

RX210 Group

R01AN1484EJ0100

Rev. 1.00

July 1, 2013

Reprogramming the On-Chip Flash Memory via UART in Single-Chip Mode

Abstract

This application note describes programming/erasing the flash memory (user area) for code storage using the erase block number, programming data size, and programming data which are transmitted by the master device (master) through asynchronous serial communication.

This application note uses sample codes in the following application notes:

- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ0200)
- RX200 Series Simple Flash API for RX200 Rev. 1.01 (R01AN0823EU0101)

Products

- RX210 Group 100-pin package with a ROM size between 128 KB and 512 KB
- RX210 Group 80-pin package with a ROM size between 128 KB and 512 KB
- RX210 Group 64-pin package with a ROM size between 128 KB and 512 KB
- RX210 Group 48-pin package with a ROM size between 128 KB and 256 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	4
2. Operation Confirmation Conditions	6
3. Reference Application Notes	6
4. Hardware	7
4.1 Hardware Configuration	7
4.2 Pins Used	8
5. Software	9
5.1 Operation Overview	9
5.1.1 Specifications of Asynchronous Serial Communication	9
5.1.2 Specifications of Communication Commands	9
5.1.3 Communications between the Master and Slave	10
5.1.4 Erase Block Number	13
5.1.5 Programming Data Size	13
5.1.6 Overrun Error	14
5.1.7 Framing Error	14
5.2 Programming/Erasing the User Area	14
5.2.1 Erasing the User Area	14
5.2.2 Programming the User Area	14
5.2.3 Change in the Simple Flash API	14
5.2.4 Notes on Using Interrupts	15
5.3 Slave Ready Processing	15
5.4 LED Display Patterns	16
5.5 Handshaking Control	17
5.6 Configuring Sections	17
5.7 File Composition	18
5.8 Option-Setting Memory	19
5.9 Constants	20
5.10 Structure List	21
5.11 Variables	21
5.12 Functions	21
5.13 Function Specifications	22
5.14 Flowcharts	25
5.14.1 Initial Setting	25
5.14.2 MCU Initial Setting	26
5.14.3 Main Processing	28
5.14.4 Program/Erase Operations	29
5.14.5 Normal End Processing	32
5.14.6 Error Processing	33
5.14.7 1-Byte Data Reception	34
5.14.8 n-Byte Data Reception	35
5.14.9 1-Byte Data Transmission	36

6.	Usage Notes.....	37
6.1	Note on Reprogramming the Erase Block EB00	37
6.2	Changing the ROM size.....	37
6.3	Operation Mode Settings	37
6.4	Endian Settings.....	37
6.4.1	When Using Little Endian	37
6.4.2	When Using Big Endian.....	37
7.	Sample Code.....	38
8.	Reference Documents.....	38

1. Specifications

This application note describes programming/erasing the user area in single-chip mode.

To program/erase the user area, the slave device (slave) receives the erase block number, programming data size, and programming data from the master through asynchronous serial communication.

Asynchronous serial communication between the master and slave is performed using channel 0 in the SCI module (SCI0).

Settings for asynchronous serial communication are as follows:

- Bit rate: 31250 bps
- Data length: 8 bits
- Parity bit: None
- Stop bit: 1 bit

1. When the slave is ready for communication, the output level of the P13 pin is switched to low to notify the master of the slave status. Connect a pin to monitor the slave status in the master side.
2. Handshaking between slave and master is used for communication control. The slave performs processing required for data received from the master, then it transmits the ACCEPTABLE command (55h) to the master. When the master receives the ACCEPTABLE command, it starts the next serial transmission.
3. The slave erases the specified block and writes the received data from the start address of the erased block.

In this application note, programs are located in blocks EB00 to EB03, thus erase/program operations cannot be performed to these blocks. If the erase block number corresponding to blocks EB00 to EB03 is specified, an error occurs.

4. When completed erasing/programming the user area successfully, the slave notifies of the normal end with four LEDs connected to the I/O ports, and switches the output level of the P13 pin back to high (slave is not ready for communication).

If an error occurs while communicating with the master or performing erase/program operations, the slave switches the output level of the P13 pin back to high and notifies of the error status with LEDs.

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

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
ROM (flash memory for code storage)	Reprogramming on-chip flash memory in ROM P/E mode
Serial communication interface	Asynchronous serial communication for communicating with the master

