RX630 Group

On-chip Flash Memory Reprogramming in Single-chip Mode via an UART Interface (Slave)

R01AN1251EJ0101
Rev.1.01
Dec 04, 2014

Introduction

This application note describes the write and erase processing for the flash memory (user MAT) using the erasure block number, write data size, and write data transferred by asynchronous serial communication from the master.

When using this application note, see also the RX630 Group document “On-chip Flash Memory Reprogramming in Single-chip Mode via an UART Interface (Master)” (R01AN1271EJ) for details on the processing used to transfer the erasure block number, write data size, and write data using asynchronous serial communication.

Note that this application note utilizes sample code from the following application notes:

RX630 initial settings:
RX630 Group: Initial Setting, rev. 1.00 (R01AN1004EJ0100)

Erasing/programming on-chip flash memory:
RX600 Series: Simple Flash API for RX600, rev. 2.20 (R01AN054EU0220)

Target Device

- RX630 group, 177- or 176- pin version, ROM capacity: 768 KB to 2 MB
- RX630 group, 145- or 144- pin version, ROM capacity: 768 KB to 2 MB
- RX630 group, 100-pin version, ROM capacity: 384 KB to 2 MB
- RX630 group, 80-pin version, ROM capacity: 384 KB to 512 KB

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

Contents

1. Specifications

This application note describes programming and erasing the user MAT using an RX630 Group MCU (R5F5630EDDFP) in single-chip mode.

The slave receives the erasure block number, write data size, and write data using asynchronous serial communication from the master, and performs programming and erase operations required on the user MAT.

The SCI channel 0 (SCI0) module is used for asynchronous serial communication between the master and the slave.

Asynchronous serial communication specifications are as follows:

- Bit rate: 31,250 bps
- Data length: 8 bits
- Parity bits: none
- Stop bits: 1 bit

1. In this application note's sample code, the output level of the P15 pin switches to low when the slave device is ready for communication, hereby notifying the master device that it is ready. The master should be connected to the pin used to monitor the ready-to-communicate state of the slave.
2. Handshaking is used to control communication between the master and slave. After processing the data received from the master, the slave returns an [ACCEPTABLE] command (55h) to the master. The master starts the next serial transfer after receiving the [ACCEPTABLE] command from the slave.
3. The slave erases the specified erasure block and writes the received write data to the erased block, starting at the start address of that block. In this application the program codes are located in EB00 and EB01, so these blocks cannot be erased or programmed. Specifying EB00 or EB01 as an erasure block number results in an error.
4. When the slave successfully finishes erasing or programming the user MAT, it reports the normal end by means of four LEDs connected to I/O ports, and the output level of the P15 pin returns to high (slave in not-ready state). Note that if an error occurs during communication with the master or during erasing or programming, the slave returns to the not-ready state and notification of the error is made by means of the LEDs.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Specifications.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
ROM (flash memory for code storage)	ROM P/E mode is used to reprogram the on-chip flash memory.
Serial communication interface	Used for asynchronous serial communication with the master device.

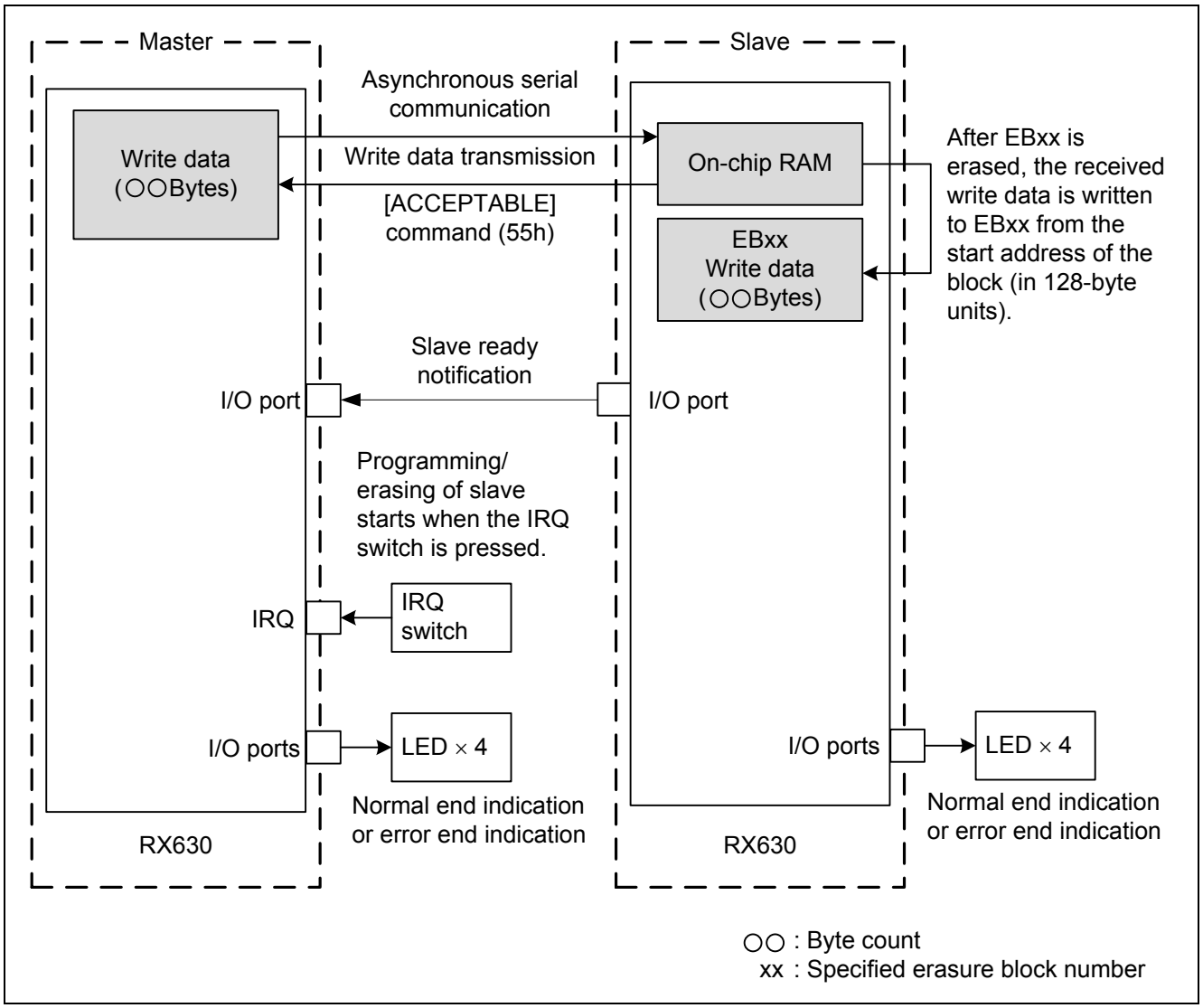


Figure 1.1 Specifications

2. Operation Confirmation Conditions

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

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F5630EDDFP (RX630 Group)
Operating frequency	Main clock: 12.0 MHz PLL: 192 MHz (main clock divided by 1 and multiplied by 16) System clock (ICLK): 96 MHz (PLL divided by 2) Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4) USB clock supplied to USB (UCLK): 48 MHz (PLL divided by 4) Flash IF clock (FCLK): 48 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics RX Standard Toolchain Version 1.2.1.0
	Compiler options -cpu=rx600 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo
iodefine.h version	1.0A
Endian	Little endian
Operating mode	Single-chip mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit for RX630 (Product No. R0K505630C000BE)
Optimizing Linkage Editor (rom option)*	-rom=D=R,D_1=R_1,D_2=R_2,PFRAM=RPFRAM

Note: * See 6.5, rom Option.

3. Reference Application Notes

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

- RX600 Series Simple Flash API for RX600, rev. 2.20 (R01AN0544EU)
- RX630 Group Initial Setting, rev. 1.00 (R01AN1004EJ)

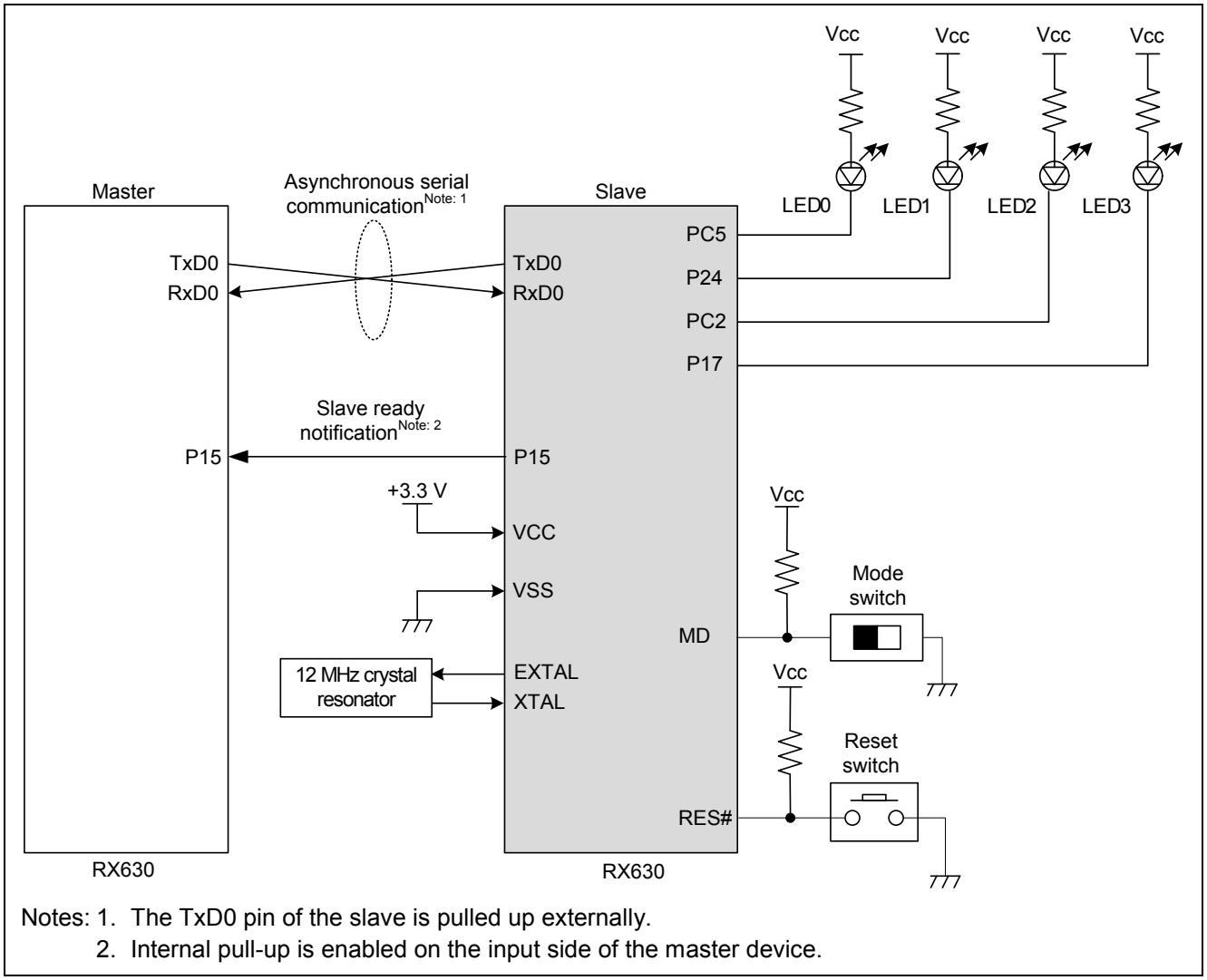
The functions from the above application notes are used in the sample code of this application note. The revision numbers indicated are current as of the time this application note was made.

If a more recent version is available, use it in place of the version provided here. The latest version can be obtained from the Renesas Electronics Web site.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a hardware configuration diagram of the slave device as used in this application note.



Notes: 1. The TxD0 pin of the slave is pulled up externally.
2. Internal pull-up is enabled on the input side of the master device.

