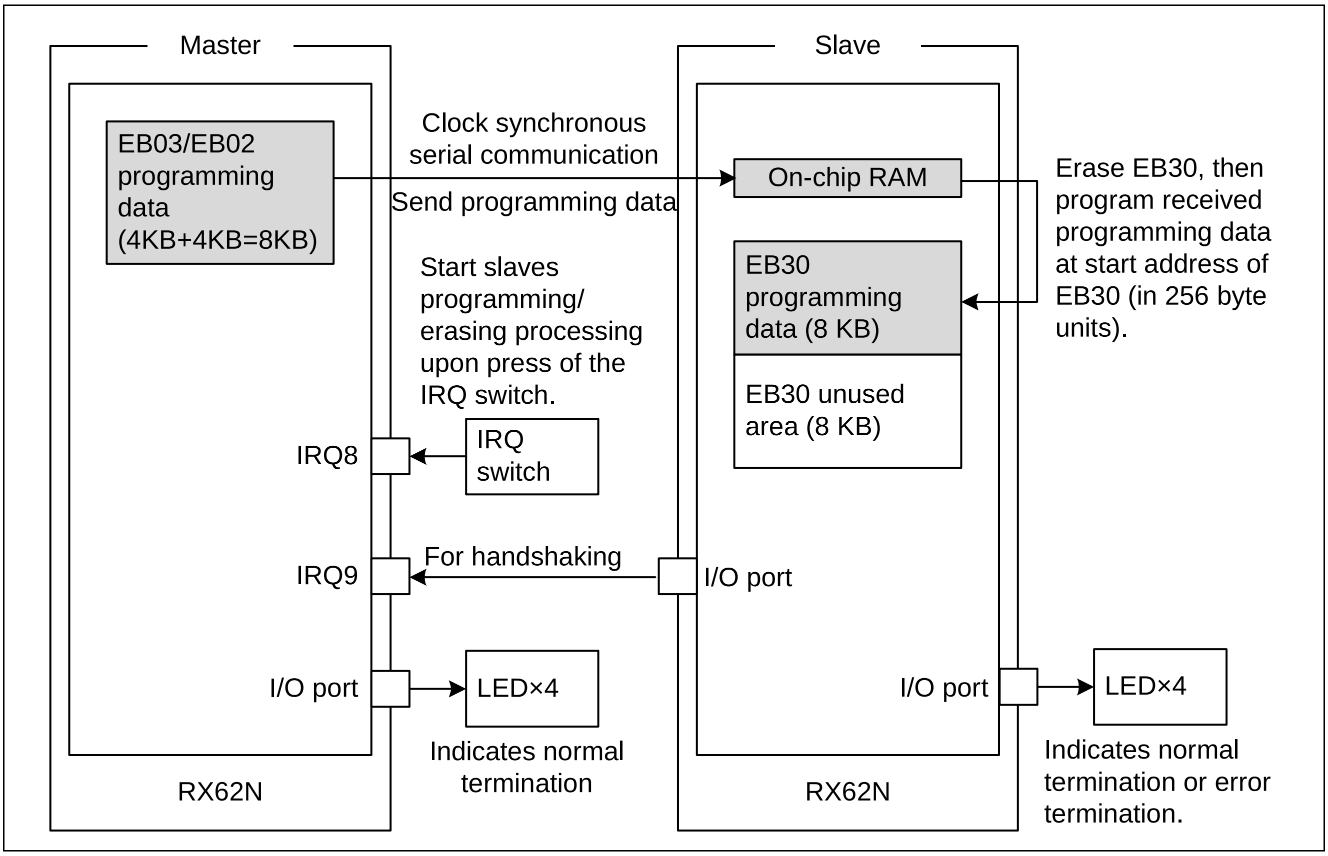

Figure 1 Specification Outline

Figure 2 shows the hardware configuration diagram for the slave device referred to in this application note.

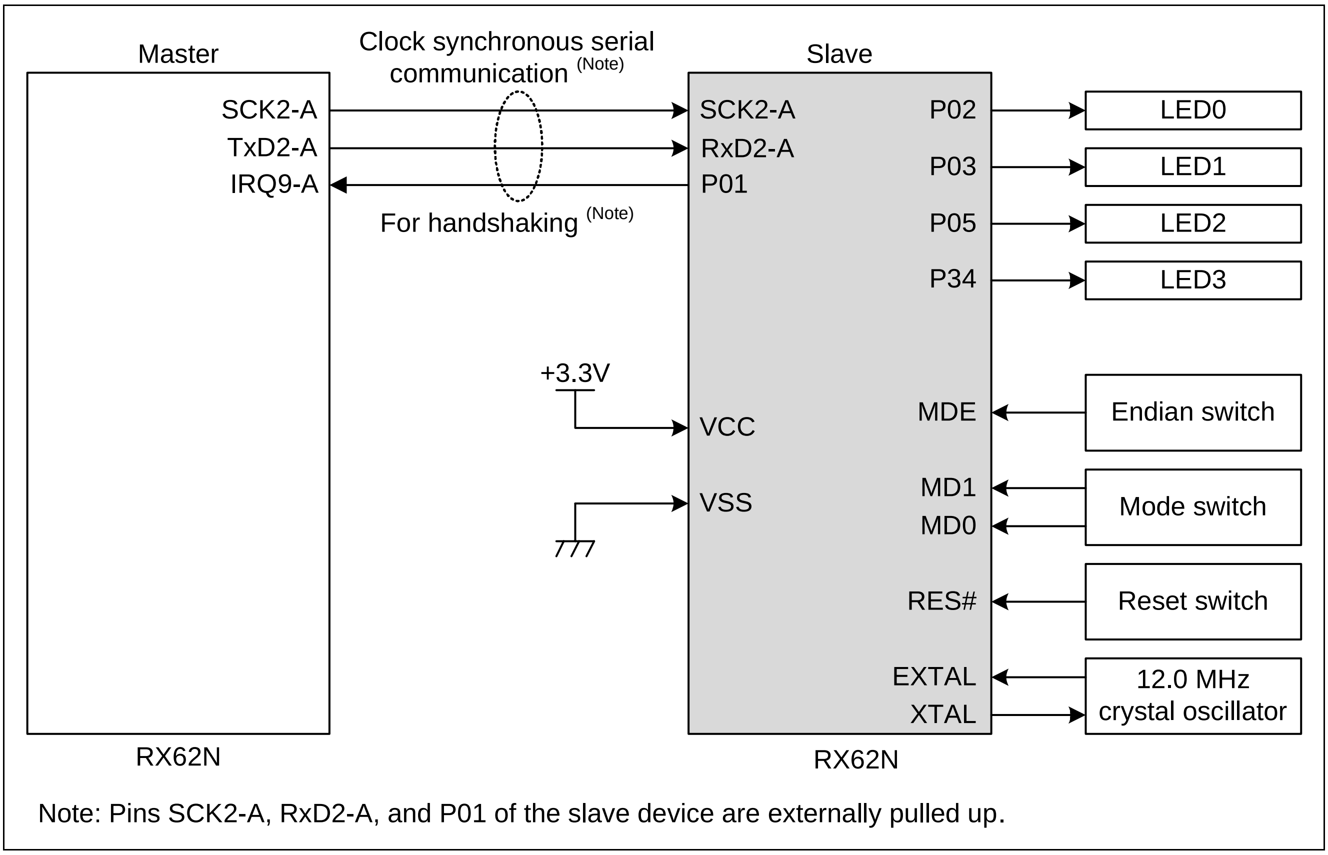

Figure 2 Slave Hardware Configuration Diagram

2. Operating Environment

Table 1 summarizes the major characteristics of the environment in which the slave is run.

Table 1 Slave Operating Environment

Item	Description
Device	RX62N group: R5F562N8BDBG (ROM size: 512 K bytes, RAM size: 96 K bytes)
Board	Renesas starter kit (R0K5562N0S000BE)
Power voltage	5.0 V (CPU operating voltage is 3.3 V.)
Input clock	12.0 MHz (ICLK = 96 MHz, PCLK = 48 MHz, BCLK = 24 MHz)
Operating temperature	Room temperature
High-performance Embedded Workshop	Version 4.07.00.007
Toolchain	RX Standard Toolchain (V.1.0.0.0)
FDT	V.4.06 Release 00

3. Functions Used

- Clock Generation Circuit
- Low Power Consumption
- Interrupt Controller Unit (ICU)
- I/O ports
- Serial communications interface
- ROM (Flash Memory for Code Storage)

See "User's Manual" listed in section 7, Reference Documents, for details.

4. Description of Operation

4.1 Setting the Operating Mode

In the example given in this application note, the slave mode pin MD1 is set to 1 and mode pin MD0 to 0 to set the operating mode to user boot mode and the ROME bit of the system control register 0 (SYSCR0) is set to 1 to enable the on-chip ROM, and the EXBE bit of the SYSCR0 register is set to 0 to disable the external bus.

The user boot MAT for the RX62N stores the program that is started in USB boot mode (the mode pin settings are the same as those for the user boot mode). To program a user-supplied program in the user boot MAT, erase the user boot MAT in boot mode using a flash memory programming tool (e.g., FDT) before carrying out the programming process.

The slave is activated in user boot mode from the user boot MAT.

Table 2 summarizes the operating mode settings for the slave used in the example given in this application note.

Table 2 Slave Operating Mode Settings

Mode Pin		SYSCR0 Register		Operating Mode	On-chip ROM	External Bus
MD1	MD0	ROME	EXBE			
1	0	1	0	User boot mode	Enabled	Disabled

Note: The SYSCR0 register should never be set up during program execution since the ROME and EXBE bits of the SYSCR0 register are initialized as follows: SYSCR0.ROME = 1, SYSCR0.EXBE = 0

4.2 Setting up the Clocks

The evaluation board referred to in this application note is provided with a 12.0 MHz crystal oscillator.

Accordingly, the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK) are set to ×8 (96 MHz), ×4 (48 MHz), and ×2 (24 MHz), respectively, in the example given in this application note.

4.3 Setting up Endian

This application note is compatible with both of big endian and little endian. The endian settings that can be set up by hardware (MDE pin) are listed in table 3. The endian settings of the master and slave must be identical.

Table 3 Endian Settings (Hardware)

MDE Pin	Endian
0	Little endian
1	Big endian

Table 4 lists the endian settings that can be set up using a compiler option.

Table 4 Endian Settings (Compiler Option)

Microcontroller Option	Endian
endian = little	Little endian
endian = big	Big endian

Note: Set up the MDE pin according to the endian setting that is selected using the compiler option.

4.4 Clock Synchronous Communication Specifications

In the example given in this application note, the master and slave exchange the communications commands, erase block number, programming data size, programming data through a clock synchronous communications interface. The transfer clock is transmitted by the master. The pins SCK2-A and RxD2-A of the SCI2 which is used are externally pulled up.

Table 5 lists the major clock synchronous communication specifications.

Table 5 Clock Synchronous Communication Specifications

Item	Specification
Channel	SCI channel 2 (SCI2)
Communications mode	Clock synchronous mode
Bit rate	2.4 Mbps (at PCLK = 48 MHz)
Direction of data transfer	LSB first
Error	Overrun error

4.4.1 Communications Command Specifications

Table 6 lists the major specifications for the communications commands exchanged between the master and slave.

Table 6 Communications Command Specifications

Command	Code	Description	Direction of Communication
FSTART	10h	Starts the user MAT programming/erasure processing on the slave.	Master → Slave
ERASE	11h	Starts the user MAT erasing processing on the slave.	Master → Slave
WRITE	12h	Starts the user MAT programming on the slave.	Master → Slave

4.4.2 Communications Flows

Figures 3 to 6 show the flows of communications between the master and slave devices.

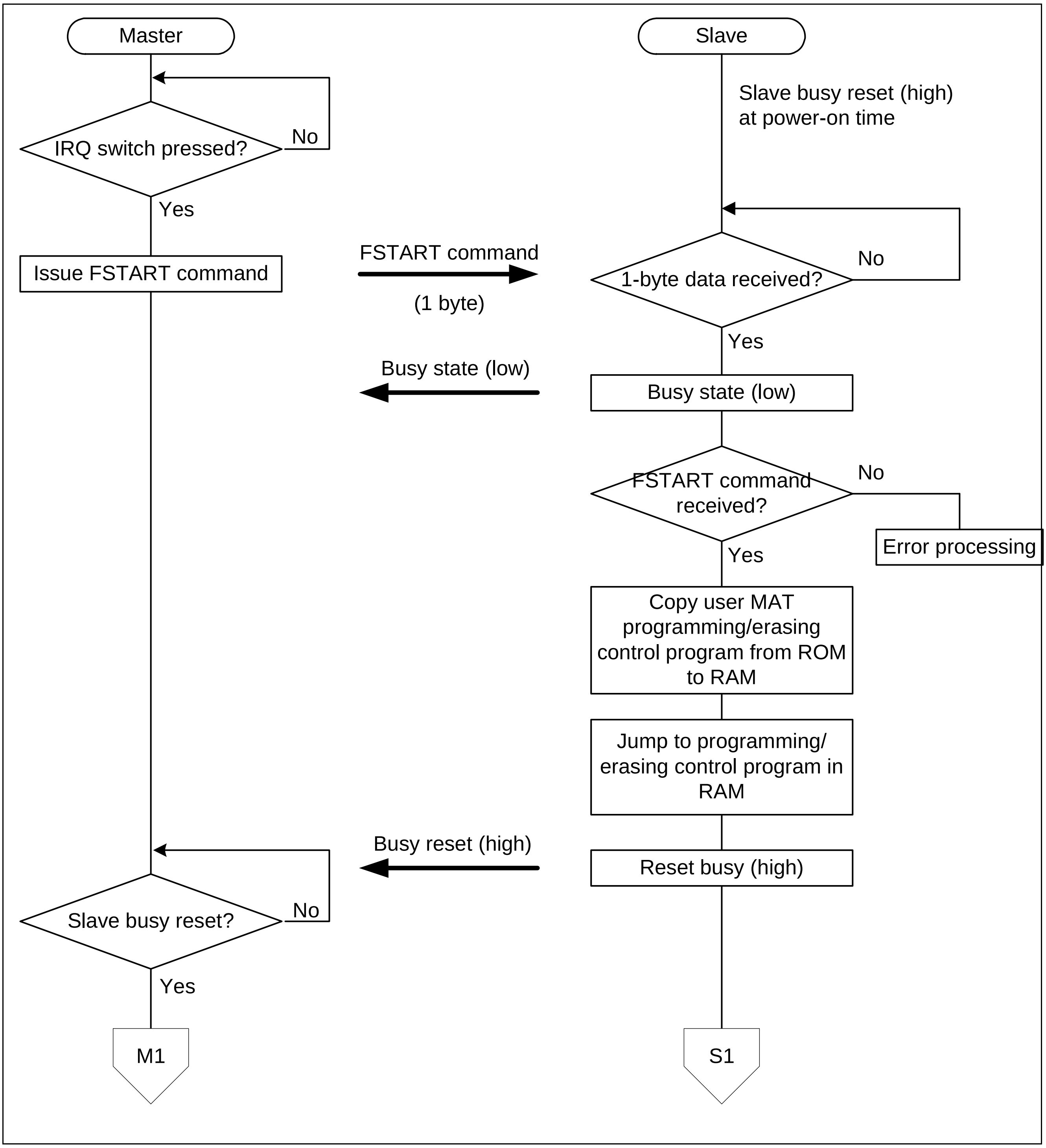

Figure 3 Communications Flow (1)

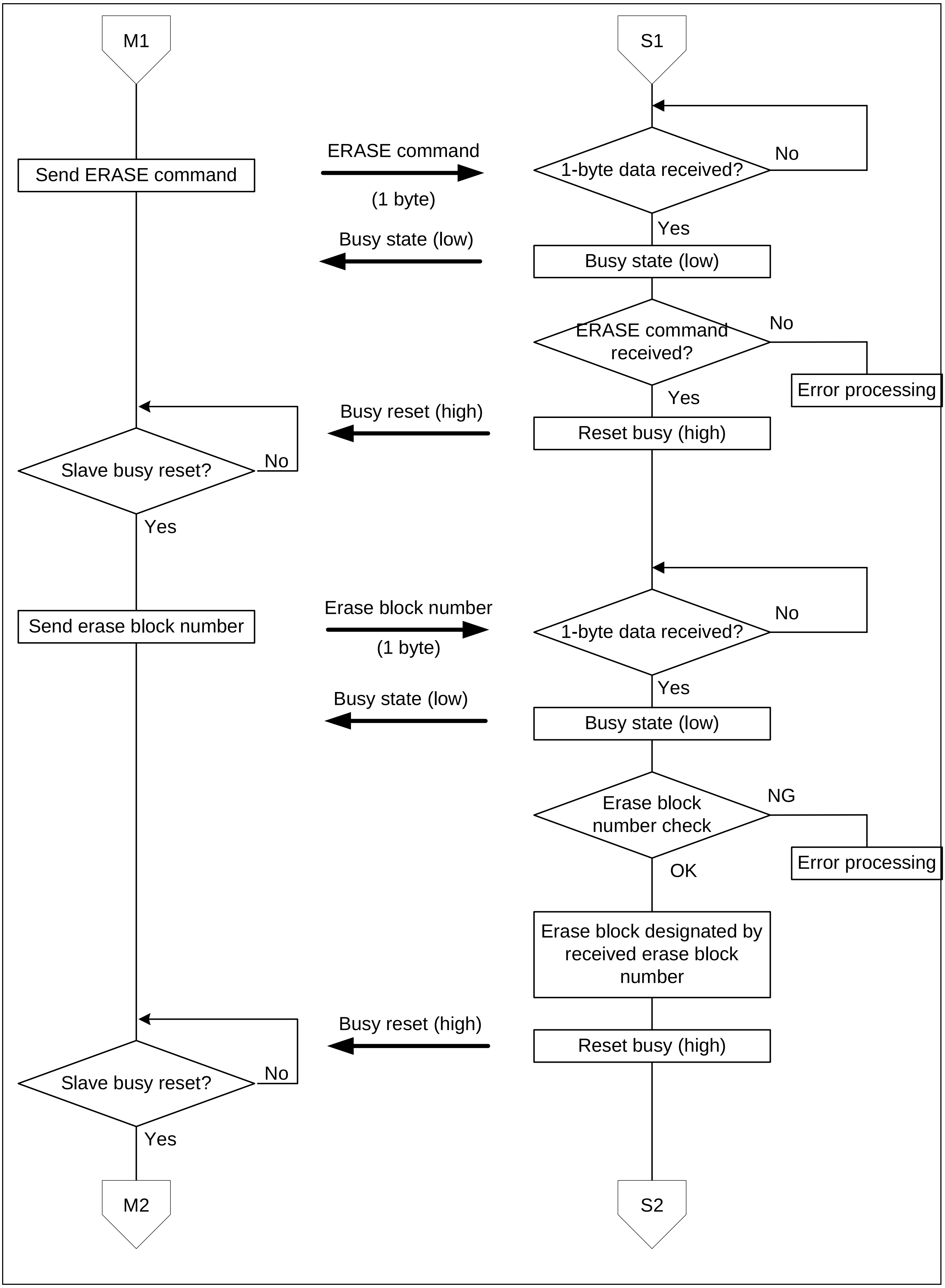

Figure 4 Communications Flow (2)

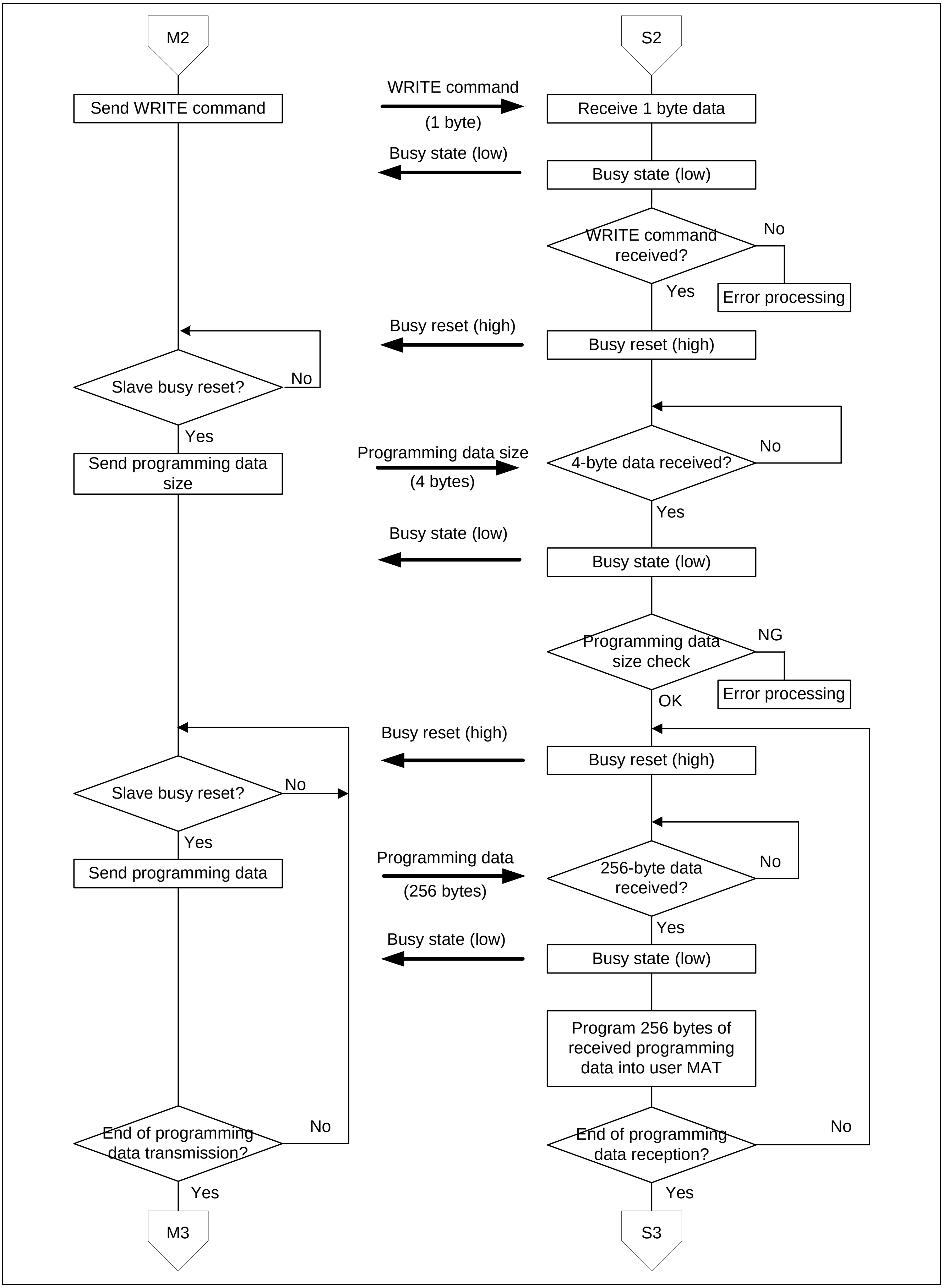

Figure 5 Communications Flow (3)