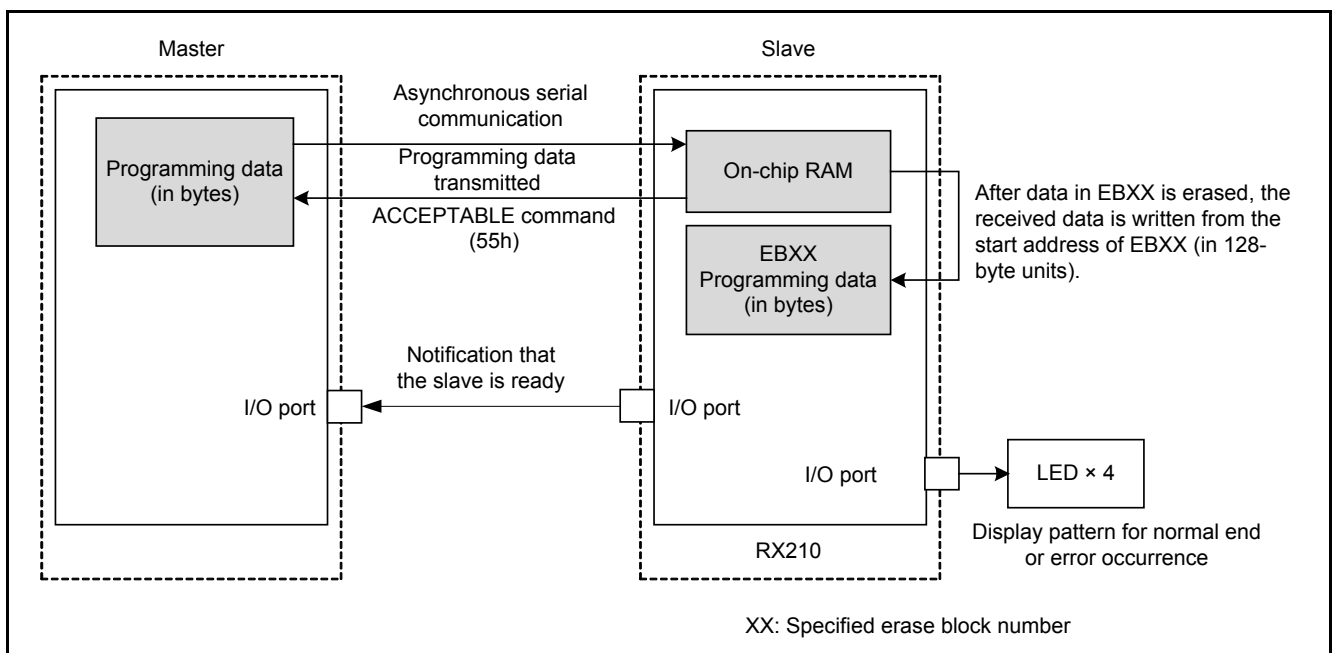


Figure 1.1 Usage Example

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	R5F52108ADFP (RX210 Group)
Operating frequencies	- Main clock: 20 MHz - PLL: 100 MHz (main clock divided by 2 and multiplied by 10) - System clock (ICLK): 50 MHz (PLL divided by 2) - Peripheral module clock B (PCLKB): 25 MHz (PLL divided by 4) - FlashIF clock (FCLK): 25 MHz (PLL divided by 4)
Operating voltage	5.0 V
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01 Compile options -cpu=rx200 -output=obj="\$ (CONFIGDIR)\\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.)
iodefine.h version	Version 1.2A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Device used	Renesas Starter Kit for RX210 (product part no.: R0K505210C000BE)

3. Reference Application Notes

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

- RX200 Series Simple Flash API for RX200 Rev. 1.01 (R01AN0823EU0101_RX200)
- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ0200_RX210)

The functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are the ones when these application notes were made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows the Hardware Configuration of the Slave.