Figure 4.1 Hardware Configuration Diagram of Slave Device

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

The pins described here are for 100-pin products. When using an MCU version with fewer than 100 pins, adjust the pin assignments as appropriate.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P20/TXD0	Output	Master communication serial transmit pin
P21/RXD0	Input	Master communication serial receive pin
P15	Output	Slave ready notification pin
PC5	Output	LED0 connection pin (high-level output: turned off, low-level output: turned on)
P24	Output	LED1 connection pin (high-level output: turned off, low-level output: turned on)
PC2	Output	LED2 connection pin (high-level output: turned off, low-level output: turned on)
P17	Output	LED3 connection pin (high-level output: turned off, low-level output: turned on)

5. Software

5.1 Operation Overview

5.1.1 Asynchronous Serial Communication Specifications

This application note uses asynchronous serial communication between the master and the slave to receive communication commands ([FSTART], [ERASE], [WRITE]), the erasure block number, the write data size, and the write data. Note that the slave transmits the [ACCEPTABLE] command (55h) as a status command for handshaking.

Table 5.1 lists the asynchronous serial communication specifications.

Table 5.1 Asynchronous Serial Communication Specifications

Item	Specification
Channel	SCI channel 0 (SCI0)
Communication mode	Asynchronous mode
Bit rate	31,250 bps (PCLKB = 48 MHz)
Data length	8 bits
Parity bit	None
Stop bit	1 bit
Error	Overrun error, framing error

5.1.2 Communication Command Specifications

Table 5.2 lists the specifications of the communication commands sent between the master and slave.

Table 5.2 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
ACCEPTABLE	55h	Status command used by the slave to inform the master that it is able to receive data from the master	Slave → Master

5.1.3 Communication Sequence

Figures 5.1 to 5.4 show the communication sequence between master and slave.

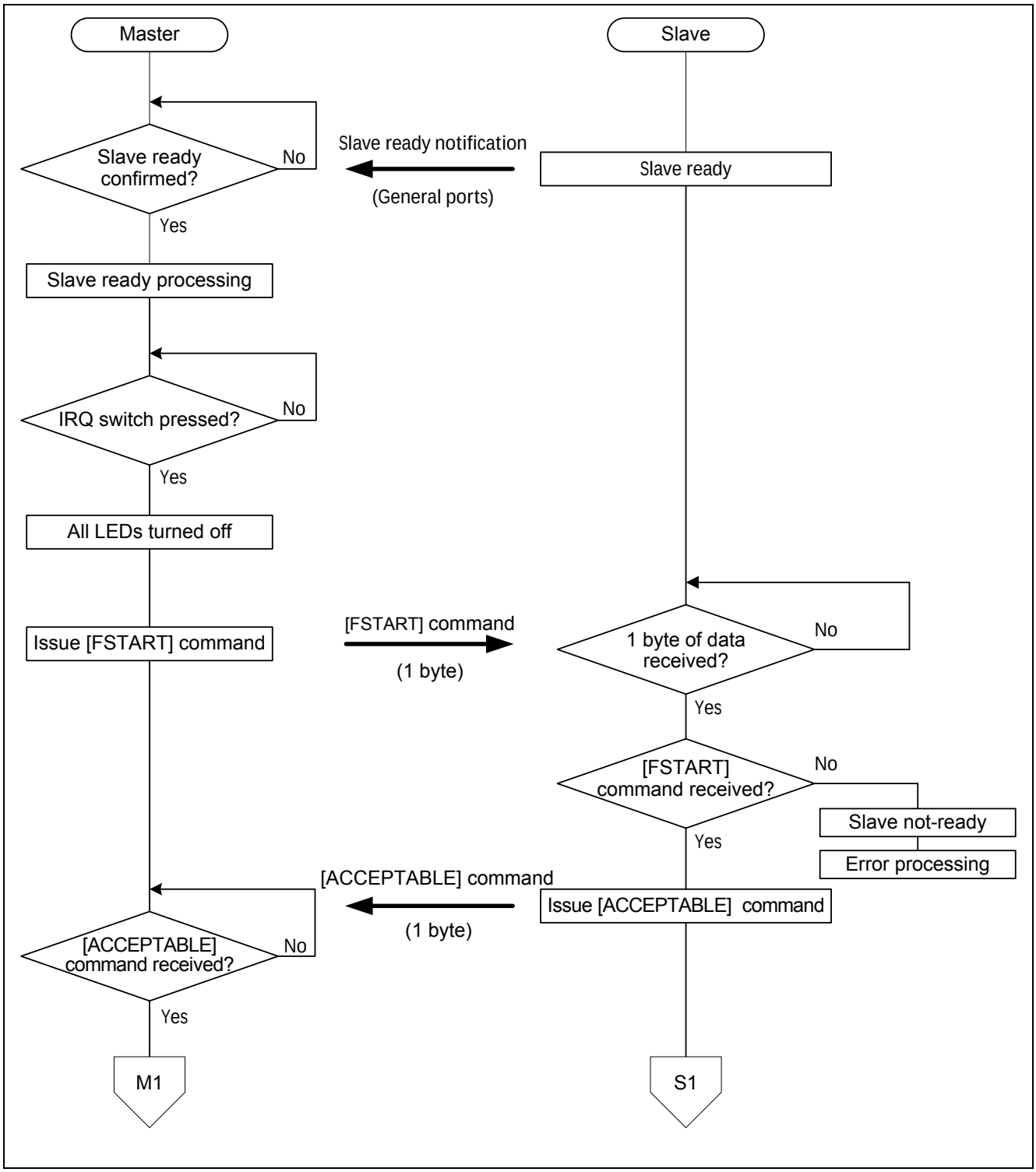


Figure 5.1 Communication Sequence (1)

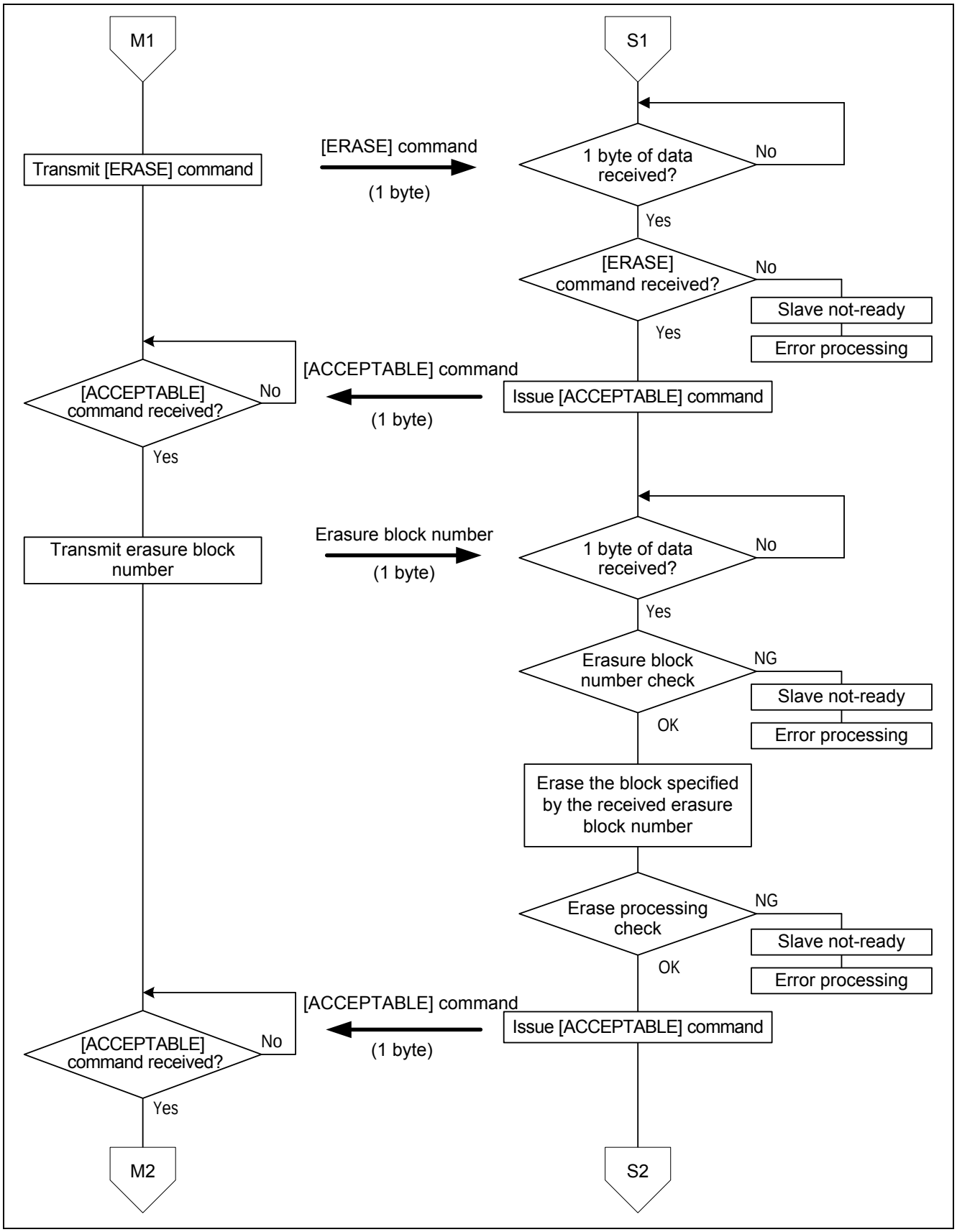
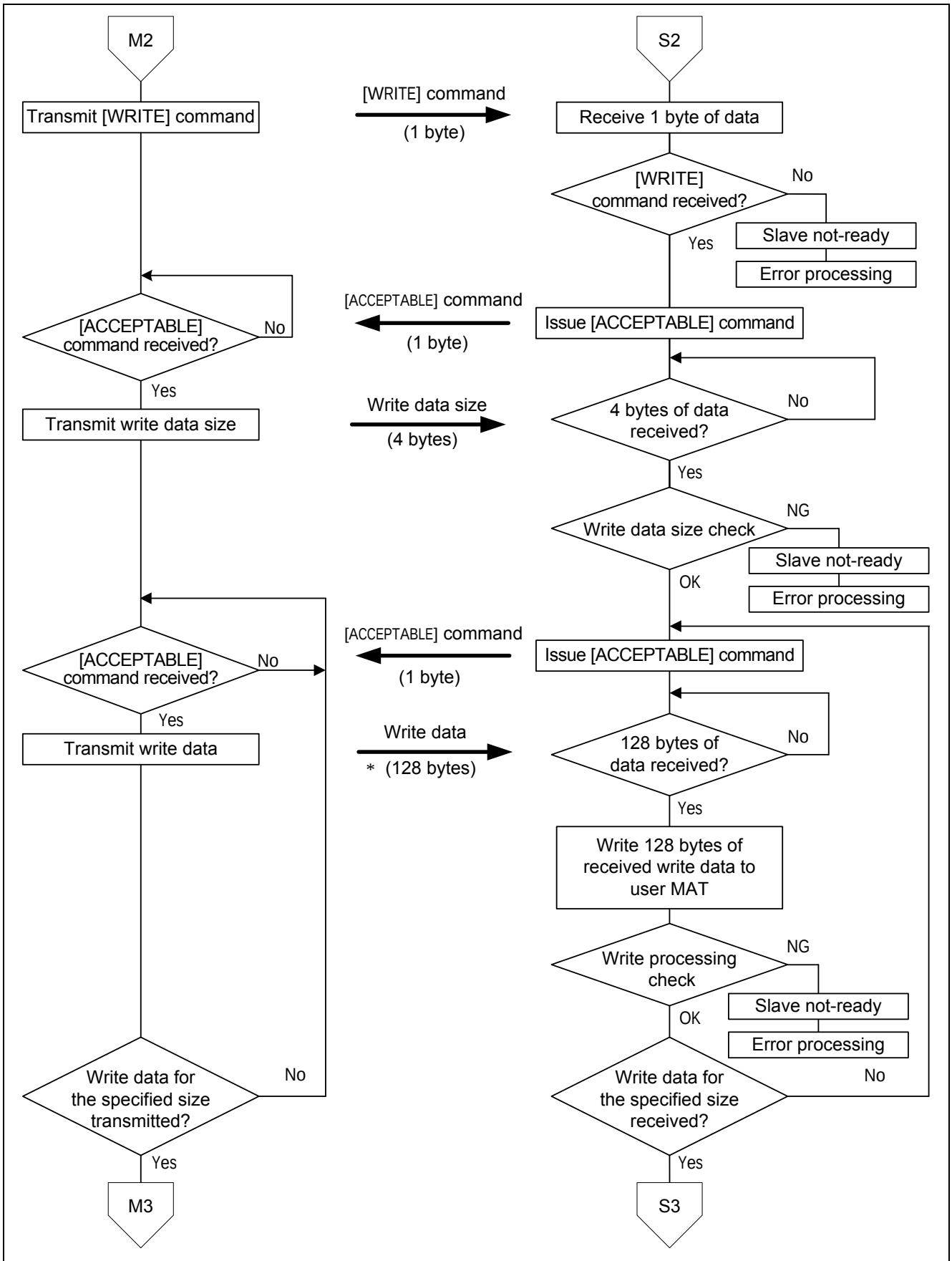


Figure 5.2 Communication Sequence (2)



Note: * Operation may not work correctly when transmitting more than 129 bytes of data.