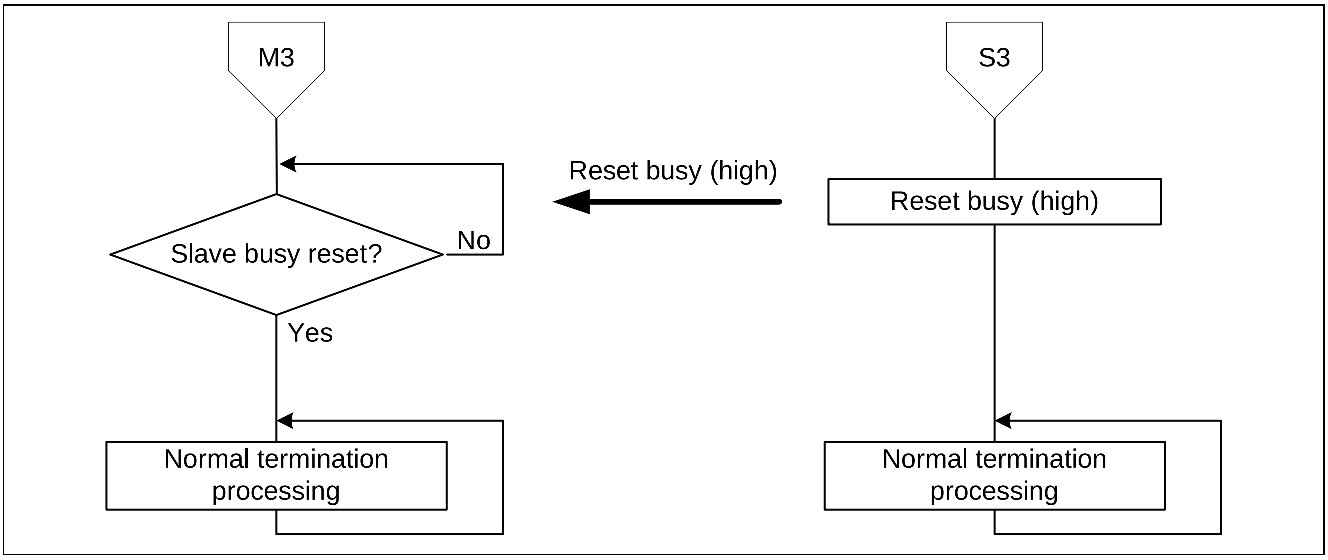

Figure 6 Communications Flow (4)

4.4.3 Erasure Block Number

The slave receives 1-byte erase block numbers (1-byte data defined by a symbolic constant) after receiving the ERASE command from the master. Table 7 gives a list of erase block numbers and figure 7 shows the major specifications for the erase block numbers.

Table 7 List of Erasure Block Numbers

Erasure Block Number		
Symbolic Constant Name	Value	**Description**
EB37_INDEX	00h	Specifies erase block 37 (size: 16 K bytes).
EB36_INDEX	01h	Specifies erase block 36 (size: 16 K bytes).
EB35_INDEX	02h	Specifies erase block 35 (size: 16 K bytes).
EB34_INDEX	03h	Specifies erase block 34 (size: 16 K bytes).
EB33_INDEX	04h	Specifies erase block 33 (size: 16 K bytes).
EB32_INDEX	05h	Specifies erase block 32 (size: 16 K bytes).
EB31_INDEX	06h	Specifies erase block 31 (size: 16 K bytes).
EB30_INDEX	07h	Specifies erase block 30 (size: 16 K bytes).
EB29_INDEX	08h	Specifies erase block 29 (size: 16 K bytes).
EB28_INDEX	09h	Specifies erase block 28 (size: 16 K bytes).
EB27_INDEX	0Ah	Specifies erase block 27 (size: 16 K bytes).
EB26_INDEX	0Bh	Specifies erase block 26 (size: 16 K bytes).
EB25_INDEX	0Ch	Specifies erase block 25 (size: 16 K bytes).
EB24_INDEX	0Dh	Specifies erase block 24 (size: 16 K bytes).
EB23_INDEX	0Eh	Specifies erase block 23 (size: 16 K bytes).
EB22_INDEX	0Fh	Specifies erase block 22 (size: 16 K bytes).
EB21_INDEX	10h	Specifies erase block 21 (size: 16 K bytes).
EB20_INDEX	11h	Specifies erase block 20 (size: 16 K bytes).
EB19_INDEX	12h	Specifies erase block 19 (size: 16 K bytes).
EB18_INDEX	13h	Specifies erase block 18 (size: 16 K bytes).
EB17_INDEX	14h	Specifies erase block 17 (size: 16 K bytes).
EB16_INDEX	15h	Specifies erase block 16 (size: 16 K bytes).
EB15_INDEX	16h	Specifies erase block 15 (size: 16 K bytes).
EB14_INDEX	17h	Specifies erase block 14 (size: 16 K bytes).
EB13_INDEX	18h	Specifies erase block 13 (size: 16 K bytes).
EB12_INDEX	19h	Specifies erase block 12 (size: 16 K bytes).
EB11_INDEX	1Ah	Specifies erase block 11 (size: 16 K bytes).
EB10_INDEX	1Bh	Specifies erase block 10 (size: 16 K bytes).
EB09_INDEX	1Ch	Specifies erase block 09 (size: 16 K bytes).
EB08_INDEX	1Dh	Specifies erase block 08 (size: 16 K bytes).
EB07_INDEX	1Eh	Specifies erase block 07 (size: 4 K bytes).
EB06_INDEX	1Fh	Specifies erase block 06 (size: 4 K bytes).
EB05_INDEX	20h	Specifies erase block 05 (size: 4 K bytes).
EB04_INDEX	21h	Specifies erase block 04 (size: 4 K bytes).
EB03_INDEX	22h	Specifies erase block 03 (size: 4 K bytes).
EB02_INDEX	23h	Specifies erase block 02 (size: 4 K bytes).
EB01_INDEX	24h	Specifies erase block 01 (size: 4 K bytes).
EB00_INDEX	25h	Specifies erase block 00 (size: 4 K bytes).

Erase block data (unsigned char type)

b7	b6	b5	b4	b3	b2	b1	b0
BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0

This application note assumes an erase block data of EB30_INDEX (07h) for the slave to program or erase the erase block EB30.

Note: Specify erase block numbers between EB37_INDEX (00h) and EB00_INDEX (25h) which are listed in table 7. If an erase block number 26h to FFh is specified, the slave will signal an error and perform error processing.

Figure 7 Erasure Block Number Specifications

4.4.4 Programming Data Size

The slave receives 4 bytes of programming data size data after receiving the WRITE command from the master. Figure 8 shows the major specifications for the programming data size.

Programming data size (unsigned long type)

b31	b30	b29	b28	b27	b26	b25	b24
SZ31	SZ30	SZ29	SZ28	SZ27	SZ26	SZ25	SZ24

b23	b22	b21	b20	b19	b18	b17	b16
SZ23	SZ22	SZ21	SZ20	SZ19	SZ18	SZ17	SZ16

b15	b14	b13	b12	b11	b10	b9	b8
SZ15	SZ14	SZ13	SZ12	SZ11	SZ10	SZ09	SZ08

b7	b6	b5	b4	b3	b2	b1	b0
SZ07	SZ06	SZ05	SZ04	SZ03	SZ02	SZ01	SZ00

This application note assumes a programming data size of 0000 2000h since the programming size of block data is set to 8 K bytes.

Notes:
1. The programming data size must be greater than 0 but not greater than the size of the erase block designated by the erase block number. If a 0 is specified or a size value greater than the size of the erase block designated by the erase block number is specified, the slave will signal an error and perform error processing.
2. The size of programming data that is to be transmitted is fixed at 256 bytes. If the size of the programming data is not a multiple of 256 bytes, the master sends to the slave device 256 bytes in every transmission operation with the last data block, which is less than 256 bytes long, padded with FFh bytes to make up a 256-byte programming data block.

Figure 8 Programming Data Size Specifications

4.4.5 Overrun Error Processing

In the example given in this application note, the slave performs error processing if it encounters an overrun error (SCI2.SSR.ORER bit is set to 1) during clock synchronous communication.

4.5 Normal Termination Processing

The slave indicates a normal termination condition with the four LEDs connected to its I/O port when the user MAT programming/erasure processing terminates normally. On normal termination, LED0 to LED3 are turned on sequentially and repeatedly, one at a time.

4.6 Error Processing

Table 8 shows a list of errors that can occur on the slave device referred to in this application note. During slave error processing, the error status is displayed on the four LEDs.

Table 8 List of Slave Errors

○: On, ●: Off

Error Number	Description	LED Display			
		LED3	LED2	LED1	LED0
Error No. 01	An overrun error occurred.	●	●	●	○
Error No. 02	A command other than FSTART was received from the master while waiting for a FSTART command.	●	●	○	●
Error No. 03	A command other than ERASE was received from the master while waiting for an ERASE command.	●	●	○	○
Error No. 04	The erase block number received from the master did not fall between EB00 and EB37.	●	○	●	●
Error No. 05	A timeout (tE16K × 1.1) occurred while switching into ROM read mode before transferring the FCU firmware.	●	○	●	○
Error No. 06	Either the ILGLERR, ERSERR, PRGERR, or FCUERR bit is set to 1 while switching into ROM P/E mode before issuing a peripheral clock notification command.	●	○	○	●
Error No. 07	A timeout (tPCKA) occurred or the ILGLERR bit is set to 1 when a peripheral clock notification command is issued.	●	○	○	○
Error No. 08	A timeout (tE16K × 1.1) occurred or either the ILGLERR or ERSERR bit is set to 1 while erasing an erase block.	○	●	●	●
Error No. 09	A command other than WRITE was received from the master while waiting for a WRITE command.	○	●	●	○
Error No. 10	The programming data size received from the master was 0 or greater than the size of the block designated by the erase block number.	○	●	○	●
Error No. 11	A timeout (tP256 × 1.1) occurred or either the ILGLERR or PRGERR bit is set to 1 while programming data.	○	●	○	○
Error No. 12	A timeout (tE16K × 1.1) occurred while switching into ROM read mode after finishing data programming.	○	○	●	●

4.7 LED Cabling

Figure 9 shows the cabling diagram for LED0 to LED3 that are connected to I/O ports of the slave device.

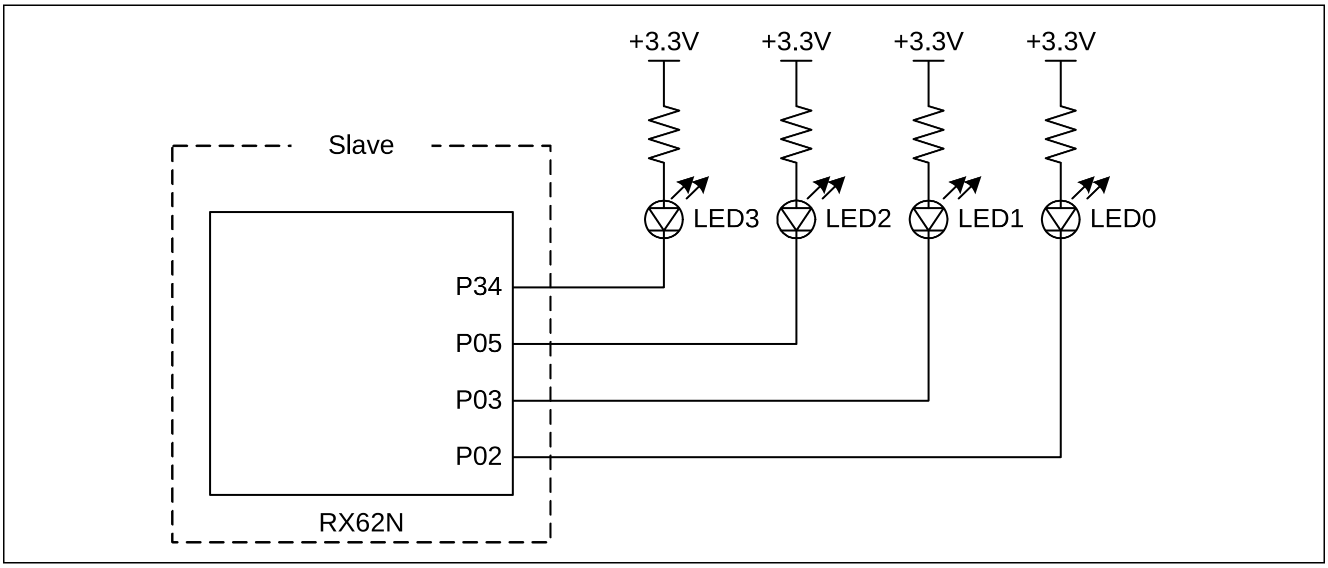

Figure 9 Slave LED Cabling Diagram

As seen from figure 9, LED0 to LED3 turn off when the I/O ports (P02, P03, P05, and P34) are set high and on when the I/O ports are set low. Table 9 shows the relationship between the I/O port outputs and LED states.

Table 9 Slave I/O Port Outputs and LED States

I/O Port	Register Setting	I/O Port State	LED State	
P02	PORT0.DR.B2 = 1, PORT0.DDR.B2 = 1	High output	LED0	Off
	PORT0.DR.B2 = 0, PORT0.DDR.B2 = 1	Low output		On
P03	PORT0.DR.B3 = 1, PORT0.DDR.B3 = 1	High output	LED1	Off
	PORT0.DR.B3 = 0, PORT0.DDR.B3 = 1	Low output		On
P05	PORT0.DR.B5 = 1, PORT0.DDR.B5 = 1	High output	LED2	Off
	PORT0.DR.B5 = 0, PORT0.DDR.B5 = 1	Low output		On
P34	PORT3.DR.B4 = 1, PORT3.DDR.B4 = 1	High output	LED3	Off
	PORT3.DR.B4 = 0, PORT3.DDR.B4 = 1	Low output		On

4.8 Handshake Control

The slave makes handshakes with the master to control the communication between them and generates the signal for handshaking from its Busy port (P01).

For handshake control, the slave asserts the Busy port (low) after receiving a serial communication from the master. It negates the Busy port (for busy reset) when it becomes ready for receiving the next serial communication. Table 10 shows the relationship between the slave's Busy port outputs and I/O port states.

Table 10 Slave Busy Port Outputs

I/O Port	Register Setting	I/O Port State	Busy Port	
P01	PORT0.DR.B1 = 1, PORT0.DDR.B1 = 1	High output	Negate	Busy reset
	PORT0.DR.B1 = 0, PORT0.DDR.B1 = 1	Low output	Assert	Busy state

4.9 User MAT Programming/Erasing

This section explains the procedures for programming and erasing user MATs. For details, see "User's Manual" listed in section 7, Reference Documents.

4.9.1 RX62N Group (R5F562N8) Memory MAT Configuration

The flash memory of the R5F562N8 available for storing code consists of a 512K-byte user MAT and a 16K-byte user boot MAT. Figure 10 shows the memory map of the user MAT and user boot MAT of the R5F562N8.

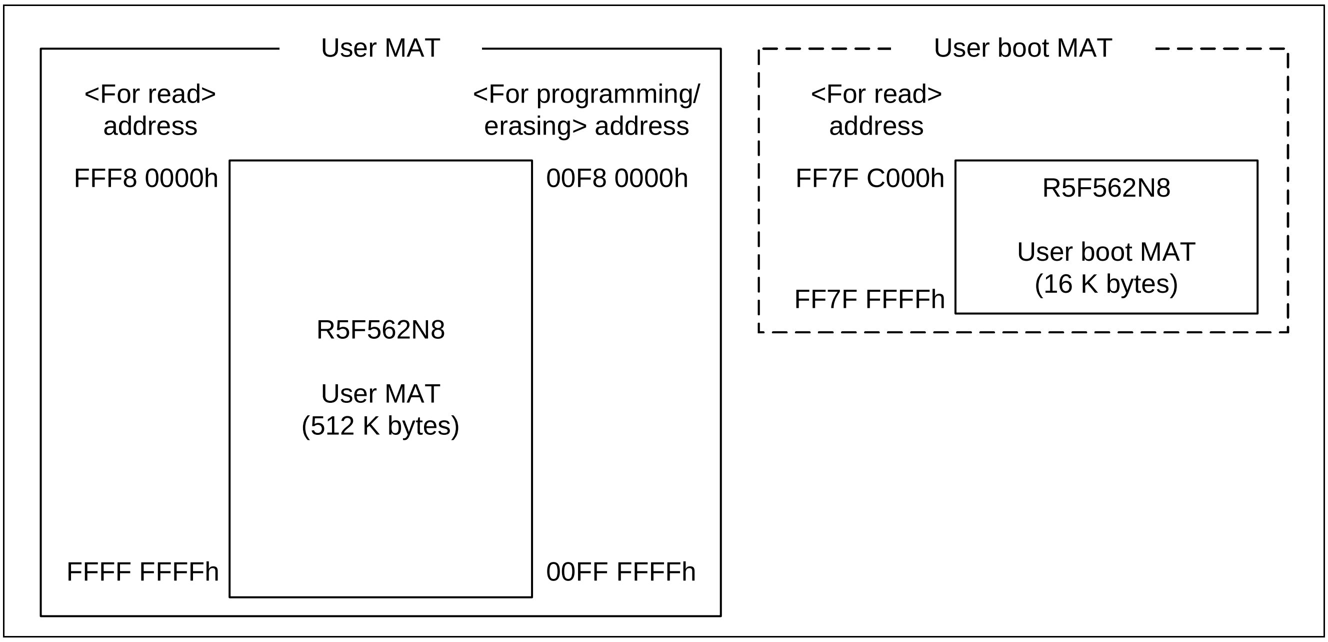

Figure 10 R5F562N8 Memory MAT Configuration

4.9.2 RX62N Group (R5F562N8) Erasure Block Configuration

The user MAT of the R5F562N8 is divided into 16K-byte blocks (30 blocks) and 4K-byte blocks (8 blocks). The user MAT is erased in units of this size block.

Programming into the user MAT is done in 256 byte units in which their lowest address starts at 00h.

Table 11 shows the erase blocks of the R5F562N8's user MAT.