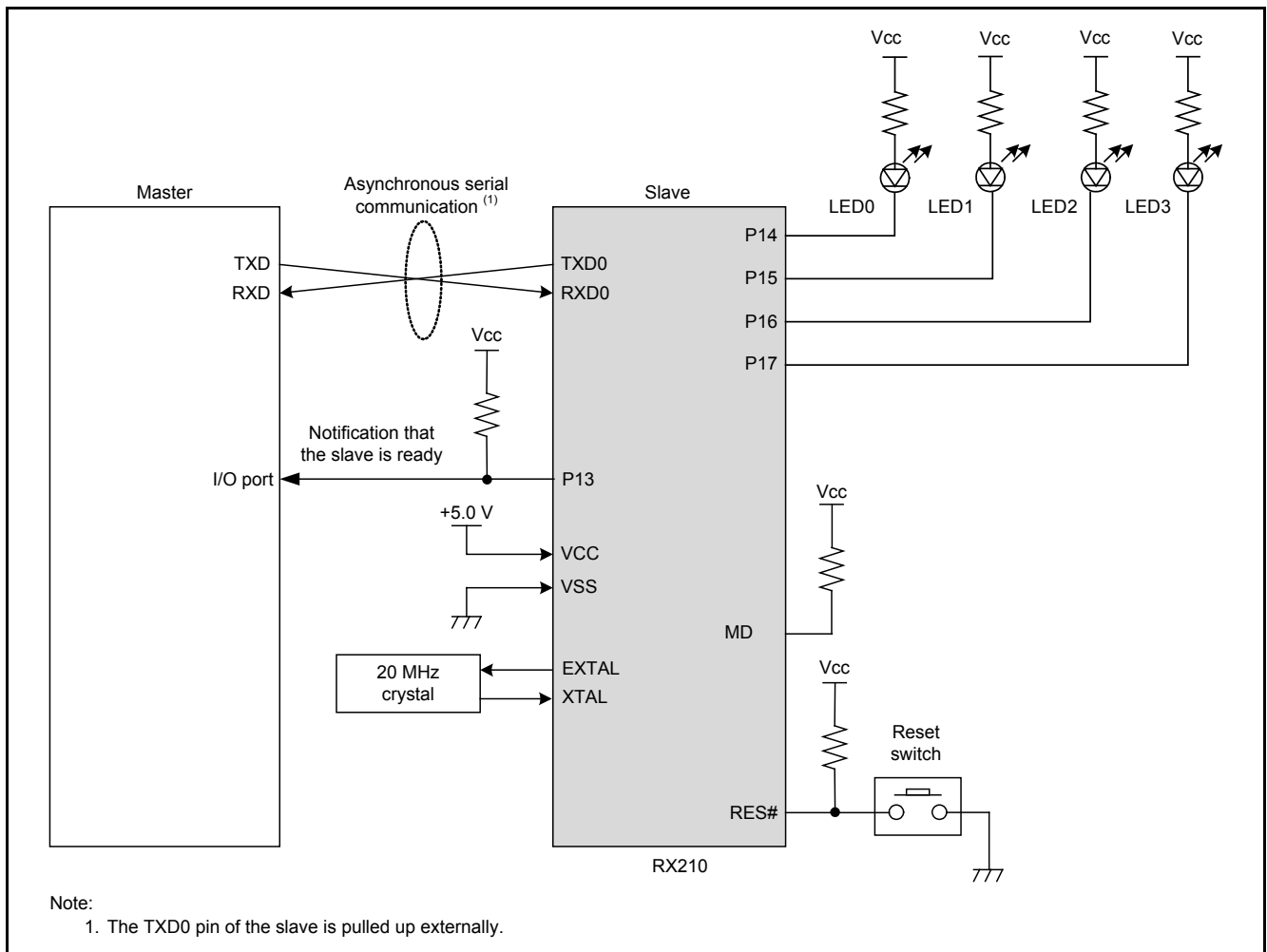


Figure 4.1 Hardware Configuration of the Slave

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

The pins described here are for 100-pin products. When the product with less than 100 pins is used, select appropriate pins for the product used.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P20/TXD0	Output	Serial transmission pin for communication with the master
P21/RXD0	Input	Serial reception pin for communication with the master
P13	Output	Pin used to notify that the slave is ready
P14	Output	Pin connected to LED0 (high output: turned off, low output: turned on)
P15	Output	Pin connected to LED1 (high output: turned off, low output: turned on)
P16	Output	Pin connected to LED2 (high output: turned off, low output: turned on)
P17	Output	Pin connected to LED3 (high output: turned off, low output: turned on)

5. Software

5.1 Operation Overview

5.1.1 Specifications of Asynchronous Serial Communication

In this application note, the slave receives communication commands (FSTART, ERASE, and WRITE), the erase block number, programming data size, and programming data using asynchronous serial communication between the master and slave. The slave transmits the ACCEPTABLE command (55h) as the status command for handshaking.

Table 5.1 lists the Specifications of Asynchronous Serial Communication.

Table 5.1 Specifications of Asynchronous Serial Communication

Item	Specification
Channel	SCI channel 0 (SCI0)
Serial communication mode	Asynchronous operation
Transfer rate	31250 bps (PCLKB is 25 MHz)
Data length	8 bits
Parity bit	None
Stop bit	1 bit
Errors	Overrun and framing errors

5.1.2 Specifications of Communication Commands

Table 5.2 Specifications of Communication Commands

Command	Value	Description	Communication Direction
FSTART	10h	Command to start programming/erasing the user area in the slave.	Master → slave
ERASE	11h	Command to start erasing the user area in the slave.	Master → slave
WRITE	12h	Command to start programming the user area in the slave.	Master → slave
ACCEPTABLE	55h	Status command to notify the master that the slave is ready to receive data.	Slave → master

5.1.3 Communications between the Master and Slave

Figure 5.1 to Figure 5.3 show the Communications between the Master and Slave.

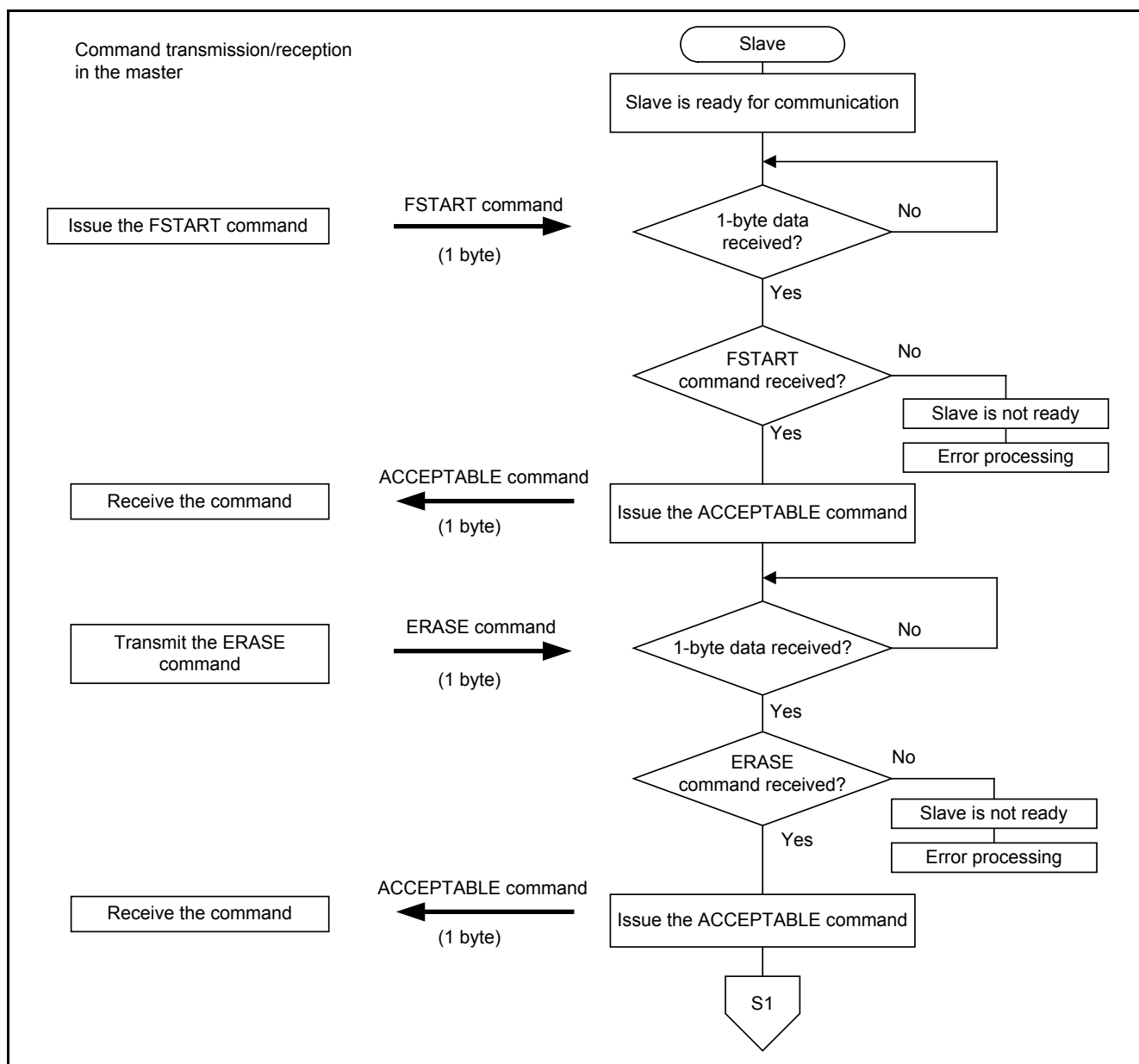


Figure 5.1 Communications between the Master and Slave (1/3)

