

## RX610 Group

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

R01AN0223EJ0102  
Rev.1.02  
Feb 29, 2012

#### Introduction

This application note describes programming and erasing the flash memory for code storage (user MAT) of a RX610 Group MCU by using the target erasure block number, write data size, and write data transmitted by clock synchronous serial communication from another RX610 Group MCU, as described in RX610 Group: “On-chip Flash Memory Reprogramming in the User Boot Mode (Master)” (R01AN0210EJ).

For information on transmitting of the target erasure block number, write data size, and write data by clock synchronous serial communication, see RX610 Group: “On-chip Flash Memory Reprogramming in the User Boot Mode (Master)”.

#### Target Device

RX610 Group

This program can be used with other RX Family MCUs that have the same I/O registers (peripheral device control registers) as the RX610 Group. Check the latest version of the manual for any additions and modifications to functions. Careful evaluation is recommended before using this application note.

#### Contents

1. Specifications .....	2
2. Operation Confirmation Environment.....	3
3. Functions Used .....	4
4. Operation.....	4
5. Software Description .....	19
6. Usage Notes.....	61
7. Reference Documents.....	65

### 1. Specifications

- This application note describes writing and erasing the user MAT using an RX610 Group R5F56108VNFP microcontroller in user boot mode.
- The slave receives the erase block number, write data size, and write data using clock synchronous serial communication from the master, and performs the write and erase operations required on the user MAT.
- The SCI channel 0 (SCI0) module is used for clock synchronous serial communication between the master and the slave.
- The clock synchronous serial communication specifications used are a bit rate of 2.5 Mbps, 8 data bits, LSB-first, and transfer clock output by master.
- In this application note’s sample program, the slave erases the specified erase block (EB26: 128 KB) and writes the received 8 KB (256 bytes × 32) of write data to the erased block (EB26) starting at the start address of that block.
- The slave and master perform handshaking for communication control. The slave asserts a busy state (low-level) using an I/O port (P80), and a negate signal (high-level) is output when the busy state is canceled. The master receives output from the slave on an external interrupt pin (IRQ9-A) and starts the next transmission when a rising edge is input.
- When the slave completes the erase and write processing of the user MAT normally, it reports this normal completion using the 4 LEDs connected to the I/O ports. It also reports the error state if an error occurs during communication with the master or during the write/erase processing.

Figure 1 shows the specifications of the system used in this application note.

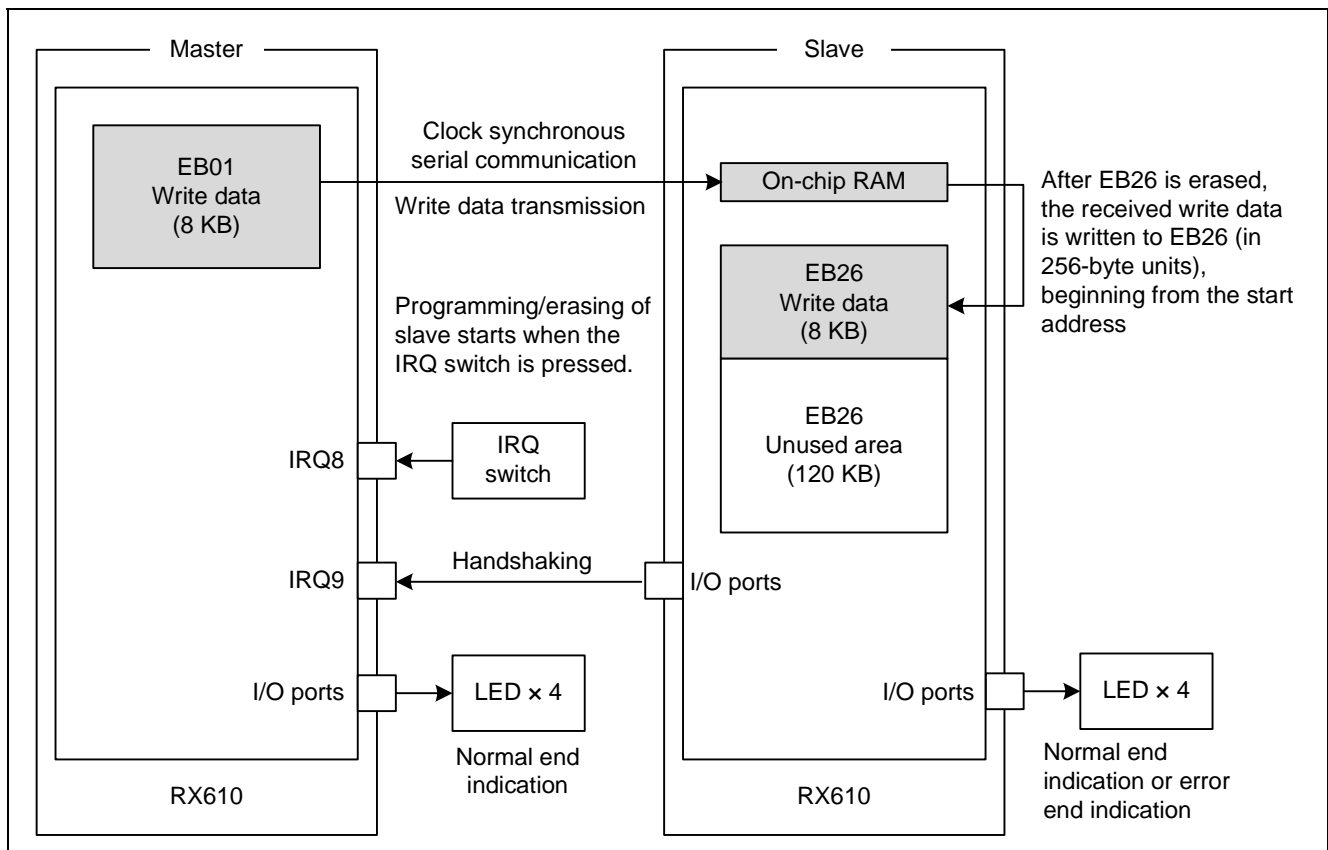
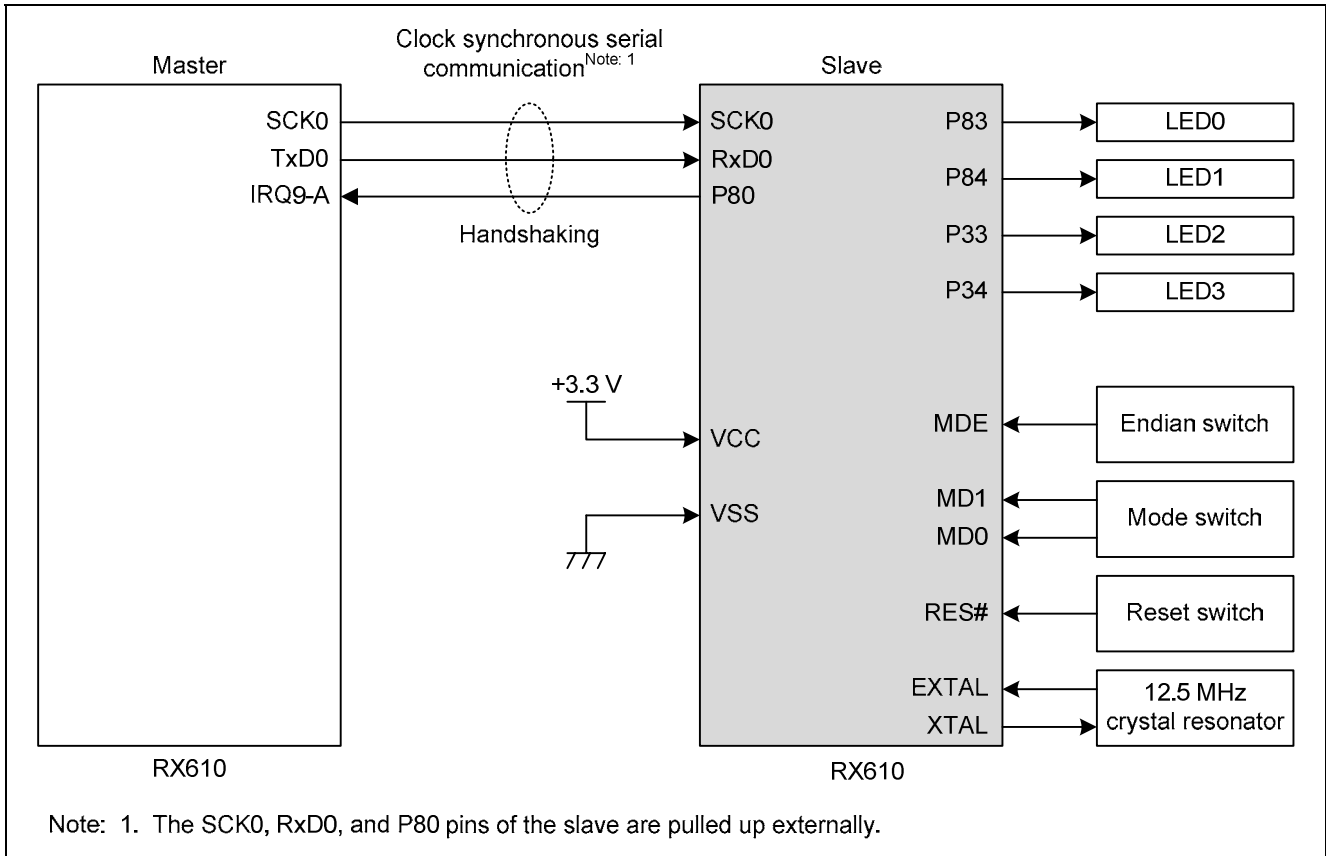


Figure 1 Specifications

Figure 2 shows a hardware configuration diagram of the slave device as used in this application note.



**Figure 2 Hardware Configuration Diagram of Slave Device**

## 2. Operation Confirmation Environment

Table 1 lists the environment required for confirming slave operation.

**Table 1 Slave Operation Confirmation Environment**

Item	Description
Device	RX610 Group: R5F56108VNFP (ROM: 2 MB, RAM: 128 KB)
Board	Evaluation board
Power supply voltage	5.0 V (CPU operating voltage: 3.3 V)
Input clock	12.5 MHz (ICLK = 100 MHz, PCLK = 50 MHz, BCLK = 25 MHz)
Operating temperature	Room temperature
HEW	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 control unit
- I/O ports
- Serial Communications Interface
- ROM (flash memory for code storage)

For details, see the Hardware Manual listed in 7, Reference Documents.

### 4. Operation

#### 4.1 Operation Mode Settings

In the sample program, the slave's mode pins are set to MD1 = 1, MD0 = 0 to select user boot mode as the operating mode, the ROME bit in system control register 0 (SYSCR0) is set to 1 to enable the on-chip ROM, and the EXBE bit in the SYSCR0 register is cleared to 0 to disable the external bus.

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

Table 2 lists the slave operating mode settings used in the sample program.

**Table 2 Operating Mode Settings of Slave Device**

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 initial settings of the ROME and EXBE bits in the SYSCR0 register are SYSCR0.ROME = 1 and SYSCR0.EXBE = 0, so it is not necessary for the sample program to make settings to the SYSCR0 register.

#### 4.2 Clock Settings

The evaluation board used for this application note includes a 12.5 MHz crystal oscillator.

Therefore this application note uses the following settings for the system clock (ICLK), the peripheral module clock (PCLK), and the external bus clock (BCLK):  $8 \times (100 \text{ MHz})$ ,  $4 \times (50 \text{ MHz})$ , and  $2 \times (25 \text{ MHz})$ .

#### 4.3 Endian Mode Setting

The sample program presented in this application note supports both big- and little-endian mode. Table 3 lists the hardware (MDE pin) endian mode settings of the slave device. Note that the master and slave must be set to the same endian mode.

**Table 3 Endian Mode Settings of Slave Device (Hardware)**

MDE pin	Endian
0	Little endian
1	Big endian

Table 4 lists the endian settings used in the compiler options.

**Table 4 Endian Mode Settings of Slave Device (Compiler Options)**

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

Note: Set the MDE bit to match the endian mode selected as a compiler option.

## 4.4 Clock Synchronous Serial Communication Specifications

The sample program uses clock synchronous serial communication for reception of communication commands, erasure block number, write data size, and write data between the master and slave. The transfer clock output by master. The pins used, SCK0 and RxD0 in SCI0, are each pulled up externally.

Table 5 lists the clock synchronous serial communication specifications.

**Table 5 Clock Synchronous Serial Communication Specifications**

Item	Description
Channel	SCI channel 0 (SCI0)
Communication mode	Clock synchronous mode
Bit rate	2.5 Mbps (PCLK = 50 MHz)
Data transfer direction	LSB-first
Error	Overflow error

### 4.4.1 Communication Command Specifications

Table 6 lists the specifications of the communication commands sent between the master and slave.

**Table 6 Communication Command Specifications**

Command	Value	Description	Communication Direction
FSTART	10h	Command to start programming/erasing of the user MAT of the slave	Master → slave
ERASE	11h	Command to start erasing of the user MAT of the slave	Master → slave
WRITE	12h	Command to start programming of the user MAT of the slave	Master → slave

4.4.2 Communication Sequence

Figures 3 to 6 show the communication sequence between master and slave.

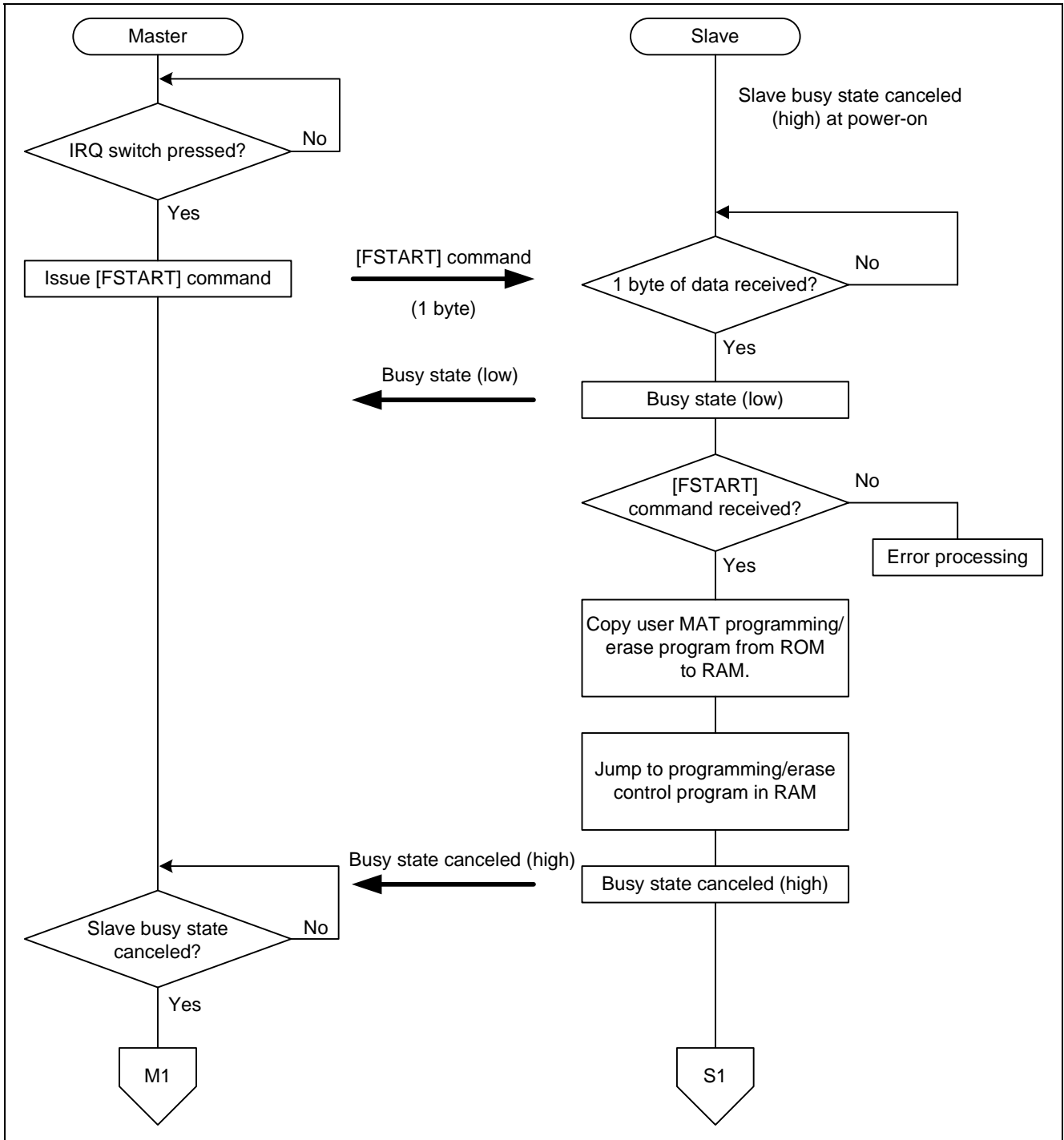


Figure 3 Communication Sequence (1)

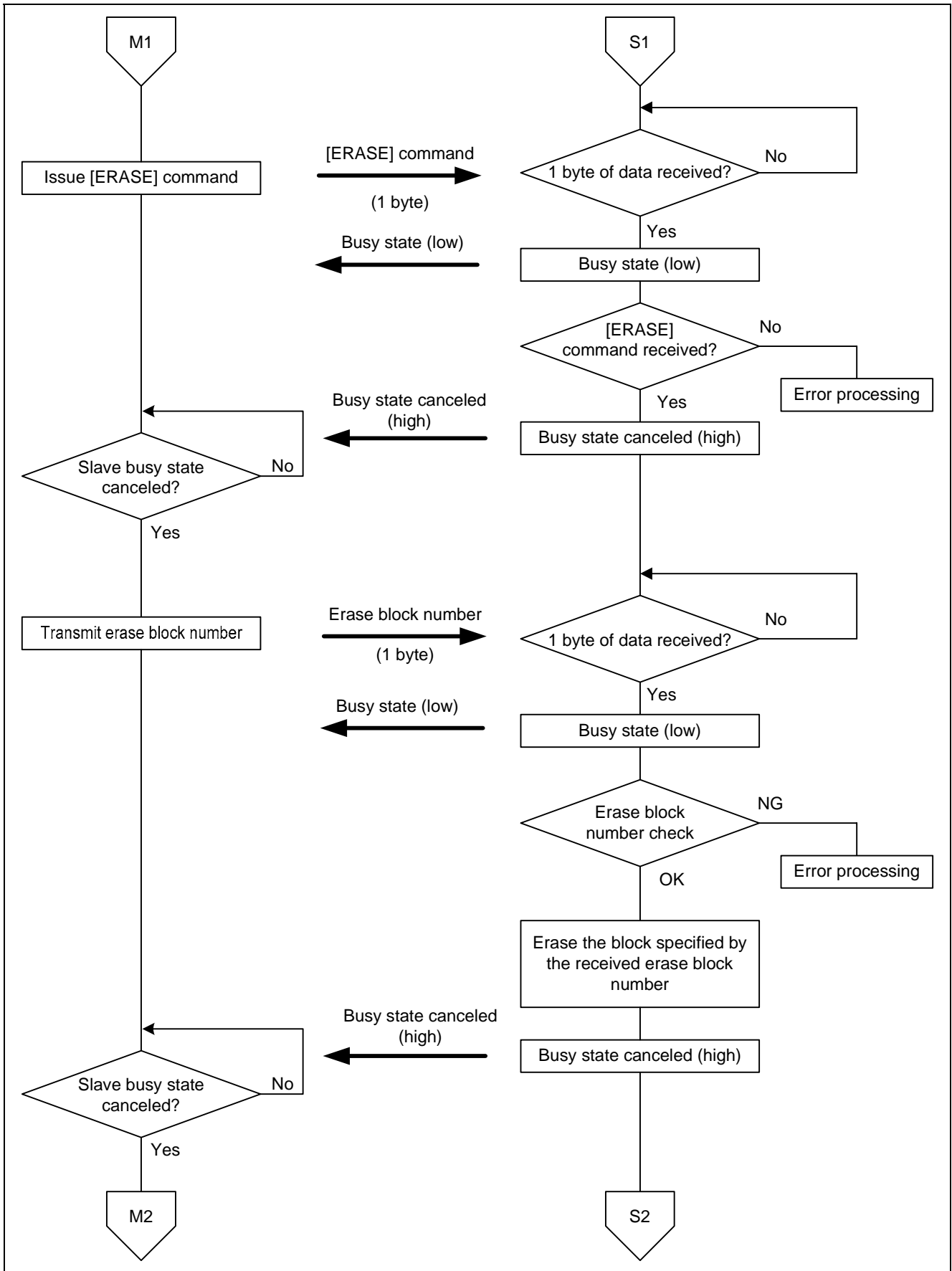


Figure 4 Communication Sequence (2)