Table 11 R5F562N8 Erasure Block Configuration

	For read		For Programming/Erasing		
Erasure Block	Start Address	End Address	Start Address	End Address	Size (in Bytes)
EB37	FFF8 0000h	FFF8 3FFFh	00F8 0000h	00F8 3FFFh	16K
EB36	FFF8 4000h	FFF8 7FFFh	00F8 4000h	00F8 7FFFh	16K
EB35	FFF8 8000h	FFF8 BFFFh	00F8 8000h	00F8 BFFFh	16K
EB34	FFF8 C000h	FFF8 FFFFh	00F8 C000h	00F8 FFFFh	16K
EB33	FFF9 0000h	FFF9 3FFFh	00F9 0000h	00F9 3FFFh	16K
EB32	FFF9 4000h	FFF9 7FFFh	00F9 4000h	00F9 7FFFh	16K
EB31	FFF9 8000h	FFF9 BFFFh	00F9 8000h	00F9 BFFFh	16K
EB30	FFF9 C000h	FFF9 FFFFh	00F9 C000h	00F9 FFFFh	16K
EB29	FFFA 0000h	FFFA 3FFFh	00FA 0000h	00FA 3FFFh	16K
EB28	FFFA 4000h	FFFA 7FFFh	00FA 4000h	00FA 7FFFh	16K
EB27	FFFA 8000h	FFFA BFFFh	00FA 8000h	00FA BFFFh	16K
EB26	FFFA C000h	FFFA FFFFh	00FA C000h	00FA FFFFh	16K
EB25	FFFB 0000h	FFFB 3FFFh	00FB 0000h	00FB 3FFFh	16K
EB24	FFFB 4000h	FFFB 7FFFh	00FB 4000h	00FB 7FFFh	16K
EB23	FFFB 8000h	FFFB BFFFh	00FB 8000h	00FB BFFFh	16K
EB22	FFFB C000h	FFFB FFFFh	00FB C000h	00FB FFFFh	16K
EB21	FFFC 0000h	FFFC 3FFFh	00FC 0000h	00FC3FFFFh	16K
EB20	FFFC 4000h	FFFC 7FFFh	00FC 4000h	00FC 7FFFh	16K
EB19	FFFC 8000h	FFFC BFFFh	00FC 8000h	00FC BFFFh	16K
EB18	FFFC C000h	FFFC FFFFh	00FC C000h	00FC FFFFh	16K
EB17	FFFD 0000h	FFFD 3FFFh	00FD 0000h	00FD 3FFFh	16K
EB16	FFFD 4000h	FFFD 7FFFh	00FD 4000h	00FD 7FFFh	16K
EB15	FFFD 8000h	FFFD BFFFh	00FD 8000h	00FD BFFFh	16K
EB14	FFFD C000h	FFFD FFFFh	00FD C000h	00FD FFFFh	16K
EB13	FFFE 0000h	FFFE 3FFFh	00FE 0000h	00FE 3FFFh	16K
EB12	FFFE 4000h	FFFE 7FFFh	00FE 4000h	00FE 7FFFh	16K
EB11	FFFE 8000h	FFFE BFFFh	00FE 8000h	00FE BFFFh	16K
EB10	FFFE C000h	FFFE FFFFh	00FE C000h	00FE FFFFh	16K
EB09	FFFF 0000h	FFFF 3FFFh	00FF 0000h	00FF 3FFFh	16K
EB08	FFFF 4000h	FFFF 7FFFh	00FF 4000h	00FF 7FFFh	16K
EB07	FFFF 8000h	FFFF 8FFFh	00FF 8000h	00FF 8FFFh	4K
EB06	FFFF 9000h	FFFF 9FFFh	00FF 9000h	00FF 9FFFh	4K
EB05	FFFF A000h	FFFF AFFFh	00FF A000h	00FF AFFFh	4K
EB04	FFFF B000h	FFFF BFFFh	00FF B000h	00FF BFFFh	4K
EB03	FFFF C000h	FFFF CFFFh	00FF C000h	00FF CFFFh	4K
EB02	FFFF D000h	FFFF DFFFh	00FF D000h	00FF DFFFh	4K
EB01	FFFF E000h	FFFF EFFFh	00FF E000h	00FF EFFFh	4K
EB00	FFFF F000h	FFFF FFFFh	00FF F000h	00FF FFFFh	4K

4.9.3 FCU Commands

The formats of the FCU commands used in the example given in this application note are summarized in table 12. For details, refer to the section on ROM (Flash Memory for Code Storage) in the companion user's manual.

Note that the FCU commands must be used with the volatile and evenaccess keywords to prevent optimization.

Table 12 FCU Command Formats

		1st Cycle		2nd Cycle		3rd Cycle		4th to 5th Cycles		6th Cycle		7th to 130th Cycles		131st Cycle	
Command	No. of Bus Cycles	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
P/E normal mode transition	1	RA	FFh	—	—	—	—	—	—	—	—	—	—	—	—
Peripheral clock setting	6	RA	E9h	RA	03h	RA	0F0Fh	RA	0F0Fh	RA	D0h	—	—	—	—
Programming	131	RA	E8h	RA	80h	WA	WDn	RA	WDn	RA	WDn	RA	WDn	RA	D0h
Block erasure	2	RA	20h	BA	D0h	—	—	—	—	—	—	—	—	—	—
Status register clearing	1	RA	50h	—	—	—	—	—	—	—	—	—	—	—	—

Legends:

Address Column	RA:	ROM programming/erasing address
	WA:	ROM programming destination address
	BA:	ROM erase block address
Data column	WDn:	Ordinal number of programming data in words (n = 1 to 128)

RENESAS

4.9.4 User MAT Programming/Erasing Procedures

Figure 11 shows the user MAT programming/erasing procedures used in the example given in this application note.

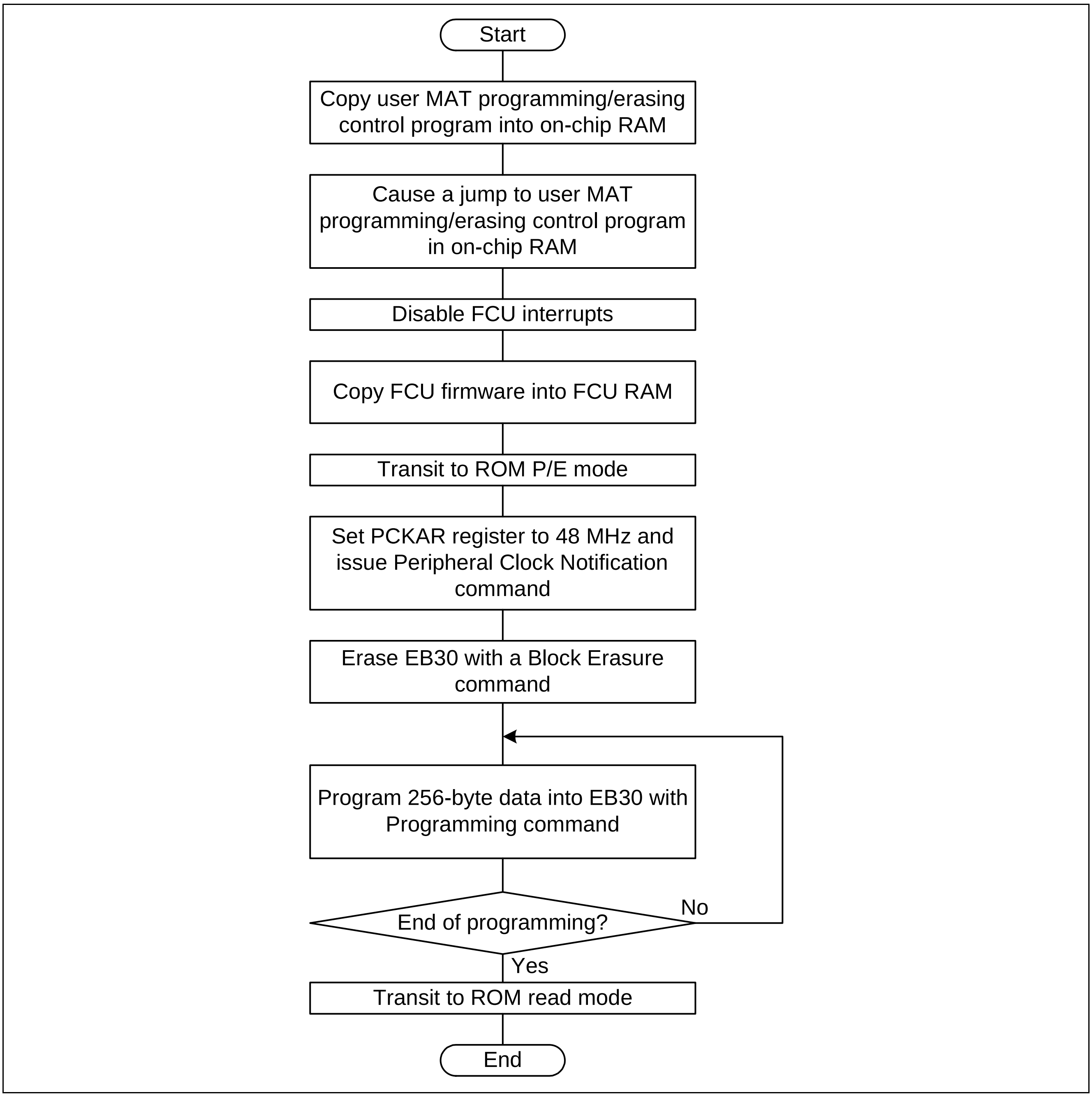

Figure 11 User MAT Programming/Erasing Procedures

4.10 Section Settings

The section settings for the slave device are listed in table 13.

Table 13 Slave Section Settings

Section Name	Start Address	Description
B_1	0000 1000h	Uninitialized data area (ALIGN = 1)
B		Uninitialized data area (ALIGN = 4)
R		RAM area in which the D section is mapped by the ROMization support option
SU		User stack area
SI		Interrupt stack area
RF_UPDATE_FUNC		RAM area in which the PF_UPDATE_FUNC section is mapped by the ROMization support option.
PResetPRG	FF7F C000h	Program area (PowerON_Reset_PC program)
C	FF7F C100h	Constant area (ALIGN = 4)
C$DSEC		Table for initializing the sections in the initialized data area
C$BSEC		Table for initializing the sections in the uninitialized data area
C$VECT		Variable vector area
D		Initialized data area (ALIGN = 4)
P		Program area
PIntPRG		Program area (interrupt program)
PF_UPDATE_FUNC		Program area (user MAT programming/erasing control program)
FIXEDVECT	FF7F FFFCh	Fixed vector area (reset vector)

5. Software Description

5.1 File Organization

The file organization for the slave device is summarized in table 14. For the files that are not listed in table 14, files that are automatically generated by HEW are used.

Table 14 Slave File Organization

File Name	Description
resetprg.c (*1)	Performs initialization.
vecttbl.c (*2)	Defines the fixed vector table.
vect.h (*3)	Defines the interrupt handling functions.
intprg.c (*4)	Performs interrupt processing.
main.c	Controls the processes of receiving the communications command, erase block number, programming data size, and programming data from the master through clock synchronous communication, of block-erasing and programming the user MAT, and of displaying the LEDs in the event of errors.

Notes: *1 A file that is automatically generated by High-performance Embedded Workshop. For the example given in this application note, the call for the HardwareSetup function in the PowerON_Reset_PC function is uncommented so that the HardwareSetup function in the main.c can be called from the PowerON_Reset_PC function.

*2 A file that is automatically generated by High-performance Embedded Workshop. For the example given in this application note, the definitions for the privileged instruction exception, undefined instruction exception, floating-point exception, nonmaskable interrupts, and the definition for the Dummy function in the reserved area are commended out, so that only the definition for the reset vector is available.

*3 A file that is automatically generated by High-performance Embedded Workshop. For the example given in this application note, the definitions for the functions Excep_SuperVisorInst, Dummy, Excep_UndefinedIns, Excep_FloatingPoint, and NonMaskableInterrupt are commented out.

*4 A file that is automatically generated by High-performance Embedded Workshop. For the example given in this application note, the descriptions about the functions Excep_SuperVisorInst, Dummy, Excep_UndefinedInst, Excep_FloatingPoint, and NonMaskableInterrupt are commented out.

5.2 Functions

A list of functions available for the slave device is given in table 15 and the function hierarchy of the slave functions in figure 12.

Table 15 List of Functions for the Slave

Function Name	File Name	Outline
PowerON_Reset_PC	resetprg.c	Initialization function
HardwareSetup	main.c	MCU initialization function
main	main.c	Main function
Flash_Update	main.c	User MAT programming/erasing control function
fcu_Interrupt_Disable	main.c	FCU interrupt disable control function
fcu_Reset	main.c	FCU initialization function
fcu_Transfer_Firmware	main.c	FCU firmware transfer control function
fcu_Transition_RomRead_Mode	main.c	ROM read mode transition control function
fcu_Transition_RomPE_Mode	main.c	ROM P/E mode transition control function
fcu_Notify_Peripheral_Clock	main.c	FCU peripheral clock notification command issuance control function
fcu_Erase	main.c	User MAT erasing control function
fcu_Write	main.c	User MAT programming control function
Indicate_Ending_LED	main.c	Normal termination processing function
Indicate_Error_LED	main.c	Error termination processing function
SCI_Rcv1byte	main.c	1-byte data receive function
SCI_Rcvnbyte	main.c	n-byte data receive function

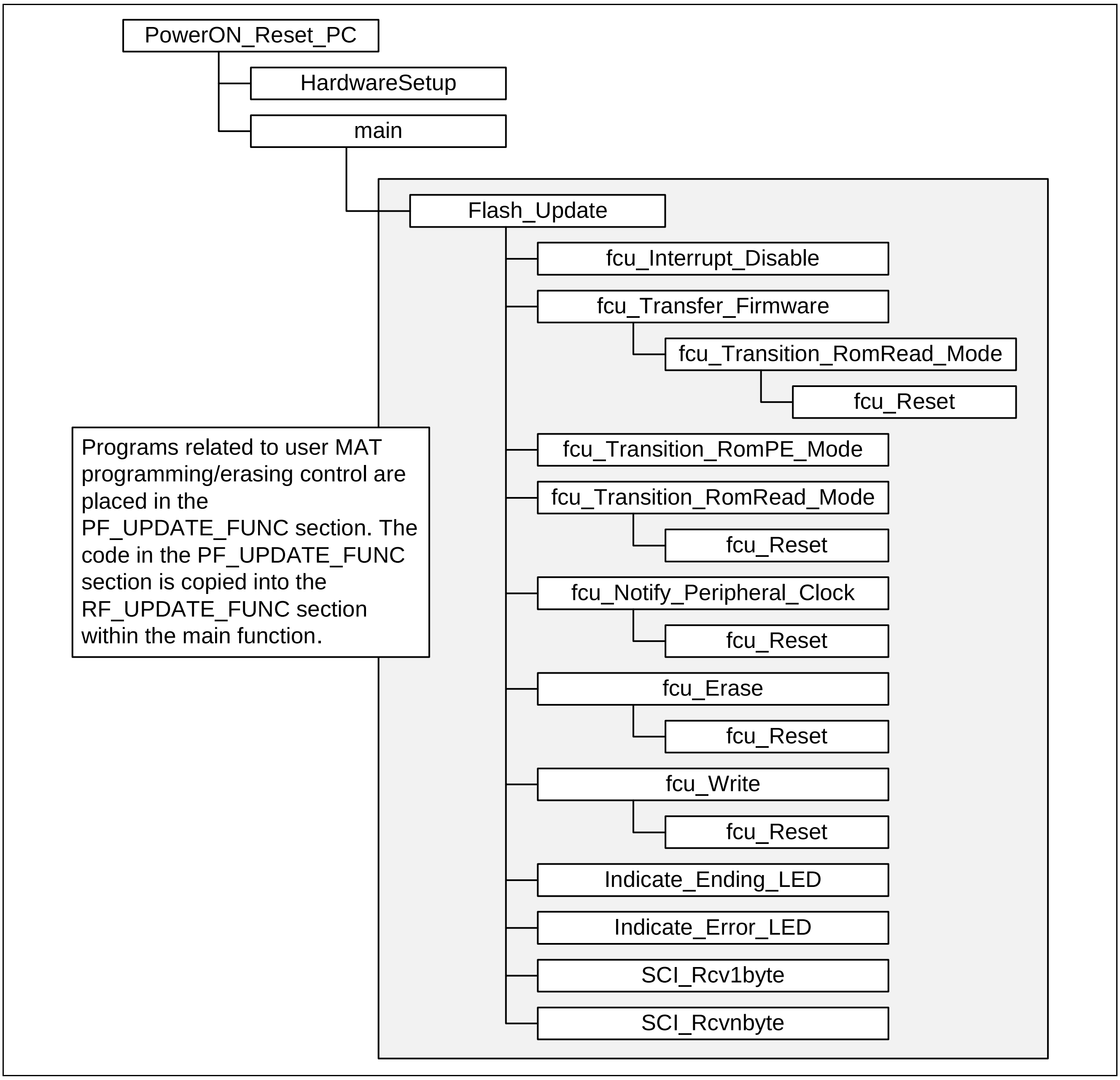

Figure 12 Slave Function Hierarchical Diagram

5.3 Symbolic Constant Description

Table 16 lists the symbolic constants that are to be used by the slave device.

Table 16 List of Slave Symbolic Constants

Symbolic Constant Name	Setting	Description	Used In
FSTART	0x10	Programming/erasure start command	main
ERASE	0x11	Erasure start command	Flash_Update
WRITE	0x12	Programming start command	Flash_Update
LED_ON	0	LED on time value	main Indicate_Ending_LED Indicate_Error_LED
LED_OFF	1	LED off time value	main Indicate_Ending_LED Indicate_Error_LED
RSK_LED0	PORT0.DR.BIT.B2	Evaluation board mounted LED0 on/off control	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED1	PORT0.DR.BIT.B3	Evaluation board mounted LED1 on/off control	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED2	PORT0.DR.BIT.B5	Evaluation board mounted LED2 on/off control	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED3	PORT3.DR.BIT.B4	Evaluation board mounted LED3 on/off control	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED0_DDR	PORT0.DDR.BIT.B2	Evaluation board mounted LED0 input/output control	HardwareSetup
RSK_LED1_DDR	PORT0.DDR.BIT.B3	Evaluation board mounted LED1 input/output control	HardwareSetup
RSK_LED2_DDR	PORT0.DDR.BIT.B5	Evaluation board mounted LED2 input/output control	HardwareSetup
RSK_LED3_DDR	PORT3.DDR.BIT.B4	Evaluation board mounted LED3 input/output control	HardwareSetup
ASSERT	0	Busy port assert time value	main Flash_Update
NEGATE	1	Busy port negate time value	main Flash_Update
SLAVE_BUSY	PORT0.DR.BIT.B1	Busy port output control	HardwareSetup main Flash_Update
SLAVE_BUSY_DDR	PORT0.DDR.BIT.B1	Busy port input/output control	HardwareSetup

Table 16 List of Slave Symbolic Constant (Continued)

Symbolic Constant Name	Setting	Description	Used In
PCKA_48MHZ	0x0030	Frequency data of the peripheral module clock (PCLK) to be set in the PCKAR register	fcu_Notify_Peripheral_Clock
WAIT_TE16K	7603200	Timeout (tE16K × 1.1) data tE16K: Erasure time for the 16K-byte to-be-erased block	fcu_Transition_RomRead_Modefcu_Erase
WAIT_TP256	345600	Timeout (tP256 × 1.1) data tP256: Programming time for 256-byte data	fcu_Write
WAIT_TRESW2	2520	Wait (tRESW2) data tRESW2: Programming/erasing reset pulse width	fcu_Reset
WAIT_TPCKA	1636	Timeout (tPCKA) data	fcu_Notify_Peripheral_Clock
WAIT_LED	2000000	Time data about the LED on interval of the LEDs to be displayed when the slave's user MAT programming/erasing terminates normally	Indicate_Ending_LED Indicate_Error_LED
FCU_FIRM_TOP	0xFEFFE000	Start address of the FCU firmware storage area	fcu_Transfer_Firmware
FCU_RAM_TOP	0x007F8000	Start address of FCU RAM	fcu_Transfer_Firmware
FCU_RAM_SIZE	0x2000	Size of FCU RAM	fcu_Transfer_Firmware
SIZE_WRITE_BLOCK	128	User MAT programming size (word size)	Flash_Update fcu_Program_Verify
BUF_SIZE	256	Size of the programming data storage area	—
ERROR_NO_01	1	Data indicating error status	Flash_Update Indicate_Error_LED
ERROR_NO_02	2		
ERROR_NO_03	3		
ERROR_NO_04	4		
ERROR_NO_05	5		
ERROR_NO_06	6		
ERROR_NO_07	7		
ERROR_NO_08	8		
ERROR_NO_09	9		
ERROR_NO_10	10		
ERROR_NO_11	11		
ERROR_NO_12	12		

Table 16 List of Slave Symbolic Constant (Continued)

Symbolic Constant Name	Setting	Description	Used In
EB37_INDEX	0x00	Erase block number to be sent to designate the erase block to be programmed or erased by the slave.	Flash_Update
EB36_INDEX	0x01		
EB35_INDEX	0x02		
EB34_INDEX	0x03		
EB33_INDEX	0x04		
EB32_INDEX	0x05		
EB31_INDEX	0x06		
EB30_INDEX	0x07		
EB29_INDEX	0x08		
EB28_INDEX	0x09		
EB27_INDEX	0x0A		
EB26_INDEX	0x0B		
EB25_INDEX	0x0C		
EB24_INDEX	0x0D		
EB23_INDEX	0x0E		
EB22_INDEX	0x0F		
EB21_INDEX	0x10		
EB20_INDEX	0x11		
EB19_INDEX	0x12		
EB18_INDEX	0x13		
EB17_INDEX	0x14		
EB16_INDEX	0x15		
EB15_INDEX	0x16		
EB14_INDEX	0x17		
EB13_INDEX	0x18		
EB12_INDEX	0x19		
EB11_INDEX	0x1A		
EB10_INDEX	0x1B		
EB09_INDEX	0x1C		
EB08_INDEX	0x1D		
EB07_INDEX	0x1E		
EB06_INDEX	0x1F		
EB05_INDEX	0x20		
EB04_INDEX	0x21		
EB03_INDEX	0x22		
EB02_INDEX	0x23		
EB01_INDEX	0x24		
EB00_INDEX	0x25		

5.4 const Variable Description

Table 17 lists the const variable that is to be used by the slave device.

Table 17 List of Slave const Variables

Variable Name	Type	Description
tbl_eb_adrs[ ]	ST_EB_ADRS (*1)	Data (760 bytes) including the erase block (EB00 to EB37) starting and ending programming/erasing addresses, starting and ending read addresses, and erase block size

Note: *1 See 5.6.2, Structure ST_EB_ADRS, for details on the ST_EB_ADRS type.

5.5 RAM Variable Description

Table 18 shows the RAM variables that are to be used by the slave device.