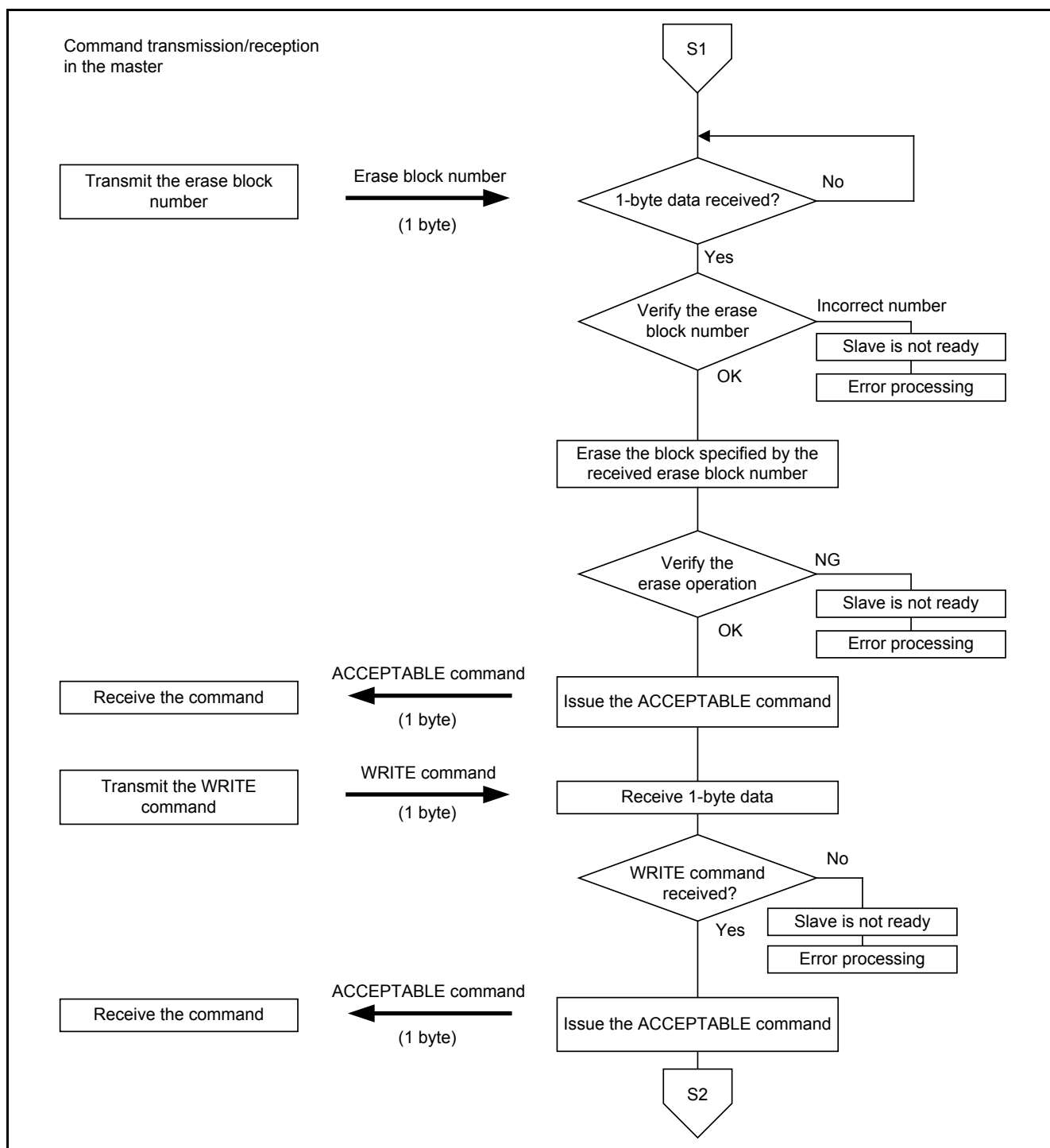


Figure 5.2 Communications between the Master and Slave (2/3)