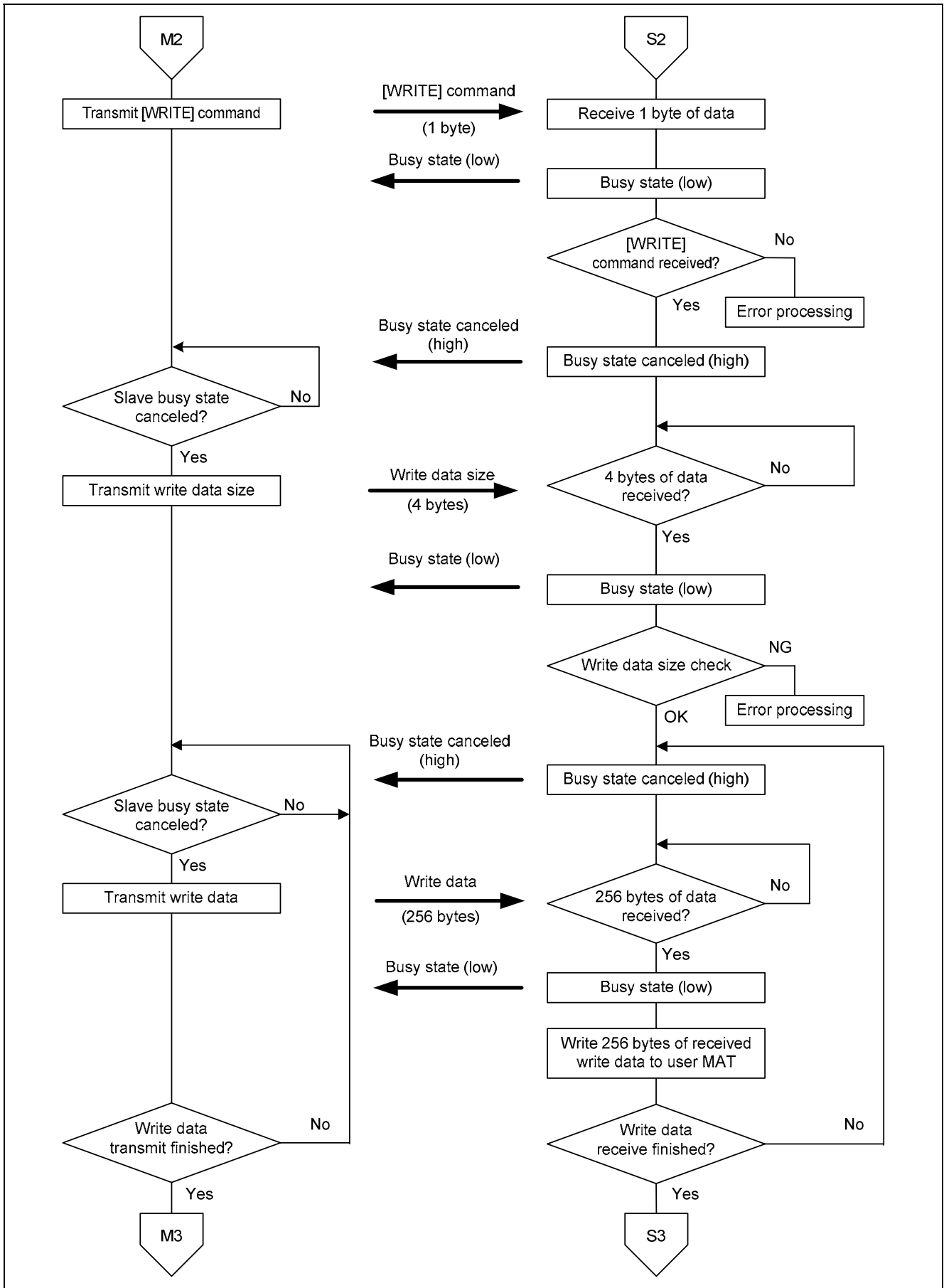


Figure 5 Communication Sequence (3)



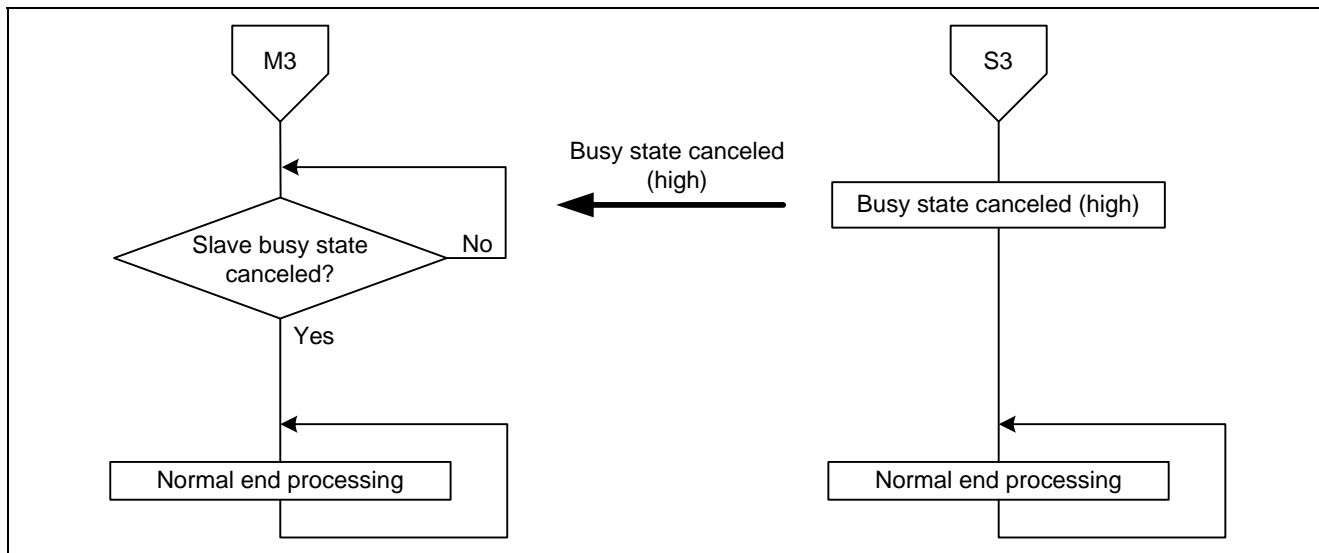


Figure 6 Communication Sequence (4)

### 4.4.3 Erasure Block Number

After receiving an [ERASE] command from the master, the slave receives 1 byte of erasure block number (1 byte of data defined by a symbolic constant). Table 7 lists the erasure block number values. Figure 7 shows the specifications of the erasure block number.

**Table 7 Erasure Block Number Values**

Erasure Block Number		
Symbolic Constant	Value	Description
EB27_INDEX	00h	Specifies erasure block EB27 (size: 128 KB)
EB26_INDEX	01h	Specifies erasure block EB26 (size: 128 KB)
EB25_INDEX	02h	Specifies erasure block EB25 (size: 128 KB)
EB24_INDEX	03h	Specifies erasure block EB24 (size: 128 KB)
EB23_INDEX	04h	Specifies erasure block EB23 (size: 128 KB)
EB22_INDEX	05h	Specifies erasure block EB22 (size: 128 KB)
EB21_INDEX	06h	Specifies erasure block EB21 (size: 128 KB)
EB20_INDEX	07h	Specifies erasure block EB20 (size: 128 KB)
EB19_INDEX	08h	Specifies erasure block EB19 (size: 128 KB)
EB18_INDEX	09h	Specifies erasure block EB18 (size: 128 KB)
EB17_INDEX	0Ah	Specifies erasure block EB17 (size: 128 KB)
EB16_INDEX	0Bh	Specifies erasure block EB16 (size: 64 KB)
EB15_INDEX	0Ch	Specifies erasure block EB15 (size: 64 KB)
EB14_INDEX	0Dh	Specifies erasure block EB14 (size: 64 KB)
EB13_INDEX	0Eh	Specifies erasure block EB13 (size: 64 KB)
EB12_INDEX	0Fh	Specifies erasure block EB12 (size: 64 KB)
EB11_INDEX	10h	Specifies erasure block EB11 (size: 64 KB)
EB10_INDEX	11h	Specifies erasure block EB10 (size: 64 KB)
EB09_INDEX	12h	Specifies erasure block EB09 (size: 64 KB)
EB08_INDEX	13h	Specifies erasure block EB08 (size: 64 KB)
EB07_INDEX	14h	Specifies erasure block EB07 (size: 8 KB)
EB06_INDEX	15h	Specifies erasure block EB06 (size: 8 KB)
EB05_INDEX	16h	Specifies erasure block EB05 (size: 8 KB)
EB04_INDEX	17h	Specifies erasure block EB04 (size: 8 KB)
EB03_INDEX	18h	Specifies erasure block EB03 (size: 8 KB)
EB02_INDEX	19h	Specifies erasure block EB02 (size: 8 KB)
EB01_INDEX	1Ah	Specifies erasure block EB01 (size: 8 KB)
EB00_INDEX	1Bh	Specifies erasure block EB00 (size: 8 KB)

Erasure block number (unsigned char type)

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

The sample program presented in this application note programs and erases erasure block EB26 of the slave, so the erasure block number value is [EB26\_INDEX(01h)].

Note: A value shown in table 7, [EB27\_INDEX(00h)] to [EB00\_INDEX(1Bh)], should be specified as the erasure block number. If a value of [1Ch] to [FFh] is specified as the erasure block number, the slave determines an error to have occurred and error handling takes place.

**Figure 7 Erasure Block Number Specifications**

#### 4.4.4 Write Data Size

After receiving a [WRITE] command from the master, the slave receives 4 bytes of write data size. Figure 8 shows the specifications of the write data size.

Write 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

The sample program uses a write size of 8 KB, so the write data size value is [0000 2000h].

Notes: 1. The write data size must be greater than zero and less than or equal to the erase block size for the specified erase block. If 0 or a value greater than the erase block size is specified, the slave will recognize an error and perform error handling.

2. The size of write data transmissions is fixed at 256 bytes. Consequently, if the write data size specifies a value that is not a multiple of 256 bytes, the master transmits write data in units of 256 bytes and then fills in the final unit of write data, which is less than 256 bytes, with bytes of value FFh as padding to reach a total of 256 bytes, which it transmits to the slave.

**Figure 8 Write Data Size Specifications**

#### 4.4.5 Overrun Error

In this application note, if an overrun error occurs during slave clock synchronous serial communication reception (the SCI0.SSR.ORER bit is set to 1), the slave will perform error handling.

### 4.5 Normal End Processing

When programming/erasing of the user MAT completes successfully, the slave makes a normal end indication by means of four LEDs connected to the I/O ports. The normal end indication consists of LED0 to LED3 illuminating one after another in a sequence that is repeated multiple times.

## 4.6 Error Handling

Table 8 lists the slave device errors that apply to the sample program. Error handling consists of indicating the error state by means of four LEDs connected to the device.

**Table 8 List of Slave Errors**

Error No.	Description	LED Indication			
		LED3	LED2	LED1	LED0
Error No. 01	An overrun error occurred.	Off	Off	Off	On
Error No. 02	In the [FSTART] command wait state, the command received from the master was not an FSTART command.	Off	Off	On	Off
Error No. 03	In the [ERASE] command wait state, the command received from the master was not an ERASE command.	Off	Off	On	On
Error No. 04	The erasure block data received from the master specifies a block other than EB00 to EB27.	Off	On	Off	Off
Error No. 05	A timeout ( $t_{E128K} \times 1.1$ ) occurred during the transition to ROM read mode prior to FCU firmware transfer.	Off	On	Off	On
Error No. 06	Bit ILGLERR, ERSERR, PRGERR, or FCUERR was set to 1 when transitioning to ROM P/E mode before the peripheral clock notification command was issued.	Off	On	On	Off
Error No. 07	A timeout ( $t_{PCKA}$ ) occurred when issuing the peripheral clock notification command, or the ILGLERR bit was set to 1.	Off	On	On	On
Error No. 08	A timeout ( $t_{E128K} \times 1.1$ ) occurred while erasing the erasure block, or the ILGLERR or ERSERR bit was set to 1.	On	Off	Off	Off
Error No. 09	In the [WRITE] command wait state, the command received from the master was not an [WRITE] command.	On	Off	Off	On
Error No. 10	The write data size received from the master was 0 or a value greater than the block size of the specified erase block.	On	Off	On	Off
Error No. 11	A timeout ( $t_{P256} \times 1.1$ ) occurred when writing data, or the ILGLERR or PRGERR bit was set to 1.	On	Off	On	On
Error No. 12	A timeout ( $t_{E128K} \times 1.1$ ) occurred during the transition to ROM read mode after completion of the data write.	On	On	Off	Off

### 4.7 LED Connections

Figure 9 shows the connections of the slave I/O ports and LED0 to LED3.

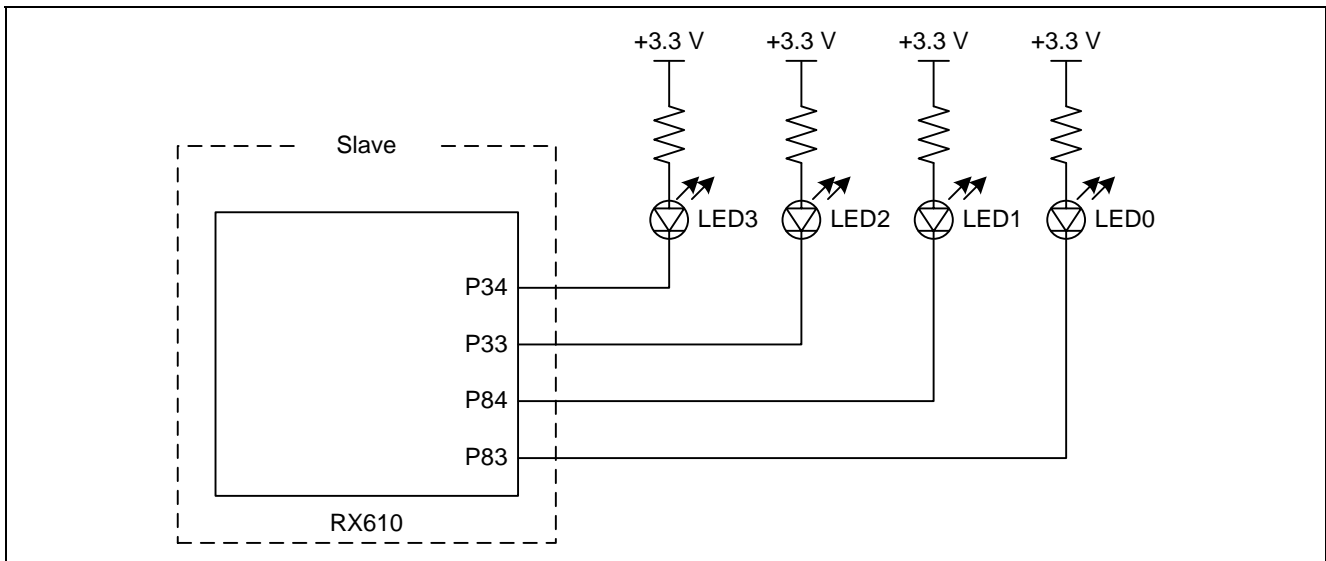


Figure 9 Slave Device LED Connection Diagram

As shown in figure 9, high-level output from an I/O port (P83, P84, P33, or P34) causes the corresponding LED among LED0 to LED3 to turn off, and low-level output causes the corresponding LED to illuminate. Table 9 shows the correspondence between I/O port output and LED states.

Table 9 Slave I/O Port Output and LED States

I/O Port	Register Setting	I/O Port State	LED State
P83	PORT8.DR.B3 = 1, PORT8.DDR.B3 = 1	High-level output	LED0 Off
	PORT8.DR.B3 = 0, PORT8.DDR.B3 = 1	Low-level output	LED0 On
P84	PORT8.DR.B4 = 1, PORT8.DDR.B4 = 1	High-level output	LED1 Off
	PORT8.DR.B4 = 0, PORT8.DDR.B4 = 1	Low-level output	LED1 On
P33	PORT3.DR.B3 = 1, PORT3.DDR.B3 = 1	High-level output	LED2 Off
	PORT3.DR.B3 = 0, PORT3.DDR.B3 = 1	Low-level output	LED2 On
P34	PORT3.DR.B4 = 1, PORT3.DDR.B4 = 1	High-level output	LED3 Off
	PORT3.DR.B4 = 0, PORT3.DDR.B4 = 1	Low-level output	LED3 On

### 4.8 Handshaking Control

The slave performs handshaking with the master for communication control. The handshaking signal is output on the Busy port (P80).

For handshaking control, the slave asserts a busy state (low-level) on the Busy port after receiving serial data from the master. When the slave is ready to receive the next serial transmission, it cancels the busy state of the Busy port (high-level). Table 10 lists the Busy port output levels and states of the slave.

Table 10 Output on Busy Port of Slave

I/O Port	Register Setting	I/O Port State	Busy Port
P80	PORT8.DR.B0 = 1, PORT8.DDR.B0 = 1	High-level output	Negate Busy canceled
	PORT8.DR.B0 = 0, PORT8.DDR.B0 = 1	Low-level output	Assert Busy state

### 4.9 Programming/Erasing User MAT

Programming and erasing of the user MAT by the sample program is described below. For details, see the Hardware Manual listed in 7, Reference Documents.

#### 4.9.1 Memory MAT Configuration of RX610 Group (R5F56108)

The flash memory for code storage of the R5F56108 comprises a 2 MB user MAT and a 16 KB user boot MAT. Figure 10 shows the addresses of the user MAT and user boot MAT of the R5F56108.

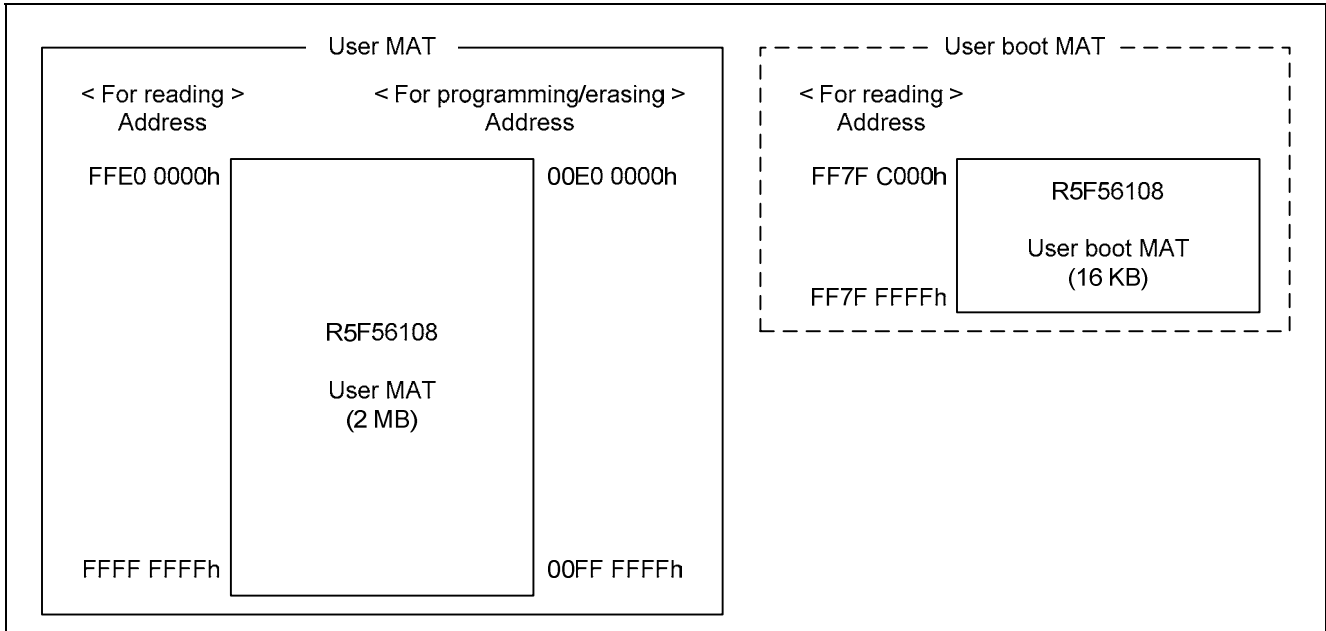


Figure 10 Memory MAT Configuration of R5F56108

**4.9.2 Erasure Block Configuration of RX610 Group (R5F56108)**

The user MAT of the R5F56108 is divided into blocks of three sizes: 128 KB (11 blocks), 64 KB (9 blocks), and 8 KB (8 blocks). Erasing is done in block units.

Programming of the user MAT is done in 256-byte units, starting from the lowest address of 00h.

Table 11 shows the user MAT erasure block configuration of the R5F56108.