Table 18 List of Slave RAM Variables

Variable Name		Type	Description
wrdata_buffer[BUF_SIZE]		unsigned char	Array storing the 256-byte programming data received from the slave (256 bytes)
fcu_info		ST_FCU_INFO (*1)	Structure storing the FCU-related address information to be used to program/erase the user MAT (28 bytes)
	p_write_buffer	unsigned short *	Address of the area for storing the programming data used during user MAT programming: 4 bytes
	p_command_adrs	unsigned char *	Address of the destination to which the FCU command is to be issued (address for programming/erasing): 4 bytes
	p_erase_adrs	unsigned short *	Start address of the block to be erased in erasure mode processing (address for programming/erasing): 4 bytes
	p_write_adrs_top	unsigned short *	Start address of the block to be erased in programming mode (address for programming/erasing): 4 bytes
	p_write_adrs_end	unsigned short *	End address of the program to be erased in programming mode (address for programming/erasing): 4 bytes
	p_write_adrs_now	unsigned short *	Destination address into which data is to be programmed in programming mode (address for programming/erasing): 4 bytes
	eb_block_size	unsigned long	Size of the block to be erased: 4 bytes

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

5.6 Structure Description

5.6.1 Structure ST_FCU_INFO

Table 19 shows the major specifications for the structure ST_FCU_INFO that is to be used by the slave device.

Table 19 Structure ST_FCU_INFO Specifications

Member Name	Type	Description
p_write_buffer	unsigned short *	Address of area for storing the programming data to be used when programming the user MAT
p_command_adrs	volatile __evenaccess unsigned char *	Destination address to which the FCU command is to be issued (address for programming/erasing)
p_erase_adrs	unsigned short *	Start address of the block to be erased in erasure mode (address for programming/erasing)
p_write_adrs_top	unsigned short *	Start address of the block to be erased in programming mode (address for programming/erasing)
p_write_adrs_end	unsigned short *	End address of the block to be erased in programming mode (address for programming/erasing)
p_write_adrs_now	unsigned short *	Destination address into which data is to be programmed in programming mode (address for programming/erasing)
eb_block_size	unsigned long	Size of block to be erased

5.6.2 Structure ST_EB_ADRS

Table 20 shows the major specifications for the structure ST_EB_ADRS that is to be used by the slave device.

Table 20 Structure ST_EB_ADRS Specifications

Member Name	Type	Description
eb_write_adrs_top	unsigned long	Start address of the block to be erased (for programming/erasing)
eb_write_adrs_end	unsigned long	End address of the block to be erased (for programming/erasing)
eb_read_adrs_top	unsigned long	Start address of the block to be erased (for read)
eb_read_adrs_end	unsigned long	End address of the block to be erased (for read)
eb_block_size	unsigned long	Size of the block to be erased

5.7 Description of the enum Type

Table 21 shows the specifications for the enum type structure FCU_STATUS that is to be used by the slave device. FCU_STATUS is used as a return value of a function to provide status information.

Table 21 enum Type FCU_STATUS Specifications

Member Name	Type	Value	Description
FCU_SUCCESS	signed long	0	Normal state
FCU_ERROR	signed long	1	Error state

5.8 Description of the I/O Registers Used

This section describes the I/O registers that are used by the program on the slave device. The settings that are described in this document are those values which are used in the example program given in this application note; they differ from their initialized values.

(1) Clock Generation Circuit

- System clock control register (SCKCR) Number of bits: 32 bits Address: 0008 0020h

Bit	Symbol	Setting	Bit Name	Description	R/W
b11-b8	PCK[3:0]	0001	Peripheral module clock(PCLK) select	0001: ×4 PCLK = 48 MHz (when EXTAL clock frequency = 12.0 MHz)	R/W
b19-b16	BCK[3:0]	0010	External bus clock (BCLK) select	0010: ×2 BCLK = 24 MHz (when EXTAL clock frequency = 12.0 MHz)	R/W
b23	PSTOP1	0	BCLK output stop	0: BCLK output	R/W
b27-b24	ICK[3:0]	0000	System clock (ICLK) select	0000: ×8 ICLK = 96 MHz (when EXTAL clock frequency = 12.0 MHz)	R/W

(2) I/O ports

- Port 0 data register (P0.DR) Number of bits: 8 bits Address: 0008 C020h

Bit	Symbol	Setting	Bit Name	Description	R/W
b2	B2	0 1	P02 output data	0: Output data = 0 1: Output data = 1	R/W
b3	B3	0 1	P03 output data	0: Output data = 0 1: Output data = 1	R/W
b5	B5	0 1	P05 output data	0: Output data = 0 1: Output data = 1	R/W

- Port 3 data register (P3.DR) Number of bits: 8 bits Address: 0008 C023h

Bit	Symbol	Setting	Bit Name	Description	R/W
b4	B4	0 1	P34 output data	0: Output data = 0 1: Output data = 1	R/W

- Port function control register F (PFFSCI) Number of bits: 8 bits Address: 0008 C10Fh

Bit	Symbol	Setting	Bit Name	Description	R/W
b2	SCI2S	0	SCI2 Pin select	0: P12 is designated as the RxD2-A pin. P11 is designated as the SCK2-A pin. P13 is designated as the TxD2-A pin.	R/W

- Port 0 data direction register (P0.DDR) Number of bits: 8 bits Address: 0008 C000h

Bit	Symbol	Setting	Bit Name	Description	R/W
b2	B2	1	P02 input/output select	1: Output port	R/W
b3	B3	1	P03 input/output select	1: Output port	R/W
b5	B5	1	P05 input/output select	1: Output port	R/W

- Port 3 data direction register (P3.DDR) Number of bits: 8 bits Address: 0008 C003h

Bit	Symbol	Setting	Bit Name	Description	R/W
b4	B4	1	P34 input/output select	1: Output port	R/W

- Port 1 input buffer control register (P1.ICR) Number of bits: 8 bits Address: 0008 C061h

Bit	Symbol	Setting	Bit Name	Description	R/W
b1	B1	1	P11 input buffer control	1: Enables the input buffer for P11.	R/W
b2	B2	1	P12 input buffer control	1: Enables the input buffer for P12.	R/W

(3) Low Power Consumption

- Module stop control register B (MSTPCRB) Number of bits: 32 bits Address: 0008 0014h

Bit	Symbol	Setting	Bit Name	Description	R/W
b29	MSTPB29	0	Serial communications interface 2 module stop	0: The SCI2 module stop state is canceled.	R/W

(4) Serial communications interface 2 (SCI2)

- SCI2 serial control register (SCI2.SCR) Number of bits: 8 bits Address: 0008 8252h
  (In serial communications interface mode (SCI2.SCMR.SMIF = 0))

Bit	Symbol	Setting	Bit Name	Description	R/W
b1-b0	CKE[1:0]	10	Clock enable	(Clock synchronous mode) 10: External clock The SCK2 pin is configured for clock input.	R/W (*1)
b2	TEIE	0	Transmit end interrupt enable	0: TEI2 interrupt requests are disabled.	R/W
b4	RE	0 1	Receive enable	0: Serial reception is disabled. 1: Serial reception is enabled.	R/W (*2)
b5	TE	0	Transmit enable	0: Serial transmission is disabled	R/W (*2)
b6	RIE	0 1	Receive interrupt enable	0: RXI2 and ERI2 interrupt requests are disabled. 1: RXI2 and ERI2 interrupt requests are enabled.	R/W
b7	TIE	0	Transmit interrupt enable	0: TXI2 interrupt requests are disabled.	R/W

Notes: *1 Writable only when TE = 0 and RE = 0.
*2 A 1 can be written only when TE = 0 and RE = 0. After setting TE or RE to 1, only 0 can be written in TE and RE.

- SCI2 serial mode register (SCI2.SMR) Number of bits: 8 bits Address: 0008 8250h
  (In serial communications interface mode (SCI2.SCMR.SMIF = 0))

Bit	Symbol	Setting	Bit Name	Description	R/W
b1-b0	CKS[1:0]	00	Clock select	00: PCLK clock (n = 0) (*1)	R/W (*2)
b7	CM	1	Communications mode	1: Run in clock synchronous mode.	R/W (*2)

Notes: *1 See "User's Manual" listed in section 7, Reference Documents, for the value of n.
*2 Writable only when SCI2.SCR.TE = 0 and SCI2.SCR.RE = 0 (serial transmission is disabled and serial reception is disabled).

- SCI2 smart card mode register (SCI2.SCMR) Number of bits: 8 bits Address: 0008 8256h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	SMIF	0	Smart card interface mode select	0: Serial communications interface mode	R/W (*1)
b3	SDIR	0	Smart card data transfer direction	0: Transmitted/received in LSB first mode.	R/W (*1)

Note: *1 Writable only when SCI2.SCR.TE = 0 and SCI2.SCR.RE = 0 (serial transmission is disabled and serial reception is disabled).

- SCI2 serial status register (SCI2.SSR) Number of bits: 8 bits Address: 0008 8254h
  (In serial communications interface mode (SCI2.SCMR.SMIF = 0))

Bit	Symbol	Setting	Bit Name	Description	R/W
b5	ORER	— (*1)	Overrun error	0: No overrun error. 1: Overrun error occurred.	R/W (*2)

Notes: *1 The ORER bit is handled only as read-only in this application note. It is never set to 0 for the purpose of clearing the flag.
*2 Only 0 can be written here to clear the flag.

- SCI2 receive data register (SCI2.RDR) Number of bits: 8 bits Address: 0008 8255h

Bit	Symbol	Setting	Bit Name	Description	R/W
b7-b0	—	—	—	These bits carry the receive data.	R

(5) Interrupt Controller Unit (ICU)

- Interrupt source priority register 82 (IPR82) Number of bits: 8 bits Address: 0008 7382h

Bit	Symbol	Setting	Bit Name	Description	R/W
b3-b0	IPR[3:0]	0000	SCI2 interrupt priority level	0000: Level 0 (interrupts disabled)	R/W

- Interrupt request enable register 1B (IER1B) Number of bits: 8 bits Address: 0008 721Bh

Bit	Symbol	Setting	Bit Name	Description	R/W
b7	IEN7	0	RXI2 interrupt request enable bit 7	0: RXI2 interrupt requests are disabled.	R/W

- Interrupt request register 223 (IR223) Number of bits: 8 bits Address: 0008 70DFh

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	IR	0	RXI2 interrupt status	0: No RXI2 interrupt request present. 1: RXI2 interrupt request present.	R/(W) (*1)

Note: *1 Only 0 can be written to clear the flag. Writing a 1 is prohibited.

- Interrupt source priority register 01 (IPR01) Number of bits: 8 bits Address: 0008 7301h

Bit	Symbol	Setting	Bit Name	Description	R/W
b3-b0	IPR[3:0]	0000	FIFERR interrupt priority level	0000: Level 0 (interrupts disabled)	R/W

- Interrupt source priority register 02 (IPR02) Number of bits: 8 bits Address: 0008 7302h

Bit	Symbol	Setting	Bit Name	Description	R/W
b3-b0	IPR[3:0]	0000	FRDYI interrupt priority level	0000: Level 0 (interrupts disabled)	R/W

- Interrupt request enable register 02 (IER02) Number of bits: 8 bits Address: 0008 7202h

Bit	Symbol	Setting	Bit Name	Description	R/W
b5	IEN5	0	FIFERR interrupt request enable bit 5	0: FIFERR interrupt requests are disabled.	R/W
b7	IEN7	0	FRDYI interrupt request enable bit 7	0: FRDYI interrupt requests are disabled.	R/W

(6) ROM (Flash Memory for Code Storage)

- Flash access status register (FASTAT) Number of bits: 8 bits Address: 007F C410h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	DFLWPE	0	Data flash programming/erasure protection violation	0: No data flash programming/erasure command is issued which conflicts with the DFLWE settings. 1: A data flash programming/erasure command is issued which conflicts with the DFLWE settings.	R/W (*1)
b1	DFLRPE	0	Data flash read protection violation	0: There is no such data flash read that conflicts with the DFLRE settings. 1: There is such a data flash read that conflicts with the DFLRE settings.	R/W (*1)
b3	DFLAE	0	Data flash access violation	0: No data flash access violation. 1: Data flash access violation.	R/W (*1)
b4	CMDLK	1	FCU command lock	0: FCU is not in the command-locked state. 1: FCU is in the command-locked state.	R
b7	ROMAE	0	ROM access violation	0: No ROM access error. 1: ROM access error.	R/W (*1)

Note: *1 Only 0 can be written after reading 1 to clear the flag.

- Flash access error interrupt enable register (FAEINT) Number of bits: 8 bits Address: 007F C411h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	DFLWPEIE	0	Data flash programming/erasure protection violation interrupt enable	0: No FIFERR interrupt request is issued when the FASTAT.DFLWPE bit is set to 1.	R/W
b1	DFLRPEIE	0	Data flash read protection violation interrupt enable	0: No FIFERR interrupt request is issued when the FASTAT.DFLRPE bit is set to 1.	R/W
b3	DFLAEIE	0	Data flash read access violation interrupt enable	0: No FIFERR interrupt request is issued when the FASTAT.DFLAE bit is set to 1.	R/W
b4	CMDLKIE	0	FCU command lock interrupt enable	0: No FIFERR interrupt request is issued when the FASTAT.CMDLK bit is set to 1.	R/W
b7	ROMAEIE	0	ROM access violation interrupt enable	0: No FIFERR interrupt request is issued when the FASTAT.ROMAE bit is set to 1.	R/W

- FCU RAM enable register (FCURAME) Number of bits: 16 bits Address: 007F C454h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	FCRME	1	FCU RAM enable	0: Access to the FCU RAM disabled 1: Access to the FCU RAM enabled.	R/W
b15-b8	KEY[7:0]	11000100	Key code	These bits are used to enable or disable the rewriting of the FCRME bit. C4h: Writing the FCRME bit is enabled only when the value of KEY[7:0] matches C4h in the word access.	R/W (*1)

Note: *1 The write data is not retained.

- Flash status register 0 (FSTATR0) Number of bits: 8 bits Address: 007F FFB0h

Bit	Symbol	Setting	Bit Name	Description	R/W
b4	PRGERR	—	Programming error	0: Programming terminated normally. 1: An error occurred during programming.	R
b5	ERSERR	—	Erasure error	0: Erasure terminated normally. 1: An error occurred during erasure.	R
b6	ILGLERR	—	Illegal command error	0: FCU detected no illegal command or ROM/data flash access. 1: FCU detected no illegal command or ROM/data flash access.	R
b7	FRDY	—	Flash ready	0: Programming/erasure in progress, programming/erasure cancelation in progress, lock bit read 2 command being processed, or data flash blank check processing in progress. 1: None of the above processing is being executed.	R

- Flash status register 1 (FSTATR1) Number of bits: 8 bits Address: 007F FFB1h

Bit	Symbol	Setting	Bit Name	Description	R/W
b7	FCUERR	—	FCU error	0: No error occurred during FCU processing. 1: An error occurred during FCU processing.	R

- Flash protection register (FPROTR) Number of bits: 16 bits Address: 007F FFB4h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	FPROTCN	1	Lock bit protection cancel	1: Protection with a lock bit disabled.	R/W
b15-b8	FPKEY[7:0]	01010101	Key code	These bits are used to enable or disable the rewriting of the FPROTCN bit. 55h: Writing the FPROTCN bit is enabled only when the value of FPKEY[7:0] matches 55h in the word access when the FENTYRY register has a value other than 0000h.	R/W (*1)

Note: *1 The write data is not retained.

- Flash reset register (FRESETR) Number of bits: 16 bits Address: 007F FFB6h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	FRESET	0	Flash reset	0: FCU is not reset.	R/W
		1		1: FCU is reset.	
b15-b8	FRKEY[7:0]	11001100	Key code	These bits are used to enable or disable the rewriting of the FRESET bit. CCh: Writing the FRESET bit is enabled only when the value of FRKEY[7:0] matches CCh in the word access.	R/W (*1)

Note: *1 The write data is not retained.

- Flash P/E mode entry register (FENTRYR) Number of bits: 16 bits Address: 007F FFB2h

Bit	Symbol	Setting	Bit Name	Description	R/W
b0	FENTRY0	0	ROM P/E mode entry 0	0: Products with ROM 512K/384K/256K bytes are in ROM read mode.	R/W
		1		1: Products with ROM 512K/384K/256K bytes are in ROM P/E mode.	
b7	FENTRYD	0	Data flash P/E mode entry	0: Products with data flash memory are in read mode.	R/W
b15-b8	FEKEY[7:0]	10101010	Key code	These bits are used to enable or disable the rewriting of the FENTRYD and FENTRY0 bits. AAh: Writing the FENTRY0 and FENTRYD bits is enabled only when the value of FEKEY[7:0] matches AAh in the word access.	R/W (*1)

Note: *1 The write data is not retained.

- Peripheral clock notification register (PCKAR) Number of bits: 16 bits Address: 007F FFE8h

Bit	Symbol	Setting	Bit Name	Description	R/W
b7-b0	PCKA[7:0]	00110000	Peripheral clock notification	0x30: PCLK frequency = 48 MHz	R/W

- Flash write erase protection register (FWEPROR) Number of bits: 8 bits Address: 0008 C289h

Bit	Symbol	Setting	Bit Name	Description	R/W
b1-b0	FLWE[1:0]	01	Flash write erase	01: Write/erase enabled	R/W
		10		10: Write/erase disabled	

5.9 Functional Specifications

This section contains the specifications for the functions that are to be used by the program on the slave device.

(1) PowerON_Reset_PC function

(a) Functional overview

The PowerON_Reset_PC function initializes the stack pointer (the ISP/USP initialization code is automatically generated by the compiler at the beginning of the function when the #pragma entry is declared for the PowerON_Reset_PC function), sets up the INTB (set_intb function: an intrinsic function), initializes the FPSW (set_fpsw function: an intrinsic function), initializes the RAM area sections (_INITSCT function: standard library function), calls the HardwareSetup function, initializes the PSW (set_psw function: an intrinsic function), and sets the processor mode to user mode. Subsequently, the function calls the main function.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

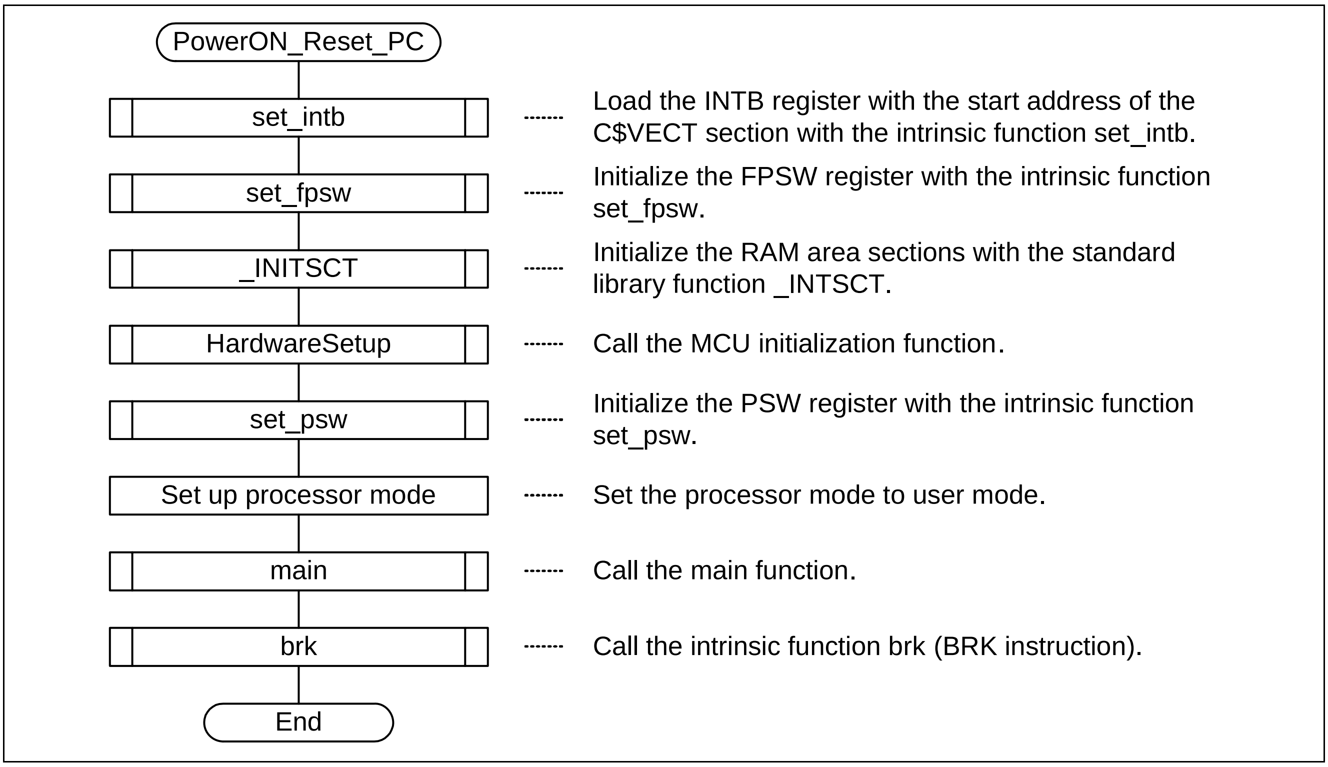

Figure 13 Flowchart (PowerON_Reset_PC) (Slave)

(2) HardwareSetup function

(a) Functional overview

The HardwareSetup function initializes the MCU. It sets up the clocks (system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK)), initializes the outputs of the I/O ports (P02, P03, P05, and P34) to which LED0 to LED3 are connected and the Busy port (P01), and initializes the SCI2.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