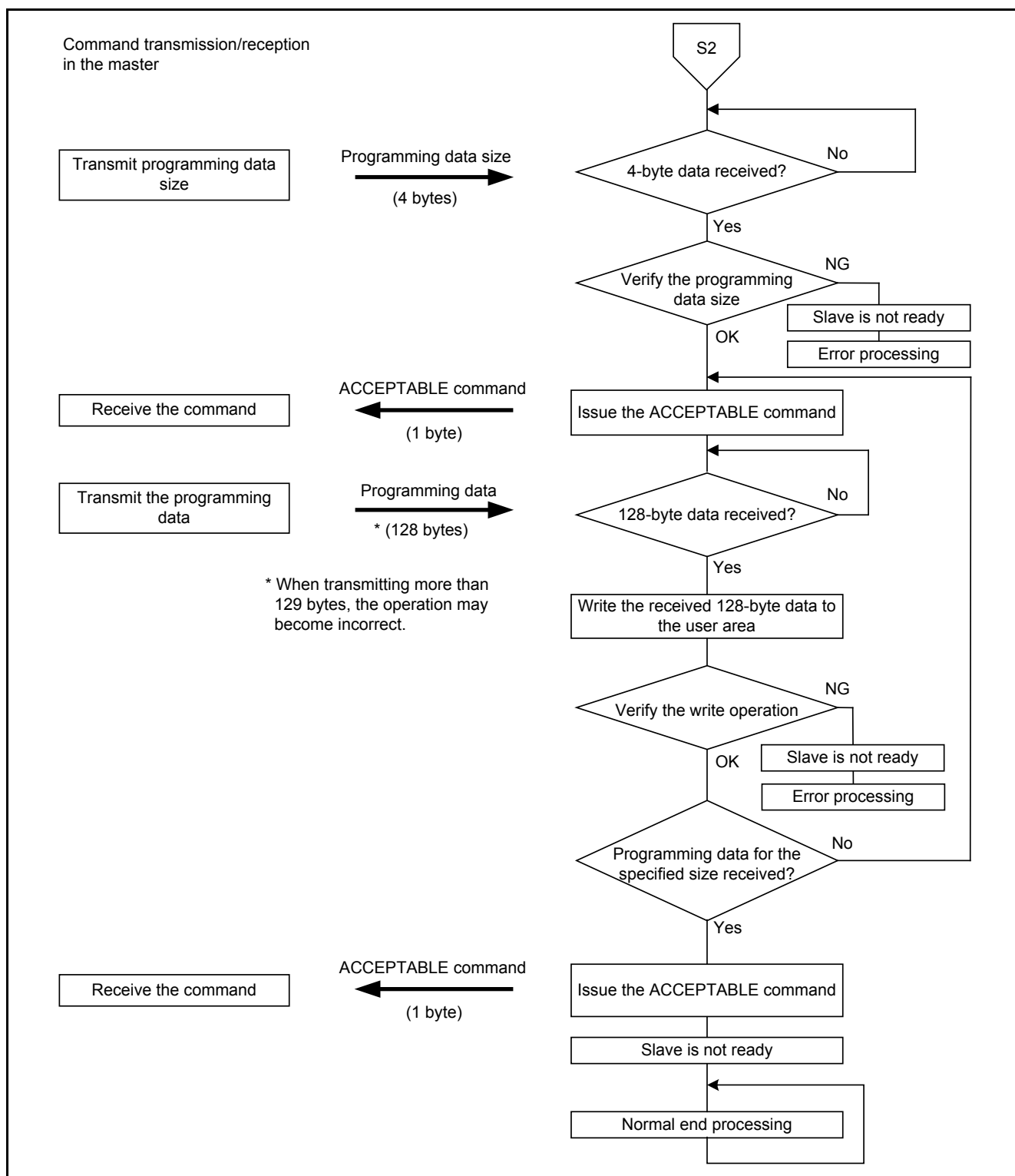


Figure 5.3 Communications between the Master and Slave (3/3)

5.1.4 Erase Block Number

The slave receives the erase block number (1-byte defined data) after receiving the ERASE command from the master.

Figure 5.4 shows the Specification of the Erase Block Number. Also refer to 5.2.1 Erasing the User Area for the erase block number.

Erase block number (uint8_t type)

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

When programming/erasing block EB08 in the slave, the erase block number becomes 08h.

Note:

1. Specify the erase block number with a value corresponding to blocks between EB04 and EB255. Otherwise the slave determines an error has occurred and performs error processing.

Figure 5.4 Specification of the Erase Block Number

5.1.5 Programming Data Size

After receiving the WRITE command from the master, the slave receives 4-byte data which signifies the programming data size.

Figure 5.5 shows the Specification of the Programming Data Size.

Programming 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 size of programming is 1 KB, the programming data size becomes 0000 0400h.

Notes:

1. Specify the programming data size with a value greater than 0, and smaller than or equal to the size specified with the erase block number. If the specified programming data size is 0 or greater than the size specified with the erase block number, the slave determines an error has occurred and performs error processing.
2. The reception size of programming data is fixed to 128 bytes. Thus the programming data is received in 128 bytes. When the programming data size is not a multiple of 128 bytes, FFh is added to the last data which has shortage of data to make the size 128 bytes. Then the data is written to the user area in the slave.

Figure 5.5 Specification of the Programming Data Size

5.1.6 Overrun Error

When an overrun error occurs (the SCI0.SSR.Over bit is set to 1) while a receive operation is being performed in the slave with asynchronous serial communication, error processing will be performed.

5.1.7 Framing Error

When a framing error occurs (the SCI0.SSR.FER bit is set to 1) while a receive operation is being performed in the slave with asynchronous serial communication, error processing will be performed.

5.2 Programming/Erasing the User Area

Refer to the RX200 Series Simple Flash API for RX200 for details on the simple Flash API for RX200 used for programming/erasing in this application note.

5.2.1 Erasing the User Area

The R_FlashErase function accompanying the RX200 Series Simple Flash API for RX200 is used to erase the user area. Therefore the erase block number and the value set in the R_FlashErase function are common. The errors are indicated by the return value from the R_FlashErase function.

5.2.2 Programming the User Area

The R_FlashWrite function accompanying the RX200 Series Simple Flash API for RX200 is used to program the user area. The errors are indicated by the return value from the R_FlashWrite function.

5.2.3 Change in the Simple Flash API

The settings in r_Flash_API_RX200_UserConfig.h have been changed in the simple flash API.