**Table 11 Erasure Block Configuration of R5F56108**

Erasure Block	Reading		Programming/Erasing		Size (Bytes)
	Start Address	End Address	Start Address	End Address	
EB27	FFE0 0000h	FFE1 FFFFh	00E0 0000h	00E1 FFFFh	128 K
EB26	FFE2 0000h	FFE3 FFFFh	00E2 0000h	00E3 FFFFh	128 K
EB25	FFE4 0000h	FFE5 FFFFh	00E4 0000h	00E5 FFFFh	128 K
EB24	FFE6 0000h	FFE7 FFFFh	00E6 0000h	00E7 FFFFh	128 K
EB23	FFE8 0000h	FFE9 FFFFh	00E8 0000h	00E9 FFFFh	128 K
EB22	FFEA 0000h	FFEB FFFFh	00EA 0000h	00EB FFFFh	128 K
EB21	FFEC 0000h	FFED FFFFh	00EC 0000h	00ED FFFFh	128 K
EB20	FFEE 0000h	FFEF FFFFh	00EE 0000h	00EF FFFFh	128 K
EB19	FFF0 0000h	FFF1 FFFFh	00F0 0000h	00F1 FFFFh	128 K
EB18	FFF2 0000h	FFF3 FFFFh	00F2 0000h	00F3 FFFFh	128 K
EB17	FFF4 0000h	FFF5 FFFFh	00F4 0000h	00F5 FFFFh	128 K
EB16	FFF6 0000h	FFF6 FFFFh	00F6 0000h	00F6 FFFFh	64 K
EB15	FFF7 0000h	FFF7 FFFFh	00F7 0000h	00F7 FFFFh	64 K
EB14	FFF8 0000h	FFF8 FFFFh	00F8 0000h	00F8 FFFFh	64 K
EB13	FFF9 0000h	FFF9 FFFFh	00F9 0000h	00F9 FFFFh	64 K
EB12	FFFA 0000h	FFFA FFFFh	00FA 0000h	00FA FFFFh	64 K
EB11	FFFB 0000h	FFFB FFFFh	00FB 0000h	00FB FFFFh	64 K
EB10	FFFC 0000h	FFFC FFFFh	00FC 0000h	00FC FFFFh	64 K
EB09	FFFD 0000h	FFFD FFFFh	00FD 0000h	00FD FFFFh	64 K
EB08	FFFE 0000h	FFFE FFFFh	00FE 0000h	00FE FFFFh	64 K
EB07	FFFF 0000h	FFFF 1FFFh	00FF 0000h	00FF 1FFFh	8 K
EB06	FFFF 2000h	FFFF 3FFFh	00FF 2000h	00FF 3FFFh	8 K
EB05	FFFF 4000h	FFFF 5FFFh	00FF 4000h	00FF 5FFFh	8 K
EB04	FFFF 6000h	FFFF 7FFFh	00FF 6000h	00FF 7FFFh	8 K
EB03	FFFF 8000h	FFFF 9FFFh	00FF 8000h	00FF 9FFFh	8 K
EB02	FFFF A000h	FFFF BFFFh	00FF A000h	00FF BFFFh	8 K
EB01	FFFF C000h	FFFF DFFFh	00FF C000h	00FF DFFFh	8 K
EB00	FFFF E000h	FFFF FFFFh	00FF E000h	00FF FFFFh	8 K

**4.9.3 FCU Commands**

Table 12 lists the formats of the FCU commands used in this application note. See the section on ROM (Flash Memory for Code Storage) in the hardware manual for details.

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

**Table 12 FCU Command Formats**

Command	Bus Cycles	First Cycle		Second Cycle		Third Cycle		4th to 5th Cycles		Sixth Cycle		7th to 130th Cycles		131st Cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
P/E normal mode transition	1	RA	FFh	—	—	—	—	—	—	—	—	—	—	—	—
Peripheral clock notification	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	—	—	—	—	—	—	—	—	—	—	—	—

Legend: Address columns: RA: ROM write/erase address  
 WA: ROM write start address  
 BA: ROM erase block address  
 Data columns: WDn: Nth word in the write data (n = 1 to 128)



4.9.4 Procedure for Programming/Erasing the User MAT

Figure 11 shows the procedure used by the sample program to program and erase the user MAT.

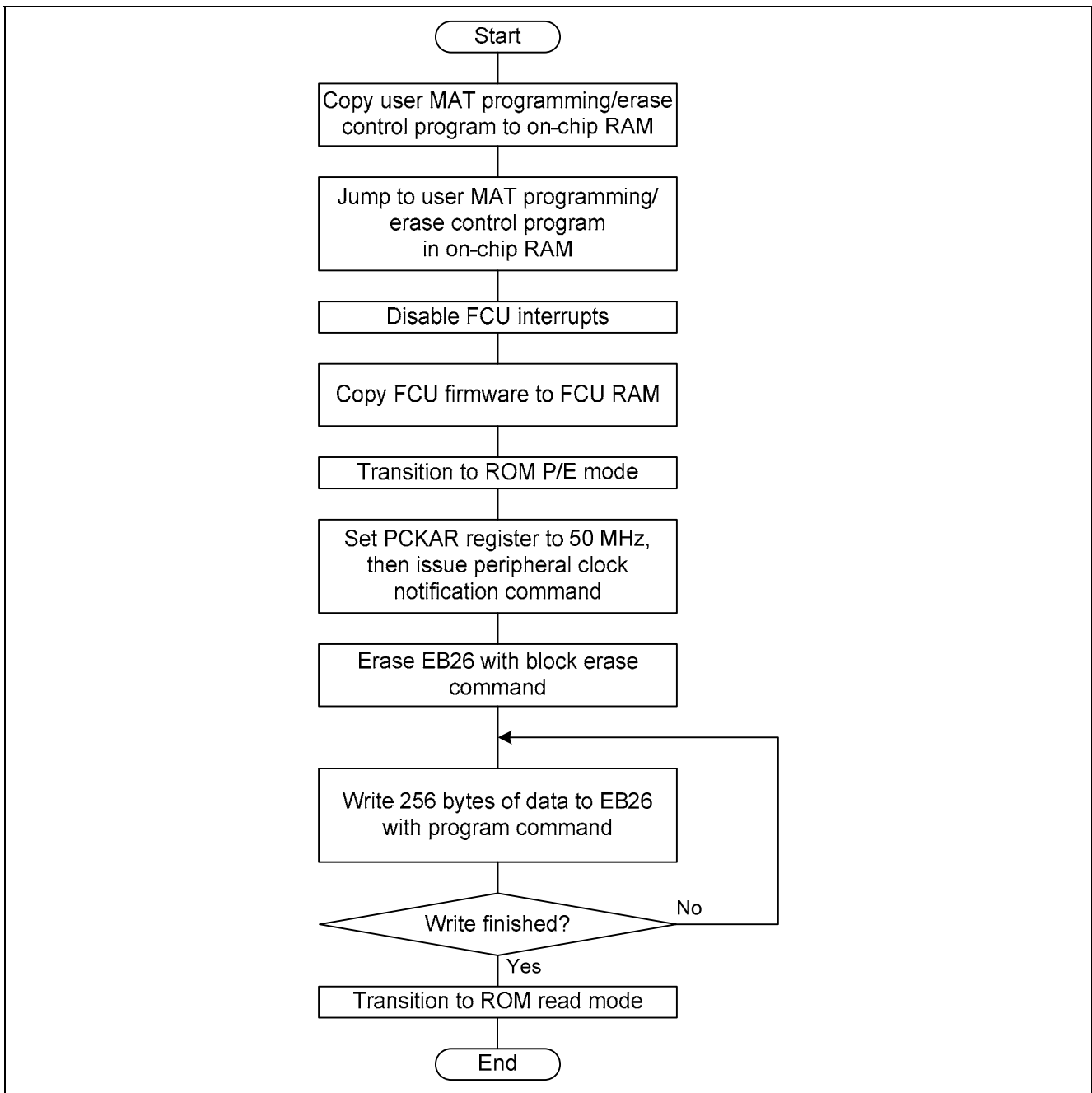


Figure 11 Sample Program Procedure for Programming/Erasing the User MAT of the Slave Device

## 4.10 Section Settings

Table 13 shows the section settings for the slave device.

**Table 13 Section Settings of Slave Device**

Section	Start Address	Description
B_1	0000 1000h	Uninitialized data area (ALIGN = 1)
B		Uninitialized data area (ALIGN = 4)
R		Area in RAM to which [D] section is mapped by ROM option
SU		User stack area
SI		Interrupt stack area
RF_UPDATE_FUNC		Area in RAM to which [PF_UPDATE_FUNC] section is mapped by ROM option
PRResetPRG	FF7F C000h	Program area (PowerON_Reset_PC program)
C	FF7F C100h	Constant area (ALIGN = 4)
C\$DSEC		Section initialization table of initialized data area
C\$BSEC		Section initialization table of uninitialized data area
C\$VECT		Relocatable vector area
D		Initialized data area (ALIGN = 4)
P		Program area
PIntPRG		Program area (interrupt program)
PF_UPDATE_FUNC		Program area (user MAT programming/control program)
FIXEDVECT	FF7F FFFCh	Fixed vector area (reset vector)

## 5. Software Description

### 5.1 File Structure

Table 14 shows the file structure of the slave device. In addition to the files listed in table 14, some files generated automatically by HEW are used as well.

**Table 14 File Structure of Slave Device**

File Name	Description
resetprg.c <sup>Note: 1</sup>	Initial settings
vecttbl.c <sup>Note: 2</sup>	Fixed vector table
vect.h <sup>Note: 3</sup>	Interrupt handler function definitions
intprg.c <sup>Note: 4</sup>	Interrupt handler
main.c	This program handles the following operations: communication commands using clock synchronous serial communication with the master; reception control for the erase block number, the write data size, and the write data; control of erase and write operations for blocks in the user MAT; control of LED display at normal completion and when an error occurs.

Notes: 1. This file is generated automatically by HEW. In the sample program it has been edited to restore a line in the PowerON\_Reset\_PC function calling the HardwareSetup function, which was originally commented out. In the edited version the HardwareSetup function in the main.c file is called from the PowerON\_Reset\_PC function.

2. This file is generated automatically by HEW. In the sample program it has been edited by commenting out the fixed vector table definitions for the privileged instruction exception, undefined instruction exception, floating point exception, non-maskable interrupt, and reserved area dummy function, leaving only the reset vector definitions.

3. This file is generated automatically by HEW. In the sample program it has been edited by commenting out the definitions for the Excep\_SuperVisorInst function, Dummy function, Excep\_UndefinedInst function, Excep\_FloatingPoint function, and NonMaskableInterrupt function.

4. This file is generated automatically by HEW. In the sample program it has been edited by commenting out the code for the Excep\_SuperVisorInst function, Dummy function, Excep\_UndefinedInst function, Excep\_FloatingPoint function, and NonMaskableInterrupt function.

## 5.2 Function Structure

Table 15 lists the functions for the slave device and figure 12 shows the hierarchy of these functions.

**Table 15 Slave Device Functions**

Function	File Name	Description
PowerON_Reset_PC	resetprg.c	Initial settings function
HardwareSetup	main.c	MCU initial settings function
main	main.c	Main function
Flash_Update	main.c	User MAT programming/erase 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 erase control function
fcu_Write	main.c	User MAT programming control function
Indicate_Ending_LED	main.c	Normal end processing function
Indicate_Error_LED	main.c	Error end processing function
SCI_Rcv1byte	main.c	1 byte data reception function
SCI_Rcvnbyte	main.c	n byte data reception function

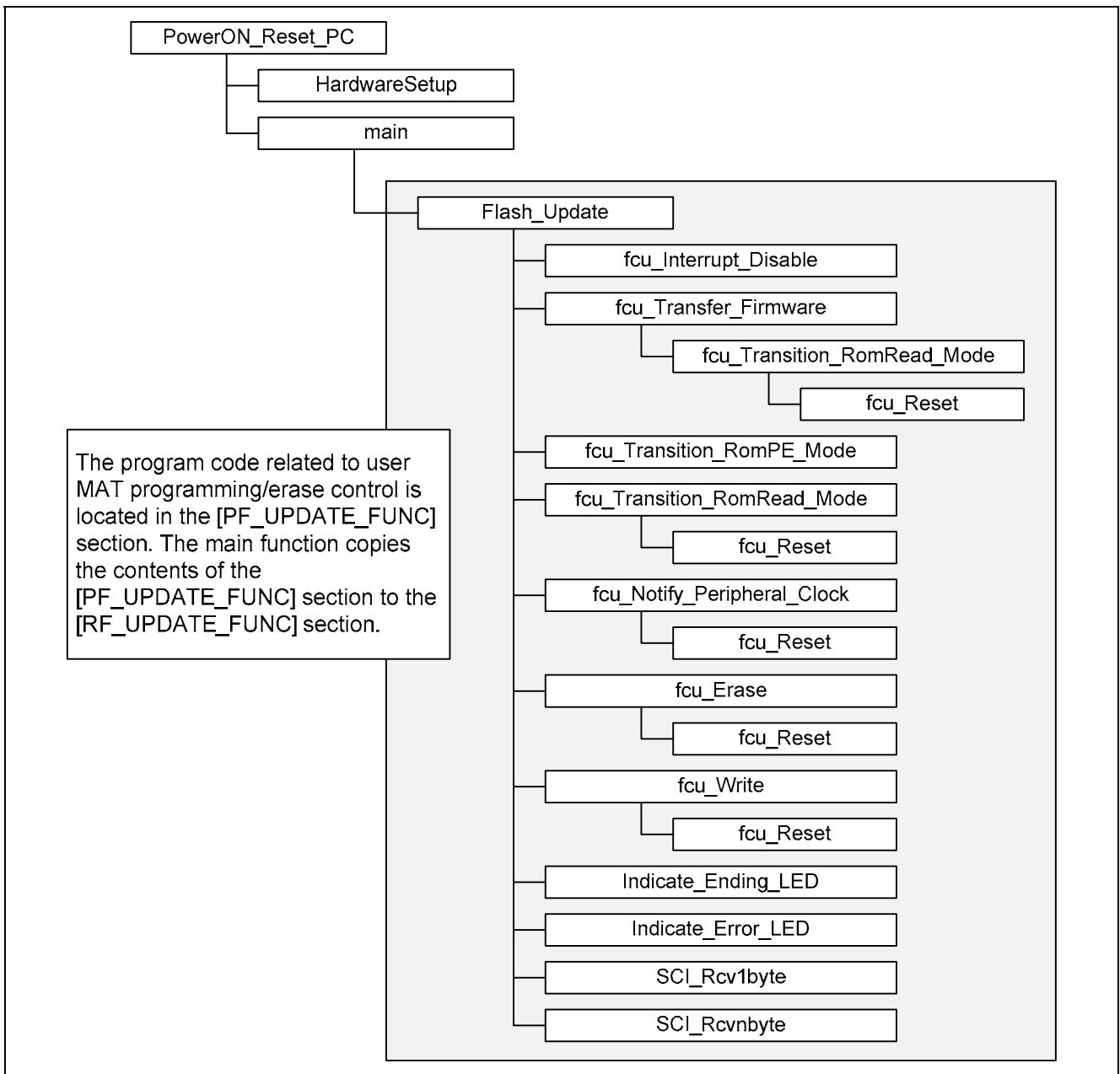


Figure 12 Hierarchy of Slave Device Functions

### 5.3 Symbolic Constants

Table 16 lists the symbolic constants used by the slave device.

**Table 16 Symbolic Constants of Slave Device**

Symbolic Constant	Setting Value	Description	Functions Used By
FSTART	0x10	Programming/erase start command	main
ERASE	0x11	Erase start command	Flash_Update
WRITE	0x12	Programming start command	Flash_Update
LED_ON	0	Set value used when the LED is on	main Indicate_Ending_LED Indicate_Error_LED
LED_OFF	1	Set value used when the LED is off	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED0	PORT8.DR.BIT.B3	On/off control of LED 0 on the evaluation board	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED1	PORT8.DR.BIT.B4	On/off control of LED 1 on the evaluation board	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED2	PORT3.DR.BIT.B3	On/off control of LED 2 on the evaluation board	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED3	PORT3.DR.BIT.B4	On/off control of LED 3 on the evaluation board	HardwareSetup main Indicate_Ending_LED Indicate_Error_LED
RSK_LED0_DDR	PORT8.DDR.BIT.B3	I/O control for LED 0 on the evaluation board	HardwareSetup
RSK_LED1_DDR	PORT8.DDR.BIT.B4	I/O control for LED 1 on the evaluation board	HardwareSetup
RSK_LED2_DDR	PORT3.DDR.BIT.B3	I/O control for LED 2 on the evaluation board	HardwareSetup
RSK_LED3_DDR	PORT3.DDR.BIT.B4	I/O control for LED 3 on the evaluation board	HardwareSetup
ASSERT	0	Setting value at Busy port assert	main Flash_Update
NEGATE	1	Setting value at Busy port negate	main Flash_Update
SLAVE_BUSY	PORT8.DR.BIT.B0	Busy port output control	HardwareSetup main Flash_Update
SLAVE_BUSY_DDR	PORT8.DDR.BIT.B0	Busy port I/O control	HardwareSetup
PCKA_50MHZ	0x0032	Peripheral module clock (PCLK) frequency data set in PCKAR register	fcu_Notify_Peripheral_Clock

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

Symbolic Constant	Setting Value	Description	Functions Used By
WAIT_TE128K	57750000	Timeout (tE128K × 1.1) data tE128K: Time required to erase a 128 KB erasure block	fcu_Transition_ RomRead_Modefcu_ Erase
WAIT_TP256	360000	Timeout (tP256 × 1.1) data tP256: Time required to write 256 bytes of data	fcu_Write
WAIT_TRESW2	2625	Wait (tRESW2) data tRESW2: Reset pulse width during programming/erase	fcu_Reset
WAIT_TPCKA	1636	Timeout (tPCKA) data	fcu_Notify_Peripheral_ Clock
WAIT_LED	2000000	LED illumination interval data for indication of successful completion of programming/erasing of slave user MAT	Indicate_Ending_LED Indicate_Error_LED
FCU_FIRM_TOP	0xFEFFE000	Start address of 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	Write size for programming user MAT (word units)	Flash_Update fcu_Program_Verify
BUF_SIZE	256	Size of write data storage area	—
ERROR_NO_01	1	Data indicating error state	Flash_Update
ERROR_NO_02	2		Indicate_Error_LED
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		
EB27_INDEX	0x00	Erasure block data transmitted to specify the erasure block of the slave to be programmed/erased	Flash_Update
EB26_INDEX	0x01		
EB25_INDEX	0x02		
EB24_INDEX	0x03		
EB23_INDEX	0x04		
EB22_INDEX	0x05		
EB21_INDEX	0x06		
EB20_INDEX	0x07		
EB19_INDEX	0x08		
EB18_INDEX	0x09		
EB17_INDEX	0x0A		
EB16_INDEX	0x0B		
EB15_INDEX	0x0C		
EB14_INDEX	0x0D		
EB13_INDEX	0x0E		
EB12_INDEX	0x0F		
EB11_INDEX	0x10		

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

Symbolic Constant	Setting Value	Description	Functions Used By
EB10_INDEX	0x11	Erasure block data transmitted to specify the erasure block of the slave to be programmed/erased	Flash_Update
EB09_INDEX	0x12		
EB08_INDEX	0x13		
EB07_INDEX	0x14		
EB06_INDEX	0x15		
EB05_INDEX	0x16		
EB04_INDEX	0x17		
EB03_INDEX	0x18		
EB02_INDEX	0x19		
EB01_INDEX	0x1A		
EB00_INDEX	0x1B		

### 5.4 Constant Variables

Table 17 lists the constant variables used by the slave device.

**Table 17 Constant Variables of Slave Device**

Constant	Type	Description
tblEbAdrs[]	ST_EB_ADRS <sup>Note: 1</sup>	Data for the erasure block (EB00 to EB27) start address and end address for programming/erasing, start address and end address for reading, and erasure block size (560 bytes)

Note: 1. For details on the ST\_EB\_ADRS type, see 5.6.2, ST\_EB\_ADRS Structure.

### 5.5 RAM Variables

Table 18 lists the RAM variables used by the slave device.

**Table 18 RAM Variables of Slave Device**

Variable	Type	Description
wrdata_buffer [BUF_SIZE]	unsigned char	Array for storing 256 bytes of write data received from the slave (256 bytes)
fcu_info	ST_FCU_INFO <sup>Note: 1</sup>	Structure for storing FCU-related address information used to program/erase the user MAT (28 bytes)
p_write_buffer	unsigned short *	Address of write data storage area used when programming user MAT: 4 bytes
p_command_adrs	unsigned char *	FCU command destination address (programming/erase address): 4 bytes
p_erase_adrs	unsigned short *	Start address of erase target erasure block (programming/erase address): 4 bytes
p_write_adrs_top	unsigned short *	Start address of programming target erasure block (programming/erase address): 4 bytes
p_write_adrs_end	unsigned short *	End address of programming target erasure block (programming/erase address): 4 bytes
p_write_adrs_now	unsigned short *	Programming target address (programming/erase address): 4 bytes
eb_block_size	unsigned long	Block size of the target erase block: 4 bytes

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.



## 5.6 Structures

### 5.6.1 ST\_FCU\_INFO Structure

Table 19 lists the specifications of the ST\_FCU\_INFO structure used by the slave device.