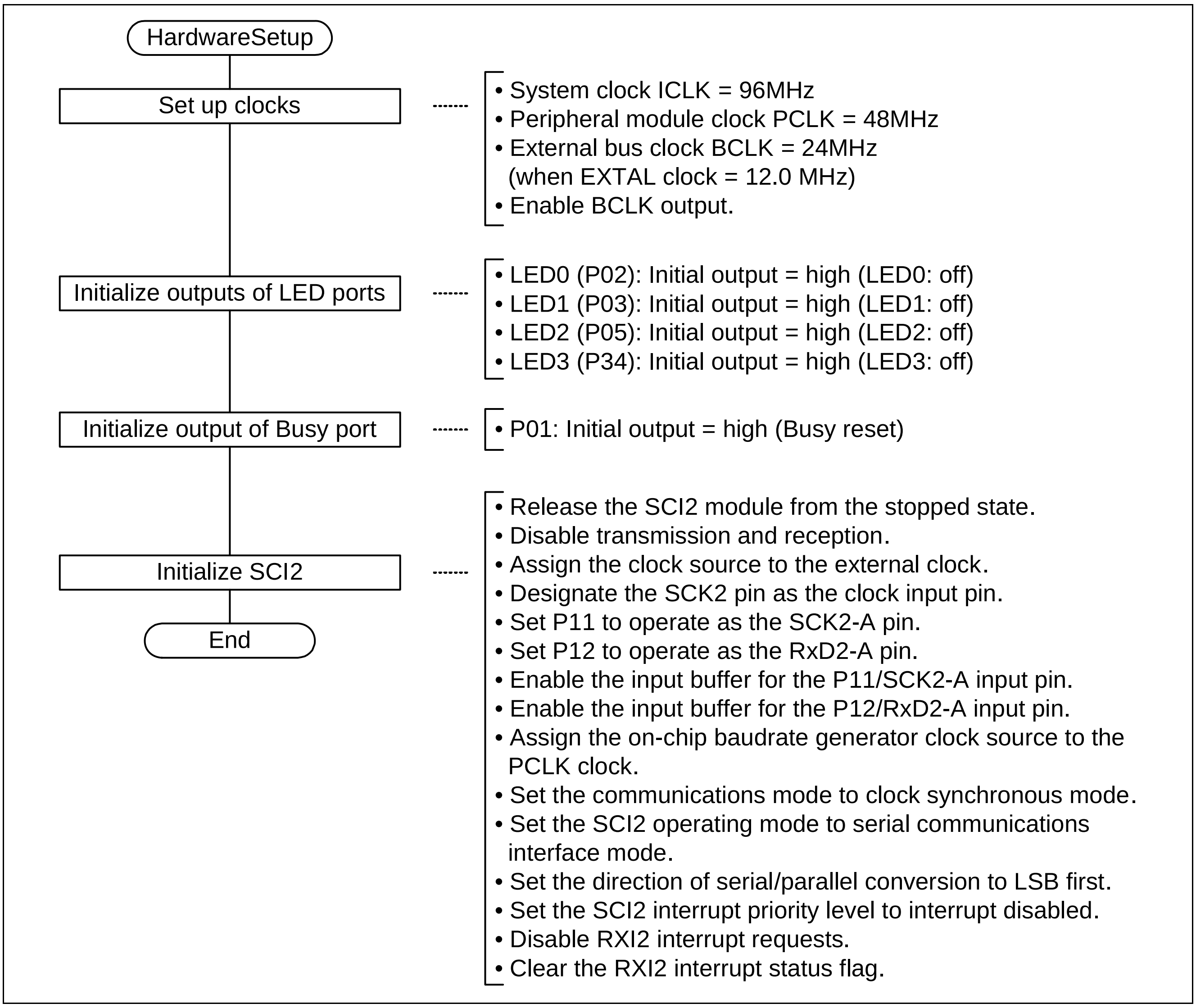

Figure 14 Flowchart (HardwareSetup) (Slave)

(3) main function

(a) Functional overview

The main function controls the reception of 1-byte data from the master, copies the user MAT programming/erasing control program from the user boot MAT (PF_UPDATE_FUNC section) to the on-chip RAM (RF_UPDATE_FUNC section), and calls the user MAT programming control program (Flash_Update function) in the on-chip RAM.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

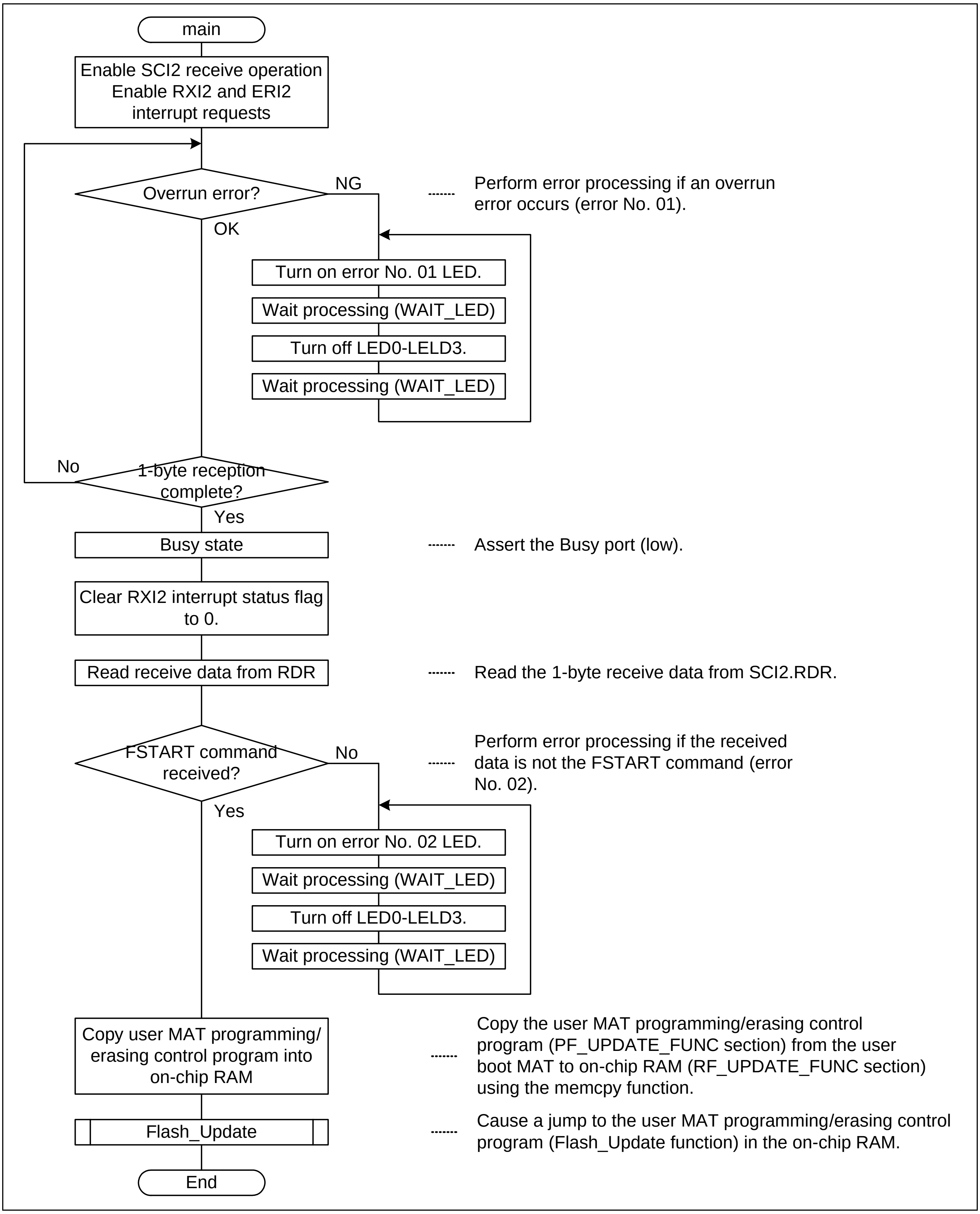

Figure 15 Flowchart (main) (Slave)

(4) Flash_Update function

(a) Functional overview

The Flash_Update function controls the reception of the communications command, erase block number, programming data size, and programming data that are sent from the master through clock synchronous communication. It also controls the Busy port for serial communication, user MAT programming and erasing. The function calls the Indicate_End_LED function when programming or erasure of the user MAT terminates normally and the Indicate_Error_LED function in the event of an error termination.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

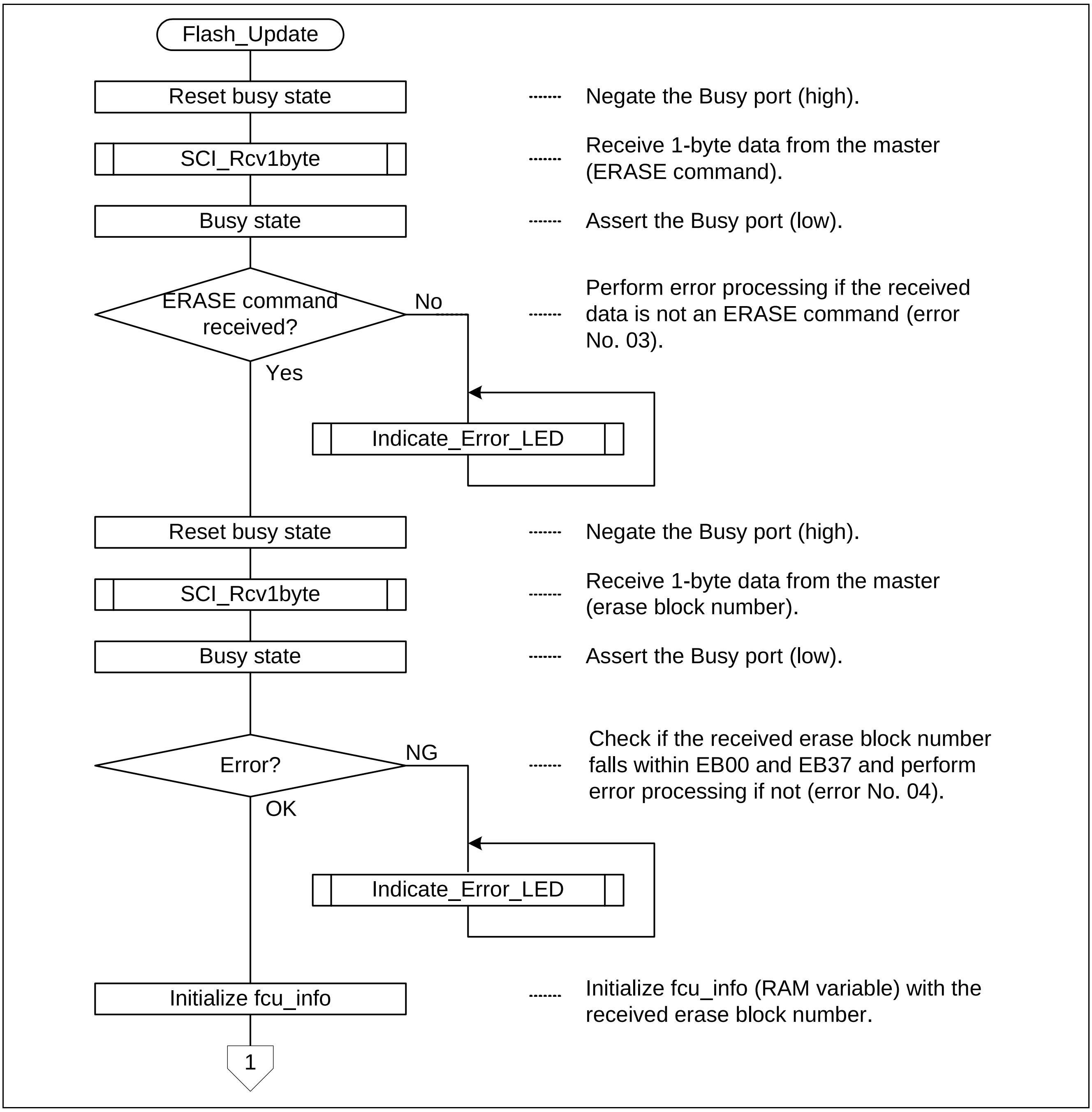

Figure 16 Flowchart (Flash_Update) (1) (Slave)

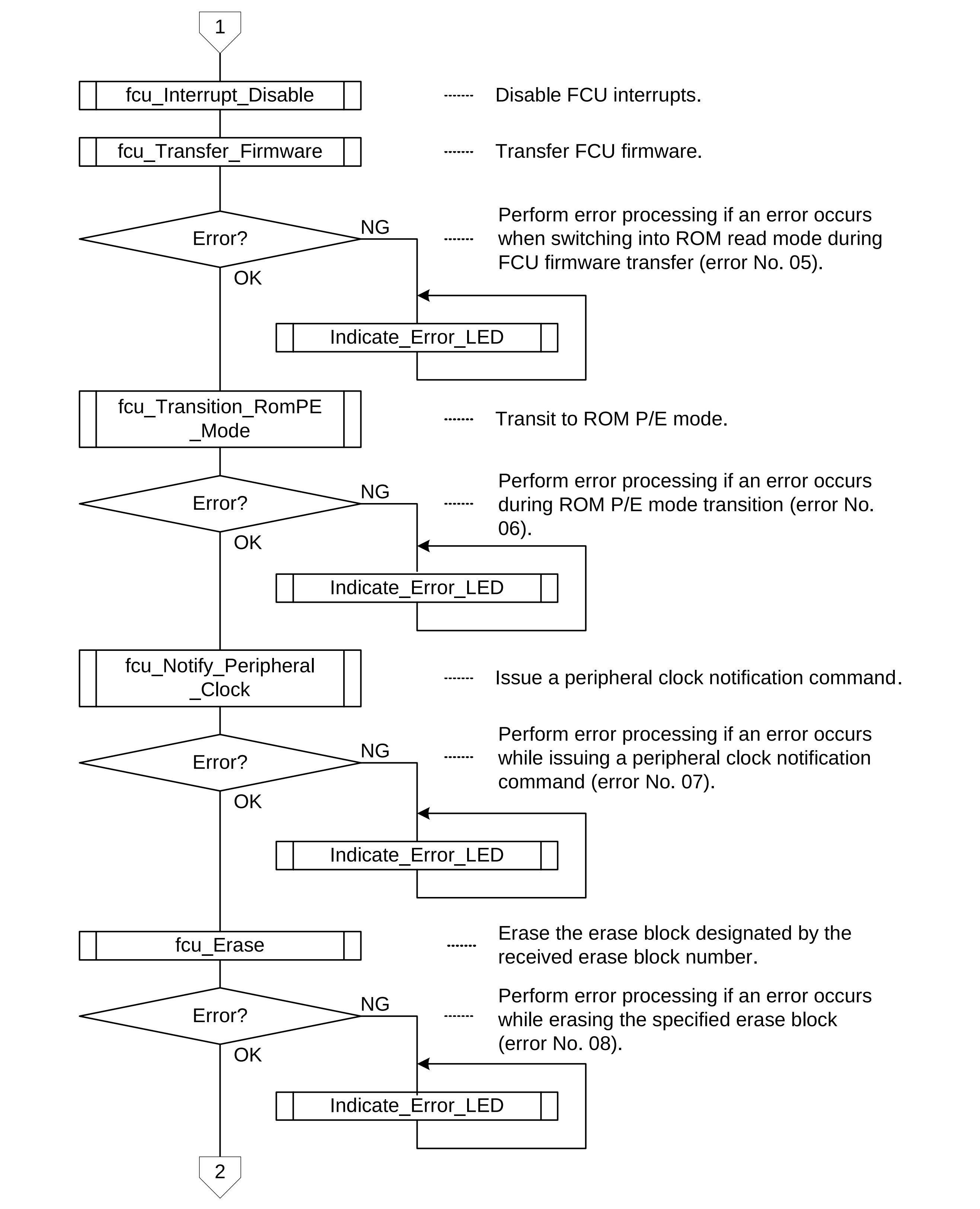

Figure 17 Flowchart (Flash_Update) (2) (Slave)

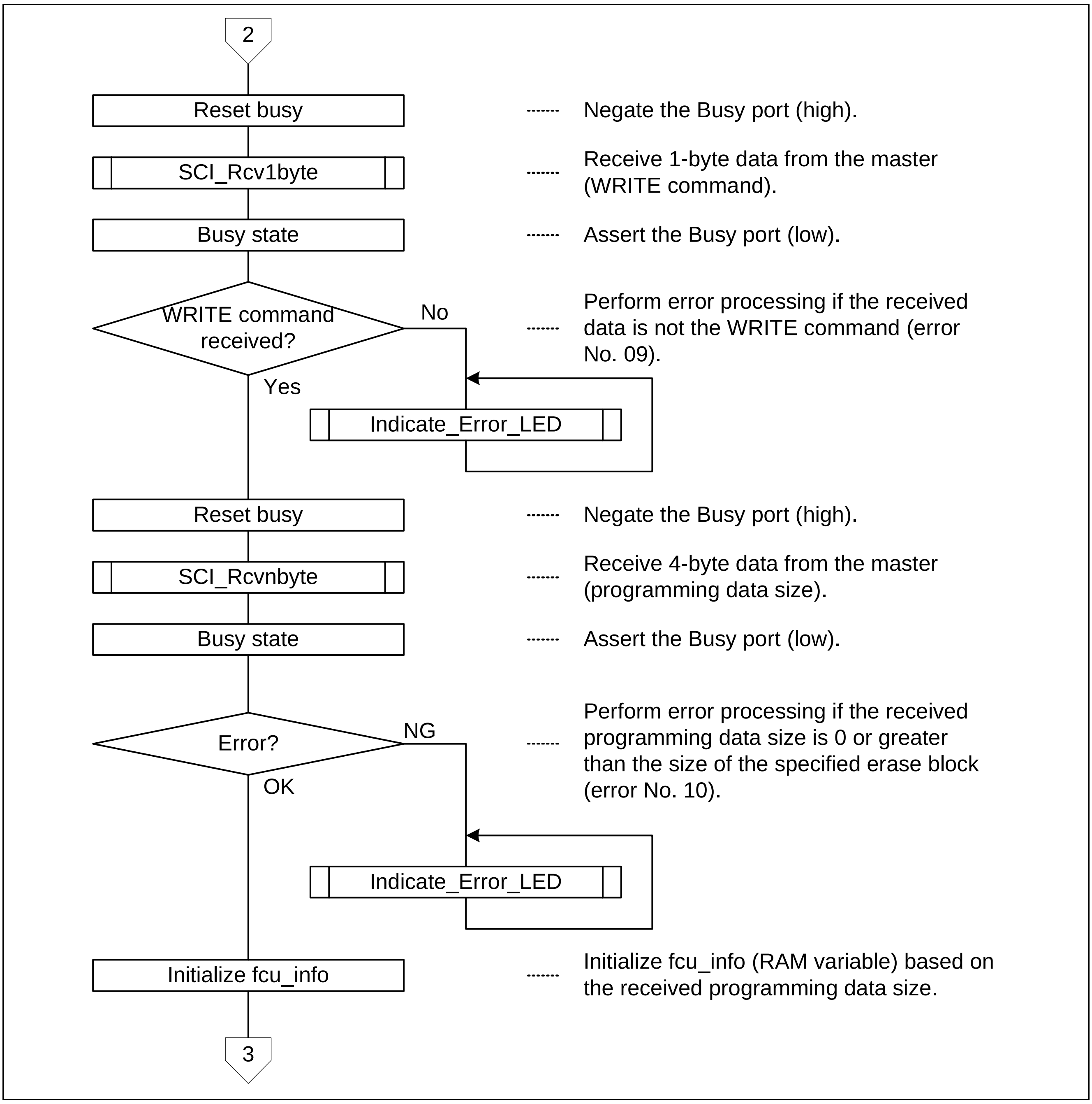

Figure 18 Flowchart (Flash_Update) (3) (Slave)

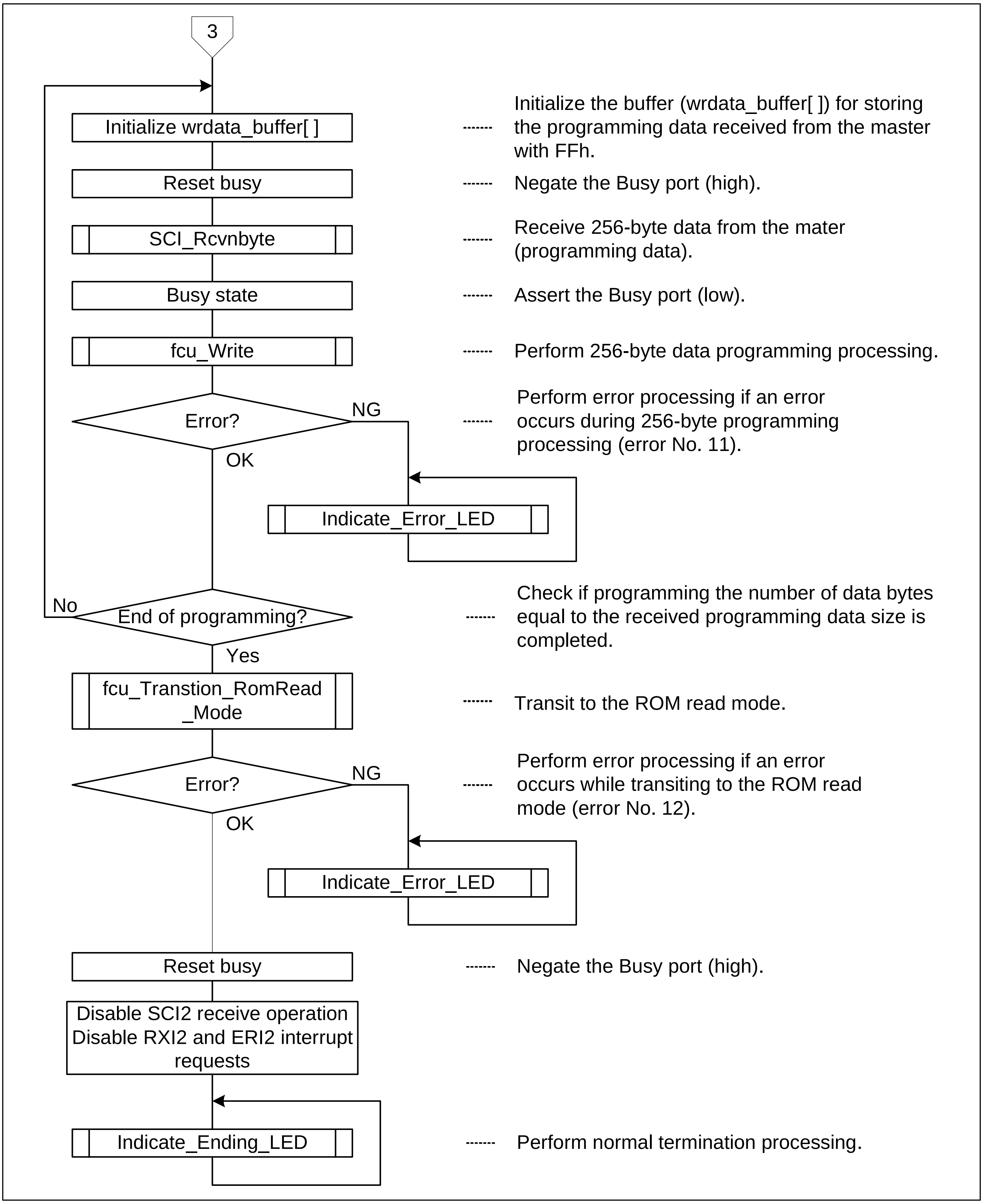

Figure 19 Flowchart (Flash_Update) (4) (Slave)