Figure 5.3 Communication Sequence (3)

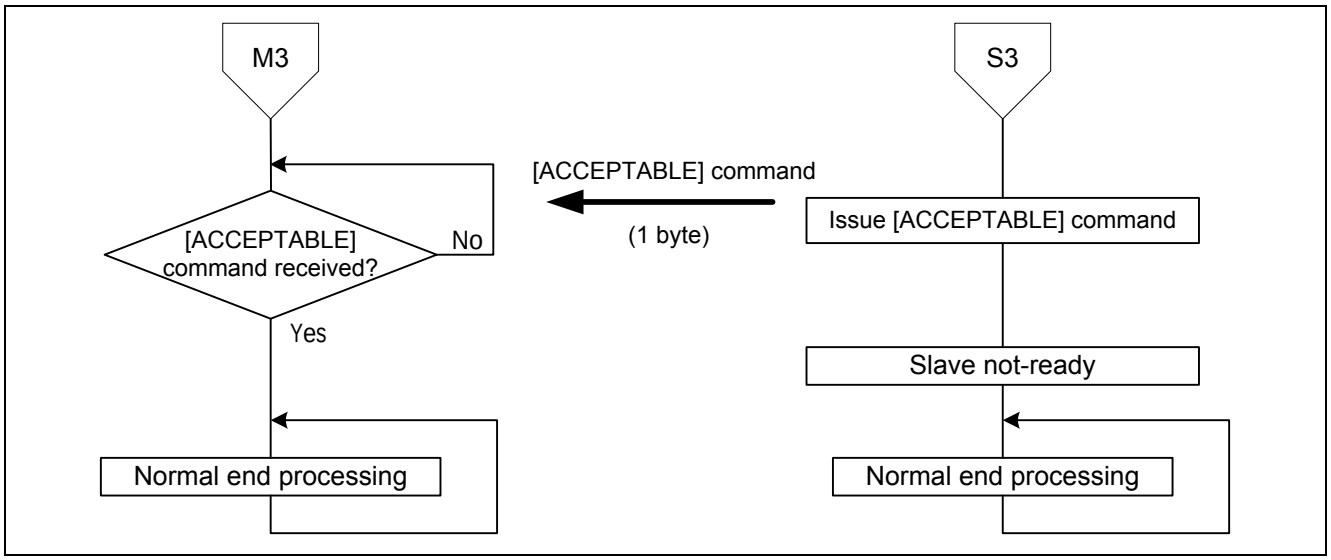


Figure 5.4 Communication Sequence (4)

5.1.4 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).

Figure 5.5 shows the specifications of the erasure block number. See section 5.2.1, User MAT Erase Processing, for details on the erasure block numbers.

Erasure block number (uint8_t type)

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

The erasure block number is [08h] when programming or erasing erasure block EB08 of the slave.

Note: A value corresponding to EB02 to EB69 must be specified as the erasure block number. If a value other than an erasure block number is specified, the slave will recognize an error and perform error handling.

Figure 5.5 Erasure Block Number Specifications

5.1.5 Write Data Size

After receiving a [WRITE] command from the master, the slave receives 4 bytes of write data size. Figure 5.6 shows the specifications of the write data size.

Write data size (uint32_t 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

When the write size is 8 KB 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 erasure block size for the specified erasure block. If 0 or a value greater than the erasure block size is specified, the slave will recognize an error and perform error handling.

2. The size of write data reception is fixed to 128 bytes, so write data is received 128 bytes at a time. If the write data size is not a multiple of 128 bytes, the final unit of write data, which is less than 128 bytes, is filled in with bytes of value FFh as padding, and this is written to the user MAT of the slave.

Figure 5.6 Write Data Size Specifications

5.1.6 Overrun Error

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

5.1.7 Framing Error

In this application note, if a framing error occurs during slave asynchronous serial communication reception (the SCI0.SSR.FER bit is set to 1), the slave will perform error handling.

5.2 User MAT Write and Erase

See the “RX600 Simple Flash API” application note listed in section 8, Reference Documents, for details on the RX600 Simple Flash API used for programming and erase in this application note.

5.2.1 User MAT Erase Processing

The R_FlashErase() function provided by the RX600 Simple Flash API is used for user MAT erase processing. Therefore the block numbers for erasing are the same as the values specified to the R_FlashErase() function. Erase errors are indicated in the return value from the R_FlashErase() function.

5.2.2 User MAT Write Processing

The R_FlashWrite() function provided by the RX600 Simple Flash API is used for user MAT write processing. Write errors are indicated in the return value from the R_FlashWrite() function.

5.2.3 Changes to the Simple Flash API

The Simple Flash API r_flash_api_rx600_config.h file is modified for this application note.

Table 5.3 lists the changes to r_flash_api_rx600_config.h.

Table 5.3 Changes to r_flash_api_rx600_config.h

Item Changed	Place Changed
Changes to Simple Flash API settings	//#define IGNORE_LOCK_BITS //#define COPY_CODE_BY_API //#define FLASH_API_USE_R_BSP

5.2.4 Notes on Using Interrupts

Flash ROM cannot be access during ROM programming or erase operations. Therefore it is necessary to prevent access to the ROM by interrupts that occur during programming or erase operations. The sample code of this application note does not use interrupts, and interrupts are not suppressed. If interrupts are used, the definition “#define FOR_INTERRUPTION” may be added to the sample code to enable the lines enclosed in red boxes below, which suppress interrupts. (The sample code as issued does not include the “#define FOR_INTERRUPTION” definition, so the lines enclosed in red boxes have no effect.)

The processing performed by (1) to (3), which are enclosed in red boxes in figure 5.7, is as follows:

(1) The current interrupt priority level (IPL) is saved to variable flash_pipl.
(2) Before programming/erasing starts, the IPL is changed to FLASH_READY_IPL. The value of FLASH_READY_IPL is defined as 5 in the sample code, so interrupts with an IPR setting of 5 or below are disabled. For details of FLASH_READY_IPL, see Simple Flash API for RX600, listed in 8, Reference Documents.
(3) After programming/erasing finishes, IPL is restored to its previous value.

The above process suppresses interrupts which access the ROM by temporarily changing the IPL setting.

```
#ifdef FOR_INTERRUPTION
/* Holds IPL of processor before flash operation */
 static unsigned char flash_pipl;

/* Save current processor IPL */
(1) → flash_pipl = get_ipl();                /* If your system is using the interrupt, enable this line. */

/* Set the processor IPL so that interrupts that access ROM will not occur during ROM program/erase operations. */
(2) → set_ipl(FLASH_READY_IPL);        /* If your system is using the interrupt, enable this line. */
#endif

/* Erasure process by using "simple API" */
    fcu_status = R_FlashErase((uint8_t)target_eb);

/* Programming process by using "simple API" */
      fcu_status = R_FlashWrite((uint32_t)fcu_info.p_write_adrs_now,
      (uint32_t)wrdata_buffer,ROM_PROGRAM_SIZE);  /* Program 128 byte data to the target EB by using "simpleAPI"
                                                     * (call of R_FlashWrite function in "simpleAPI") */
#ifdef FOR_INTERRUPTION
/* Restore processor IPL */
(3) → set_ipl(flash_pipl);                /* If your system is using the interrupt, enable this line. */
#endif
```

Figure 5.7 Interrupt Suppression Example

5.3 Slave Ready Processing

When the slave is ready for communication, a low-level signal is output from P15 which is the slave ready notification pin. The master should be connected to the pin used to monitor the ready-to-communicate state of the slave.

High-level output from P15 indicates a slave not-ready condition, and commands to program or erase the user MAT are not acknowledged.

5.4 LED Indications

Figure 5.8 lists the LED indications corresponding to the operating states of the sample code.

Error Number	Operating State	LED Indication				
		LED3	LED2	LED1	LED0	Sequence
	Normal end (Indication shifts at fixed intervals.)	●	●	●	○	
		●	●	○	●	
		●	○	●	●	
		○	●	●	●	
Error No.01	[FSTART] command error	●	●	●	⊗	
Error No.02	[ERASE] command error	●	●	⊗	●	
Error No.03	Erasure block number error	●	●	⊗	⊗	
Error No.04	Erase processing error	●	⊗	●	●	
Error No.05	[WRITE] command error	●	⊗	●	⊗	
Error No.06	Write data size error	●	⊗	⊗	●	
Error No.07	Program operation error	●	⊗	⊗	⊗	
Error No.08	Overrun error	⊗	●	●	●	
Error No.09	Framing error	⊗	●	●	⊗	

On: ○ Flashing: ⊗ Off: ●

Figure 5.8 LED Indications

5.5 Handshaking Control

The slave uses handshaking with the master for communications control.

For handshaking control, after the slave receives serial communication from the master, performs the processing for the received data, and then returns an [ACCEPTABLE] command (55h) when it is ready to receive the next serial communication. The master starts the next serial communication after it has received an [ACCEPTABLE] command from the slave.

5.6 Section Settings

Table 5.4 shows the section settings for the slave device.

Table 5.4 Section Settings of Slave Device

Section	Start Address	Description
RPFRAM	0000 0000h	Area mapped in RAM as the [PFRAM] section by the ROM support option.
B_1	0000 1000h	Uninitialized data area (ALIGN = 1)
R_1		Area mapped in RAM as the [D-1] section by the ROM support option.
B_2		Uninitialized data area (ALIGN = 2)
R_2		Area mapped in RAM as the [D-2] section by the ROM support option.
B		Uninitialized data area (ALIGN = 4)
R		Area mapped in RAM as the [D] section by the ROM support option.
SI		Interrupt stack area
PResetPRG	FFFF E000h	Program area (PowerON_Reset_PC program)
P		Program area
PIntPRG		Program area (interrupt program)
C_1	FFFF EE00h	Constant area (ALIGN = 1)
C_2		Constant area (ALIGN = 2)
C		Constant area (ALIGN = 4)
C$*		Section initialization table of uninitialized data area, relocatable vector area
D_1		Initialized data area (ALIGN = 1)
D_2		Initialized data area (ALIGN = 2)
D		Initialized data area (ALIGN = 4)
W*		Switch statement branch table area
PFRAM	FFFF F400h	Program area (user MAT programming/control program)
FIXEDVECT	FFFF FFD0h	Fixed vector area

5.7 File Composition

Table 5.5 lists the files used in the sample code. Files automatically generated by the integrated development environment should not be listed in this table.

Table 5.5 Files Used in the Sample Code

File Name	Description	Remarks
r_init_stop_module.c	Stop processing for active peripheral functions after a reset.	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Initial settings of nonexistent ports	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Initial clock settings	
r_init_clock.h	Header file for r_init_clock.c	
resetprg.c	Initial settings	In the PowerON_Reset_PC function, uncomment the call to the HardwareSetup function, so that the HardwareSetup function in the main.c file is called by the PowerON_Reset_PC function.
main.c	This program handles the following operations: main processing; reception and transmission control for communication commands using asynchronous serial communication with the master, reception control for the erasure block number, the write data size, and the write data, control of LED display at normal end and when an error occurs.	
r_flash_api_rx600.c	RX600 Series: Simple Flash API program for RX600	For details, see the application note RX600 Series: Simple Flash API for RX600.
r_flash_api_rx600.h r_flash_api_rx600_private.h	Include header file of RX600 Series: Simple Flash API program for RX600 for external reference	
r_flash_api_rx600_config.h mcu_info.h	Include header file of RX600 Series: Simple Flash API for RX600 for parameter settings	

5.8 Option-Setting Memory

Table 5.6 lists the states of the option settings memory used in the sample code. These settings should be changed as needed to values optimized for the target system. Also note that the sample code assumes that the same endianness settings are used for the master and slave.