**Table 19 ST\_FCU\_INFO Structure Specifications**

Member	Type	Description
p_write_buffer	unsigned short *	Address of write data storage area used when programming user MAT
p_command_adrs	volatile __evenaccess unsigned char *	FCU command destination address (programming/erase address)
p_erase_adrs	unsigned short *	Start address of erase target erasure block (programming/erase address)
p_write_adrs_top	unsigned short *	Start address of programming target erasure block (programming/erase address)
p_write_adrs_end	unsigned short *	End address of programming target erasure block (programming/erase address)
p_write_adrs_now	unsigned short *	Address of write data storage area used when programming user MAT
eb_block_size	unsigned long	Block size of target erase block

### 5.6.2 ST\_EB\_ADRS Structure

Table 20 lists the specifications of the ST\_EB\_ADRS structure used by the slave device.

**Table 20 ST\_EB\_ADRS Structure Specifications**

Member	Type	Description
eb_write_adrs_top	unsigned long	Start address of erasure block (programming/erase)
eb_write_adrs_end	unsigned long	End address of erasure block (programming/erase)
eb_read_adrs_top	unsigned long	Start address of erasure block (read)
eb_read_adrs_end	unsigned long	End address erasure block (read)
eb_block_size	unsigned long	Size of erasure block

## 5.7 Enumerated Types

Table 21 lists the members of the enumerated type FCU\_STATUS used by the slave device. The return value of the function indicates the status.

**Table 21 Enumerated Type FCU\_STATUS Specifications**

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

## 5.8 I/O Registers

The I/O registers of the slave device used by the sample program are listed below. Note that the setting values shown are those used in the sample program and differ from the initial setting values.

### (1) Clock Generation Circuit

#### System Clock Control Register (SCKCR)

Number of Bits: 32    Address: 0008 0020h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b11 to b8	PCK[3:0]	0001	Peripheral module clock (PCLK) select bits	0001: × 4 PCLK = 50 MHz (when EXTAL clock = 12.5 MHz)	R/W
b19 to b16	BCK[3:0]	0010	External bus clock (BCLK) select bits	0010: × 2 BCLK = 25 MHz (when EXTAL clock = 12.5 MHz)	R/W
b23	PSTOP1	0	BCLK output stop bit	0: BCLK output	R/W
b27 to b24	ICK[3:0]	0000	System clock (ICLK) select bits	0000: × 8 ICLK = 100 MHz (when EXTAL clock = 12.5 MHz)	R/W

### (2) I/O Ports

#### Port 8 Data Register (P8.DR)

Number of Bits: 8    Address: 0008 C028h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	B0	0 1	P80 output data storage bit	0: Output data = 0 1: Output data = 1	R/W
b3	B3	0 1	P83 output data storage bit	0: Output data = 0 1: Output data = 1	R/W
b4	B4	0 1	P84 output data storage bit	0: Output data = 0 1: Output data = 1	R/W

#### Port 3 Data Register (P3.DR)

Number of Bits: 8    Address: 0008 C023h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b3	B3	0 1	P33 output data storage bit	0: Output data = 0 1: Output data = 1	R/W
b4	B4	0 1	P34 output data storage bit	0: Output data = 0 1: Output data = 1	R/W

#### Port 8 Data Direction Register (P8.DDR)

Number of Bits: 8    Address: 0008 C008h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	B0	1	P80 I/O select bit	1: Output port	R/W
b3	B3	1	P83 I/O select bit	1: Output port	R/W
b4	B4	1	P84 I/O select bit	1: Output port	R/W

## **RX610 Group      On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)**

### **Port 3 Data Direction Register (P3.DDR)**

**Number of Bits: 8**

**Address: 0008 C003h**

<b>Bit</b>	<b>Symbol</b>	<b>Setting Value</b>	<b>Bit Name</b>	<b>Function</b>	<b>R/W</b>
b3	B3	1	P33 I/O select bit	1: Output port	R/W
b4	B4	1	P34 I/O select bit	1: Output port	R/W

### **Port 2 Input Buffer Control Register (P2.ICR)**

**Number of Bits: 8**

**Address: 0008 C062h**

<b>Bit</b>	<b>Symbol</b>	<b>Setting Value</b>	<b>Bit Name</b>	<b>Function</b>	<b>R/W</b>
b1	B1	1	P21 input buffer control bit	1: P21 input buffer enabled	R/W
b2	B2	1	P22 input buffer control bit	1: P22 input buffer enabled	R/W

### **(3) Low Power Consumption**

#### **Module Stop Control Register B (MSTPCRB)**

**Number of Bits: 32**

**Address: 0008 0014h**

<b>Bit</b>	<b>Symbol</b>	<b>Setting Value</b>	<b>Bit Name</b>	<b>Function</b>	<b>R/W</b>
b31	MSTPB31	0	Serial communication interface 0 module stop setting bit	0: SC0 module stop state canceled	R/W

### **(4) Serial Communications Interface 0 (SCI0)**

#### **SCI0 Serial Control Register (SCI0.SCR)**

**Number of Bits: 8**

**Address: 0008 8242h**

**(Serial communication interface mode (SMIF bit in SCI0.SCMR = 0))**

<b>Bit</b>	<b>Symbol</b>	<b>Setting Value</b>	<b>Bit Name</b>	<b>Function</b>	<b>R/W</b>
b1, b0	CKE[1:0]	10	Clock enable bits	(Clock synchronous mode) 10: External clock SCK0 functions as clock input pin.	R/W Note: 1
b2	TEIE	0	Transmit end interrupt enable bit	0: TEI0 interrupt disabled	R/W
b4	RE	0 1	Receive enable bit	0: Serial reception disabled 1: Serial reception enabled	R/W Note: 2
b5	TE	0	Transmit enable bit	0: Serial transmission disabled	R/W Note: 2
b6	RIE	0 1	Receive interrupt enable bit	0: RXI0 and ERI0 interrupts disabled 1: RXI0 and ERI0 interrupts enabled	R/W
b7	TIE	0	Transmit interrupt enable bit	0: TXI0 interrupt disabled	R/W

Notes: 1. Writing to these bits is possible only when the TE and RE bits are both cleared to 0.

2. A value of 1 may be written to either these bits only when the TE and RE bits are both cleared to 0. Also, 0 may be written to both the TE and RE bits after one of them has been set to 1.

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

**SCI0 Serial Mode Register (SCI0.SMR)**                      **Number of Bits: 8**      **Address: 0008 8240h**  
**(Serial communication interface mode (SMIF bit in SCI0.SCMR = 0))**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b1, b0	CKS[1:0]	00	Clock select bit	00: PCLK clock (n = 0) <sup>Note: 1</sup>	R/W Note: 2
b7	CM	1	Communication mode bit	1: Clock synchronous mode	R/W Note: 2

Notes: 1. For information on *n* setting values, see the Hardware Manual listed in 7, Reference Documents.  
 2. Writing to these bits is possible only when the TE and RE bits in SCI0.SCR are both cleared to 0 (serial transmission and serial reception both disabled).

**SCI0 Smart Card Mode Register (SCI0.SCMR)**                      **Number of Bits: 8**      **Address: 0008 8246h**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	SMIF	0	Smart card interface mode select bit	0: Serial communication interface mode	R/W Note: 1
b3	SDIR	0	Bit order selection bit	0: LSB-first transmission/reception	R/W Note: 1

Note: 1. Writing to this bit is possible only when the TE and RE bits in SCI0.SCR are both cleared to 0 (serial transmission and serial reception both disabled).

**SCI0 Serial Status Register (SCI0.SSR)**                      **Number of Bits: 8**      **Address: 0008 8244h**  
**(Serial communication interface mode (SMIF bit in SCI0.SCMR = 0))**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b5	ORER	— <sup>Note: 1</sup>	Overrun error flag	0: No overrun error occurred 1: An overrun error occurred	R/(W) Note: 2

Notes: 1. The sample program only reads the ORER bit. It does not write 0 to clear the flag.  
 2. Only 0 may be written to this bit to clear the flag.

**SCI0 Receive Data Register (SCI0.RDR)**                      **Number of Bits: 8**      **Address: 0008 8245h**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b7 to b0	—	—	—	Stores receive data.	R

### (5) Interrupt Control Unit (ICU)

**Interrupt Priority Register 80 (IPR80)**                      **Number of Bits: 8**      **Address: 0008 7380h**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b2 to b0	IPR[2:0]	000	SCI0 Interrupt priority level setting bits	000: Level 0 (interrupt disabled)	R/W

**Interrupt Request Enable Register 1A (IER1A)**                      **Number of Bits: 8**      **Address: 0008 721Ah**

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b7	IEN7	0	RX10 Interrupt enable bit 7	0: RX10 interrupt disabled	R/W

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

Interrupt Request Register 215 (IR215) Number of Bits: 8 Address: 0008 70D7h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	IR	0	RX10 Interrupt status flag	0: No RX10 interrupt request 1: RX10 interrupt request	R/(W) Note: 1

Note: 1. Only 0 may be written to this bit to clear the flag. Writing 1 is prohibited.

Interrupt Priority Register 01 (IPR01) Number of Bits: 8 Address: 0008 7301h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b2 to b0	IPR[2:0]	000	FIFERR Interrupt priority level setting bits	000: Level 0 (interrupt disabled)	R/W

Interrupt Priority Register 02 (IPR02) Number of Bits: 8 Address: 0008 7302h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b2 to b0	IPR[2:0]	000	FRDYI Interrupt priority level setting bits	000: Level 0 (interrupt disabled)	R/W

Interrupt Request Enable Register 02 (IER02) Number of Bits: 8 Address: 0008 7202h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b5	IEN5	0	FIFERR Interrupt enable bit 5	0: FIFERR interrupt disabled	R/W
b7	IEN7	0	FRDYI Interrupt enable bit 7	0: FRDYI interrupt disabled	R/W

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

### (6) ROM (Flash Memory for Code Storage)

#### Flash Access Status Register (FASTAT)

Number of Bits: 8

Address: 007F C410h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	DFLWPE	0	Data flash programming/erasure protection violation bit	0: No violation of programming/erasure protection set in DFLWE register occurred. 1: A violation of programming/erasure protection set in DFLWE register occurred.	R/(W) Note: 1
b1	DFLRPE	0	Data flash read protection violation bit	0: No violation of the read protection set in the DFLRE register occurred. 1: A violation of the read protection set in the DFLRE register occurred.	R/(W) Note: 1
b3	DFLAE	0	Data flash access violation interrupt bit	0: No access violation interrupt generated. 1: Access violation interrupt generated.	R/(W) Note: 1
b4	CMDLK	1	FCU command lock bit	0: FCU is not in command locked state. 1: FCU is in command locked state.	R
b7	ROMAE	0	ROM access violation bit	0: No ROM access error occurred. 1: A ROM access error occurred.	R/(W) Note: 1

Note: 1. To clear the flag, write 0 to the bit after reading it as 1.

#### Flash Access Error Interrupt Enable Register (FAEINT)

Number of Bits: 8

Address: 007F C411h

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

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

### FCU RAM Enable Register (FCURAME)

Number of Bits: 16 Address: 007F C454h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	FCRME	1	FCU RAM enable bit	0: FCU RAM access disabled 1: FCU RAM access enabled	R/W
b15 to b8	KEY[7:0]	11000100	Key code	These bits enable or disable overwriting of the FCRME bit. C4h: Writing to the FCRME bit is enabled only when C4h is written to bits KEY7 to KEY0 using word access.	R/(W) Note: 1

Note: 1. Write data is not retained.

### Flash Status Register 0 (FSTATR0)

Number of Bits: 8 Address: 007F FFB0h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b4	PRGERR	—	Programming error bit	0: Normal end of programming 1: Error occurred during programming	R
b5	ERSERR	—	Erase error bit	0: Normal end of erasing 1: Error occurred during erasing	R
b6	ILGLERR	—	Illegal command error bit	0: No illegal command or illegal ROM/data flash access detected by FCU 1: Illegal command or illegal ROM/data flash access detected by FCU	R
b7	FRDY	—	Flash ready bit	0: Programming/erase, programming/erase suspension, lock bit read 2 command, or data flash blank check processing in progress 1: Above processing not being executed	R

### Flash Status Register 1 (FSTATR1)

Number of Bits: 8 Address: 007F FFB1h

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

## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

### Flash Protection Register (FPROTR)

Number of Bits: 16 Address: 007F FFB4h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	FPROTCN	1	Lock bit protection cancel bit	1: Protection with a lock bit disabled	R/W
b15 to b8	FPKEY[7:0]	01010101	Key code	These bits enable or disable overwriting of the FPROTCN bit. 55h: Writing to the FPROTCN bit is enabled only when the value of the FENTYRY register is other than 0000h and 55h is written to bits FPKEY7 to FPKEY0 using word access.	R/(W) Note: 1

Note: 1. Write data is not retained.

### Flash Reset Register (FRESETR)

Number of Bits: 16 Address: 007F FFB6h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	FRESET	0 1	Flash reset bit	0: FCU is not reset. 1: FCU is reset.	R/W
b15 to b8	FRKEY[7:0]	11001100	Key code	These bits enable or disable overwriting of the FRESET bit. CCh: Writing to the FRESET bit is enabled only when CCh is written to bits FRKEY7 to FRKEY0 using word access.	R/(W) Note: 1

Note: 1. Write data is not retained.



## RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Slave)

### Flash P/E Mode Entry Register (FENTRYR)

Number of Bits: 16 Address: 007F FFB2h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b0	FENTRY0	0	ROM P/E mode entry bit 0	0: 1 MB of ROM (read address range: FFF0 0000h to FFFF FFFFh, programming/erase address range: 00F0 0000h to 00FF FFFFh) set to ROM read mode	R/W
		1		1: 1 MB of ROM (read address range: FFF0 0000h to FFFF FFFFh, programming/erase address range: 00F0 0000h to 00FF FFFFh) set to ROM P/E mode	
b1	FENTRY1	0	ROM P/E mode entry bit 1	0: 1 MB of ROM (read address range: FFE0 0000h to FFEF FFFFh, programming/erase address range: 00E0 0000h to 00EF FFFFh) set to ROM read mode	R/W
		1		1: 1 MB of ROM (read address range: FFE0 0000h to FFEF FFFFh, programming/erase address range: 00E0 0000h to 00EF FFFFh) set to ROM P/E mode	
b7	FENTRYD	0	Data flash P/E mode entry bit	Sets the data flash to read mode.	R/W
b15 to b8	FEKEY[7:0]	10101010	Key code	These bits enable or disable overwriting of the FENTRY0, FENTRY1, and FENTRYD bits. AAh: Writing to the FENTRY0, FENTRY1, and FENTRYD bits is enabled only when AAh is written to bits FEKEY7 to FEKEY0 using word access.	R/(W) Note: 1

Note: 1. Write data is not retained.

### Peripheral Clock Notification Register (PCKAR)

Number of Bits: 16 Address: 007F FFE8h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b7 to b0	PCKA[7:0]	00110010	Peripheral clock notification bits	0x32: PCLK frequency = 50 MHz	R/W

### Flash Write Erase Protection Register (FWEPROR)

Number of Bits: 8 Address: 0008 C289h

Bit	Symbol	Setting Value	Bit Name	Function	R/W
b1, b0	FLWE[1:0]	01	Flash write/erase bits	01: Write/erase enabled	R/W
		10		10: Write/erase disabled	

### 5.9 Function Specifications

The specifications of the slave device functions are as follows.

**(1) PowerON\_Reset\_PC Function**

**(a) Functional overview**

The PowerON\_Reset\_PC function initializes the stack pointer (a #pragma entry declaration causes the compiler automatically to generate ISP/USP initialization code at the start of the PowerON\_Reset\_PC function), sets INTB (set\_intb function: embedded function), initializes FPSW (set\_fpsw function: embedded function), initializes the RAM area section (\_INITSCT function: standard library function), calls the HardwareSetup function, initializes PSW (set\_psw function: embedded function), and sets user mode as the processor mode. Then it calls the main function.

**(b) Arguments**

None

**(c) Return values**

None

**(d) Flowchart**

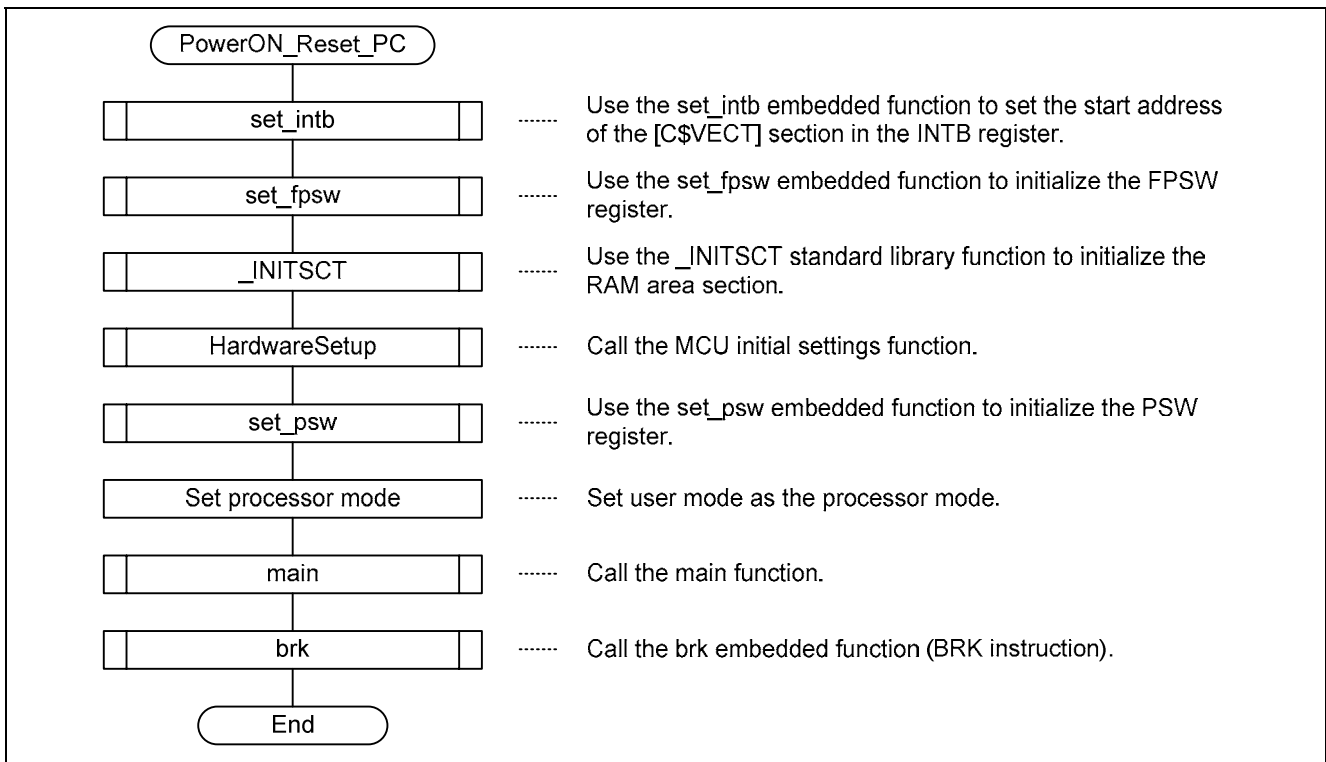


Figure 13 Flowchart (PowerON\_Reset\_PC) (Slave)

(2) **HardwareSetup Function**

(a) **Functional overview**

The HardwareSetup function makes initial settings to the MCU. It makes initial clock settings (system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK)), initial output settings for the I/O ports (P83, P84, P33, and P34) connected to LED0 to LED3, the initial output setting for the Busy port (P80), and initial settings to SCIO.