(5) fcu_Interrupt_Disable function

(a) Functional overview

The fcu_Interrupt_Disable function disables FCU interrupts (the FRDYI interrupt, data flash programming/erasure protection violation interrupt, data flash read protection violation interrupt, data flash access violation interrupt, FCU command lock interrupt, ROM access violation interrupt, and FIFERR interrupt) before user MAT programming erasing processing.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

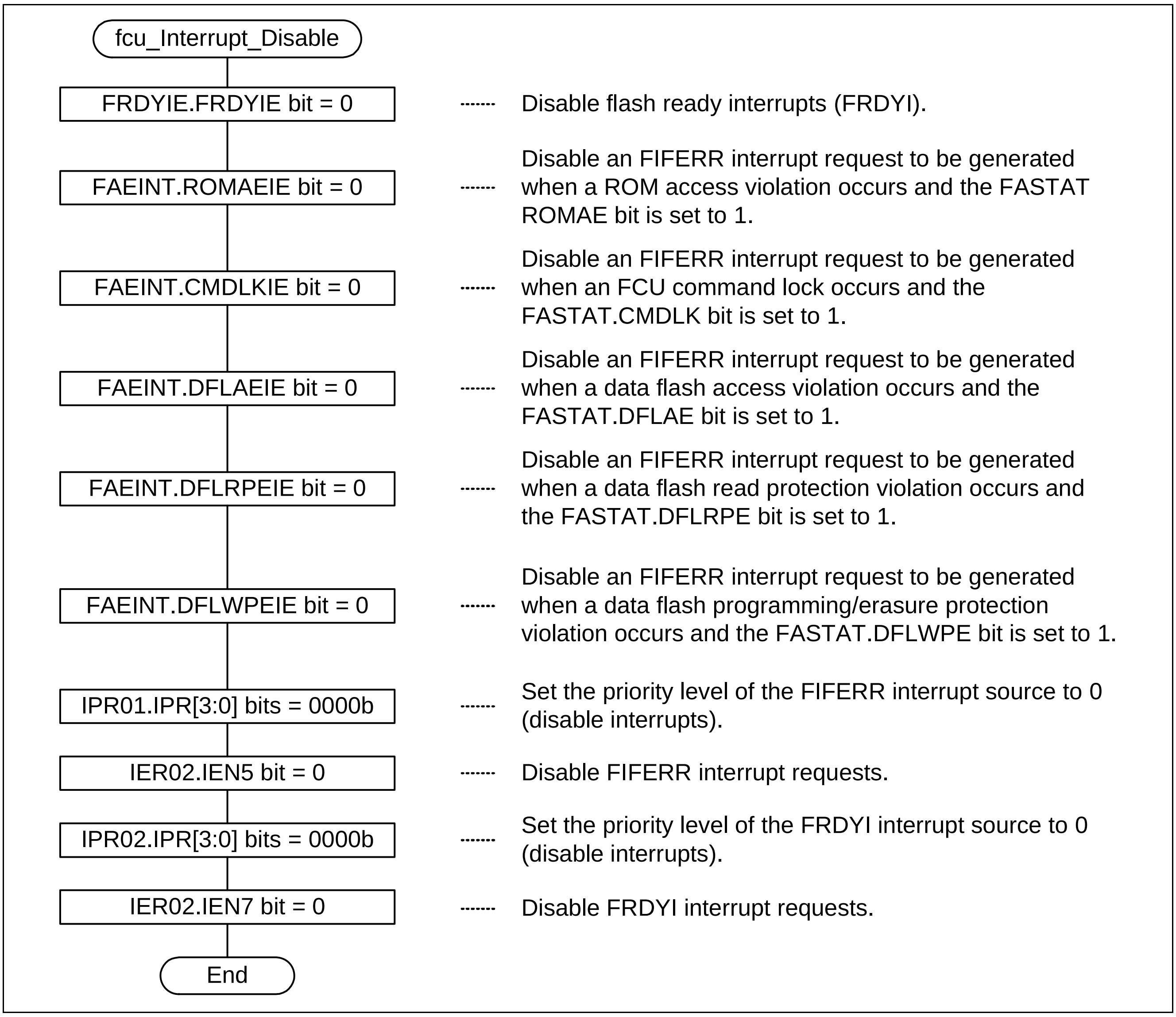

Figure 20 Flowchart (fcu_Interrupt_Disable) (Slave)

(6) fcu_Reset function

(a) Functional overview

The fcu_Reset function initializes the FCU according to the state of the FRESETR.FRESET bit.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

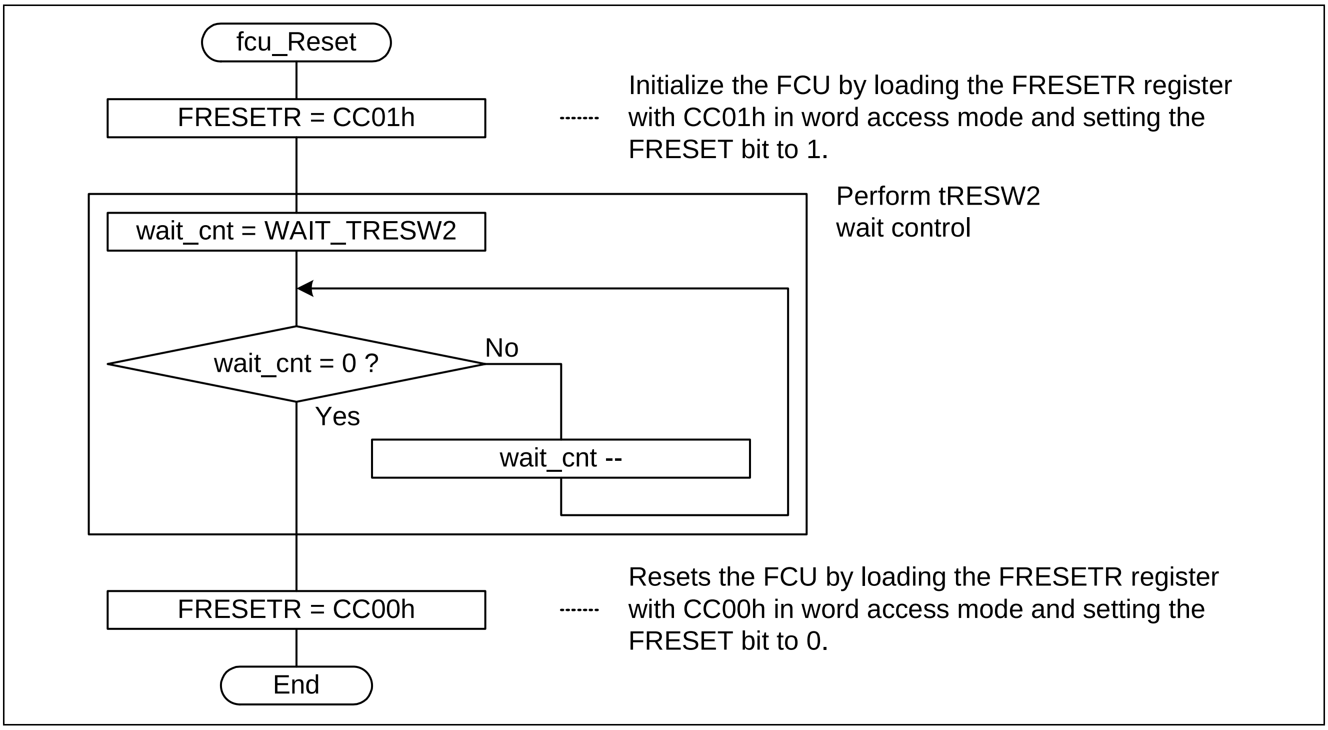

Figure 21 Flowchart (fcu_Reset) (Slave)

(7) fcu_Transfer_Firmware function

(a) Functional overview

The fcu_Transfer_Firmware function copies the FCU firmware from the FCU firmware storage area (FEFF E000h to FEFF FFFFh) to the FCU RAM area (007F 8000h to 007F 9FFFh).

(b) Arguments

Table 22 lists the argument that is used by this function.

Table 22 List of fcu_Transfer_Firmware Function Arguments

Argument	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 23 lists the return value that is returned by this function.

Table 23 List of fcu_Transfer_Firmware Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

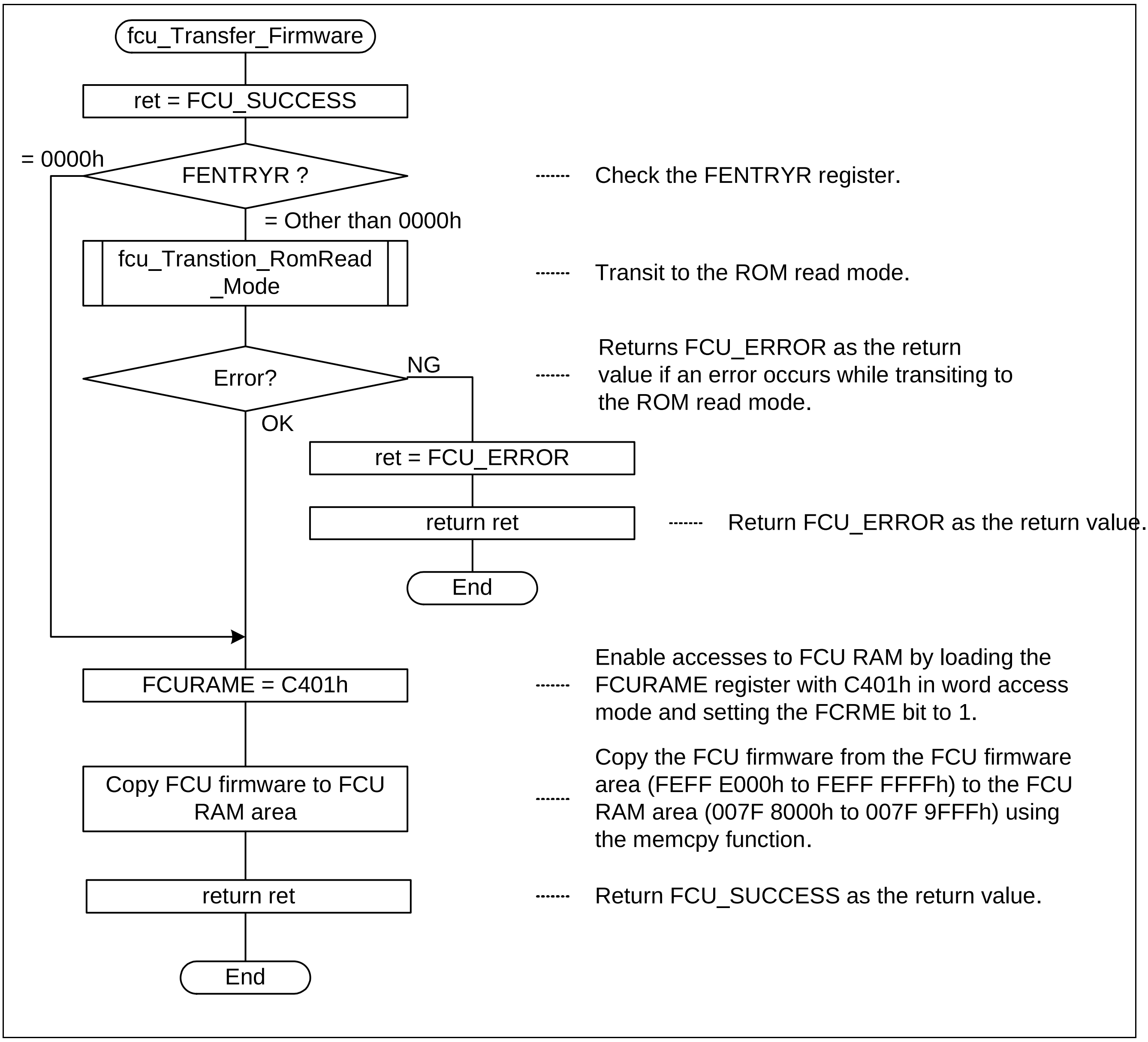

Figure 22 Flowchart (fcu_Transfer_Firmware) (Slave)

(8) fcu_Transition_RomRead_Mode function

(a) Functional overview

The fcu_Transition_RomRead_Mode function transits the FCU to the ROM read mode.

(b) Arguments

Table 24 lists the argument that is used by this function.

Table 24 List of fcu_Transition_RomRead_Mode Function Arguments

Arguments	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 25 lists the return value that is returned by this function.

Table 25 List of fcu_Transition_RomRead_Mode Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

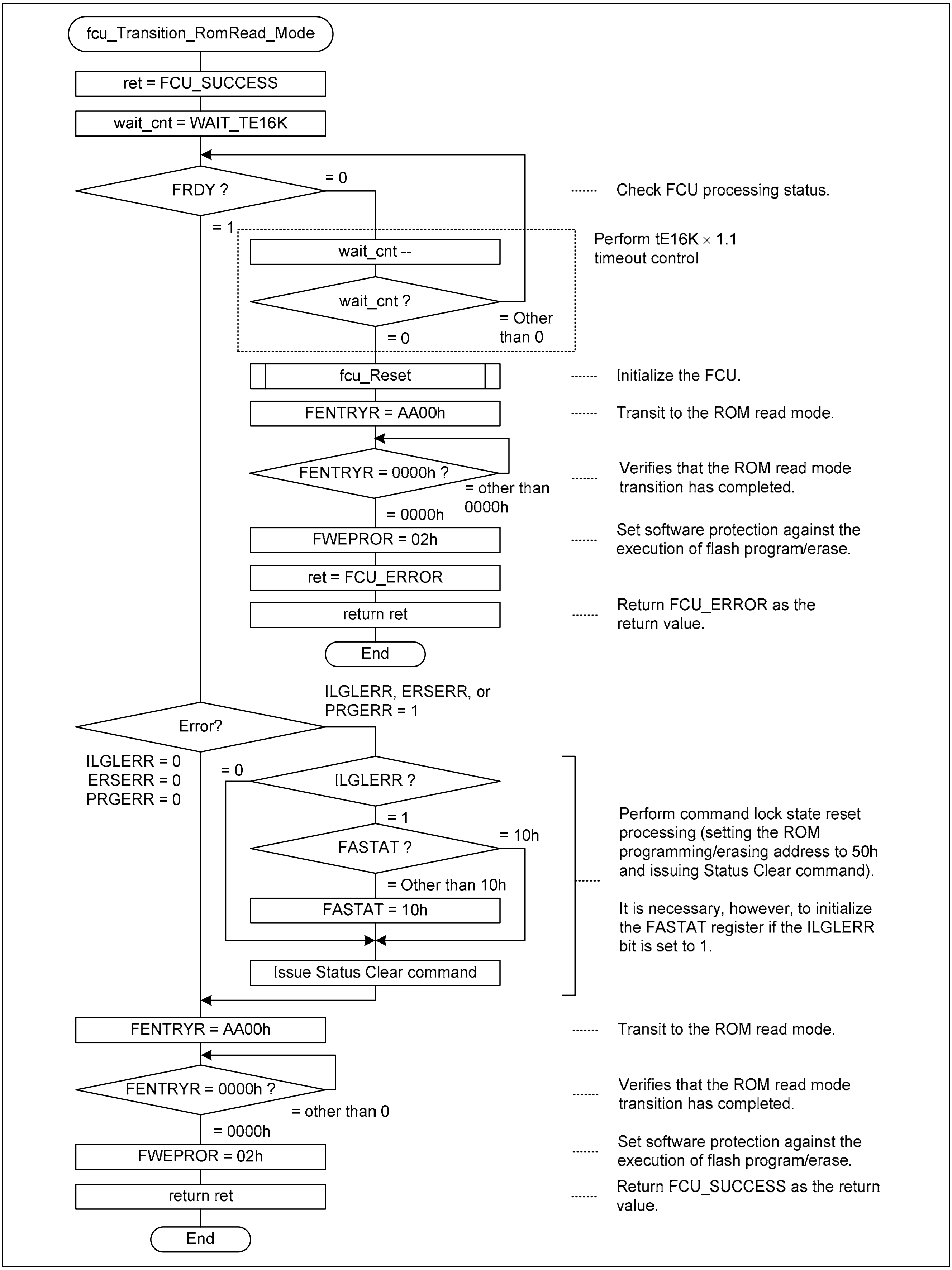

Figure 23 Flowchart (fcu_Transition_RomRead_Mode) (Slave)

(9) fcu_Transition_RomPE_Mode function

(a) Functional overview

The fcu_Transition_RomPE_Mode function transits the FCU to the ROM P/E mode.

(b) Arguments

Table 26 lists the argument that is used by this function.

Table 26 List of fcu_Transition_RomPE_Mode Function Arguments

Argument	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 27 lists the return value that is returned by this function.

Table 27 List of fcu_Transition_RomPE_Mode Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

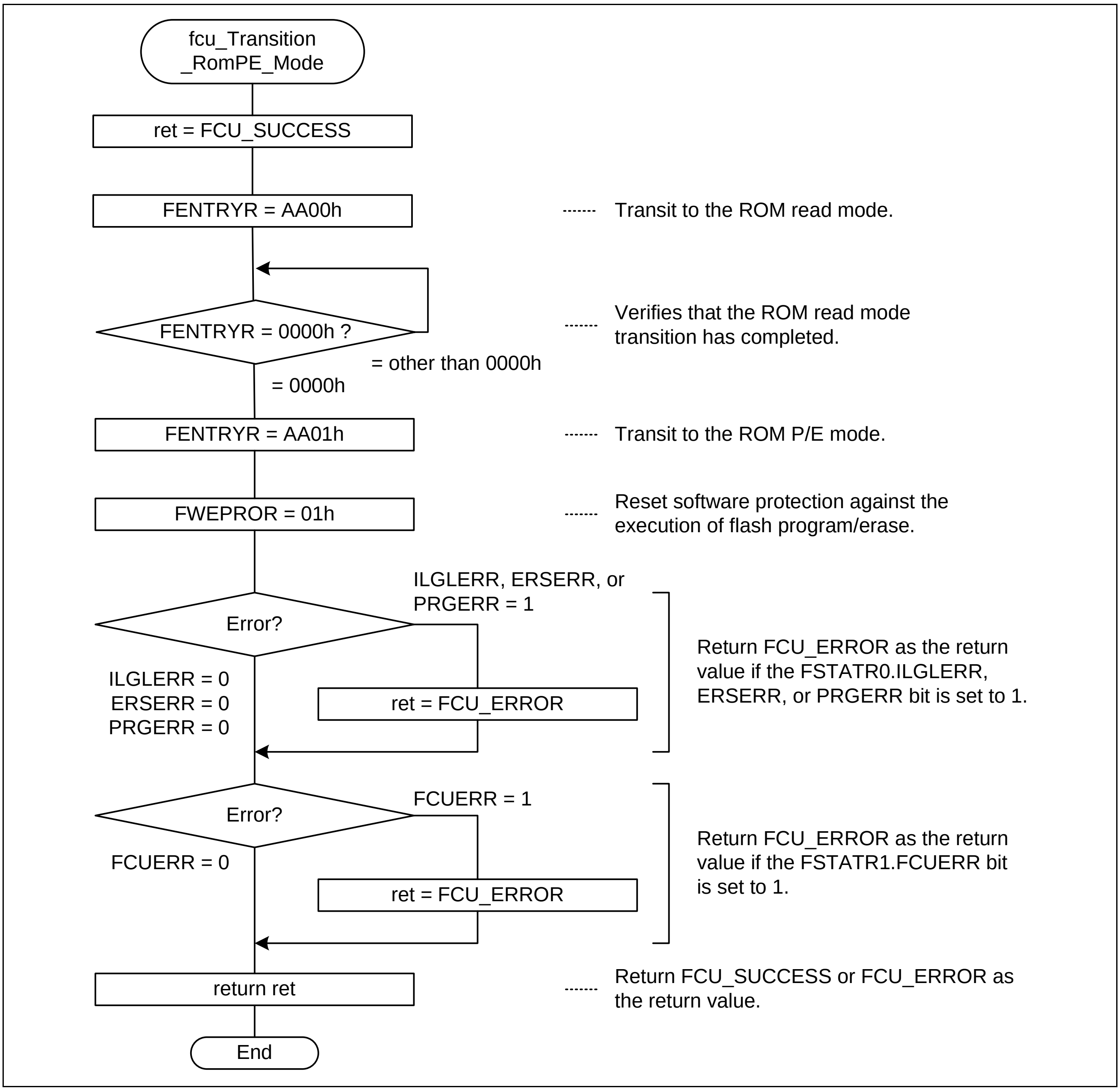

Figure 24 Flowchart (fcu_Transition_RomPE_Mode) (Slave)

(10) fcu_Notify_Peripheral_Clock function

(a) Functional overview

The fcu_Notify_Peripheral_Clock function places the frequency of the peripheral module clock (PCLK) in the PCKAR register and issues a peripheral clock notification command.