Table 5.3 Changes in r_Flash_API_RX200_UserConfig.h

Changed Item	Description
Settings in the simple flash API	#define ROM_SIZE 524288
	//#define ROM_SIZE 131072
	#define FLASH_CLOCK_FREQUENCY 25
	//#define DATA_FLASH_BGO

5.2.4 Notes on Using Interrupts

The ROM cannot be accessed during programming and erasing the ROM (flash memory). Therefore accessing the ROM by interrupts generated during programming or erasing must be avoided. Processing for avoiding interrupts is not performed in this application note as interrupts are not used. To avoid interrupts, add the definition ‘#define FOR_INTERRUPTION’ to the sample code to enable codes within red boxes in Figure 5.6 (these codes are currently disabled in the sample code without having the definition ‘#define FOR_INTERRUPTION’).

Here describes processing (1) to (3) in Figure 5.6.

- (1) Store the value of the current processor interrupt priority level (IPL) in the flash_pipl variable.
- (2) Change the IPL to FLASH_READY_IPL before starting program/erase operations. In the sample code, the FLASH_READY_IPL value is defined as 5. Thus interrupts with the IPL less than or equal to 5 are disabled. Refer to the RX200 Series Simple Flash API for RX200 for details of FLASH_READY_IPL.
- (3) Set the IPL back to the original value stored.

With the above processing, interrupts which access the ROM can be avoided by temporarily changing the IPL.

```

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

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

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

/* Erasure process using "simple API" */
fcu_status = R_FlashErase((uint16_t)target_eb);

/* Programming process using "simple API" */
fcu_status = R_FlashWrite((uint32_t)fcu_info.p_write_adrs_now,
(uint32_t)wdata_buffer, ROM_PROGRAM_SIZE); /* Write 128-byte data to the target erase block using "simpleAPI"
* (call of R_FlashWrite function in "simpleAPI") */

(3) → #ifdef FOR_INTERRUPTION
/* Restore processor IPL */
set_ipl(flash_pipl); /* If your system uses an interrupt, enable this line. */
#endif

```

Figure 5.6 Avoiding Interrupts


5.3 Slave Ready Processing

After the slave becomes ready for communication, a low level signal is output from the P13 pin which is used to notify the slave status. Connect a pin to monitor the slave status in the master side.

Outputting a high level signal from the P13 pin indicates that the slave is not ready, and then the user area cannot be programmed/erased.

5.4 LED Display Patterns

Figure 5.7 shows the LED display patterns according to the operation statuses in the sample code.

Error No.	Operation Status	LED Display				
		LED3	LED2	LED1	LED0	Sequence
	Normal end (shift display at regular intervals)	●	●	●	○	
		●	●	○	●	
		●	○	●	●	
		○	●	●	●	
Error No. 01	FSTART command error has occurred	●	●	●	●	
Error No. 02	ERASE command error has occurred	●	●	●	●	
Error No. 03	Erase block number error has occurred	●	●	●	●	
Error No. 04	Erase operation error has occurred	●	●	●	●	
Error No. 05	WRITE command error has occurred	●	●	●	●	
Error No. 06	Programming data size error has occurred	●	●	●	●	
Error No. 07	Program operation error has occurred	●	●	●	●	
Error No. 08	Overrun error has occurred	●	●	●	●	
Error No. 09	Framing error has occurred	●	●	●	●	

○ : On ● : Off ● : Blink

Figure 5.7 LED Display Patterns

5.5 Handshaking Control

Handshaking between slave and master is used for communication control.

For handshaking control, after the slave receives serial communication from the master, the slave processes the received data and returns the ACCEPTABL command (55h) to the master when it is ready to receive the next serial communication. The master starts the next serial transmission when it receives the ACCEPTABLE command from the slave.

5.6 Configuring Sections

Table 5.4 lists the Slave Section Configuration.

Table 5.4 Slave Section Configuration

Section Name	Start Address	Description
RPFRAM	0000 0000h	Area mapped the PFRAM section to the RAM using the ROM-support function.
B_1	0000 1000h	Uninitialized data area (ALIGN = 1)
R_1		Area mapped the D_1 section to the RAM using the ROM-support function.
B_2		Uninitialized data area (ALIGN = 2)
R_2		Area mapped the D_2 section to the RAM using the ROM-support function.
B		Uninitialized data area (ALIGN = 4)
R		Area mapped the D section to the RAM using the ROM-support function.
SU		User stack area
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 for 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 F800h	Program area (programming the user area/control program)
FIXEDVECT	FFFF FFD0h	Fixed vector area

5.7 File Composition

Table 5.5 lists the Files Used in the Sample Code. The table does not include files that are generated by the integrated development environment and have no change.

Table 5.5 Files Used in the Sample Code

File Name	Outline	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	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	
resetprg.c	Initial setting	Remove comments for disabling the call function for the HardwareSetup function in the PowerON_Reset_PC function, so that the HardwareSetup function in main.c can be called from the PowerON_Reset_PC function.
dbstc.c	Initialization of RPFRAM area	Add the following: { __sectop("PFRAM"), __sectend("PFRAM"), __sectop("RPFRAM") }
main.c	<ul style="list-style-type: none"> - Main processing - Transmission/reception control of communication command for asynchronous serial communication with the master - Receive operation control for the erase block number, programming data size, and programming data - LED display control for normal end and error occurrence 	
r_Flash_API_RX200.c	RX200 Series simple flash API program for RX200	Refer to the RX200 Series Simple Flash API for RX200 application note.
r_Flash_API_RX200.h	Include header of RX200 Series simple flash API program for RX200 for external reference	
r_Flash_API_RX200_User Config.h	Include header of RX200 Series simple flash API program for RX200 for configuring parameters	

5.8 Option-Setting Memory

Table 5.6 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system. Endian settings must be same in the master side and slave side.

Table 5.6 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES ⁽¹⁾	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

Note:

1. The little endian is used in the sample code. Refer to 6.4 Endian Settings for details on switching the endian.

5.9 Constants

Table 5.7 lists the Constants Used in the Sample Code.