Table 5.6 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	IWDT stopped after a reset WDT stopped after a reset
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	Voltage monitor reset 0 disabled after a reset HOCO oscillation disabled after a reset
MDES*	FFFF FF83h to FFFF FF80h		(Single-chip mode)
		FFFF FFFFh	Little-endian
		FFFF FFF8h	Big-endian

Note: * The sample code uses the little-endian setting. For information on changing the endianness, see 6.4, Endianness.

5.9 Constants

Table 5.7 lists the constants used in the sample code.

Table 5.7 Constants Used in the Sample Code (1)

Constant Name	Setting Value	Contents
FSTART	0x10	Programming/erase start command
ERASE	0x11	Erase start command
WRITE	0x12	Programming start command
ACCEPTABLE	0x55	Status command sent to the slave
LED_ON	0	Set value used when the LED is on
LED_OFF	1	Set value used when the LED is off
RSK_LED0	PORTC.PODR.BIT.B5	On/off control of LED 0 on the evaluation board
RSK_LED1	PORT2.PODR.BIT.B4	On/off control of LED 1 on the evaluation board
RSK_LED2	PORTC.PODR.BIT.B2	On/off control of LED 2 on the evaluation board
RSK_LED3	PORT1.PODR.BIT.B7	On/off control of LED 3 on the evaluation board
RSK_LED0_PDR	PORTC.PDR.BIT.B5	I/O control for LED 0 on the evaluation board
RSK_LED1_PDR	PORT2.PDR.BIT.B4	I/O control for LED 1 on the evaluation board
RSK_LED2_PDR	PORTC.PDR.BIT.B2	I/O control for LED 2 on the evaluation board
RSK_LED3_PDR	PORT1.PDR.BIT.B7	I/O control for LED 3 on the evaluation board
NOT_READY	1	Output level indicating slave not-ready
READY	0	Output level indicating slave ready
SLAVE_READY_PODR	PORT1.PODR.BIT.B5	I/O control of slave ready confirmation port
SLAVE_READY_PDR	PORT1.PDR.BIT.B5	I/O control of slave ready confirmation port
WAIT_LED	2000000	Time data that is the interval for an LED to be turned on when the slave has completed programming/erasing the user MAT successfully.
ROM_PROGRAM_SIZE	128*	Sets the write unit appropriate to the device used for programming the user MAT. This value is specified in r_flash_api_rx600_private.h, and this file is included so that it can be referenced.
RxD0_PMR	PORT2.PMR.BIT.B1	RxD0 port mode register setting
TxD0_PMR	PORT2.PMR.BIT.B0	TxD0 port mode register setting
ERROR_NO_01	1	Data indicating error state
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	

Note: * These values are valid when an RX630 group MCU is used.

Table 5.8 Constants Used in the Sample Code (2)

Constant Name	Setting Value	Contents
WRITE_ADRS_TOP_64K	0x00E00000	Start address of the 64 KB block size area in the programming/erase address space
WRITE_ADRS_TOP_32K	0x00F00000	Start address of the 32 KB block size area in the programming/erase address space
WRITE_ADRS_TOP_16K	0x00F80000	Start address of the 16 KB block size area in the programming/erase address space
WRITE_ADRS_TOP_4K	0x00FF8000	Start address of the 4 KB block size area in the programming/erase address space
BLK_SIZE_64K	64 × 1024	The size of each block in EB54 to EB69
BLK_SIZE_32K	32 × 1024	The size of each block in EB38 to EB53
BLK_SIZE_16K	16 × 1024	The size of each block in EB08 to EB37
BLK_SIZE_4K	4 × 1024	The size of each block in EB00 to EB07

5.10 Structure/Union List

Figure 5.9 shows the structure used in the sample code.

```
typedef struct
{
    uint8_t *p_write_adrs_top;      /* Start address of the erasure block for programming */
    uint8_t *p_write_adrs_end;      /* End address of the erasure block for programming */
    uint8_t *p_write_adrs_now;      /* Target address for programming */
    uint32_t eb_block_size;         /* Block size of the target erasure block */
}st_fcu_info_t;
```

Figure 5.9 Structure Used in the Sample Code

5.11 Variables

Table 5.9 lists the global variables.

Table 5.9 Global Variables

Type	Variable Name	Contents	Function Used
uint8_t	wrdata_buffer [ROM_PROGRAM_SIZE]	Array (128 bytes) for storing 128 bytes of write data received from master	Flash_Update
st_fcu_info_t	fcu_info	Structure (16 bytes) for storing FCU-related address information used when programming/erasing the user MAT	Flash_Update

5.12 Functions

Table 5.10 lists the Functions. Note that functions used by the sample flash API are omitted.

Table 5.10 Functions

Function Name	Outline
HardwareSetup	MCU initial settings function
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Initial settings of nonexistent ports
R_INIT_Clock	Initial clock settings
main	Main function
Flash_Update	User MAT programming/erasing function
Indicate_Ending_LED	Normal end processing function
Indicate_Error_LED	Error end processing function
SCI_Rcv1byte	1 byte data reception function
SCI_Rcvnbyte	n byte data reception function
SCI_Trs1byte	1 byte data transmission function
mpc_init	MPC initial settings function
pmr_init	PMR initial settings function

5.13 Function Specifications

The following tables list the sample code function specifications.

HardwareSetup	
Outline	MCU initial settings function
Header	iodefine.h, r_init_clock.h, r_init_non_existent_port.h
Declaration	void HardwareSetup(void)
Description	The HardwareSetup function makes initial settings to the MCU. • Initial settings for nonexistent ports (100-pin version). • Clock settings (settings including system clock (ICLK) and peripheral module clock B (PCLKB)). • Initial output settings for I/O ports (PC5, P24, PC2, and P17) connected to LED0 to LED3. • Cancels the module stop state. • Multifunction pin controller (MPC) and port mode register (PMR) settings. • Initial settings to the SCI0.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configures the setting to enter the module-stop state.
Arguments	None
Return Value	None
Remarks	In the sample code no transition to the module stop state occurs. For details of this function, see the application note RX630 Group: Initial Setting, rev. 1.00.

R_INIT_NonExistentPort	
Outline	Initial settings of nonexistent ports
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Performs initial settings to the port direction register for nonexistent ports on MCU products with fewer than 176 pins.
Arguments	None
Return Value	None
Remarks	The settings in the sample code are for the 100-pin version (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. In the sample code no transition to the module stop state occurs. For details of this function, see the application note RX630 Group: Initial Setting, rev. 1.00.

R_INIT_Clock	
Outline	Initial clock settings
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Configures initial clock settings.
Arguments	None
Return Value	None
Remarks	The sample code uses the PLL as the system clock and selects processing without using a subclock. In the sample code no transition to the module stop state occurs. For details of this function, see the application note RX630 Group: Initial Setting, rev. 1.00.

main	
Outline	Main processing
Header	iodefine.h
Declaration	void main(void)
Description	The main function notifies of the ready state by outputting a low-level signal from P15. After this, it controls reception of one byte of data from the master, calls the Indicate_Error_LED function if an error occurs, and calls the user MAT programming/control program (Flash_Update function) in the on-chip RAM.
Arguments	None
Return Value	None

Flash_Update	
Outline	User MAT programming/erasing function
Header	iodefine.h, mcu_info.h, r_flash_api_rx600.h, r_flash_api_rx600_private.h
Declaration	void Flash_Update(void)
Description	The Flash_Update function receives the erasure block number, write data size, and write data from the master by asynchronous serial communication, and performs programming/erasing of the user MAT. In case of normal end, it calls the Indicate_Ending_LED function, and in case of error end, it calls the Indicate_Error_LED function. • Receives the FSTART communication command from the master and starts processing. • After receiving the ERASE communication command from the master, receives a 1-byte erasure block number and calls the R_FlashErase function to erase the block. • After receiving the WRITE communication command from the master, receives a 4-byte write data size. • Receives 128 bytes of data from the master, calls the R_FlashWrite as many times as required for the programming data size to write the data to the user MAT. • When programming/erasing of the user MAT finishes successfully, notifies of the slave not-ready state and calls the Indicate_Ending_LED function to display a normal end indication on the LEDs.
Arguments	None
Return Value	None

Indicate_Ending_LED	
Outline	Normal end processing function
Header	None
Declaration	void Indicate_Ending_LED(void)
Description	When programming/erasing finishes successfully, the Indicate_Ending_LED function displays a normal end indication on LED0 to LED3. LED0 to LED3 are turned on one at a time in sequence.
Arguments	None
Return Value	None

SCI_Rcv1byte	
Outline	1 byte data receive function
Header	iodefine.h
Declaration	uint8_t SCI_Rcv1byte(void)
Description	The SCI_Rcv1byte function uses SCI0 asynchronous serial communication to control reception of one byte of data.
Arguments	None
Return Value	One byte of data received using SCI0 asynchronous serial communication.
Remarks	None

Indicate_Error_LED		
Outline	Error end processing function	
Header	None	
Declaration	void Indicate_Error_LED(uint8_t error_no)	
Description	When an error occurs during programming/erasing, the Indicate_Error_LED function displays an indication of the error number on LED0 to LED3. LEDs are turned on to indicate the error number and all turned off repeatedly.	
Arguments	uint8_t error_no	: Number of error that occurred during programming/erasing of user MAT*
Return Value	None	
Remarks	None	

Note: * See 5.4, LED Indications, for a list of the error numbers.

SCI_Rcvnbyte		
Outline	*n* byte data receive function	
Header	iodefine.h	
Declaration	void SCI_Rcvnbyte(uint16_t size, uint8_t *rcv_buffer)	
Description	The SCI_Rcvnbyte function controls reception of n-byte data with SCI0 asynchronous serial communication (*n* is the first argument in uint16_t type).	
Arguments	uint16_t size	: Number of bytes to be received using SCI0 asynchronous serial communication
	uint8_t *rcv_buffer	: Start address of receive data storage location
Return Value	None	

SCI_Trs1byte	
Outline	1 byte data transmit function
Header	iodefine.h
Declaration	void SCI_Trs1byte(uint8_t data)
Description	The SCI_Trs1byte function controls transmission of 1 byte data with SCI0 asynchronous serial communication.
Arguments	uint8_t data : One byte of data transmitted using SCI0 asynchronous serial communication
Return Value	None

mpc_init	
Outline	MPC initial settings
Header	iodefine.h
Declaration	void mpc_init(void)
Description	Selects the following MPC functions: P20 → TXD0 P21 → RXD0
Arguments	None
Return Value	None

pmr_init	
Outline	PMR initial settings
Header	iodefine.h
Declaration	void pmr_init(void)
Description	Configures initial settings for the PMR. • P20 and P21 are used as peripheral functions.
Arguments	None
Return Value	None

5.14 Flowcharts

5.14.1 Initial Settings Function

Figure 5.10 is a flowchart of the initial settings function.

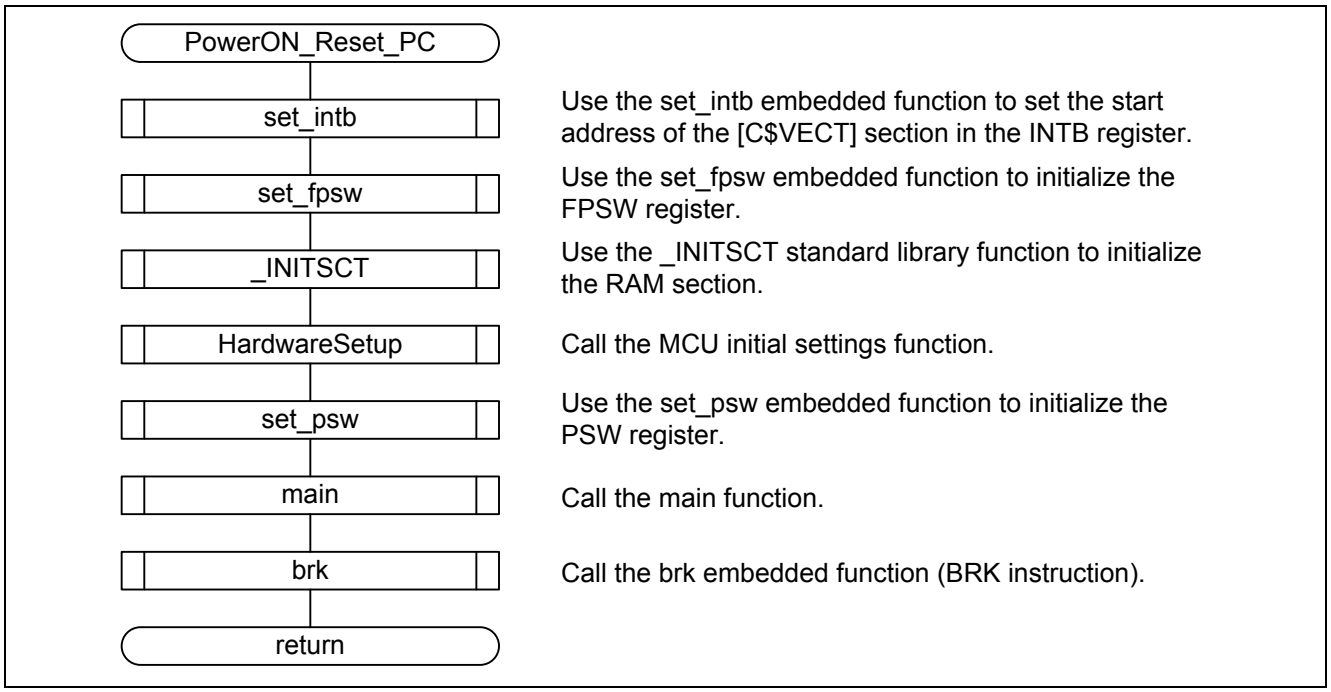


Figure 5.10 Initial Settings Function

5.14.2 Initialization Processing

Figure 5.11 is a flowchart of the initialization processing.