(b) Arguments

Table 28 lists the argument that is used by this function.

Table 28 List of fcu_Notify_Peripheral_Clock Function Arguments

Argument	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 29 lists the return value that is returned by this function.

Table 29 List of fcu_Notify_Peripheral_Clock Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

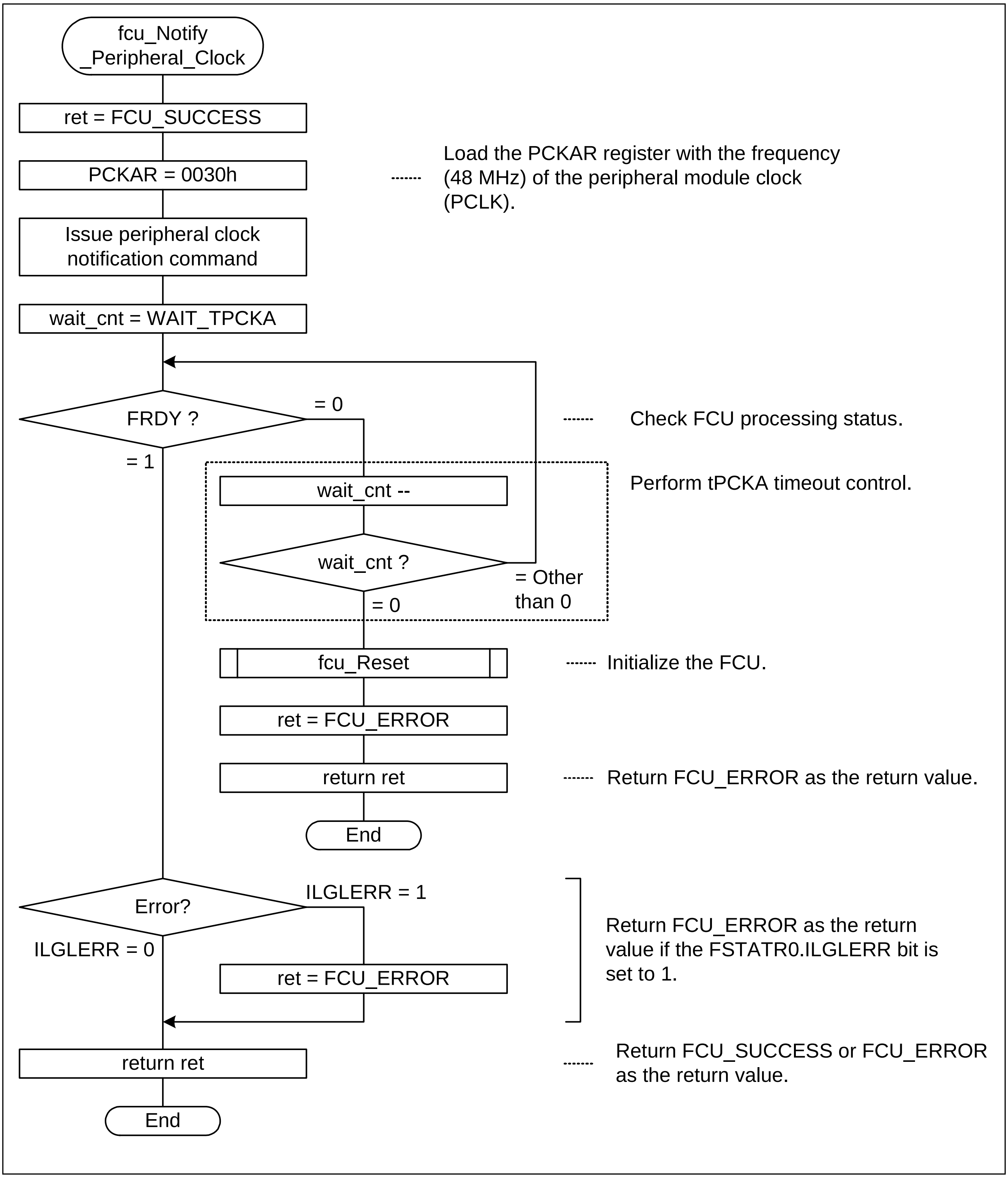

Figure 25 Flowchart (fcu_Notify_Peripheral_Clock) (Slave)

(11) fcu_Erase function

(a) Functional overview

The fcu_Erase function erases the user MAT (in erase block units) using the block erase command.

(b) Arguments

Table 30 lists the argument that is used by this function.

Table 30 List of fcu_Erase Function Arguments

Arguments	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 31 lists the return value that is returned by this function.

Table 31 List of fcu_Erase Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

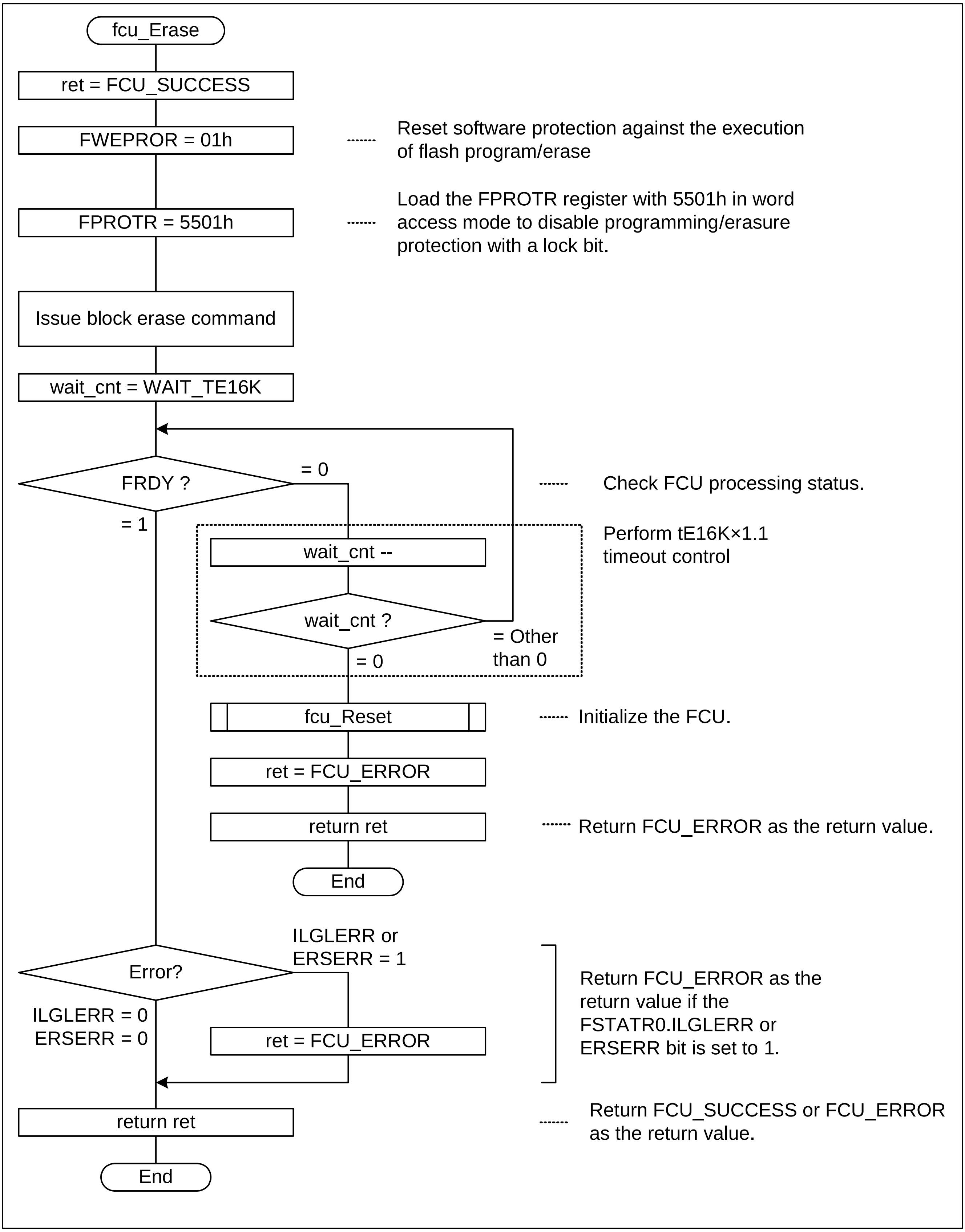

Figure 26 Flowchart (fcu_Erase) (Slave)

(12) fcu_Write function

(a) Functional overview

The fcu_Write function programs data into the user MAT (in 256 byte units) using the program command.

(b) Arguments

Table 32 lists the argument that is used by this function.

Table 32 List of fcu_Write Function Arguments

Argument	Type	Description
First argument	ST_FCU_INFO * (*1)	Address of the structure storing the FCU-related address information to be used during user MAT programming/erasure processing

Note: *1 See 5.6.1, Structure ST_FCU_INFO, for details on the ST_FCU_INFO type.

(c) Return value

Table 33 lists the return value that is returned by this function.

Table 33 List of fcu_Write Function Return Values

Type	Description
FCU_STATUS (*2)	Status established by the execution of the function

Note: *2 See section 5.7, Description of the enum Type, for details on the FCU_STATUS type.

(d) Flowchart

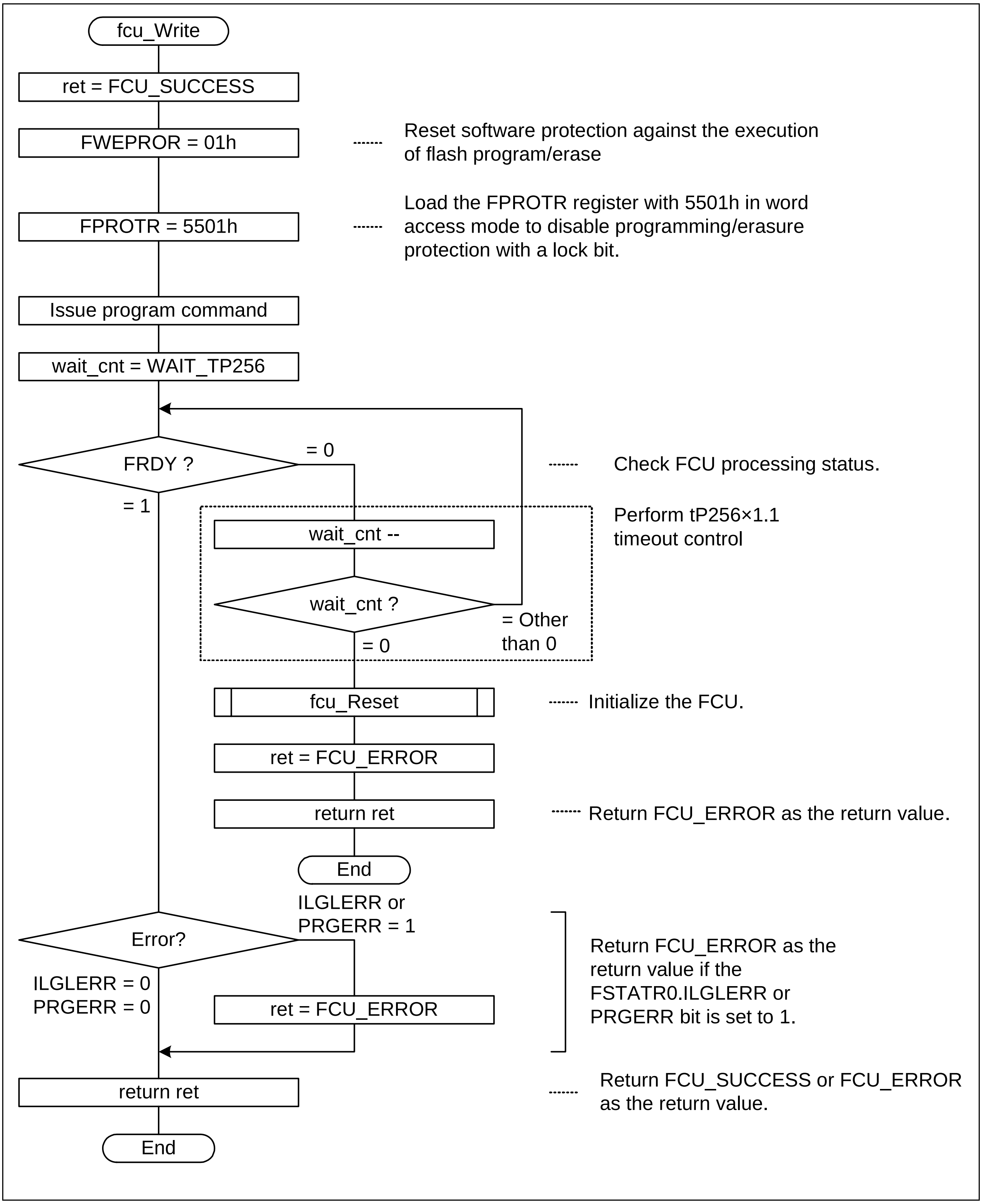

Figure 27 Flowchart (fcu_Write) (Slave)

(13) Indicate_End_LED function

(a) Functional overview

The Indicate_End_LED function displays the normal termination status on LED0 to LED3 when the programming/erasure processing terminates normally. It turns on LED0 to LED3 sequentially, one at a time.

(b) Arguments

None

(c) Return value

None

(d) Flowchart

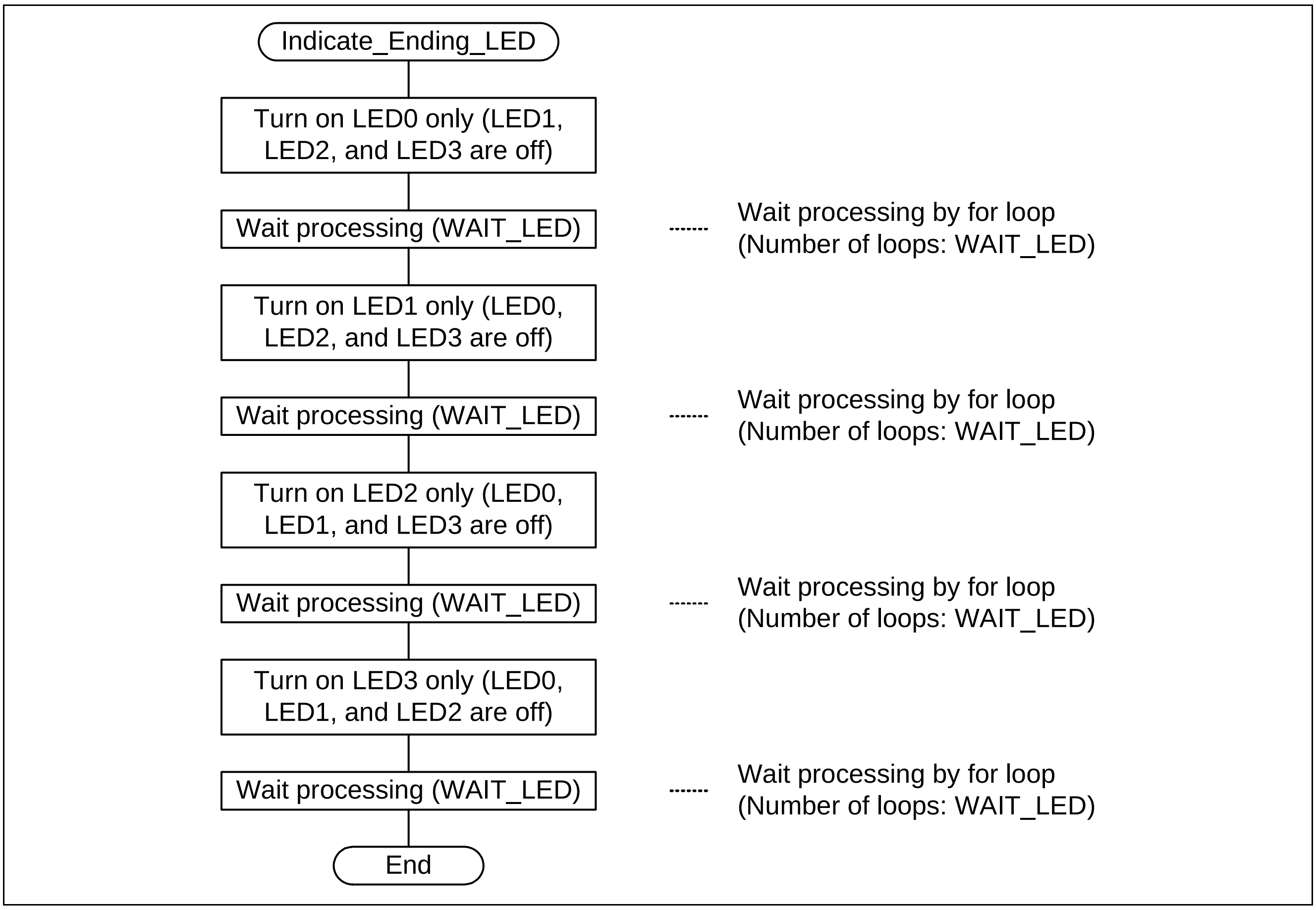

Figure 28 Flowchart (Indicate_End_LED) (Slave)

(14) Indicate_Error_LED function

(a) Functional overview

The Indicate_Error_LED function displays the error number of any error occurring during user MAT programming/erasure processing on LED0 to LED3. It repeats the cycle of displaying the error number on the LEDs and turning off all LEDs.

(b) Arguments

Table 34 lists the argument that is used by this function.

Table 34 Indicate_Error_LED Function Arguments

Argument	Type	Description
First argument	unsigned char	Error number of the error occurring during user MAT programming/erasure processing (*1)

Note: *1 See section 4.6, Error Processing, for the error number.

(c) Return value

None

(d) Flowchart

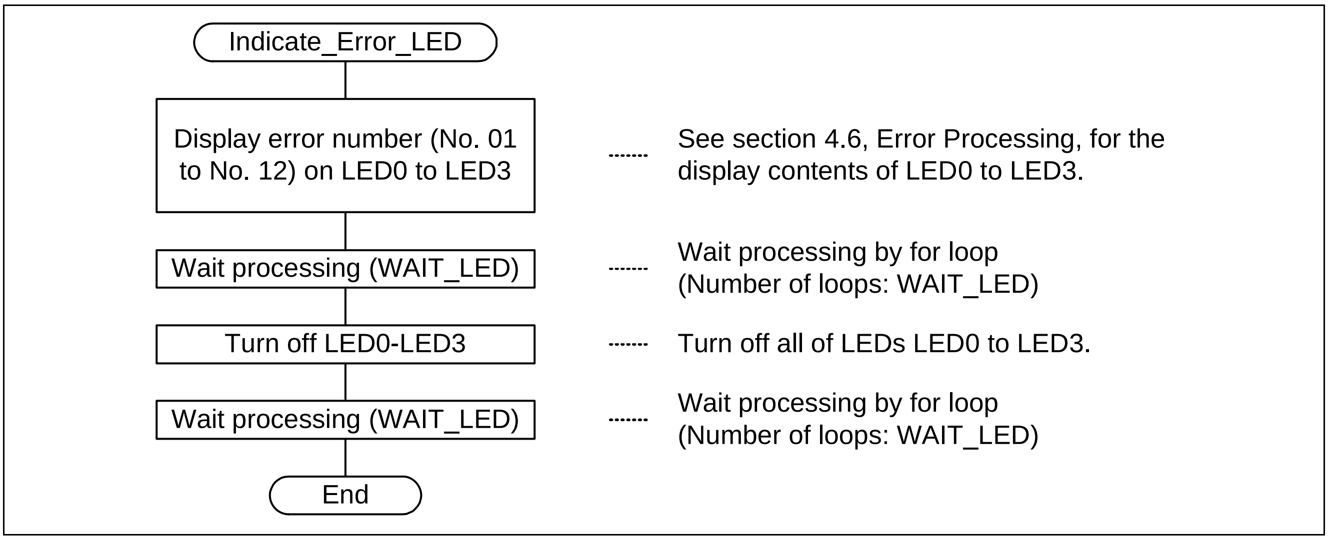

Figure 29 Flowchart (Indicate_Error_LED) (Slave)

(15) SCI_Rcv1byte function

(a) Functional overview

The SCI_Rcv1byte function controls the reception of 1-byte data through the SCI2 clock synchronous communications interface.

(b) Arguments

None

(c) Return value

Table 35 lists the return value that is returned by this function.

Table 35 SCI_Rcv1byte Function Return Values

Type	Description
unsigned char	1-byte data received through the SCI2 clock synchronous communications interface.