Table 5.7 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
FSTART	0x10	Command to start programming/erasing
ERASE	0x11	Command to start erasing
WRITE	0x12	Command to start programming
ACCEPTABLE	0x55	Status command transmitted by the slave
LED_ON	0	Setting value when an LED is on.
LED_OFF	1	Setting value when an LED is off.
RSK_LED0	PORT1.PODR.BIT.B4	On/off control for LED0 on the evaluation board
RSK_LED1	PORT1.PODR.BIT.B5	On/off control for LED1 on the evaluation board
RSK_LED2	PORT1.PODR.BIT.B6	On/off control for LED2 on the evaluation board
RSK_LED3	PORT1.PODR.BIT.B7	On/off control for LED3 on the evaluation board
RSK_LED0_PDR	PORT1.PDR.BIT.B4	I/O control for LED0 on the evaluation board
RSK_LED1_PDR	PORT1.PDR.BIT.B5	I/O control for LED1 on the evaluation board
RSK_LED2_PDR	PORT1.PDR.BIT.B6	I/O control for LED2 on the evaluation board
RSK_LED3_PDR	PORT1.PDR.BIT.B7	I/O control for LED3 on the evaluation board
NOT_READY	1	Output level which indicates the slave is not ready
READY	0	Output level which indicates the slave is ready
SLAVE_READY_PODR	PORT1.PODR.BIT.B3	I/O control for the port to verify the slave status
SLAVE_READY_PDR	PORT1.PDR.BIT.B3	I/O control for the port to verify the slave status
WAIT_LED	2000000	Time data that is the interval for an LED to be turned on when the slave has completed programming/erasing successfully.
ROM_PROGRAM_SIZE_LARGE	128 ⁽¹⁾	Programming size for one operation set according to the device used. This setting is in r_Flash_API_RX200.h and can be referred with include header.
RXD0_PMR	PORT2.PMR.BIT.B1	Setting for the RXD0 port mode register
TXD0_PMR	PORT2.PMR.BIT.B0	Setting for the TXD0 port mode register
RXD0_PDR	PORT2.PDR.BIT.B1	Setting for the RXD0 port direction register
TXD0_PODR	PORT2.PODR.BIT.B0	Setting for the TXD0 port output data register
TXD0_PDR	PORT2.PDR.BIT.B0	Setting for the TXD0 port direction register
ERROR_NO_01	1	Data to indicate the error statuses
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	
WRITE_ADRS_TOP_2K	ROM_PE_ADDR	Start address of an area, whose block size is 2 KB, for programming/erasing in the address space
BLK_SIZE_2K	2 × 1024	Block size of each block from EB00 to EB255

Note:

1. The value set here is for the RX210 Group products.

5.10 Structure List

Figure 5.8 shows the Structure Used in the Sample Code.

```
typedef struct
{
    uint8_t *p_write_adrs_top; /* Start address of the target erase block when programming */
    uint8_t *p_write_adrs_end; /* End address of the target erase block when programming */
    uint8_t *p_write_adrs_now; /* Destination address for programming */
    uint32_t eb_block_size; /* Size of the target erase block */
}st_fcu_info_t;
```

Figure 5.8 Structure Used in the Sample Code

5.11 Variables

Table 5.8 lists the Global Variables.

Table 5.8 Global Variables

Type	Variable Name	Contents	Function Used
uint8_t	wrdata_buffer [ROM_PROGRAM_SIZE_LARGE]	128-byte array to store 128-byte programming data received from the master	Flash_Update
st_fcu_info_t	fcu_info	16-byte structure to store the address information associated with the flash control unit (FCU) used when programming/erasing the user area	Flash_Update

5.12 Functions

Table 5.9 lists the Functions.

Table 5.9 Functions

Function Name	Outline
HardwareSetup	MCU initial setting
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
main	Main processing
Flash_Update	Program/erase operation
Indicate_Ending_LED	Normal end processing
Indicate_Error_LED	Error processing
SCI_Rcv1byte	1-byte data reception
SCI_Rcvnbyte	n-byte data reception
SCI_Trns1byte	1-byte data transmission

5.13 Function Specifications

The following tables list the sample code function specifications.

HardwareSetup	
Outline	MCU initial setting
Header	iodefine.h, r_init_stop_module.h, r_init_clock.h, and r_init_non_existent_port.h
Declaration	void HardwareSetup (void)
Description	Configures the MCU initial setting. <ul style="list-style-type: none"> - Stop processing for active peripheral functions after a reset - Nonexistent port initialization (100-pin version) - Clock initialization - Initial output setting for I/O ports (P14, P15, P16, and P17) that connect to LED0 to LED3 - Canceling module stop state - Configuring the multi-function pin controller (MPC) and port mode register (PMR) - SCIO initialization
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	Transition to the module-stop state is not performed in the sample code. Refer to the RX210 Group Initial Setting Rev. 2.00 application note for details on this function.
R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initializes port direction registers for ports that do not exist in products with less than 100 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 100-pin package (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. Refer to the RX210 Group Initial Setting Rev. 2.00 application note for details on this function.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initializes the clock.
Arguments	None
Return Value	None
Remarks	<p>The sample code selects processing which uses PLL as the system clock without using the sub-clock.</p> <p>Refer to the RX210 Group Initial Setting Rev. 2.00 application note for details on this function.</p>
main	
Outline	Main processing
Header	iodefine.h
Declaration	void main (void)
Description	<p>Notifies that the slave is ready by outputting low level signal from P13. Controls 1-byte data reception from the master, calls the Indicate_Error_LED function on an error occurrence, and calls the Flash_Update function (programming/erasing the user area in the on-chip RAM).</p>
Arguments	None
Return Value	None
Flash_Update	
Outline	Program/erase operation
Header	iodefine.h, r_Flash_API_RX200.h, r_Flash_API_RX200_UserConfig.h
Declaration	void Flash_Update (void)
Description	<p>Receives the erase block number, programming data size, and programming data from the master with asynchronous serial communication, and performs programming/erasing the user area.</p> <p>Calls the Indicate_Ending_LED function when the operation has completed successfully, and calls the Indicate_Error_LED when an error has occurred.</p> <ul style="list-style-type: none"> - Starts processing when the FSTART communication command is received from the master. - Receives the ERASE communication command from the master, receives 1-byte erase block number, and calls the R_FlashErase function to erase the block. - Receives the WRITE communication command and receives 4-byte programming data size. - Receives 128-byte data from the master, calls R_FlashWrite function as many times as required for the programming data size to write the data to the user area. - When completed programming/erasing the user area successfully, notifies that the slave is not ready, and calls the Indicate_Ending_LED function to turn on LEDs for normal end.
Arguments	None
Return Value	None