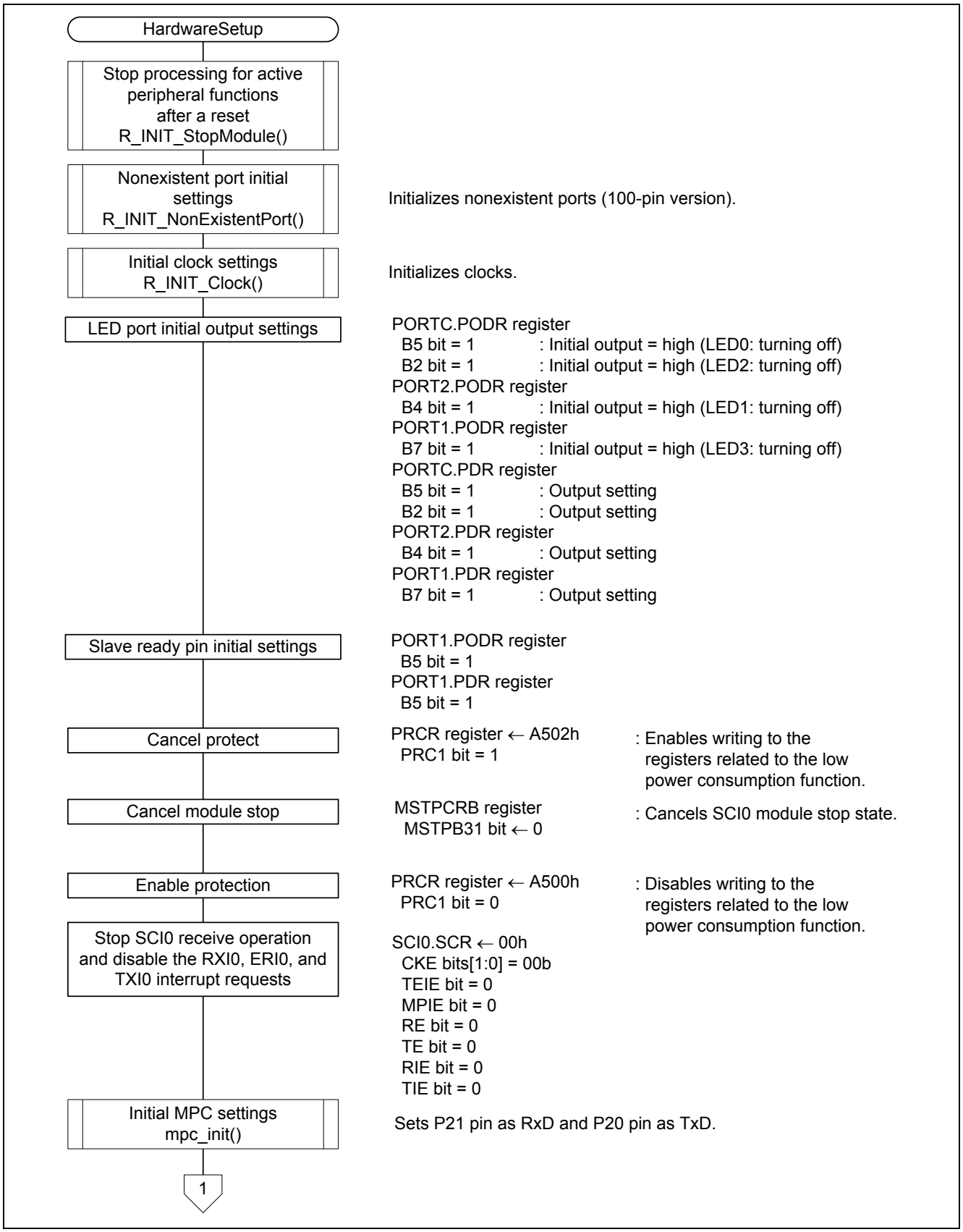


Figure 5.11 Initialization Processing (1)

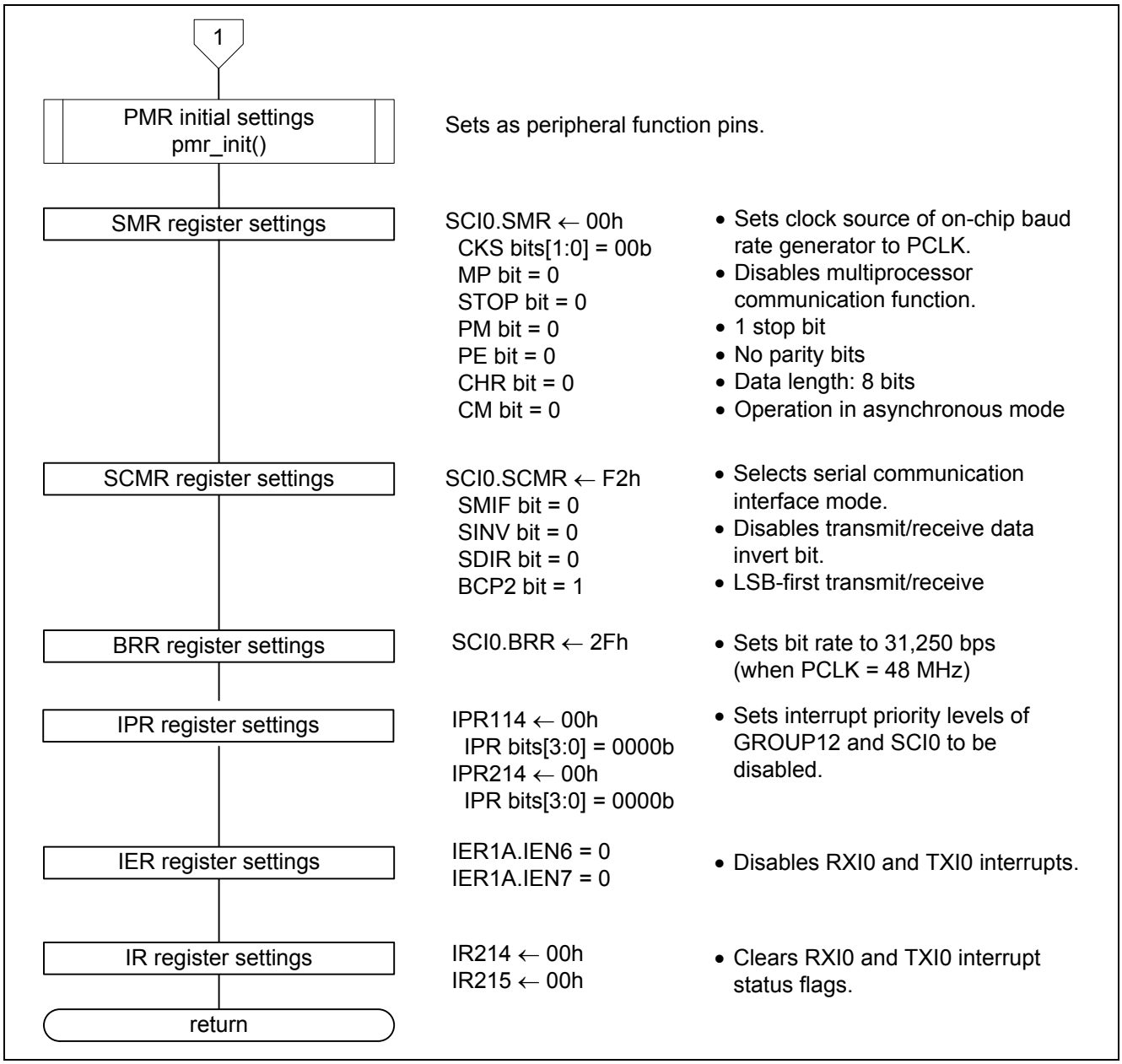


Figure 5.12 Initialization Processing (2)

5.14.3 Main Function

Figure 5.13 is a flowchart of the main function.

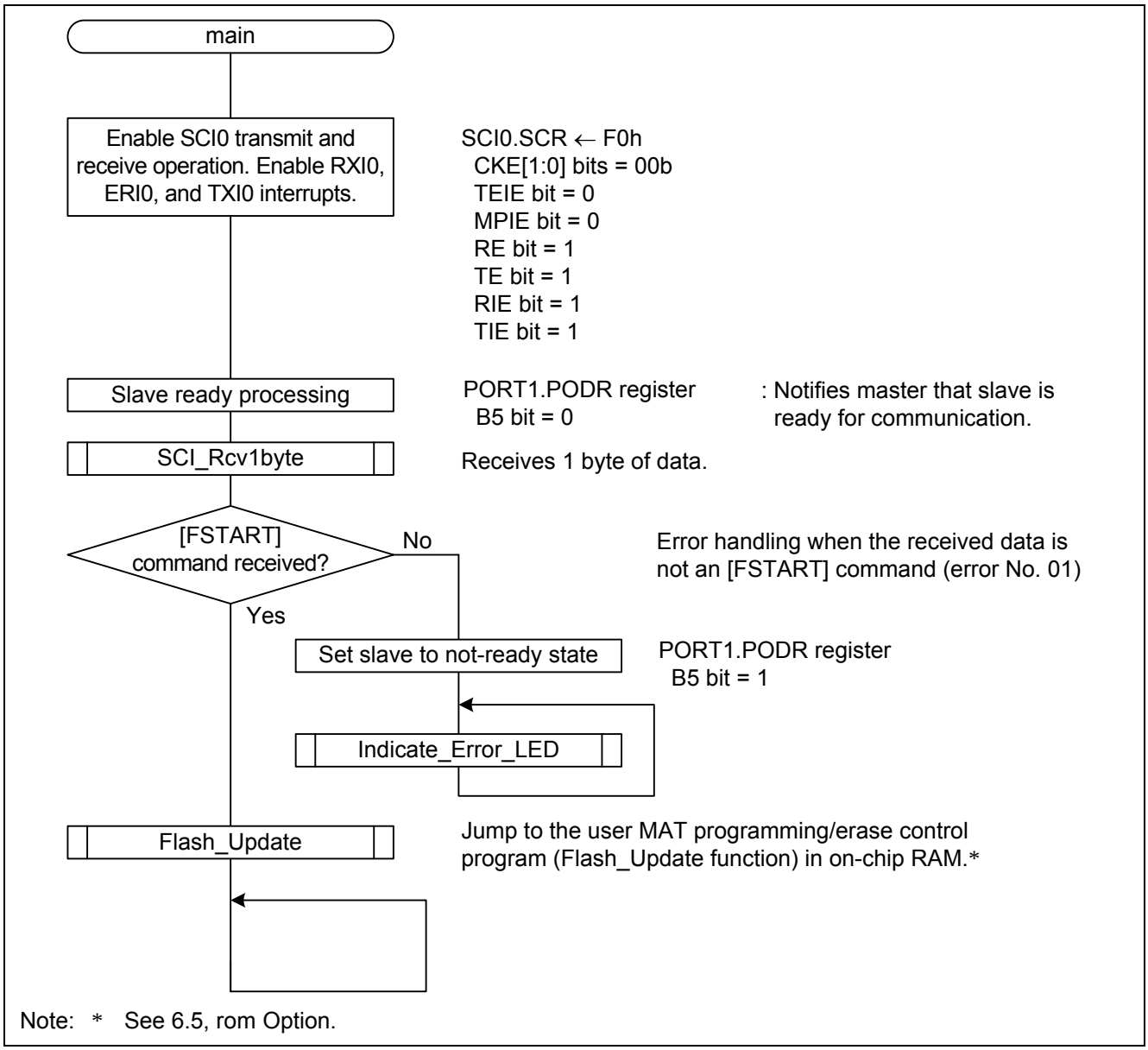


Figure 5.13 Main Function

5.14.4 Programming/Erasing

Figures 5.14 to 5.16 are flowcharts of programming/erasing.