(d) Flowchart

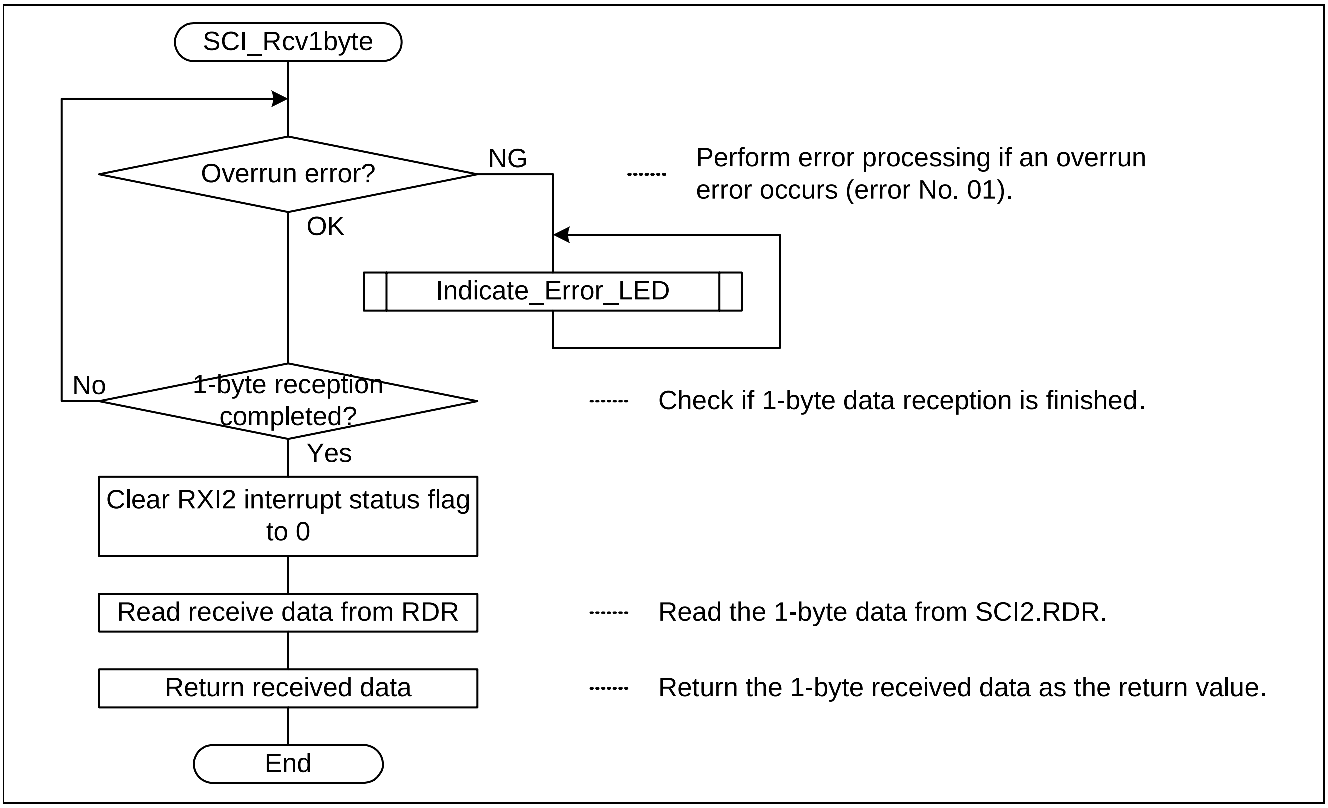

Figure 30 Flowchart (SCI_Rcv1byte) (Slave)

(16) SCI_Rcvnbyte function

(a) Functional overview

The SCI_Rcvnbyte function controls the reception of n-byte data (n is the first argument of the unsigned short type) through the SCI2 clock synchronous communications interface.

(b) Arguments

Table 36 lists the argument that is used by this function.

Table 36 SCI_Rcvnbyte Function Arguments

Argument	Type	Description
First argument	unsigned short	Number of bytes to receive through the SCI2 clock synchronous communications interface.
Second argument	unsigned char *	Start address of the area for storing the received data

(c) Return value

None

(d) Flowchart

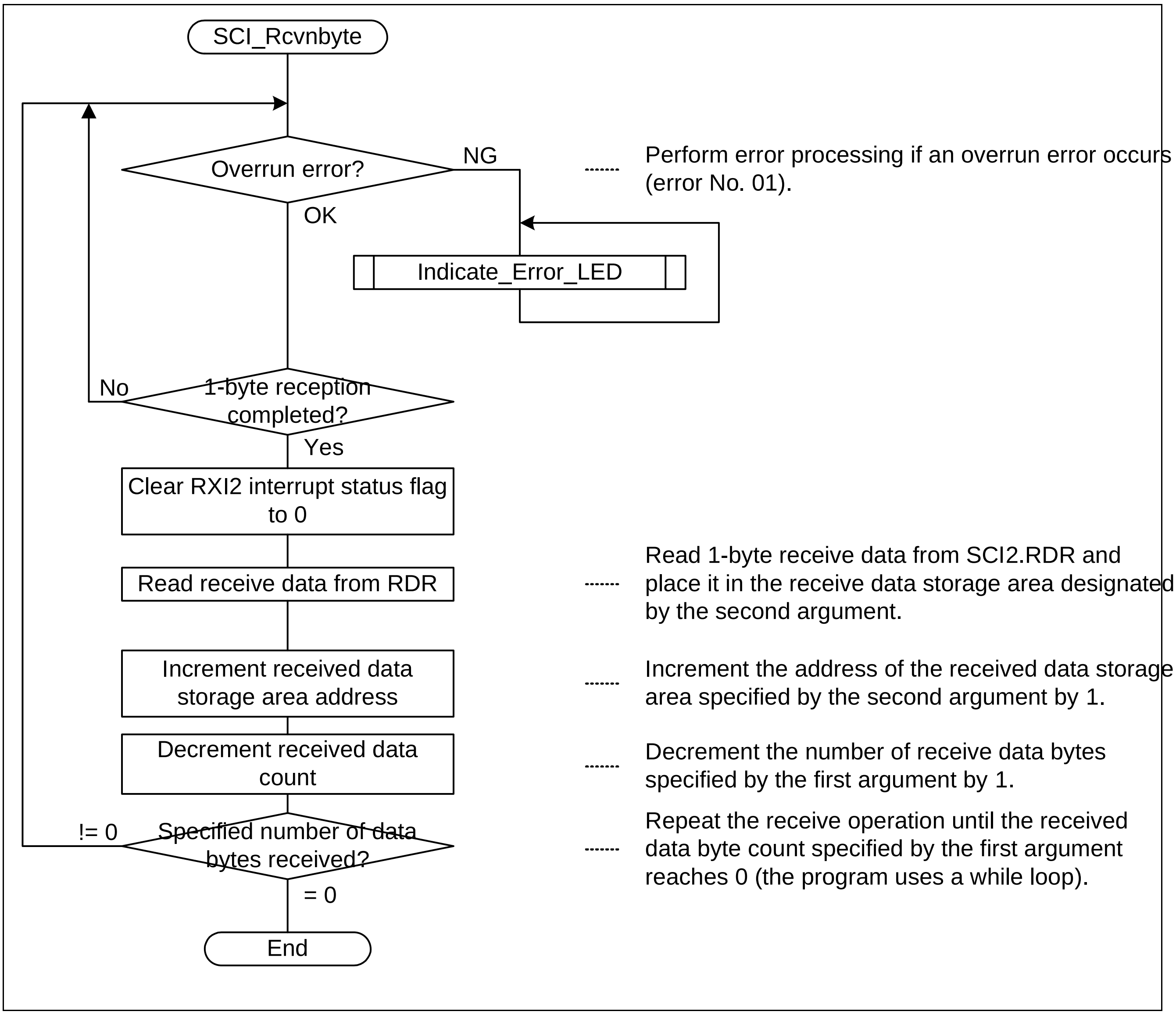

Figure 31 Flowchart (SCI_Rcvnbyte) (Slave)

6. Usage Notes

6.1 Timeout Processing

The example given in this application note exercises some timeout control during user MAT programming/erasure processing. The time measurement for this purpose is accomplished using software timers.

This section explains the types of timeout control used for the example given in this application note.

6.1.1 t_{PCKA} Timeout Control

t_{PCKA} timeout control is exercised when the FCU peripheral clock notification command is issued. In the example given in this application note, the FCU is initialized and error processing is performed if a time longer than t_{PCKA} elapses after a peripheral clock notification command and till the FSTATR0.FRDY bit is set to 1.

In the example given in this application note, the t_{PCKA} wait time is created by cycling through the while loop the number of times defined by the symbolic constant WAIT_TPCKA. Given that the number of cycles taken in one pass through the while loop is 11 cycles (the user can check this in the assembly listing that is generated by the compiler), the number of cycles through the while loop can be calculated using the following formula:

Number of cycles through the while loop = Wait time / (Number of cycles taken in one pass through the while loop × ICLK cycle time)

Since the CPU's instruction execution time varies depending on the type of pipeline processing, the above-mentioned number of cycles taken in one pass through the while loop (11 cycles) becomes equal to an approximate instruction execution time.

tPCKA is 60[μs] for a PCLK frequency of 50 MHz and 120[μs] for a PCLK frequency of 25 MHz. For the example given in this application note, tPCKA is 62.5[μs] since PCLK = 48 MHz.

Since the wait time is calculated to be 187.5[μs] with a wide margin allowed for the example given in this application note, the number of cycles through the while loop is calculated as follows:

Number of cycles through the while loop = WAIT_TPCKA = 187.5[μs] / (11 × 10.41666[ns]) = 1636 (when ICLK = 96 MHz)

Consequently, symbolic constant WAIT_TPCKA is defined as 1636.

When using the example given in this application note, make an extensive evaluation of the CPU's instruction execution time or measure the time in question using a timer.

6.1.2 t_{RESW2} Wait Control

t_{RESW2} wait control is exercised to control, using a software timer, the reset pulse width (t_{RESW2}) occurring during the programming/erasure processing after the FRESETR.FRESET bit is set to 1 till it is cleared to 0 during FCU initialization.

Table 37 lists the reset pulse width occurring during the programming/erasure processing.

Table 37 Reset Pulse Width Occurring during The Programming/Erasure Processing

Item	Symbol	min	max	Unit	Measurement Conditions
Internal reset time (*2)	t_{RESW2} (*1)	35	—	μs	None

Notes: *1 This specification item applies to the FCU reset and WDT reset.
*2 See "Control Signal Timing" of "User's Manual" listed in section 7, Reference Documents, for details

The t_{RESW2} wait time is created by cycling through the while loop the number of times defined by the symbolic constant WAIT_TRESW2. Given that the number of cycles taken in one pass through the while loop is 4 cycles (the user can check this in the assembly listing that is generated by the compiler), the number of cycles through the while loop can be calculated using the following formula:

Number of cycles through the while loop = Wait time / (Number of cycles taken in one pass through the while loop × ICLK cycle time)

Since the CPU's instruction execution time varies depending on the type of pipeline processing, the above-mentioned number of cycles taken in one pass through the while loop (4 cycles) becomes equal to an approximate instruction execution time.

Since the wait time (t_{RESW2}) is calculated to be 105[μs] with a wide margin allowed for the example given in this application note, the number of cycles through the while loop is calculated as follows:

> Number of cycles through the while loop = WAIT_TRESW2 = 105[μs] / (4 × 10.41666 [ns]) = 2520 (when ICLK = 96 MHz)

Consequently, symbolic constant WAIT_TRESW2 is defined as 2520.

When using the example given in this application note, make an extensive evaluation of the CPU's instruction execution time or measure the time in question using a timer.

6.1.3 t_{E16K} × 1.1 Timeout Control

t_{E16K} × 1.1 timeout control is used when transiting the FCU to the ROM read mode and when erasing the user MAT. In the transition to the ROM read mode, the erasure time for the 16K byte erase block before the ROM read mode is being transited by loading the FENTRYR register with AA00h till the FSTATR0.FRDY bit is set to 1 is measured using a software timer. During erasure processing, the erasure time for the 16K byte erasure occurring since a block erase command is issued till the FSTATR0.FRDY bit is set to 1 is measured using a software timer.

Table 38 lists the erasure time for the 16K byte erasure occurring since a block erase command.

Table 38 Erasure Time for The 16K Byte Erasure Occurring Since A Block Erase Command

Item		Symbol	min	typ	max	Unit	Measurement Conditions
Erasure time (*1)	16 KB	t_{E16K}	—	100	240	ms	When PCLK = 50 MHz When No. of erasures per block ≤ 100

Note: *1 See "ROM (Flash Memory for Code Storage) Characteristics" of "User's Manual" listed in section 7, Reference Documents, for details

The t_{E16K} × 1.1 wait time is created by cycling through the while loop the number of times defined by the symbolic constant WAIT_TE16K. Given that the number of cycles taken in one pass through the while loop is 10 cycles (the user can check this in the assembly listing that is generated by the compiler), the number of cycles through the while loop can be calculated using the following formula:

> Number of cycles through the while loop = Wait time / (Number of cycles taken in one pass through the while loop × ICLK cycle time)

Since the CPU's instruction execution time varies depending on the type of pipeline processing, the above-mentioned number of cycles taken in one pass through the while loop (10 cycles) becomes equal to an approximate instruction execution time.

Since the wait time (t_{E16K} × 1.1) is calculated to be 793[ms] with a wide margin allowed for the example given in this application note, the number of cycles through the while loop is calculated as follows:

> Number of cycles through the while loop = WAIT_TE16K = 793[ms] / (10 × 10.41666 [ns]) = 7603200 (when ICLK = 96 MHz)

Consequently, symbolic constant WAIT_TE16K is defined as 7603200.

When using the example given in this application note, make an extensive evaluation of the CPU's instruction execution time or measure the time in question using a timer.

6.1.4 $t_{P256} \times 1.1$ Timeout Control

$t_{P256} \times 1.1$ timeout control is used when programming the user MAT. The 256-byte programming time after a program command is issued till the FSTATR0.FRDY bit is set to 1 is measured using a software timer.

Table 39 lists the 256-byte programming time.

Table 39 256-Byte Programming Time

Item		Symbol	min	typ	max	Unit	Measurement Conditions
Programming time (*1)	256 bytes	t_{P256}	—	2	12	ms	When PCLK = 50 MHz When No. of erasures per block ≤ 100

Note: *1 See "ROM (Flash Memory for Code Storage) Characteristics" of "User's Manual" listed in section 7, Reference Documents, for details

The $t_{P256} \times 1.1$ wait time is created by cycling through the while loop the number of times defined by the symbolic constant WAIT_TP256. Given that the number of cycles taken in one pass through the while loop is 11 cycles (the user can check this in the assembly listing that is generated by the compiler), the number of cycles through the while loop can be calculated using the following formula:

Number of cycles through the while loop = Wait time / (Number of cycles taken in one pass through the while loop × ICLK cycle time)

Since the CPU's instruction execution time varies depending on the type of pipeline processing, the above-mentioned number of cycles taken in one pass through the while loop (11 cycles) becomes equal to an approximate instruction execution time.

Since the wait time ($t_{P256} \times 1.1$) is calculated to be 39.6[ms] with a wide margin allowed for the example given in this application note, the number of cycles through the while loop is calculated as follows:

Number of cycles through the while loop = WAIT_TP256 = 39.6[ms] / (11 × 10.41666 [ns]) = 345600 (when ICLK = 96 MHz)

Consequently, symbolic constant WAIT_TP256 is defined as 345600.

When using the example given in this application note, make an extensive evaluation of the CPU's instruction execution time or measure the time in question using a timer.

6.2 Fixed Vector in User Boot Mode

Transition to the user boot mode occurs when the reset state is released after setting up the user boot mode through the MD1 and MD0 pins. The reset vector in this case is set to address FF7F FFFCh in the user boot MAT. The other entries in the vector table refer to those in the ordinary vector table.

The fixed vector table is a vector table whose address is fixed. Vectors for privileged instruction exception, undefined instruction exception, floating-point exception, nonmaskable interrupt, and reset are placed in addresses FFFF FFD0h to FFFF FFFFh.

Figure 32 illustrates how to set up the fixed vector table for the example given in this application note.

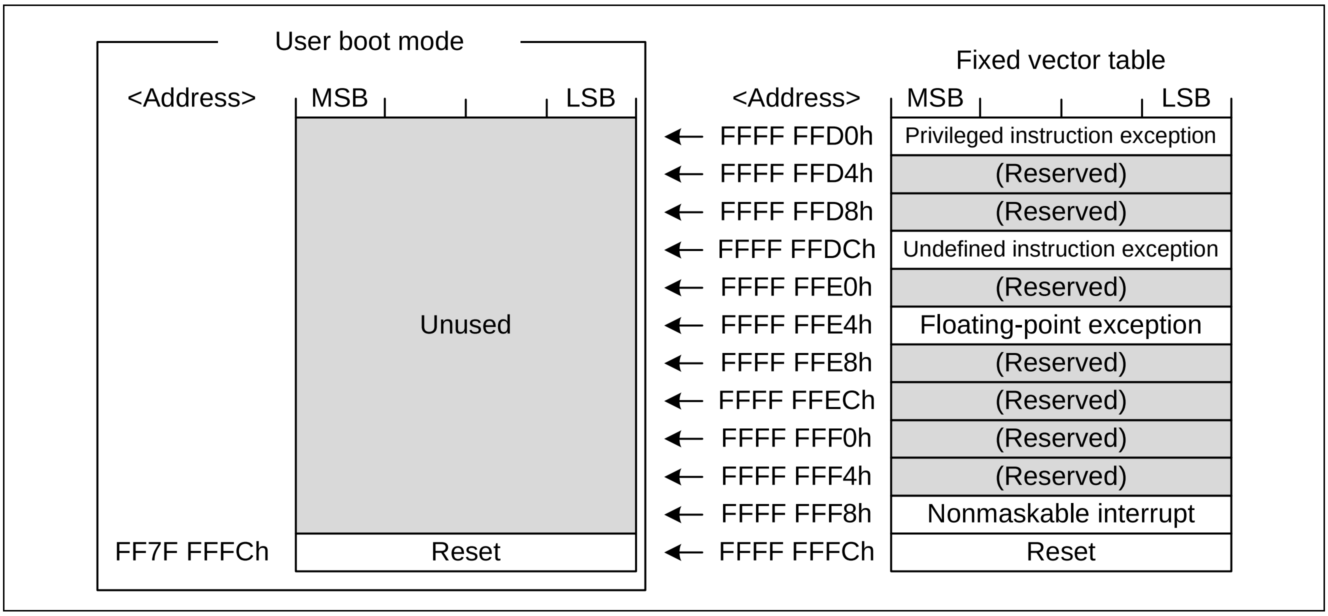

Figure 32 How to Set Up the Fixed Vector Table for this Application Note

The slave referred to in this application note runs in user boot mode. The fixed vector table for the slave is created by commenting out the privileged instruction exception (Excep_SuperVisorInst function), undefined instruction execution (Excep_UndefinedInst function), floating-point exception (Excep_FloatingPoint function), nonmaskable interrupt (NonMaskableInterrupt), and Dummy function in the reserved area in the fixed vector tables in the files (vecttbl.c, vect.h, intprg.c) that are automatically generated by HEW and leaving only the fixed vector (4 bytes) for the reset function and is placed in address FF7F FFFC in the user boot MAT.

The slave program referred to in this application note sets up only the reset vector in the user boot MAT and sets up no fixed vector in the user MAT.

When the user is to use the slave program described in this application note, he or she needs to set up the fixed vectors (for privileged instruction exception, undefined instruction exception, floating-point exception, nonmaskable interrupt, and reserved areas) in the user MAT and the corresponding exception handlers.

6.3 Notes on Reprogramming the Erasure Block EB00

Allocated to the erase block EB00 (programming/erasure addresses: 00FF F000h to 00FF FFFF, read addresses: FFFF F000h to FFFF FFFFh) are the fixed vector (FFFF FF80h to FFFF FFFFh), ID code protection codes (FFFF FFA0h to FFFF FFAFh).

The above-mentioned fixed vectors and ID code protection codes will be temporarily erased if an attempt is made to program or erase EB00 with the erase block number set to EB00_INDEX. It is therefore necessary to make settings for the fixed vectors and ID code protection again after erasing EB00.

ID code protection is a function to disable the reading, programming, and erasure by the host. It makes judgment for ID code protection using the control code and ID code written on ROM. For details on ID code protection, see "User's Manual" listed in section 7, Reference Documents.

7. Reference Documents

- User's Manuals
  RX62N Group, RX621 Group User's Manual: Hardware (R01UH0033EJ)
  (The most up-to-date versions of the documents are available on the Renesas Electronics Website.)

  RX Family User's Manual; Software (REJ09B0435)
  (The most up-to-date versions of the documents are available on the Renesas Electronics Website.)

- Development Environment Manual
  RX Family C/C++ Compiler Package User's Manual (REJ10J2062)
  (The most up-to-date versions of the documents are available on the Renesas Electronics Website.)

- Application Notes
  RX62N Group, RX621 Group
  On-chip Flash Memory Reprogramming in the User Boot Mode (Master) (R01AN0185EJ)
  (The most up-to-date versions of the documents are available on the Renesas Electronics Website.)

- Technical Updates
  (The most up-to-date versions of the documents are available on the Renesas Electronics Website.)

Website and Support

Renesas Electronics Website
  http://www.renesas.com/

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec 17, 2010	—	First edition issued
1.01	Sep.02.11		volatile __evenaccess declaration added
		17	4.9.3 FCU Commands amended
		27	Table 19 amended
		—	Source file (main.c) amended
1.02	Feb.29.12	50	Figure 23: Corrected. ("Verifies that the ROM read mode transition has completed." added.)
		52	Figure 24: Corrected. ("Verifies that the ROM read mode transition has completed." added.)
		—	Source file (main.c) amended

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 accord 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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, 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: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

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: +852-2886-9318, Fax: +852 2886-9022/9044

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1