(b) **Arguments**

None

(c) **Return values**

None

(d) **Flowchart**

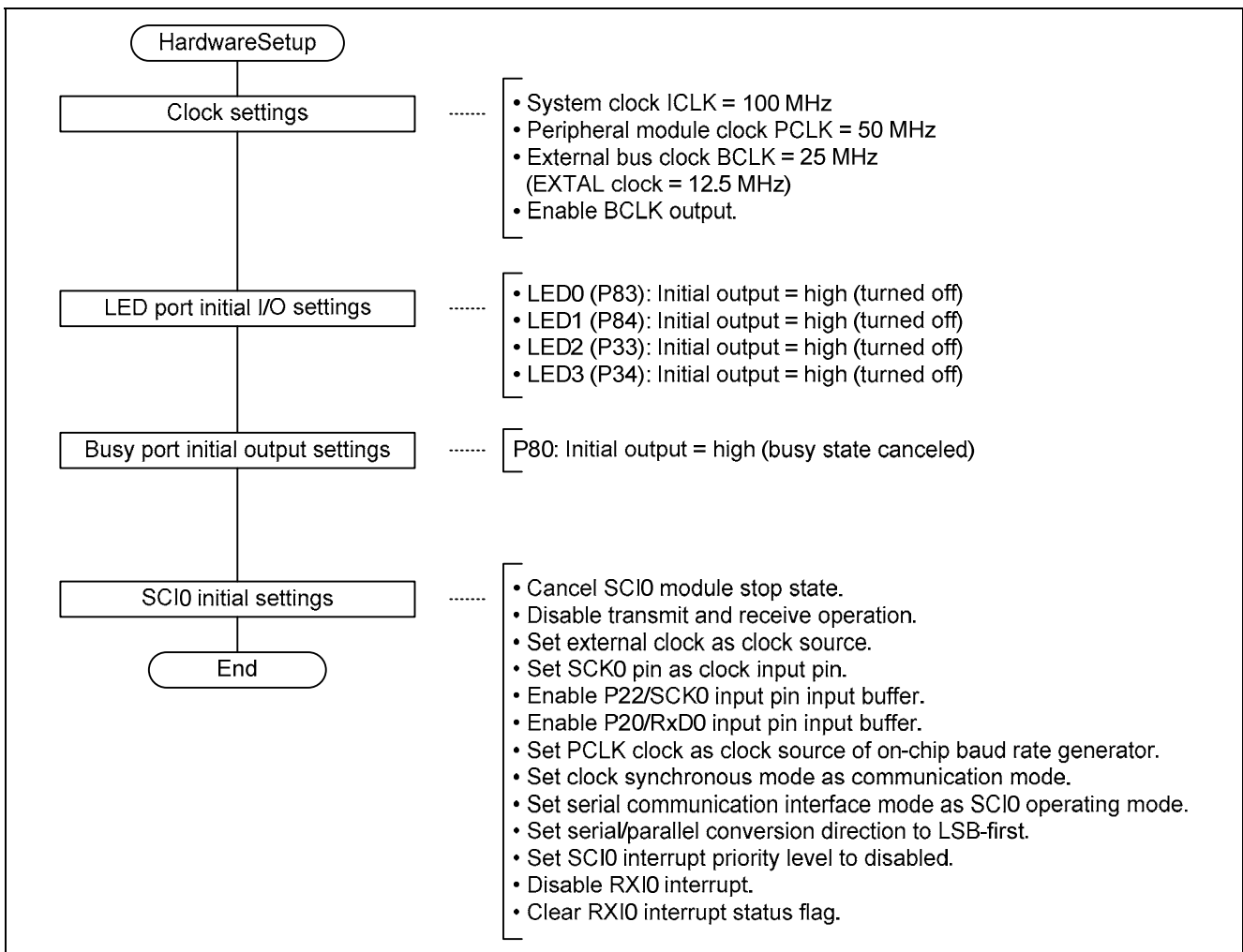


Figure 14 Flowchart (HardwareSetup) (Slave)

**(3) main Function****(a) Functional overview**

The main function controls reception of one byte of data from the master, copies the user MAT programming/erase 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 values**

None

(d) Flowchart

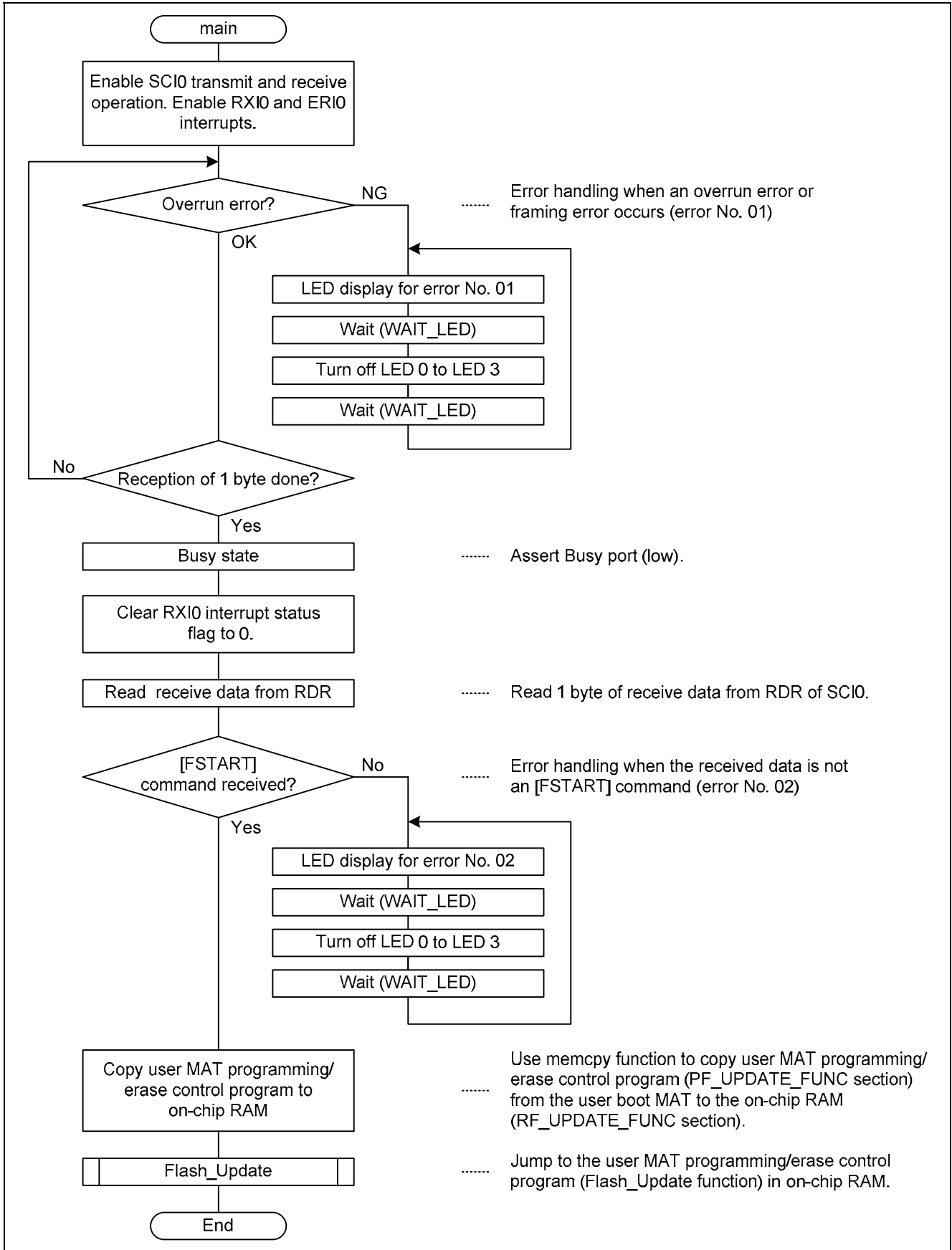


Figure 15 Flowchart (main) (Slave)

**(4) Flash\_Update Function****(a) Functional overview**

The Flash\_Update function controls reception of communication commands by clock synchronous serial communication with the master, controls reception of the erasure block number, controls reception of the write size data, controls reception of write data, controls the Busy port for serial communication, controls programming/erasing of the user MAT, calls the Indicate\_Ending\_LED function at normal end of user MAT programming/erasing, and calls the Indicate\_Error\_LED function at error end of user MAT programming/erasing.

**(b) Arguments**

None

**(c) Return values**

None

(d) Flowchart

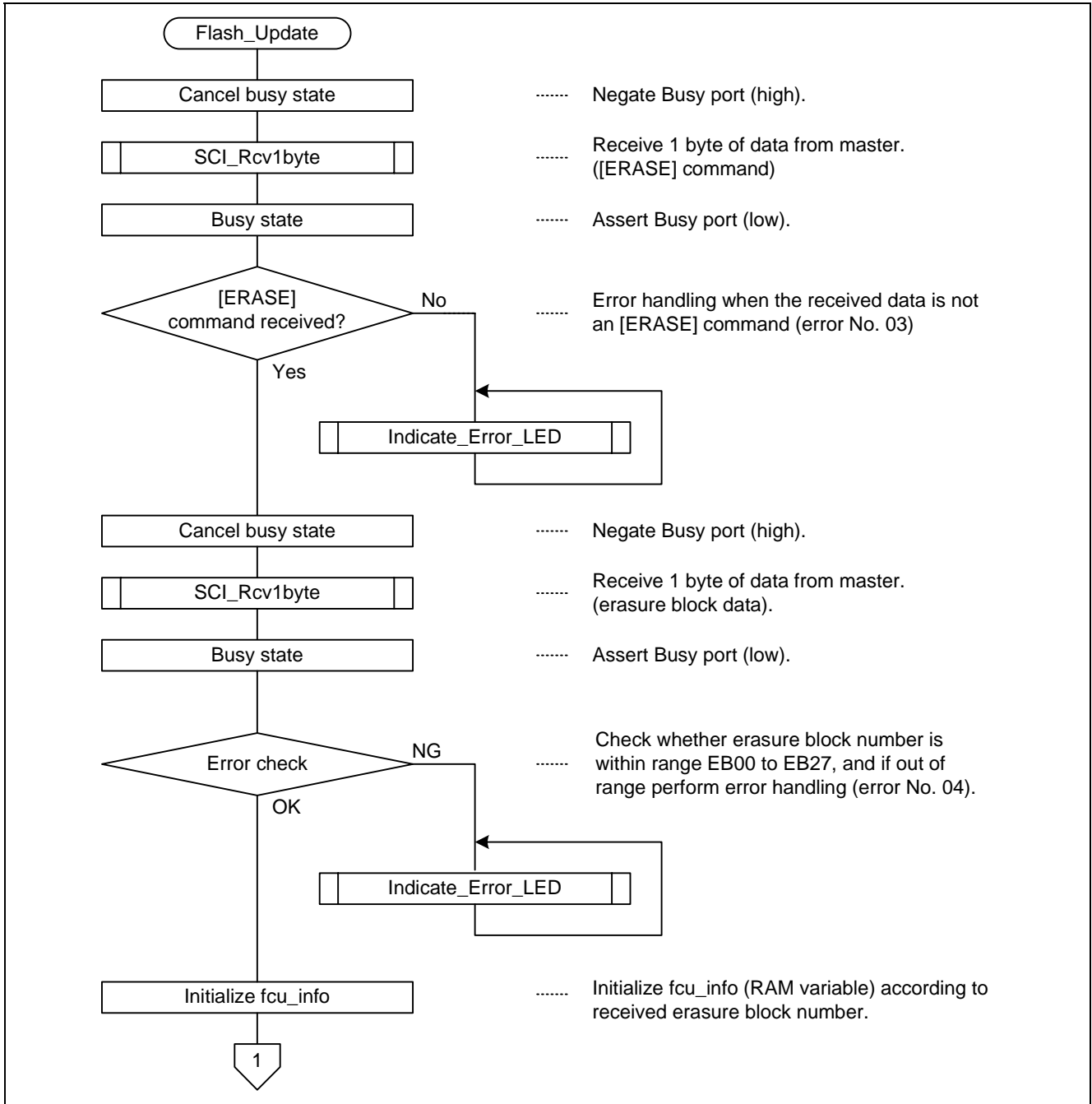


Figure 16 Flowchart (Flash\_Update) (1) (Slave)

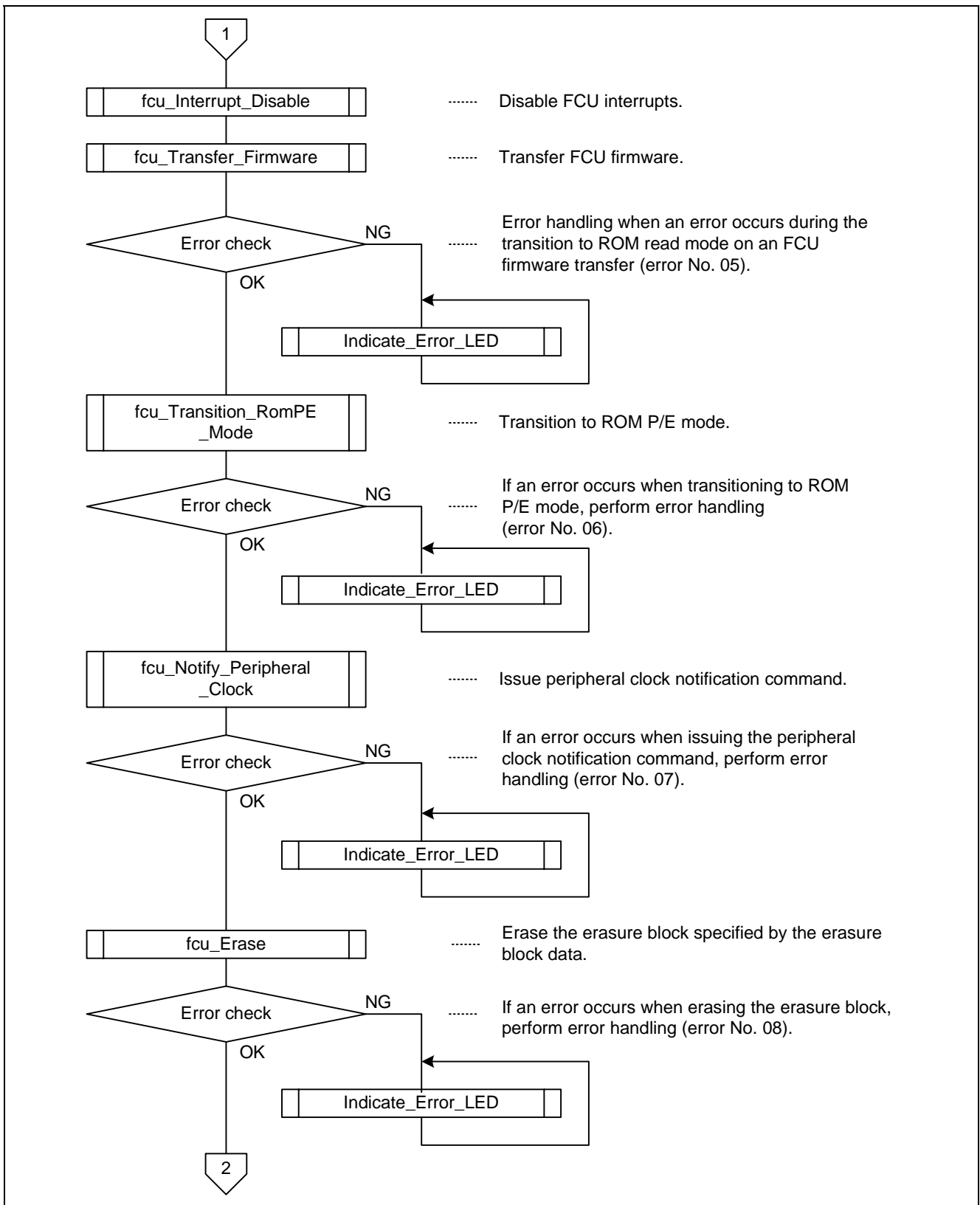


Figure 17 Flowchart (Flash\_Update) (2) (Slave)



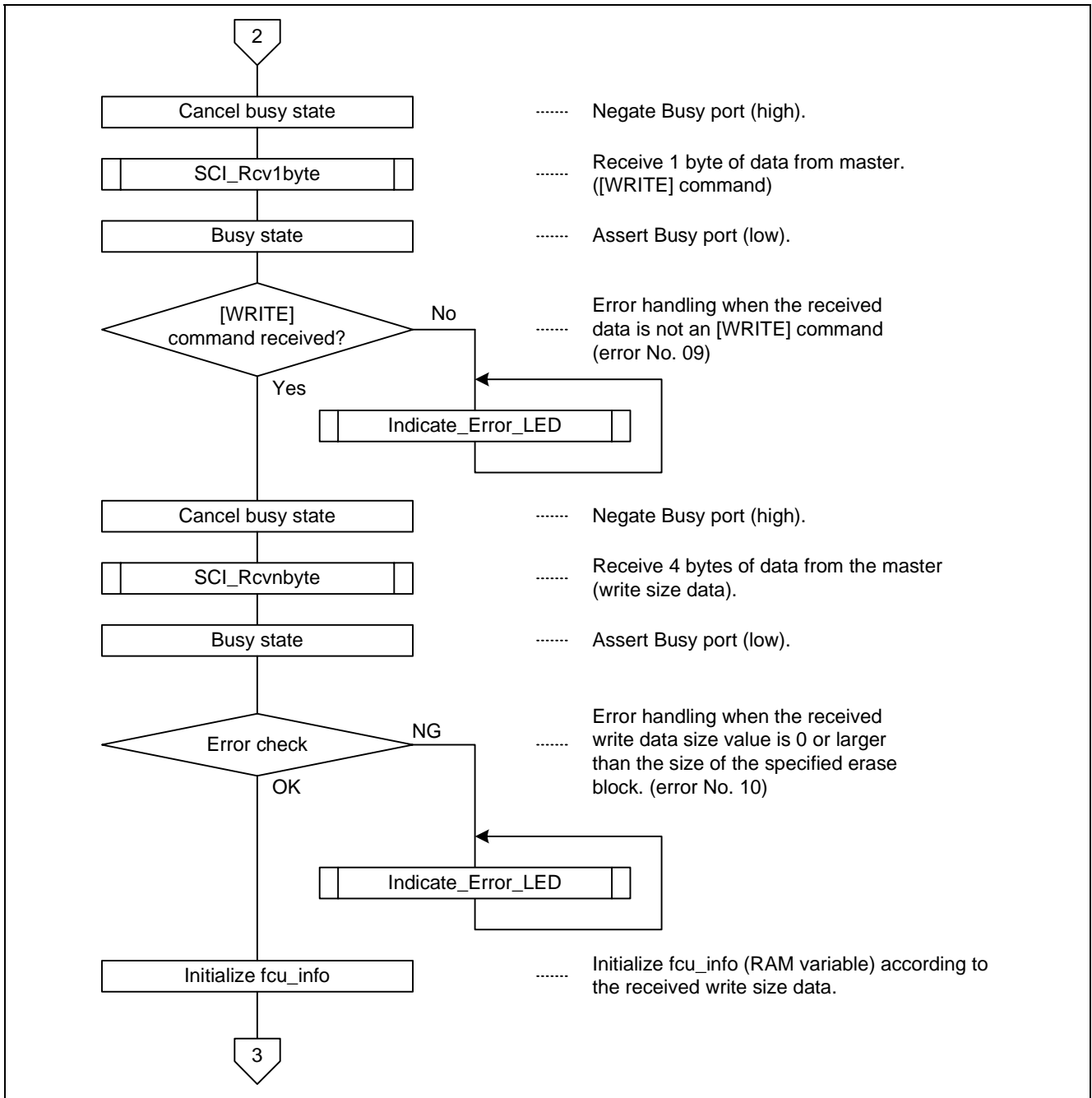


Figure 18 Flowchart (Flash\_Update) (3) (Slave)

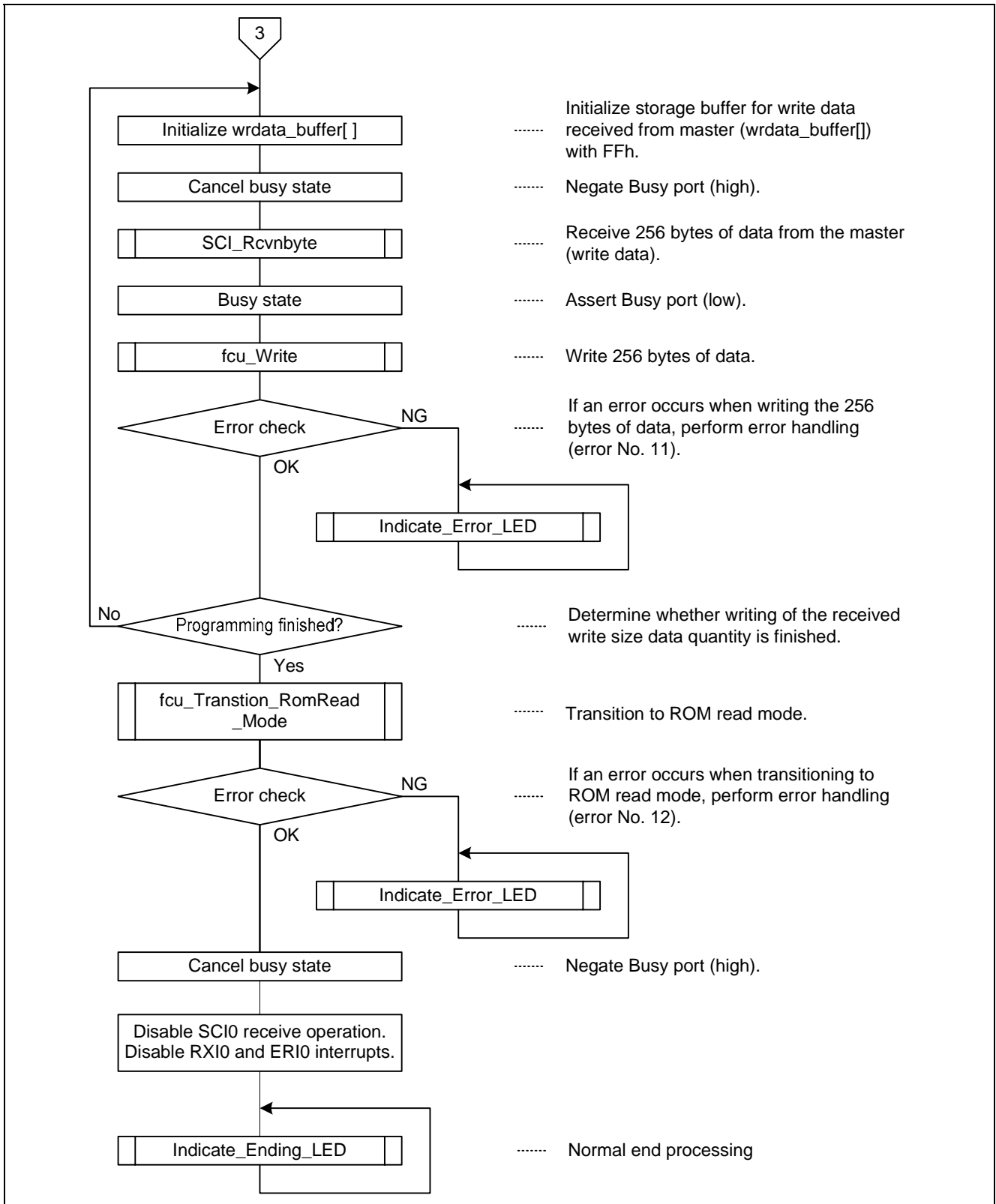


Figure 19 Flowchart (Flash\_Update) (4) (Slave)

(5) **fcu\_Interrupt\_Disable Function**

(a) **Functional overview**

The fcu\_Interrupt\_Disable function disables FCU interrupts (FRDYI interrupt, data flash programming/erasure protection violation interrupt, data flash read protection violation interrupt, access violation interrupt, FCU command lock interrupt, ROM access violation interrupt, and FIFERR interrupt) before programming/erasing of the user MAT takes place.