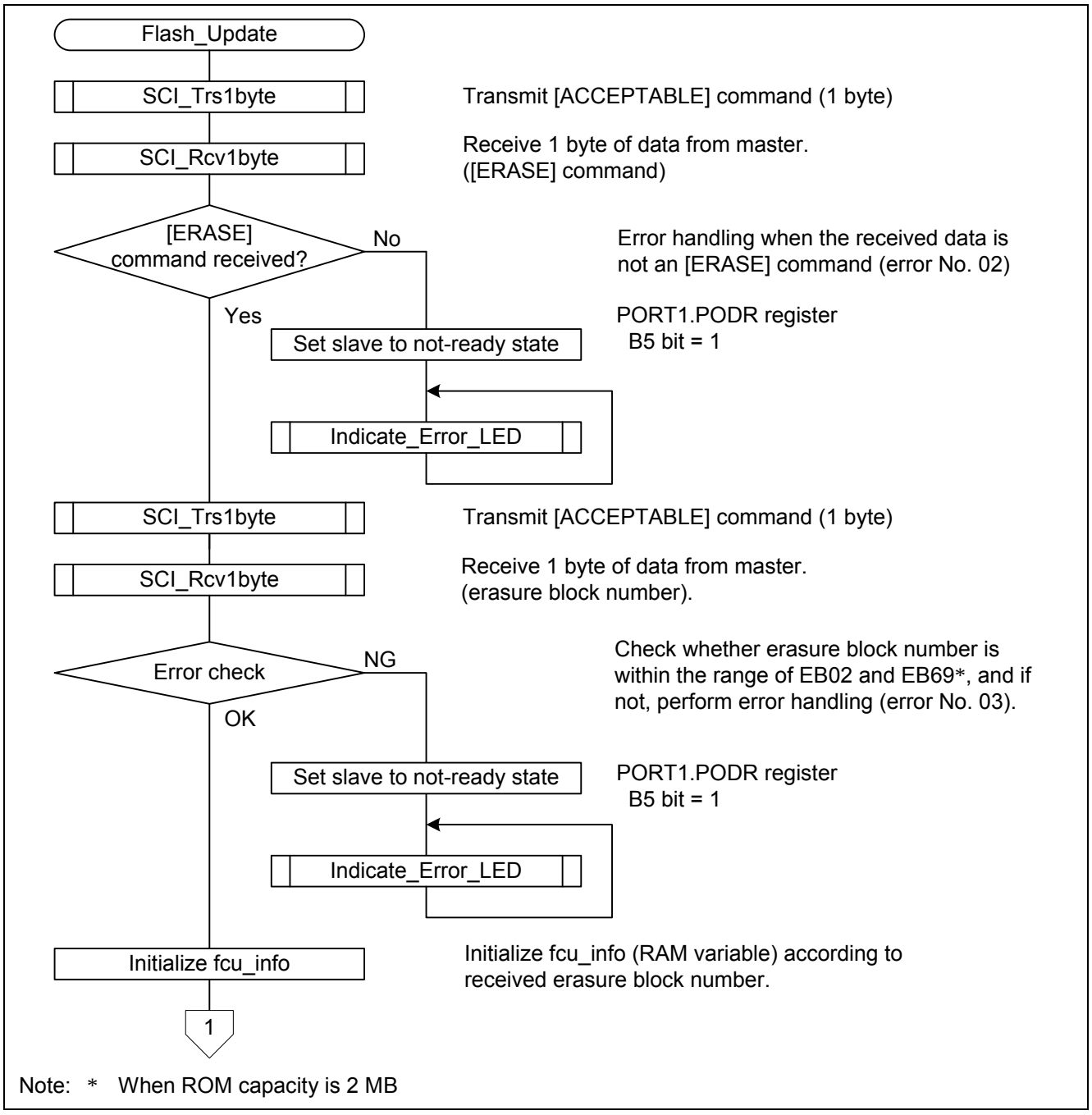


Figure 5.14 Programming/Erasing (1)

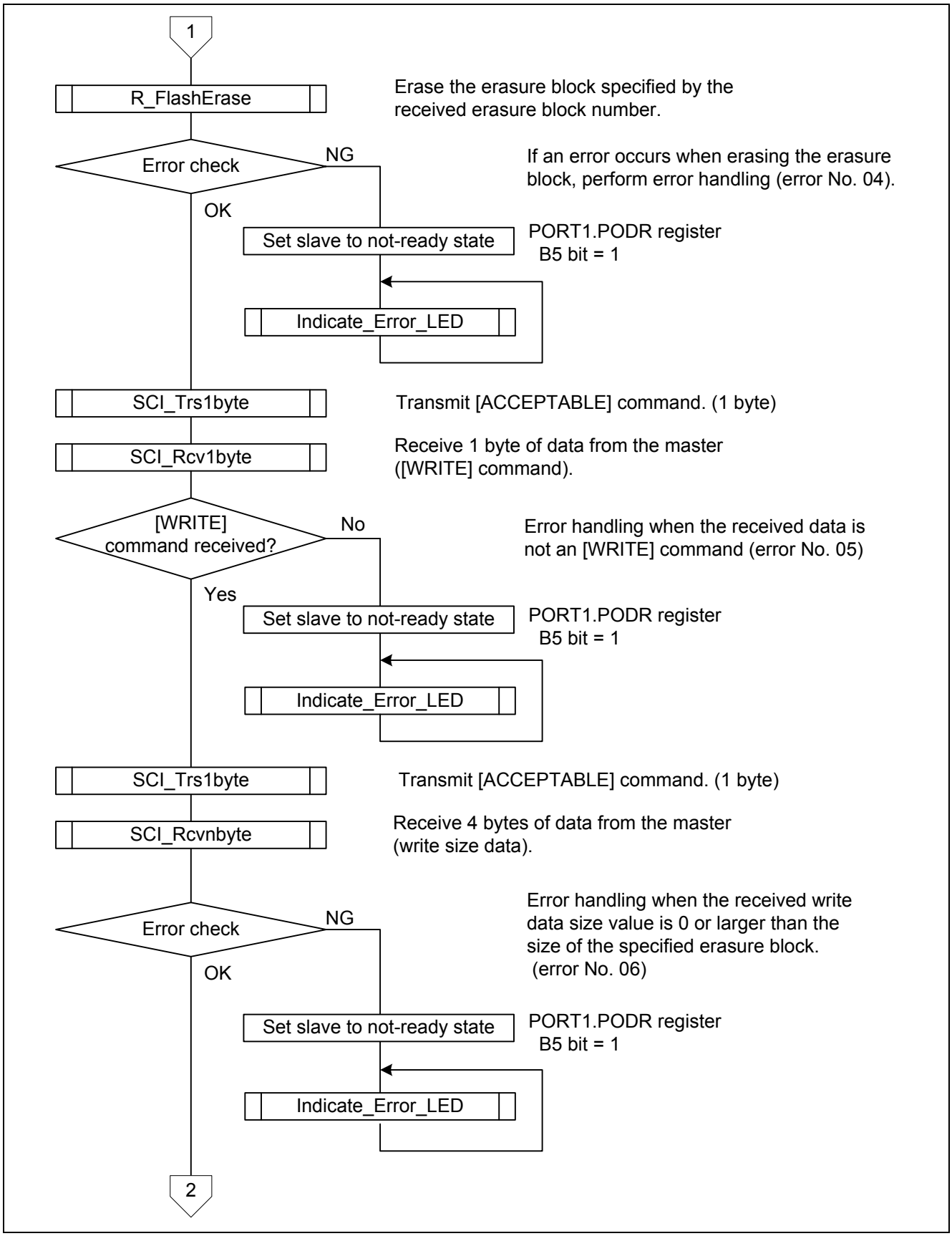


Figure 5.15 Programming/Erasing (2)

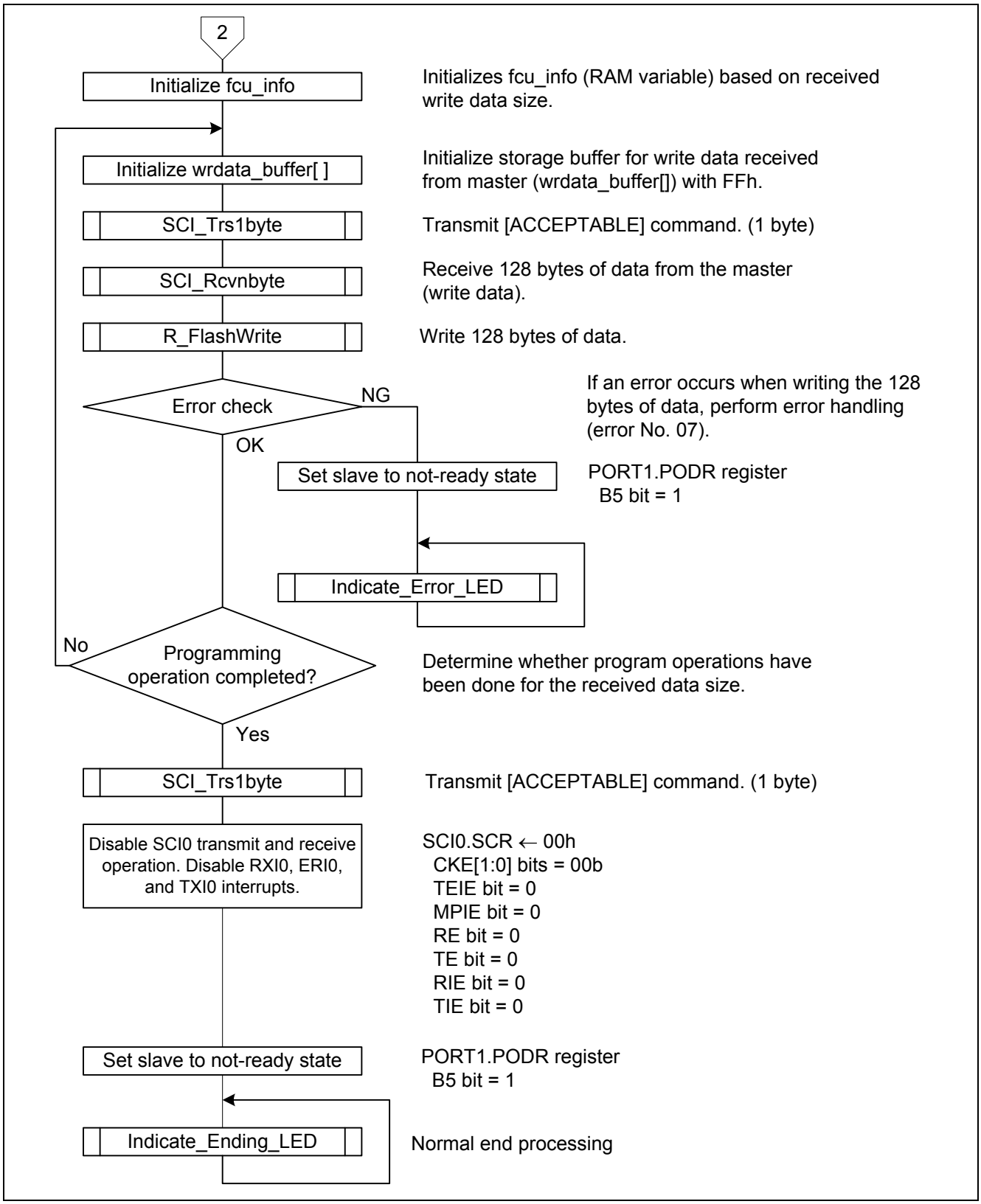


Figure 5.16 Programming/Erasing (3)

5.14.5 Normal Processing Function

Figure 5.17 is a flowchart of the normal processing function.