(b) **Arguments**

None

(c) **Return values**

None

(d) **Flowchart**

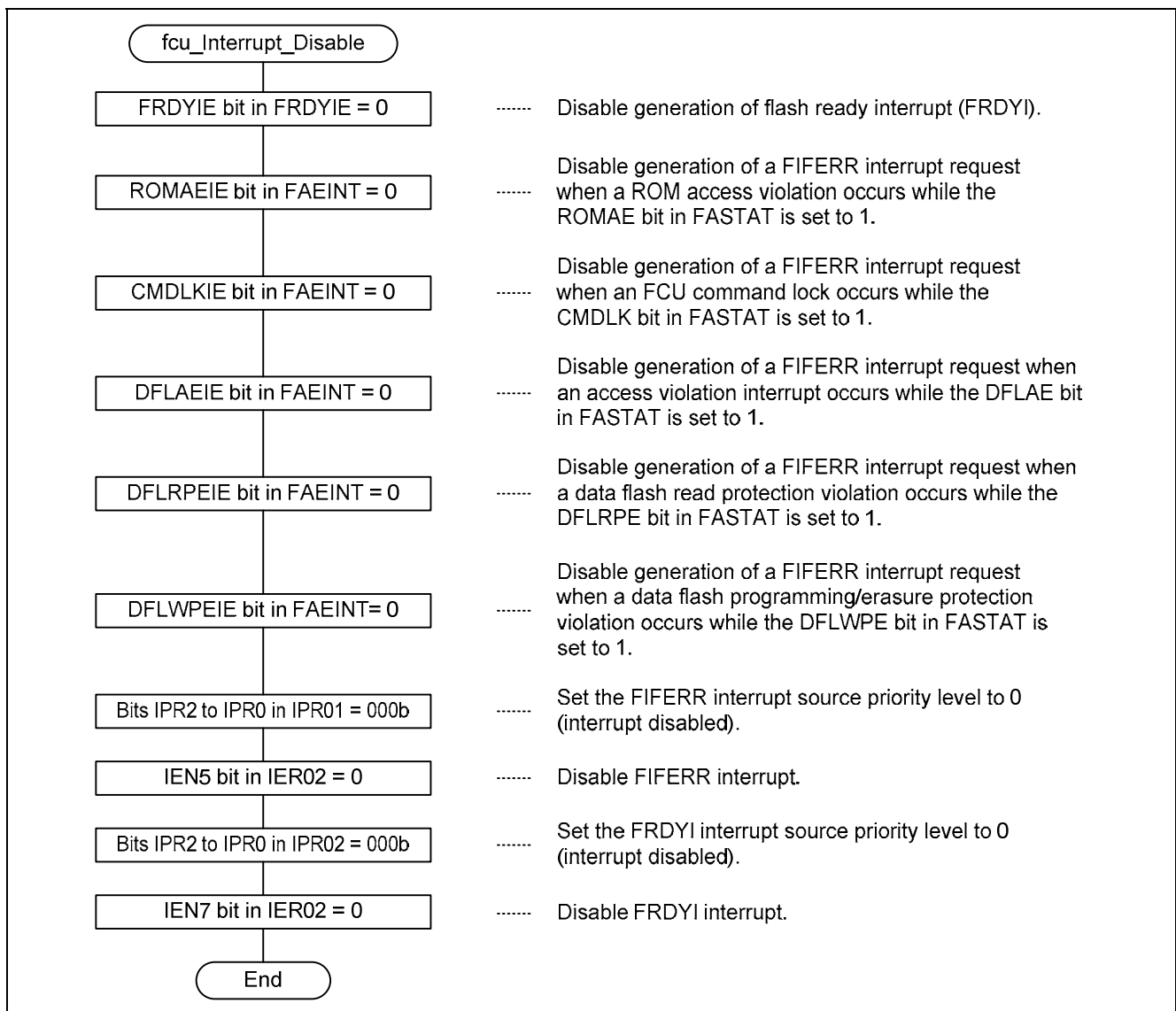


Figure 20 Flowchart (fcu\_Interrupt\_Disable) (Slave)

(6) **fcu\_Reset Function**

(a) **Functional overview**

The fcu\_Reset function uses the FRESET bit in FRESETR to initialize the FCU.

(b) **Arguments**

None

(c) **Return values**

None

(d) **Flowchart**

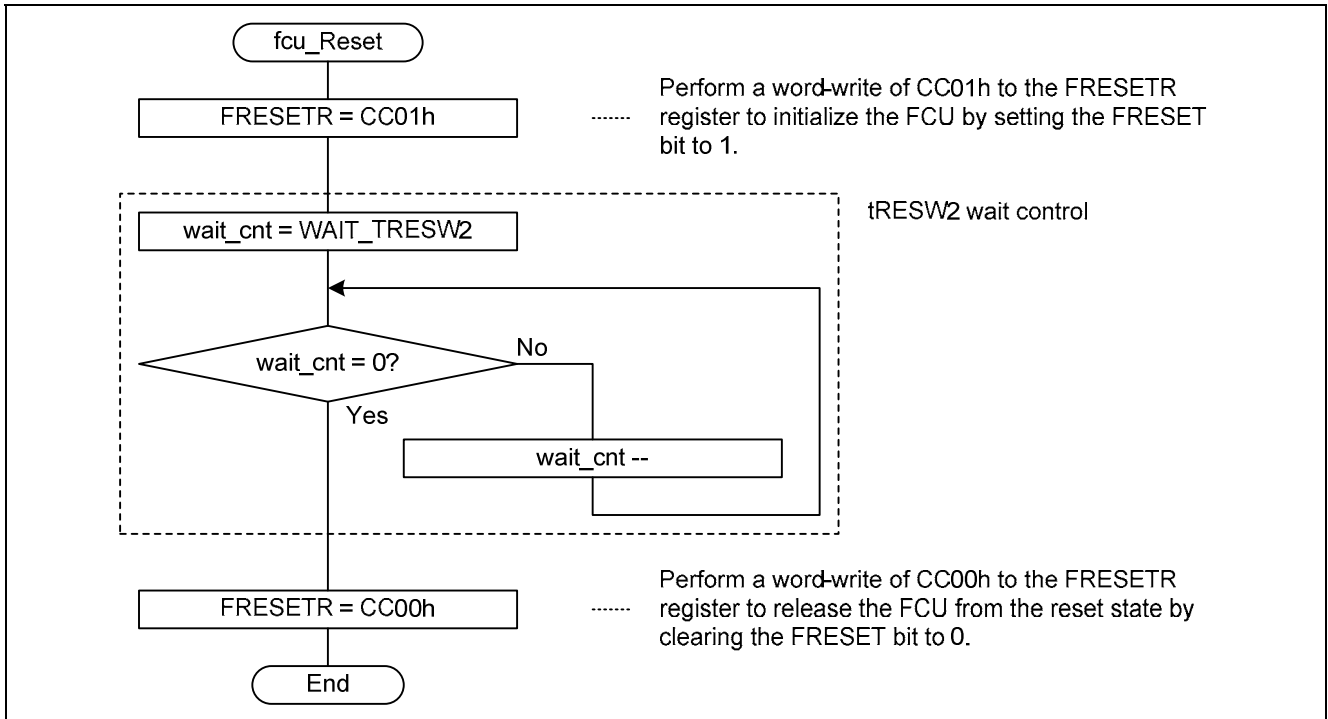


Figure 21 Flowchart (fcu\_Reset) (Slave)

### (7) fcu\_Transfer\_Firmware Function

#### (a) Functional overview

The fcu\_Transfer\_Firmware function copies the FCU firmware stored in 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 arguments used by this function.

**Table 22 Arguments of fcu\_Transfer\_Firmware Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * Note: 1	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

#### (c) Return values

Table 23 lists the return values used by this function.

**Table 23 Return Values of fcu\_Transfer\_Firmware Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

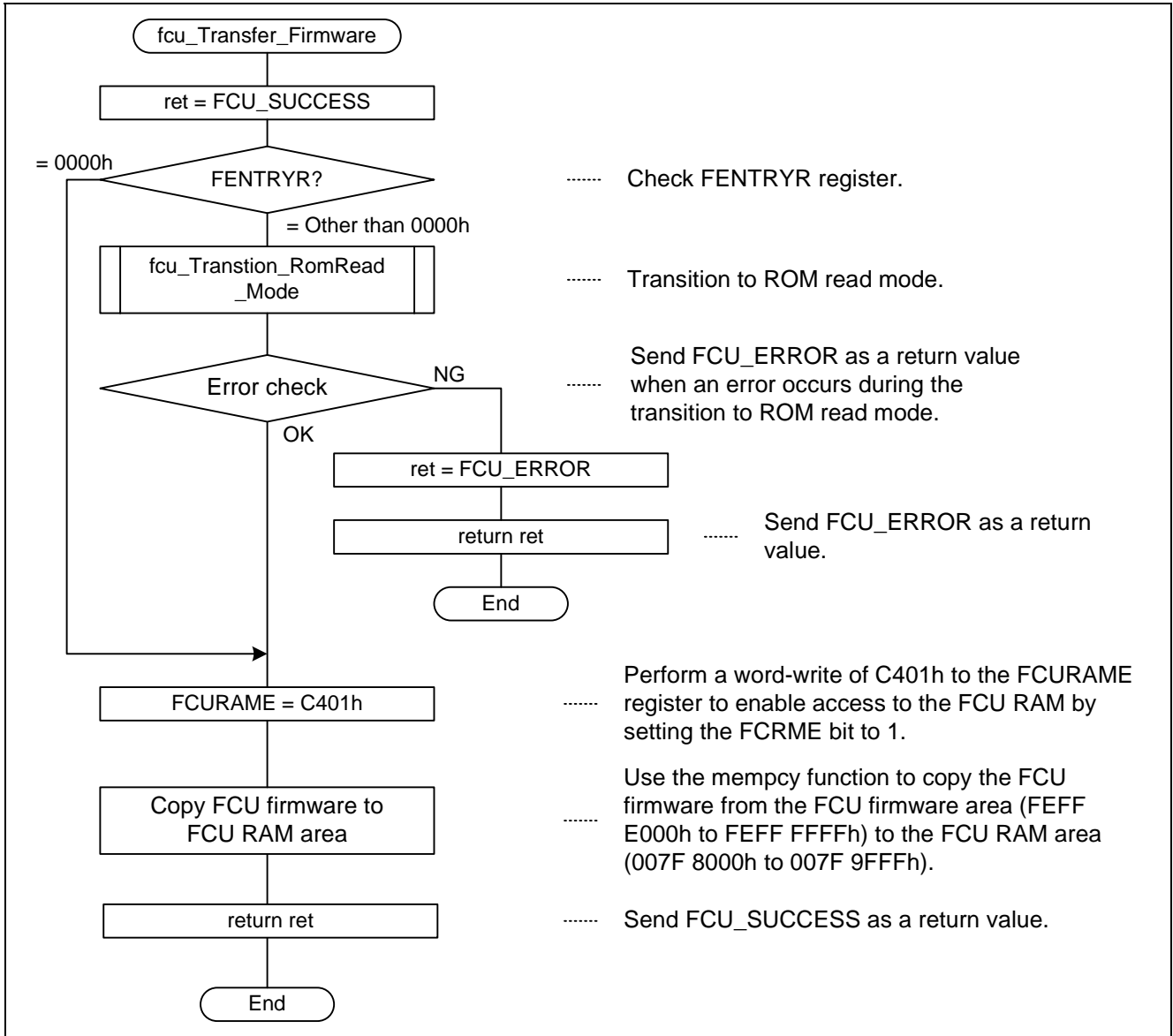


Figure 22 Flowchart (fcu\_Transfer\_Firmware) (Slave)

**(8) fcu\_Transition\_RomRead\_Mode Function**

**(a) Functional overview**

The fcu\_Transition\_RomRead\_Mode function causes the FCU to transition to ROM read mode.

**(b) Arguments**

Table 24 lists the arguments used by this function.

**Table 24 Arguments of fcu\_Transition\_RomRead\_Mode Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * <small>Note: 1</small>	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

**(c) Return values**

Table 25 lists the return values used by this function.

**Table 25 Return Values of fcu\_Transition\_RomRead\_Mode Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

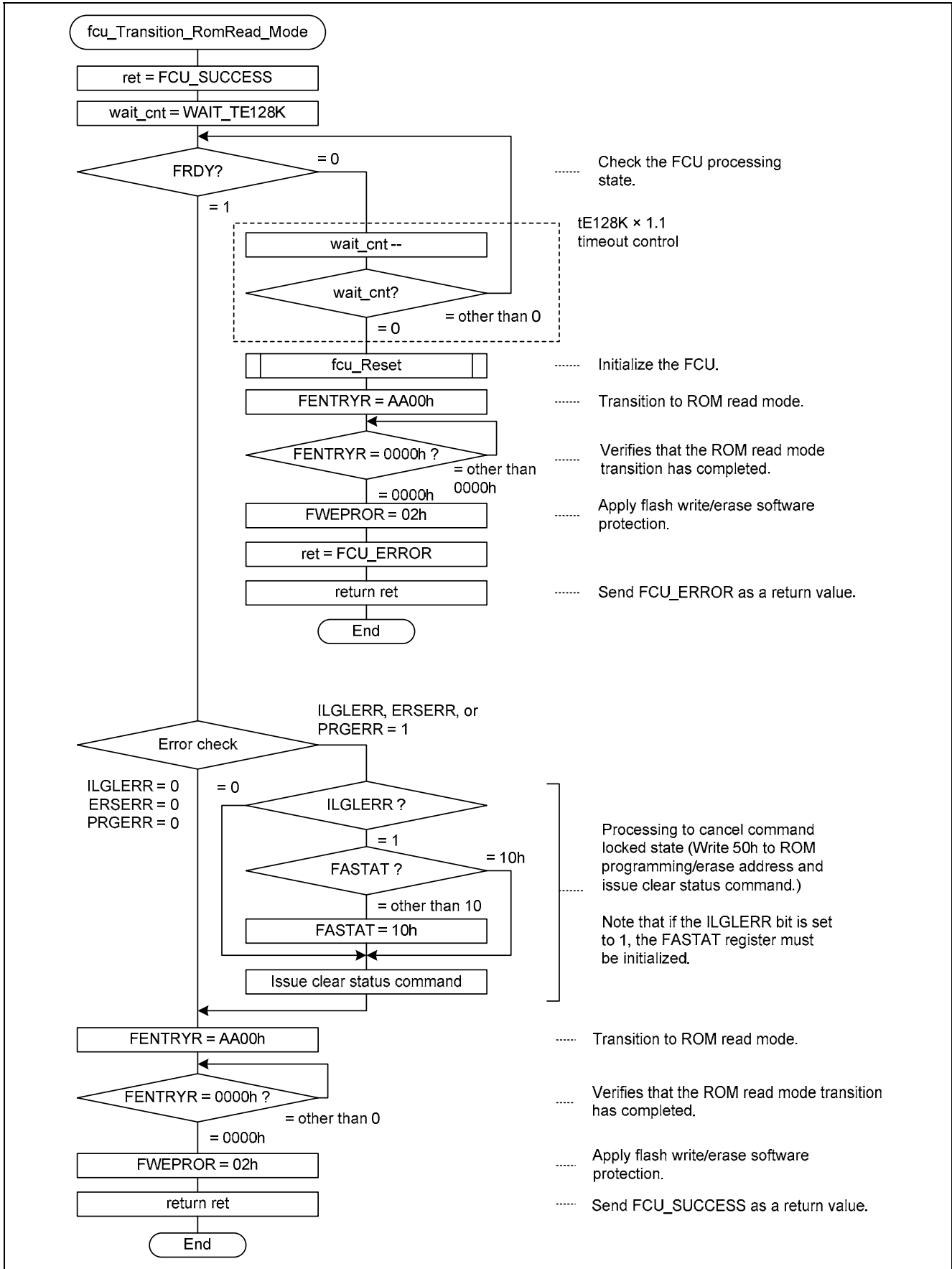


Figure 23 Flowchart (fcu\_Transition\_RomRead\_Mode) (Slave)



**(9) fcu\_Transition\_RomPE\_Mode Function**

**(a) Functional overview**

The fcu\_Transition\_RomPE\_Mode function causes the FCU to transition to ROM P/E mode.

**(b) Arguments**

Table 26 lists the arguments used by this function.

**Table 26 Arguments of fcu\_Transition\_RomPE\_Mode Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * <small>Note: 1</small>	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

**(c) Return values**

Table 27 lists the return values used by this function.

**Table 27 Return Values of fcu\_Transition\_RomPE\_Mode Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

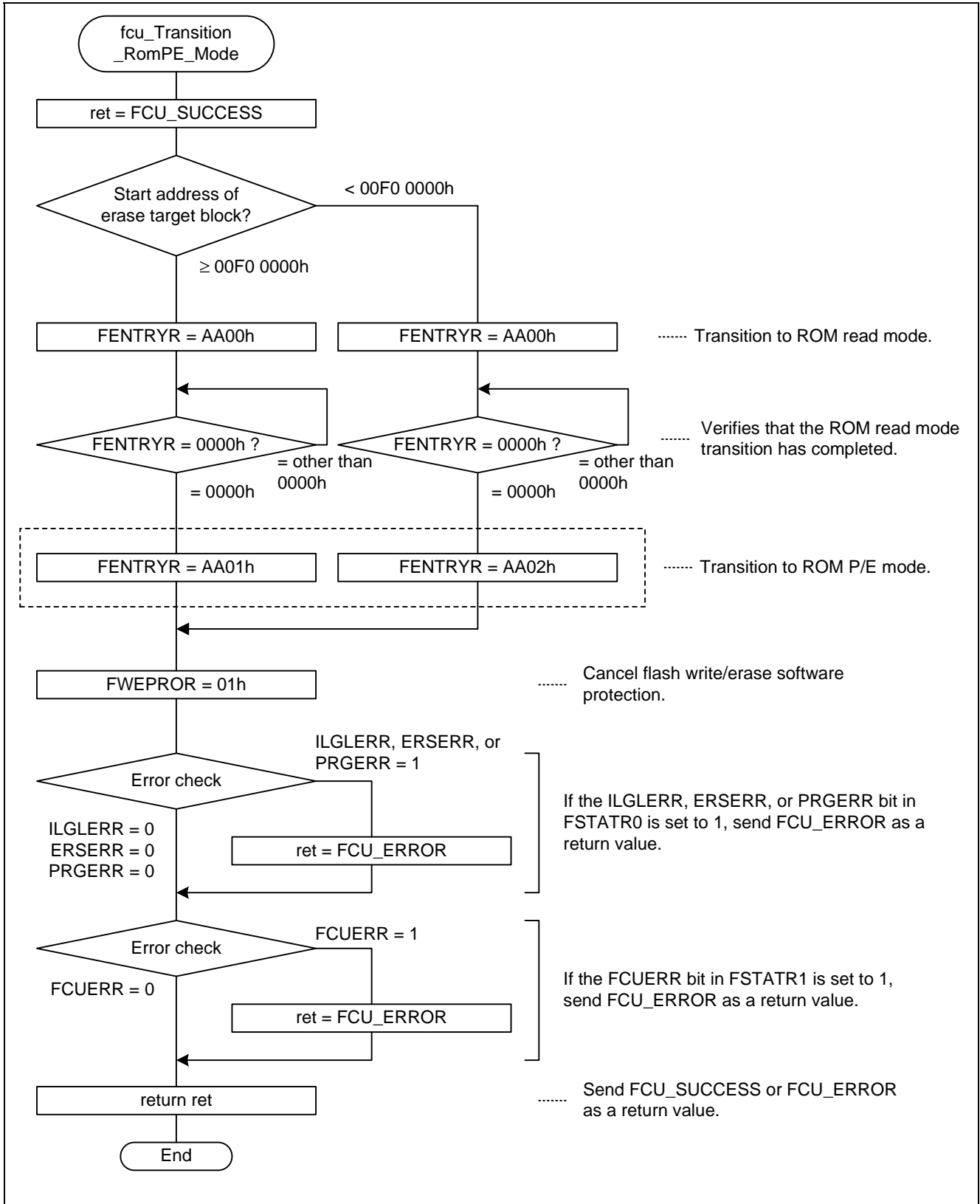


Figure 24 Flowchart (fcu\_Transition\_RomPE\_Mode) (Slave)

**(10) fcu\_Notify\_Peripheral\_Clock Function**

**(a) Functional overview**

The fcu\_Notify\_Peripheral\_Clock function sets the peripheral module clock (PCLK) frequency in the PCKAR register and issues the peripheral clock notification command.

**(b) Arguments**

Table 28 lists the arguments used by this function.

**Table 28 Arguments of fcu\_Notify\_Peripheral\_Clock Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * <small>Note: 1</small>	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

**(c) Return values**

Table 29 lists the return values used by this function.

**Table 29 Return Values of fcu\_Notify\_Peripheral\_Clock Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

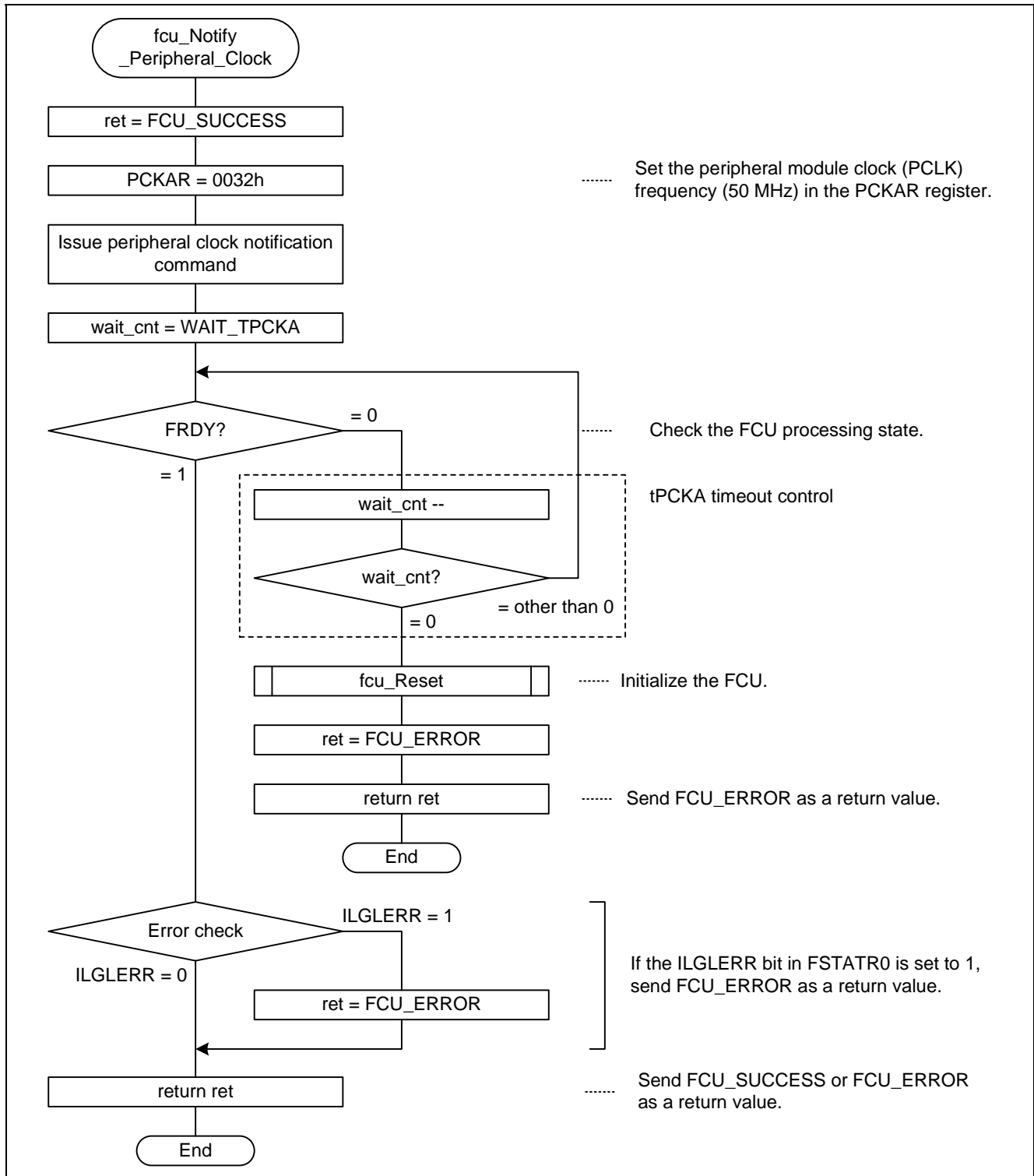


Figure 25 Flowchart (fcu\_Notify\_Peripheral\_Clock) (Slave)

**(11) fcu\_Erase Function**

**(a) Functional overview**

The fcu\_Erase function uses the block erase command to erase the user MAT (in erasure block units).

**(b) Arguments**

Table 30 lists the arguments used by this function.

**Table 30 Arguments of fcu\_Erase Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * <small>Note: 1</small>	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

**(c) Return values**

Table 31 lists the return values used by this function.

**Table 31 Return Values of fcu\_Erase Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

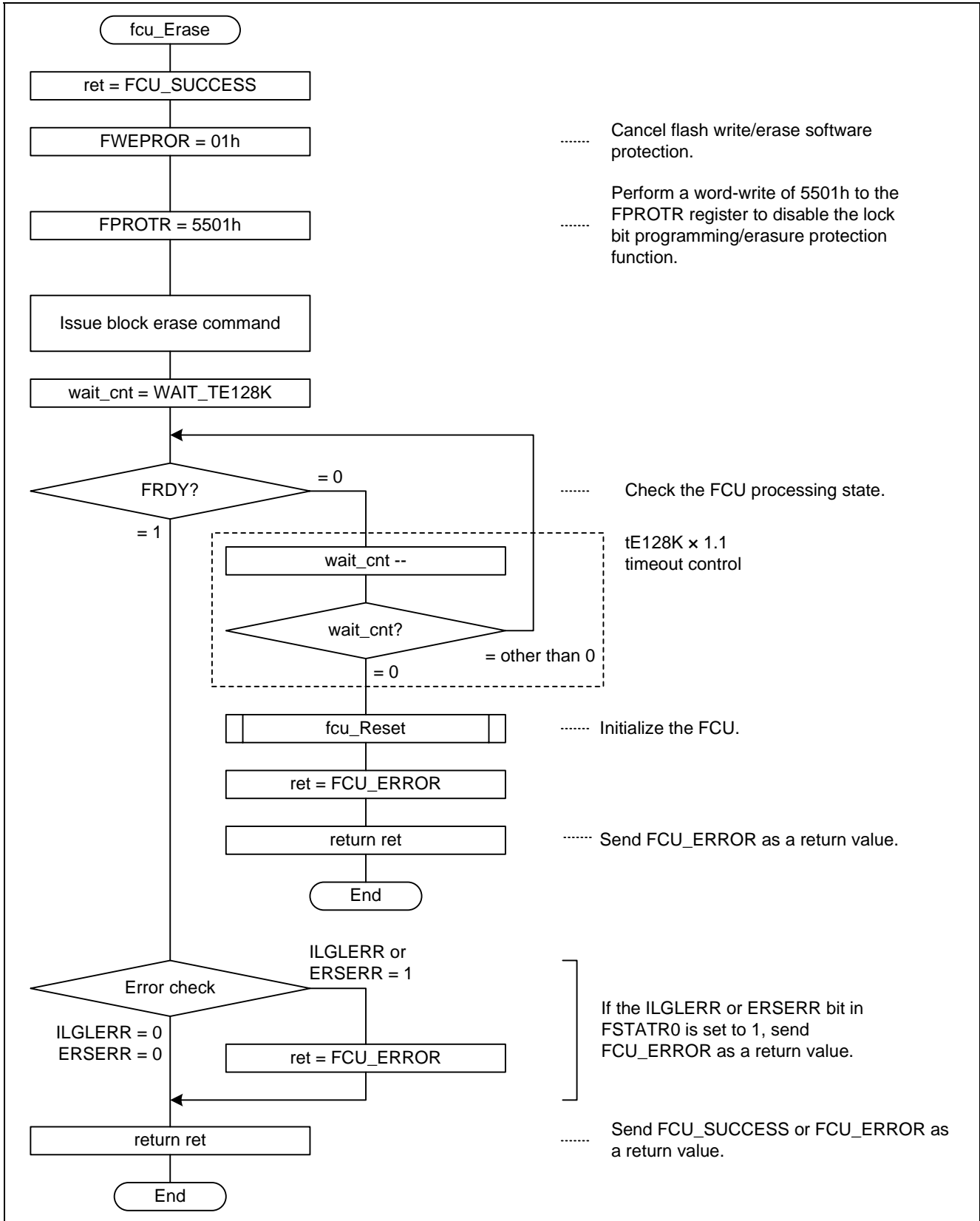


Figure 26 Flowchart (fcu\_Erase) (Slave)

**(12) fcu\_Write Function**

**(a) Functional overview**

The fcu\_Write function uses the program command to program the user MAT (in 256-byte units).

**(b) Arguments**

Table 32 lists the arguments used by this function.

**Table 32 Arguments of fcu\_Write Function**

Arguments	Type	Description
1st argument	ST_FCU_INFO * <small>Note: 1</small>	Address of structure for storing the FCU function used to program/erase the user MAT.

Note: 1. For details on the ST\_FCU\_INFO type, see 5.6.1, ST\_FCU\_INFO Structure.

**(c) Return values**

Table 33 lists the return values used by this function.

**Table 33 Return Values of fcu\_Write Function**

Type	Description
FCU_STATUS <sup>Note: 1</sup>	Status when function is executed

Note: 1. For details on the FCU\_STATUS type, see 5.7, Enumerated Types.

(d) Flowchart

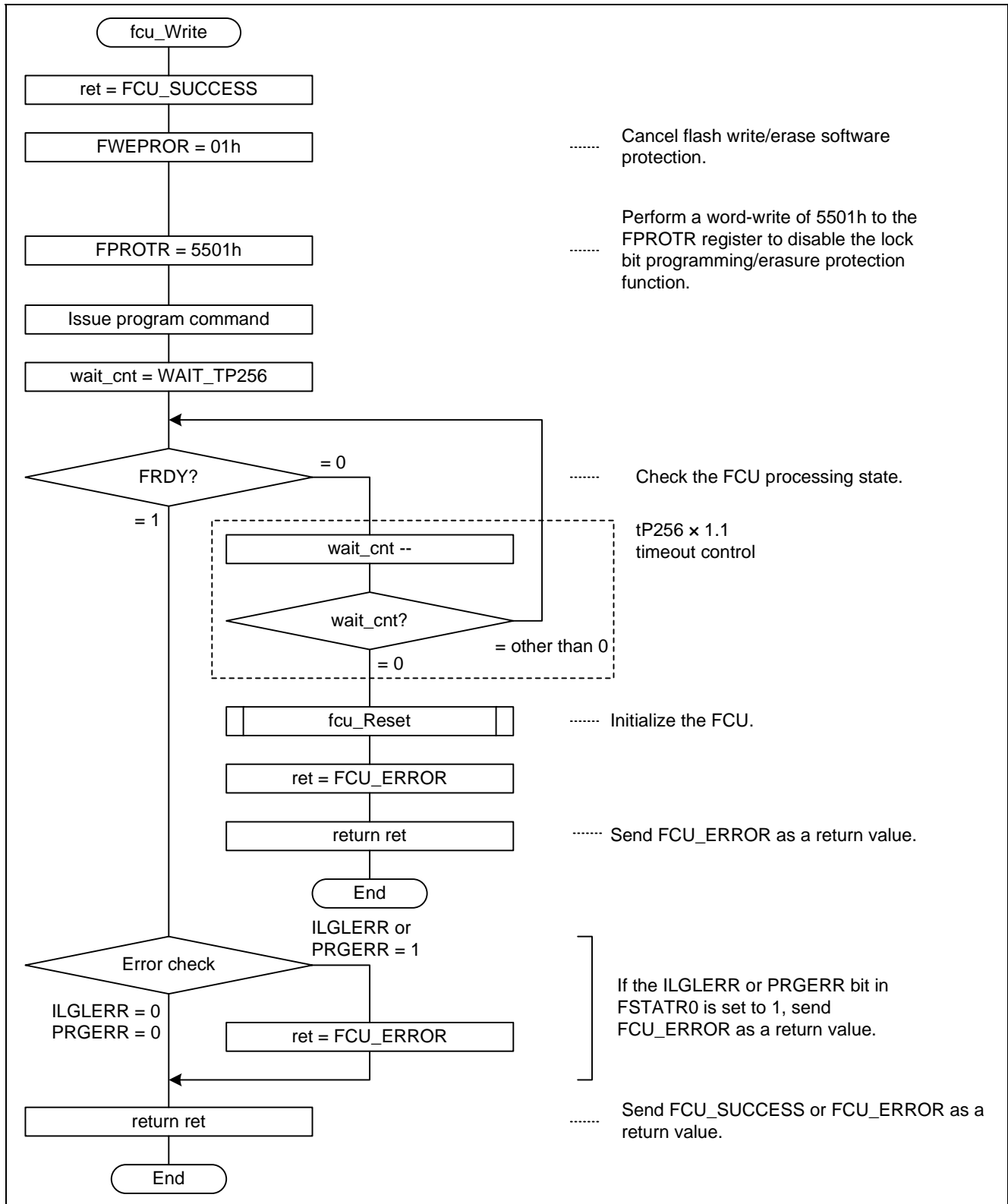


Figure 27 Flowchart (fcu\_Write) (Slave)



(13) Indicate\_Ending\_LED Function

(a) Functional overview

When programing/erasing completes successfully, the Indicate\_Ending\_LED function indicates a normal end using LED0 to LED3. The function illuminates LED0 to LED3 one at a time in sequence.

(b) Arguments

None

(c) Return values

None

(d) Flowchart

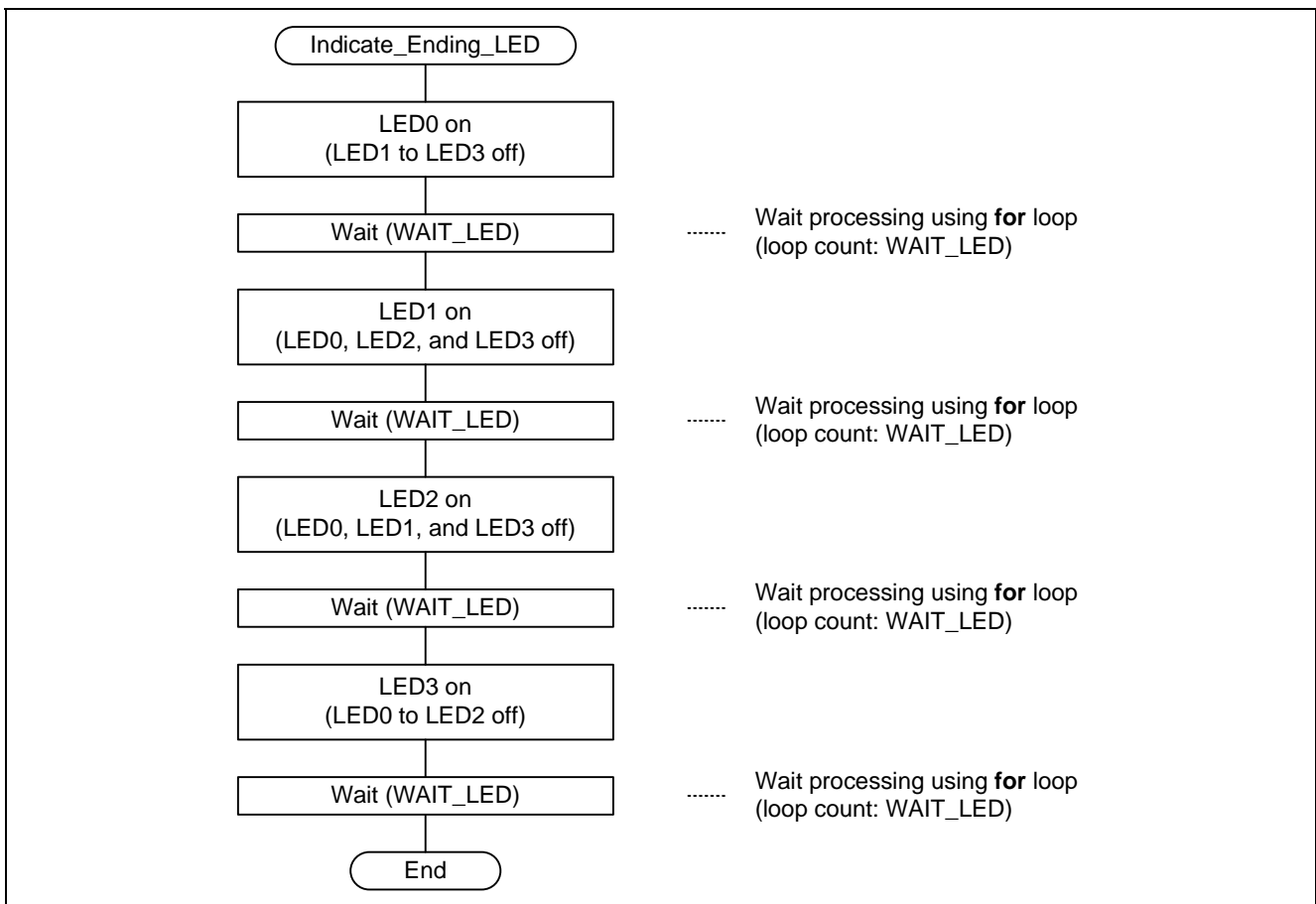


Figure 28 Flowchart (Indicate\_Ending\_LED) (Slave)

(14) Indicate\_Error\_LED Function

(a) Functional overview

When an error occurs during programing/erasing of the user MAT, the Indicate\_Error\_LED function indicates the error number using LED0 to LED3. The display alternates repeatedly between the error number indication and all LEDs off.

(b) Arguments

Table 34 lists the arguments used by this function.

Table 34 Arguments of Indicate\_Error\_LED Function

Arguments	Type	Description
1st argument	unsigned char	Error number <sup>Note: 1</sup> that occurred during programing/erasing of the user MAT

Note: 1. For information on error numbers, see 4.6, Error Handling.

(c) Return values

None

(d) Flowchart

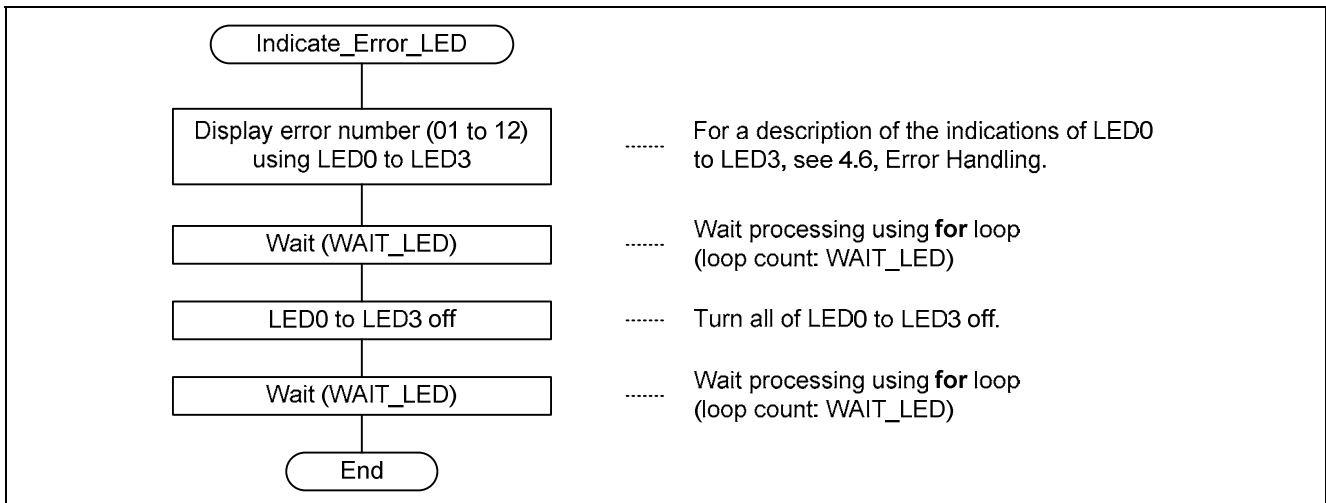


Figure 29 Flowchart (Indicate\_Error\_LED) (Slave)

**(15) SCI\_Rcv1byte Function**

**(a) Functional overview**

The SCI\_Rcv1byte function performs the reception control for receiving 1 byte of data over SCI0 clock synchronous serial communication.

**(b) Arguments**

Table 35 lists the return values used by this function.