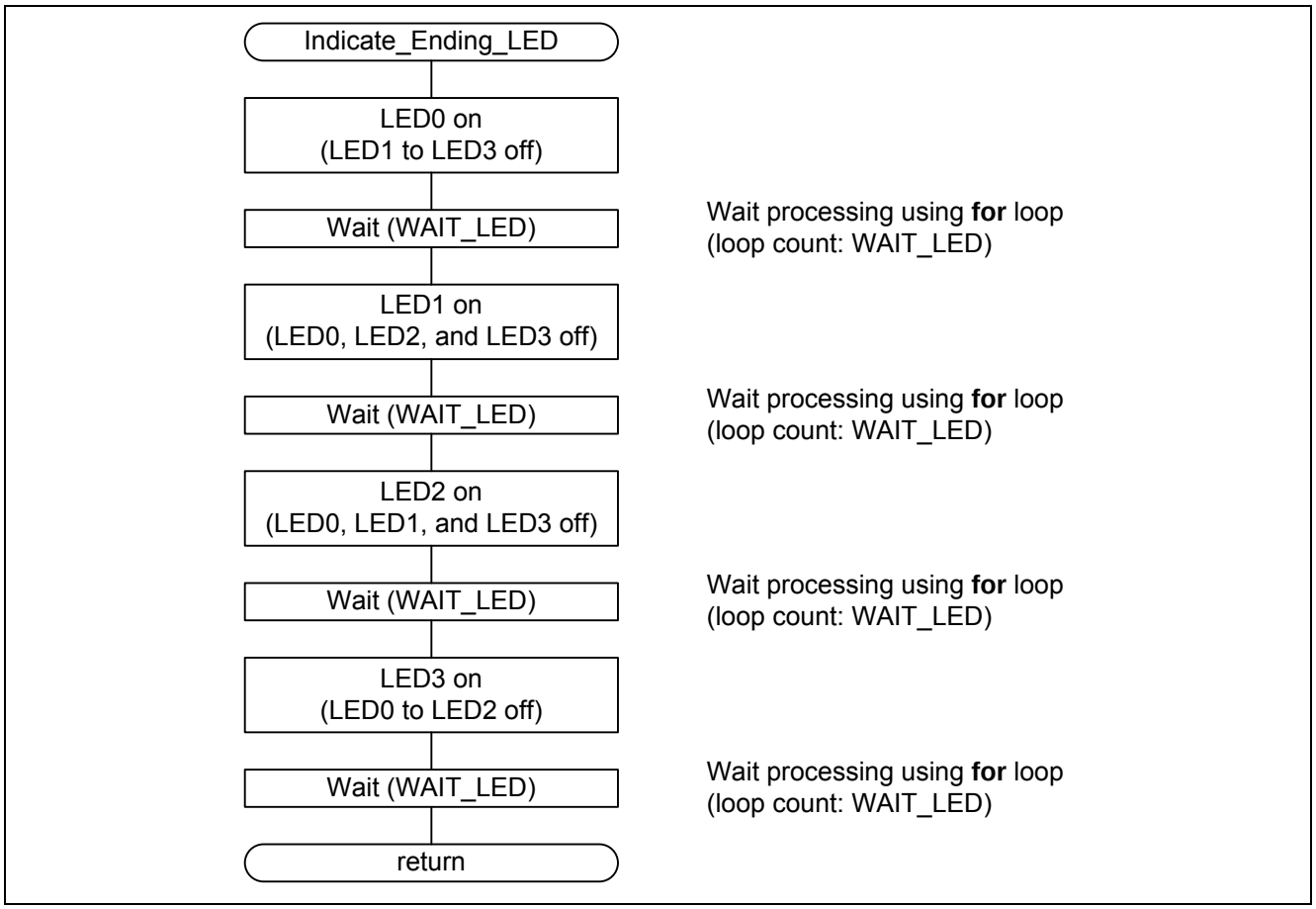


Figure 5.17 Normal Processing Function

5.14.6 Error End Processing Function

Figure 5.18 is a flowchart of the error end processing function.

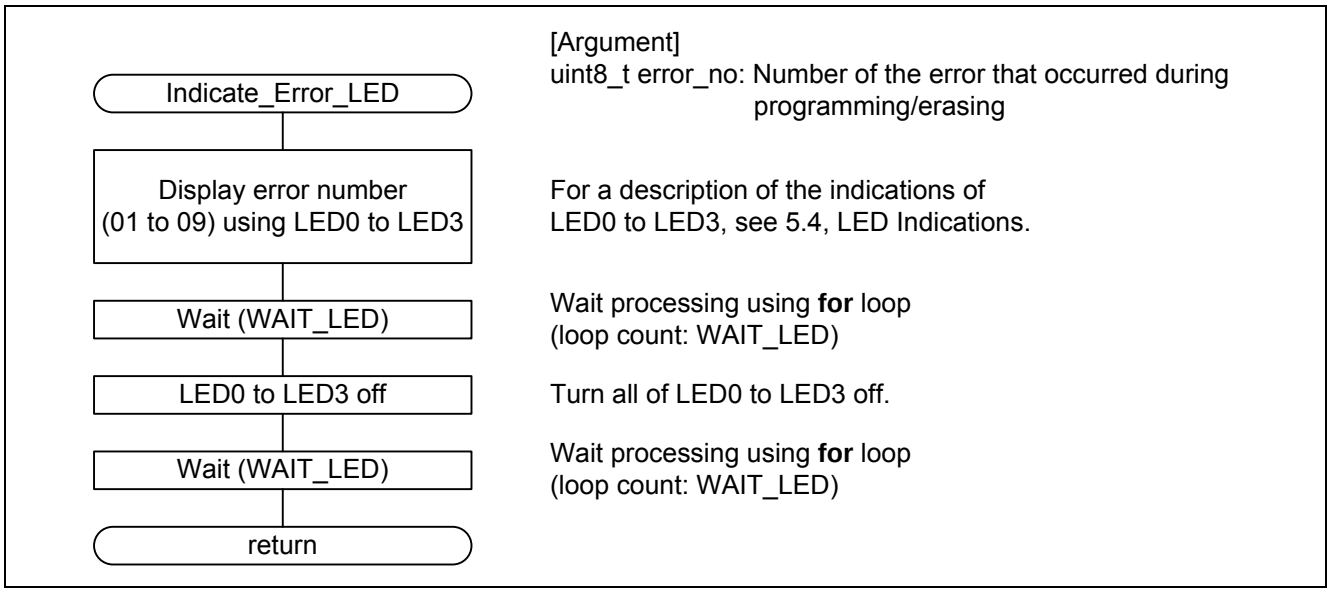


Figure 5.18 Error End Processing Function

5.14.7 1 Byte Data Receive Function

Figure 5.19 is a flowchart of the 1 byte data receive function.

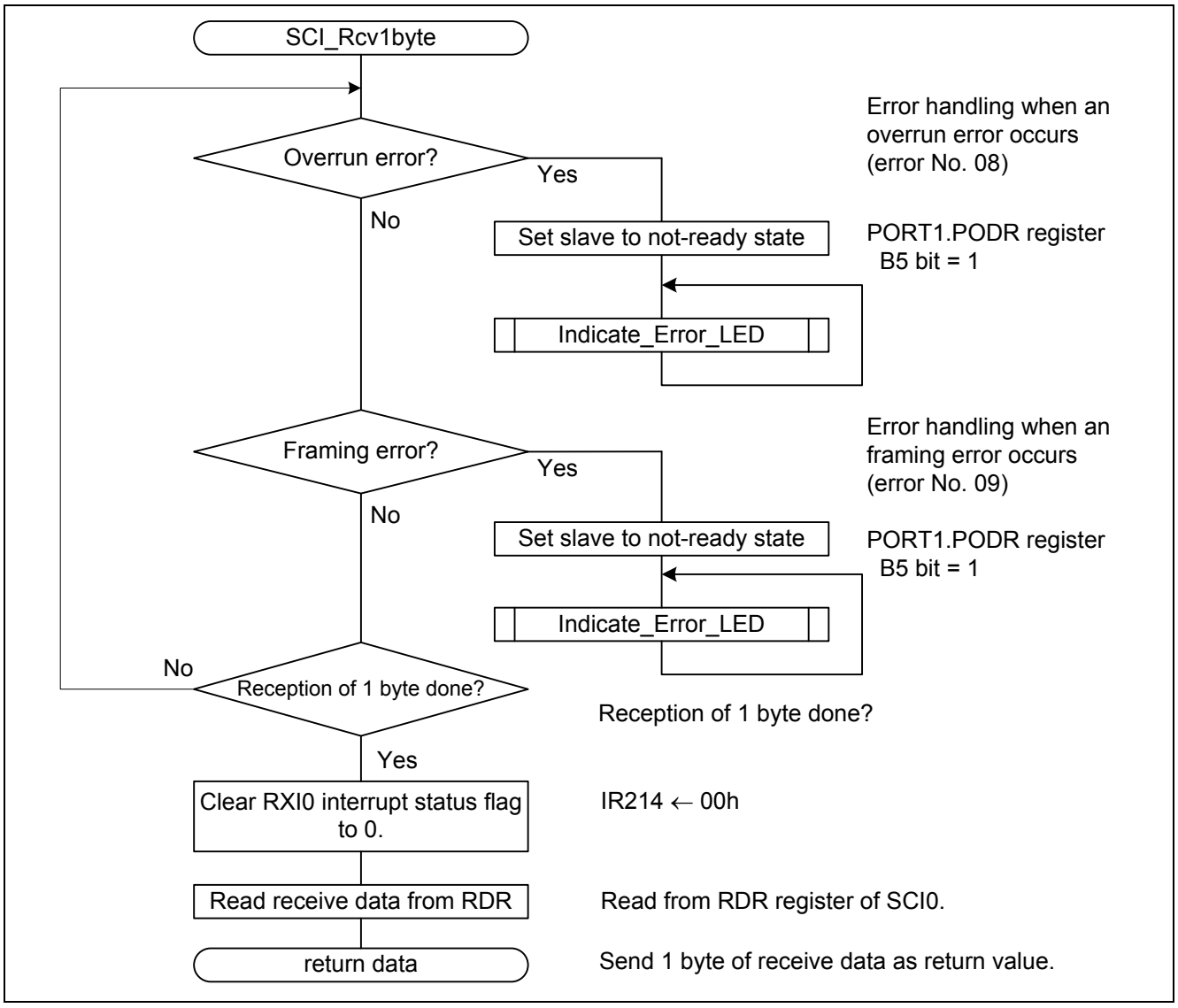


Figure 5.19 1 Byte Data Receive Function

5.14.8 *n* Byte Data Receive Function

Figure 5.20 is a flowchart of the *n* byte data receive function.