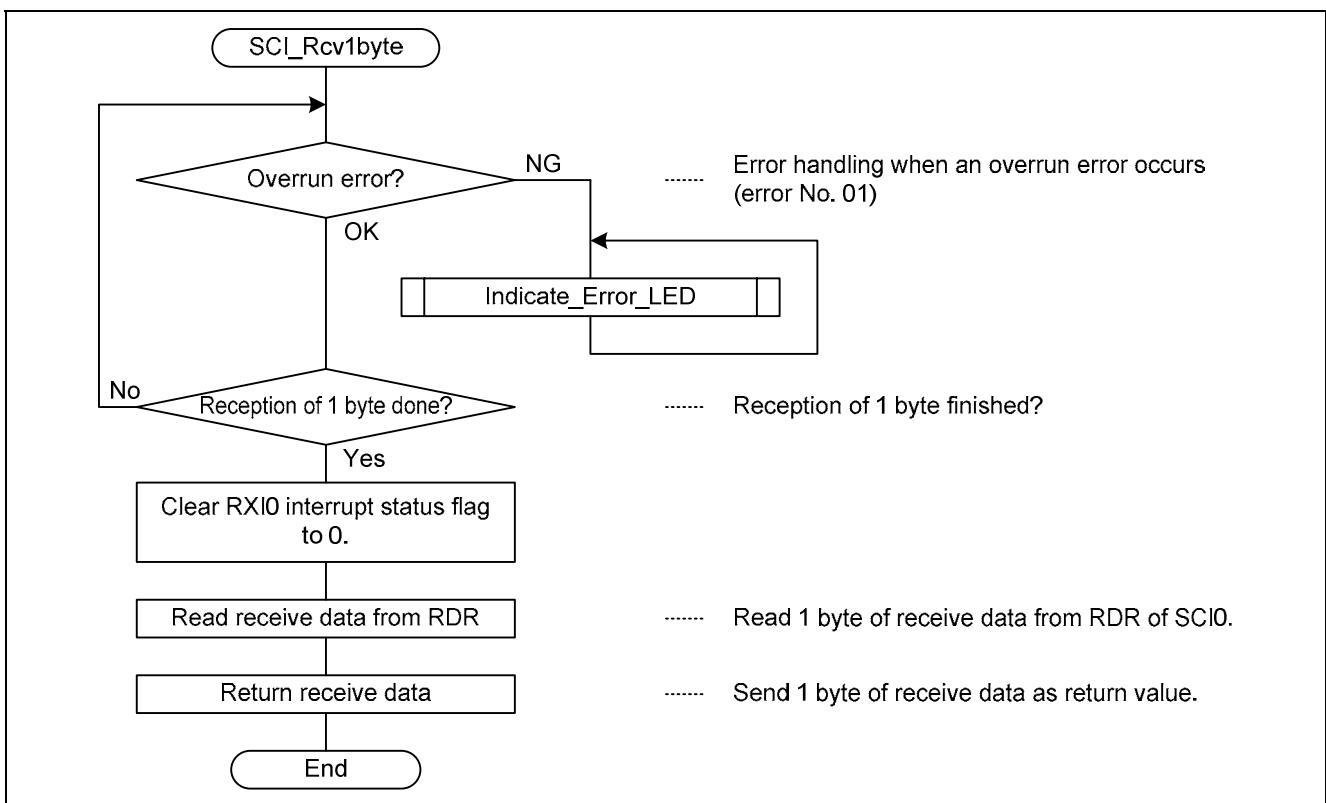
**Table 35 Return Values SCI\_Rcv1byte Function**

Type	Description
unsigned char	The one byte of receive data from the SCI0 clock synchronous serial communication.

**(c) Return values**

None

**(d) Flowchart**



**Figure 30 Flowchart (SCI\_Rcv1byte) (Slave)**

(16) SCI\_Rcvnbyte Function

(a) Functional overview

The SCI\_Rcvnbyte function controls reception of n bytes of data (n is the first argument and unsigned short type) using clock synchronous serial communication by SCI0.

(b) Arguments

Table 36 lists the arguments used by this function.

Table 36 Arguments of SCI\_Rcvnbyte Function

Arguments	Type	Description
1st argument	unsigned short	Receive data byte count obtained using clock synchronous serial communication by SCI0
2nd argument	unsigned char *	Start address of receive data storage location

(c) Return values

None

(d) Flowchart

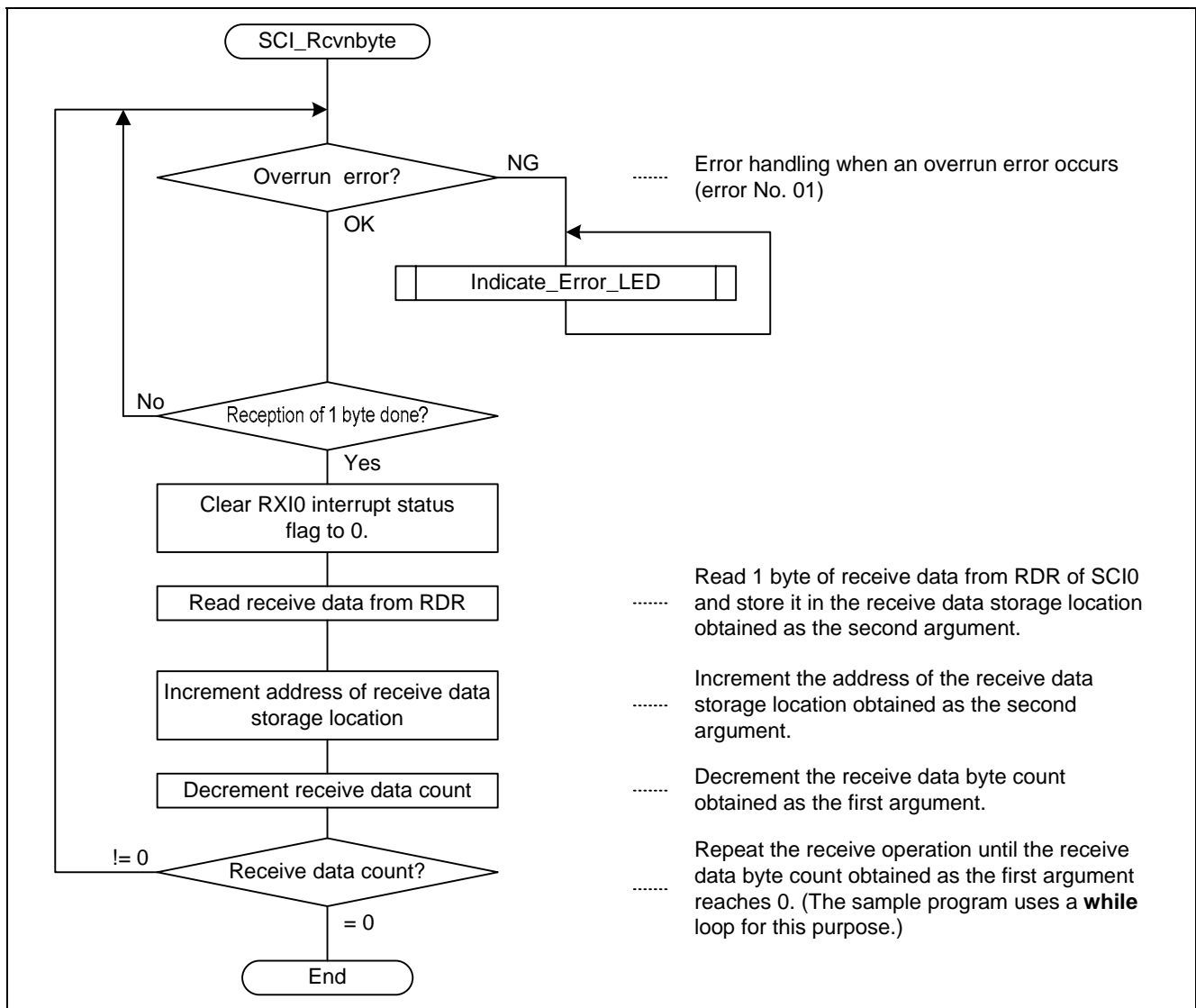


Figure 31 Flowchart (SCI\_Rcvnbyte) (Slave)

## 6. Usage Notes

### 6.1 Timeouts

The sample program performs timeout control during programming/erasing of the user MAT. A software timer is used to measure the elapsed time.

The timeout control used by the sample program is described below.

#### 6.1.1 tPCKA Timeout Control

When the FCU peripheral clock notification command is issued, tPCKA timeout control is used. If tPCKA or more time elapses after the peripheral clock notification command is issued before the FRDY bit in FSTATR0 is set to 1, the FCU is initialized and error handling takes place.

In this application note, the tPCKA wait time is implemented by iterating a while loop with the loop count defined by the WAIT\_TPCKA symbolic constant. If running the while **loop** one time requires 11 cycles (which can be confirmed by examining the assembly language output by the compiler), then the number of times it should be run can be calculated as follows:

**while** loop run count = wait duration / (cycle count per **while** loop iteration \* ICLK cycle duration)

Note that the CPU's instruction processing time can differ due to pipelining, so the above-mentioned number of cycles per **while** loop iteration (11 cycles) is a rough estimate of the instruction processing time.

When PCLK is 50 MHz, tPCKA will be 60 [μs]. In the sample program, the wait duration is calculated as 180 [μs] to provide a sufficient margin, as follows:

**while** loop run count = WAIT\_TPCKA = 180 [μs] / (11 \* 10 [ns]) = 1,636.4 (ICLK = 100 MHz)

Therefore, the symbolic constant WAIT\_TPCKA is defined as 1,636.

When using the sample program in your own applications, make sure to evaluate adequately the instruction processing time of the CPU or use a hardware timer to measure the duration.

#### 6.1.2 tRESW2 Wait Control

In tRESW2 wait control, at FCU initialization a software timer is used to control the reset pulse width (tRESW2) during program/erase operations. This period starts when the FRESETR.FRESET bit is set to 1 and extends to the time this bit is set to 0.

Table 37 lists the reset pulse width during program/erase operations.

**Table 37 Reset Pulse Width During Program/Erase Operations**

Item	Symbol	Min.	Max.	Unit	Measuring conditions
Internal reset time (during ROM, data flash programming/erasing) <sup>Note: 2</sup>	t <sub>RESW2</sub> Note: 1	35	—	μs	None

Notes: 1. This item is stipulated for FCU and WDT resets.

2. For details, see the Control Signal Timing section in the Hardware Manual listed in 7. Reference Documents.

During the tRESW2 wait duration, the sample program processes a **while** loop the number of times defined by the symbolic constant WAIT\_TRESW2. If running the while **loop** one time requires four cycles (which can be confirmed by examining the assembly language output by the compiler), then the number of times it should be run can be calculated as follows:

**while** loop run count = wait duration / (cycle count per **while** loop iteration \* ICLK cycle duration)

Note that the CPU's instruction processing time can differ due to pipelining, so the above-mentioned number of cycles per **while** loop iteration (four cycles) is a rough estimate of the instruction processing time.

In the sample program, the wait duration (tRESW2) is calculated as 105 [μs] to provide a sufficient margin, as follows:

**while** loop run count = WAIT\_TRESW2 = 105 [μs] / (4 \* 10 [ns]) = 2,625 (ICLK = 100 MHz)

Therefore, the symbolic constant WAIT\_TRESW2 is defined as 2,625.

When using the sample program in your own applications, make sure to evaluate adequately the instruction processing time of the CPU or use a hardware timer to measure the duration.

### 6.1.3 tE128K × 1.1 Timeout Control

When the FCU transitions to ROM read mode and during erasing of the user MAT, tE128K × 1.1 timeout control is used. In the case of the transition to ROM read mode, a software timer is used to measure the 128 KB erasure block erase duration until the FRDY bit in FSTATR0 is set to 1, before writing AA00h to the FENTRYR register to transition to ROM read mode. When erasing the user MAT, a software timer is used to measure the 128 KB erasure block erase duration from when the block erase command is issued until the FRDY bit in FSTATR0 is set to 1.

Table 38 lists the erase time for 128 KB erase blocks.

**Table 38 Erase Time for 128 KB Erase Blocks**

Item		Symbol	Min.	Type	Max.	Unit	Measuring conditions
Erasure time <sup>Note: 1</sup>	128 KB	t <sub>E128K</sub>	—	800	1750	ms	When PCLK = 50 MHz When the number of erase operations for each block does not exceed 100 times.

Note: 1. See the ROM (Flash Memory for Code Storage) Characteristics section in the Hardware Manual listed in 7. Reference Documents.

During the tE128K × 1.1 wait duration, the sample program processes a **while** loop the number of times defined by the symbolic constant WAIT\_TE128K. If running the **while loop** one time requires 10 cycles (which can be confirmed by examining the assembly language output by the compiler), then the number of times it should be run can be calculated as follows:

$$\text{while loop run count} = \text{wait duration} / (\text{cycle count per while loop iteration} * \text{ICLK cycle duration})$$

Note that the CPU's instruction processing time can differ due to pipelining, so the above-mentioned number of cycles per **while** loop iteration (10 cycles) is a rough estimate of the instruction processing time.

In the sample program, the wait duration (tE128K × 1.1) is calculated as 5,775 [ms] to provide a sufficient margin, as follows:

$$\text{while loop run count} = \text{WAIT\_TE128K} = 5,775 \text{ [ms]} / (10 * 10 \text{ [ns]}) = 57,750,000 \text{ (ICLK} = 100 \text{ MHz)}$$

Therefore, the symbolic constant WAIT\_TE128K is defined as 57,750,000.

When using the sample program in your own applications, make sure to evaluate adequately the instruction processing time of the CPU or use a hardware timer to measure the duration.

### 6.1.4 tP256 × 1.1 Timeout Control

When programming the user MAT, tP256 × 1.1 timeout control is used. A software timer is used to measure the 256 byte programming duration from when the program command is issued until the FRDY bit in FSTATR0 is set to 1. Table 39 lists the write time for 256 bytes.

**Table 39 Write Time for 256 Bytes**

Item		Symbol	Min.	Type	Max.	Unit	Measuring conditions
Programming time <sup>Note: 1</sup>	256 Bytes	t <sub>P256</sub>	—	2	12	ms	When PCLK = 50 MHz When the number of erase operations for each block does not exceed 100 times.

Note: 1. See the ROM (Flash Memory for Code Storage) Characteristics section in the Hardware Manual listed in 7. Reference Documents.

During the tP256 × 1.1 wait duration, the sample program processes a **while** loop the number of times defined by the symbolic constant WAIT\_TP256. If running the **while loop** one time requires 11 cycles (which can be confirmed by examining the assembly language output by the compiler), then the number of times it should be run can be calculated as follows:

$$\text{while loop run count} = \text{wait duration} / (\text{cycle count per while loop iteration} * \text{ICLK cycle duration})$$

Note that the CPU's instruction processing time can differ due to pipelining, so the above-mentioned number of cycles per **while** loop iteration (11 cycles) is a rough estimate of the instruction processing time.

In the sample program, the wait duration (tP256 × 1.1) is calculated as 39.6 [ms] to provide a sufficient margin, as follows:

$$\text{while loop run count} = \text{WAIT\_TP256} = 39.6 \text{ [ms]} / (11 * 10 \text{ [ns]}) = 360,000 \text{ (ICLK} = 100 \text{ MHz)}$$

Therefore, the symbolic constant WAIT\_TP256 is defined as 360,000.

When using the sample program in your own applications, make sure to evaluate adequately the instruction processing time of the CPU or use a hardware timer to measure the duration.

### 6.2 Fixed Vectors in User Boot Mode

Pins MD1 and MD0 are used to specify user boot mode, causing the reset state to be released and a transition to user boot mode. At this time the reset vector is address FF7F FFFCh in the user boot MAT. The normal vector table is referenced for other vectors.

A fixed vector table has fixed allocation addresses. Vectors for privileged instruction exception, undefined instruction exception, floating point exception, non-maskable interrupt, and reset are allocated within the address range FFFF FFD0h to FFFF FFFCh .

Figure 32 shows the fixed vector table settings used in the sample program.

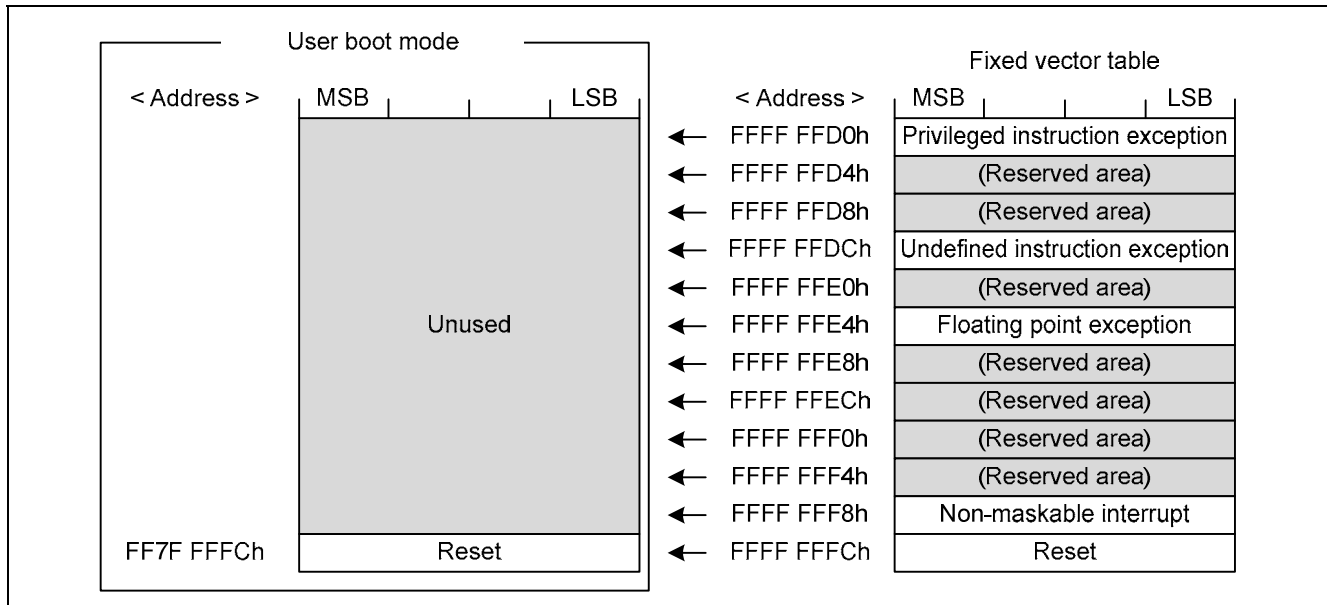


Figure 32 Fixed Vector Table Settings of Sample Program

In the sample program, the slave operates in user boot mode. In the files generated automatically by HEW that contain fixed vector settings for the slave (vecttbl.c, vect.h, and intprg.c), the fixed vector table definitions for the privileged instruction exception (Excep\_SuperVisorInst function), undefined instruction exception (Excep\_UndefinedInst function), floating point exception (Excep\_FloatingPoint function), non-maskable interrupt (NonMaskableInterrupt), and reserved area dummy function are commented out, leaving only the reset vector (4 bytes) allocation of address FF7F FFFC in the user boot MAT.

The slave program portion of the sample program only sets the reset vector in the user boot MAT and makes no fixed vector settings in the user MAT.

Note that when using the slave program portion of the sample program in in your own applications it will be necessary to make appropriate settings for fixed vectors in user MAT (privileged instruction exception, undefined instruction exception, floating point exception, non-maskable interrupt, reserved areas) and to specify exception handlers for each.

### 6.3 Note on Reprogramming Erasure Block EB00

The erasure block EB00 (programming/erase address range: 00FF E000h to 00FF FFFF, read address range: FFFF E000h to FFFF FFFh) contains areas allocated for fixed vectors (FFFF FF80h to FFFF FFFh), ID code protection (FFFF FFA0h to FFFF FFAh), etc.

When EB00 is programmed/erased by setting the erasure block number to EB00\_INDEX, the above-mentioned fixed vector and ID code protection data is erased. It is therefore necessary to make fixed vector and ID code protection settings again after erasing EB00.

ID code protection is a function that disables programming and erasing by the host. ID code protection determinations are made by using a control code and ID code programmed in the ROM. For details of ID code protection, see the Hardware Manual listed in 7, Reference Documents.



## 7. Reference Documents

- User's Manual  
RX610 Group User's Manual: Hardware  
(The latest version can be downloaded from the Renesas Electronics Web site.)  
RX Family User's Manual: Software  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Development Environment Manual  
RX Family C/C++ Compiler Package User's Manual  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates  
(The latest information can be downloaded from the Renesas Electronics Web site.)
- Application Note  
RX610 Group On-chip Flash Memory Reprogramming in the User Boot Mode (Master)  
(The latest information can be downloaded from the Renesas Electronics Web site.)

**Website and Support**

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

## Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.14.11	—	First edition issued
1.01	Sep.02.11	—	volatile __evenaccess declaration added
		16	4.9.3 FCU Commands amended
		25	Table 19 amended
1.02	Feb.29.12	—	Source file (main.c) amended
		48	Figure 23: Corrected. ("Verifies that the ROM read mode transition has completed." added.)
		50	Figure 24: Corrected. ("Verifies that the ROM read mode transition has completed." added.)
		65	Reference Documents: Modified.
—	—	—	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.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.  
"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.  
(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.  
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



### 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