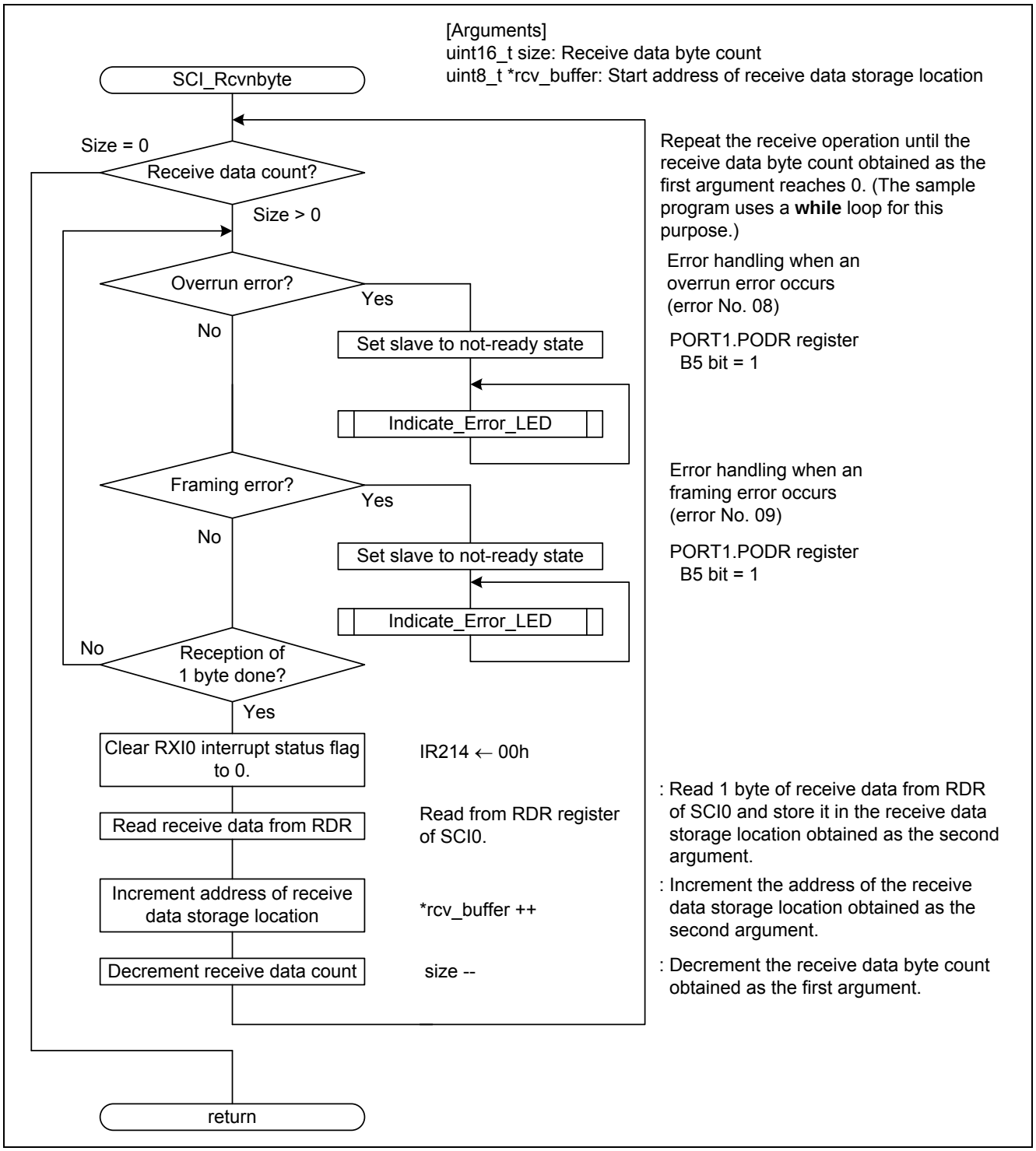


Figure 5.20 *n* Byte Data Receive Function

5.14.9 1 Byte Data Transmit Function

Figure 5.21 is a flowchart of the 1 byte data transmit function.

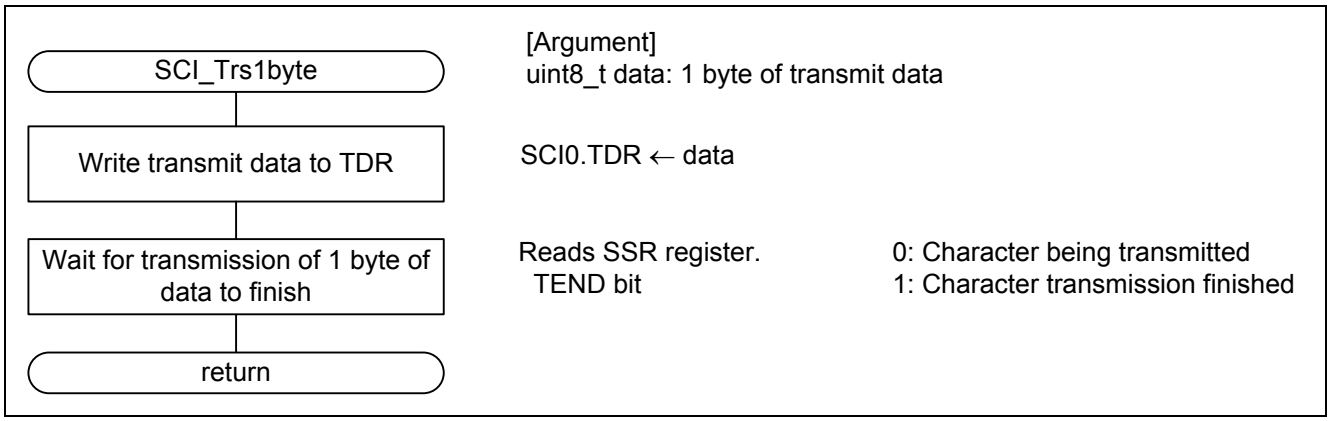


Figure 5.21 1 Byte Data Transmit Function

5.14.10 MPC Initial Settings

Figure 5.22 is a flowchart of the MPC initial settings.

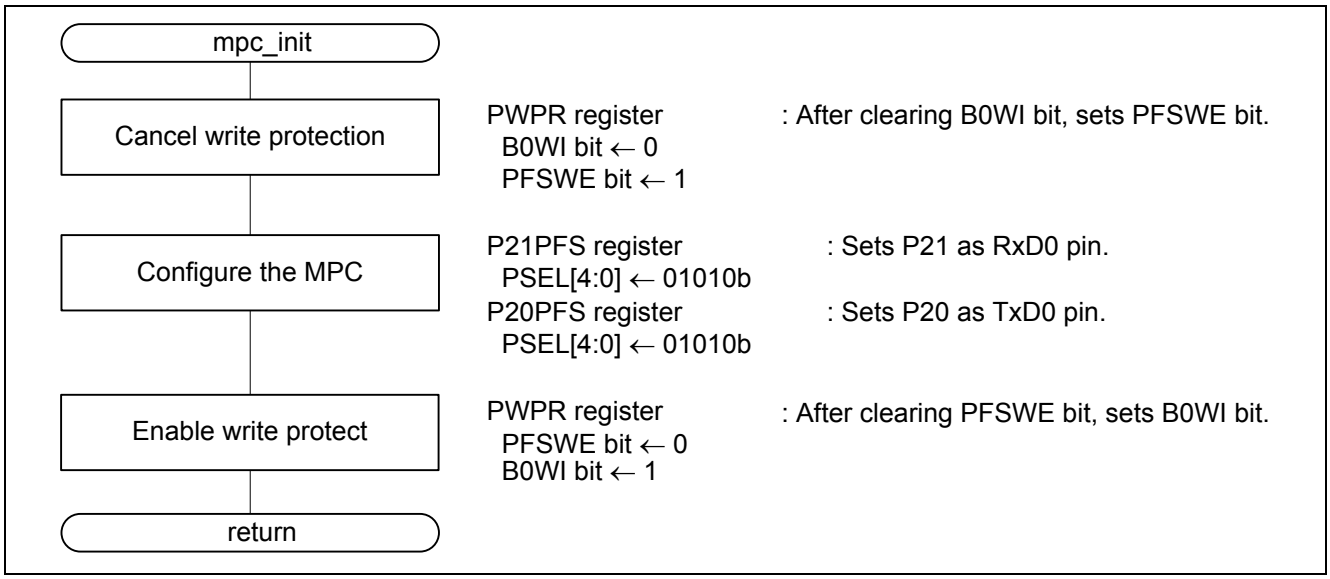


Figure 5.22 MPC Initial Settings

5.14.11 PMR Initial Settings

Figure 5.23 is a flowchart of the PMR initial settings.

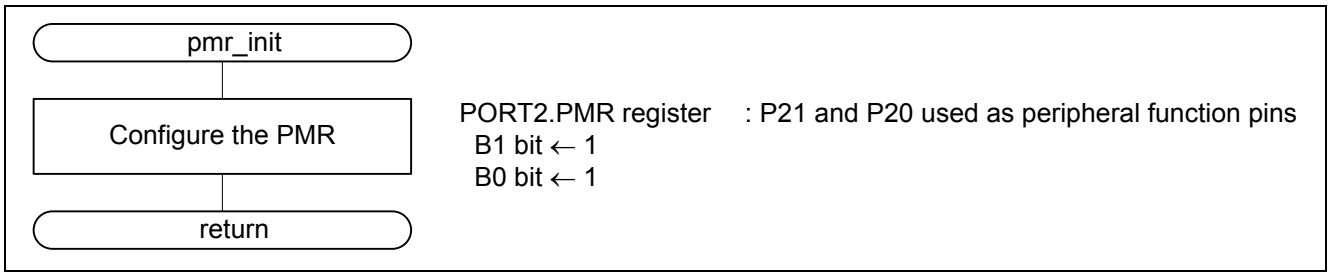


Figure 5.23 PMR Initial Settings

6. Usage Notes

6.1 Note on Reprogramming Erasure Block EB00

The erasure block EB00 (programming/erase address range: 00FF F000h to 00FF FFFF, read address range: FFFF F000h to FFFF FFFFh) contains areas allocated for fixed vectors (FFFF FF80h to FFFF FFFFh), ID code protection (FFFF FFA0h to FFFF FFAFh), etc.

When EB00 is programmed/erased, the fixed vector and ID code protection data mentioned above is erased. It is therefore necessary to configure the fixed vector and ID code protection settings again after erasing EB00.

ID code protection is a function that disables read, 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 User's Manual listed in 8, Reference Documents.

6.2 Changing the ROM Capacity

The sample code uses an MCU ROM capacity of 2 MB. To use the sample code with a ROM capacity of 1.5 MB, 1 MB, 768 KB, 512 KB, or 384 KB, change the "ROM_SIZE_BYTES" in the file "mcu_info.h" to the appropriate value. Table 6.1 lists the ROM capacities.

Example: ROM capacity of 1.5 megabytes

#define ROM_SIZE_BYTES (1572864)

Table 6.1 ROM Capacity List

Product No.	ROM Capacity	Definition Value	Usable Erasure block Numbers
R5F5630E	2 M	(2097152)	EB02 to EB69
R5F5630D	1.5 M	(1572864)	EB02 to EB61
R5F5630B	1 M	(1048576)	EB02 to EB53
R5F5630A	768 K	(786432)	EB02 to EB45
R5F56308	512 K	(524288)	EB02 to EB37
R5F56307	384 K	(393216)	EB02 to EB29

6.3 Operation Mode Settings

The sample code uses the settings shown in the Table 6.2.

Table 6.2 lists the slave operating mode settings used in the sample code.

Table 6.2 Operating Mode Settings

Mode Setting Pin	SYSCR0 Register		
MD	ROME	Operating Mode	On-Chip ROM
High	1	Single-chip mode	Enabled

Note: The initial setting of the ROME bit in the SYSCR0 register is SYSCR0.ROME = 1, so it is not necessary for the sample code to specify the SYSCR0 register.

6.4 Endianness

The sample code of this application note can be used with either big-endian or little-endian mode. Make sure to use the same endianness settings in the sample code for the master and slave devices.

6.4.1 Using the Little-Endian Setting

The settings for little-endian are as follows:

In the compiler options, select “Little-endian data” for the endianness setting. Use the little-endian value for MDES shown in 5.8, Option Settings Memory.

6.4.2 Using the Big-Endian Setting

The settings for big-endian are as follows:

In the compiler options, select “Big-endian data” for the endianness setting. Use the big-endian value for MDES shown in 5.8, Option Settings Memory.

6.5 rom Option

This application note uses the rom option of the Optimizing Linkage Editor to relocate defined symbols in the PFRAM (ROM section) to addresses in the RPFRAM (RAM section). As a result, the execution address is relocated to the RAM after the flash update function is called.

For details of the rom option, see User’s Manual: RX Family C/C++ Compiler Package, listed in section 8, Reference Documents.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware

RX630 Group User's Manual: Hardware Rev.1.50
(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

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

User's Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (including V. 1.02 supplementary materials)
(The latest version can be downloaded from the Renesas Electronics website.)

Application Note

RX630 Group
On-chip Flash Memory Reprogramming in Single-chip Mode via an UART Interface (Master) Rev.1.00
(The latest version can be downloaded from the Renesas Electronics website.)

RX600 Series Simple Flash API for RX600 Revision 2.20
(The latest version can be downloaded from the Renesas Electronics website.)

RX630 Group Initial Setting Revision 1.00
(The latest version can be downloaded from the Renesas Electronics website.)

Website and Support

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

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

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Sep. 26, 2013	—	First edition issued
1.01	Dec. 04, 2014	4	Optimizing Linkage Editor added to table 2.1
		4	Note (*) added to table 2.1
		29	Note (*) added to figure 5.13
		40	6.5 rom Option added

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 type 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. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

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

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

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

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

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

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

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

Colophon 4.0