Indicate_Ending_LED	
Outline	Normal end processing
Header	None
Declaration	void Indicate_Ending_LED (void)
Description	Displays the pattern for normal end with LED0 to LED3 when completed the programming/erasing successfully. Turns on each LED from LED0 to LED3 in order.
Arguments	None
Return Value	None
SCI_Rcv1byte	
Outline	1-byte data reception
Header	iodefine.h
Declaration	uint8_t SCI_Rcv1byte (void)
Description	Controls reception of 1-byte data with SCI0 asynchronous serial communication.
Arguments	None
Return Value	1-byte received data with SCI0 asynchronous serial communication.
Indicate_Error_LED	
Outline	Error processing
Header	None
Declaration	void Indicate_Error_LED(uint8_t error_no)
Description	Displays the pattern which indicates the error number with LED0 to LED3 when an error has occurred during programming/erasing the user area. Turns on LEDs to indicate the error number and turns off all LEDs repeatedly.
Arguments	uint8_t error_no : Error no. ⁽¹⁾ corresponding to an error occurred during programming/erasing to the user area
Return Value	None
Note:	
1. Refer to 5.4. LED Display for details of error numbers.	
SCI_Rcvnbyte	
Outline	n-byte data reception
Header	iodefine.h
Declaration	void SCI_Rcvnbyte(uint16_t size, uint8_t *rcv_buffer)
Description	Controls reception of n-byte data with SCI0 asynchronous serial communication. n: First argument in uint16_t type
Arguments	- uint16_t size : Byte counts of the receive data with SCI0 asynchronous serial communication - uint8_t *rcv_buffer : Start address of the received data storage area
Return Value	None
SCI_Tr1byte	
Outline	1-byte data transmission
Header	iodefine.h
Declaration	void SCI_Tr1byte(uint8_t data)
Description	Controls transmission of 1-byte data with SCI0 asynchronous serial communication.
Arguments	uint8_t data : 1-byte transmission data with SCI0 asynchronous serial communication
Return Value	None

5.14 Flowcharts

5.14.1 Initial Setting

Figure 5.9 shows the Initial Setting.

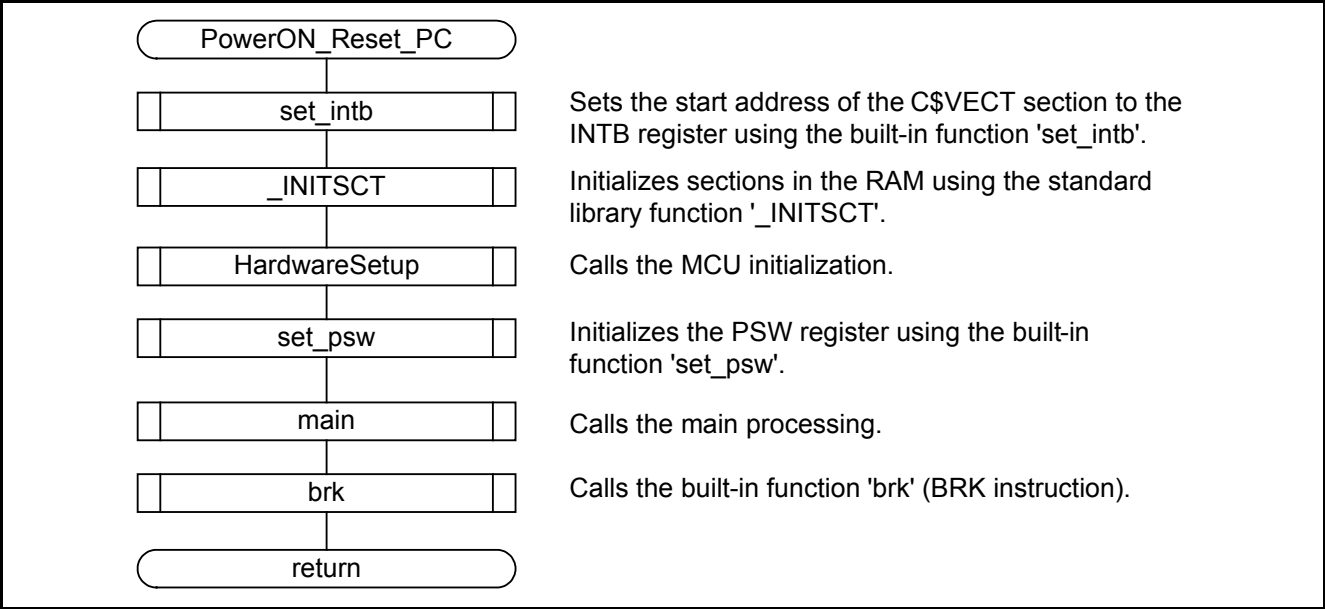


Figure 5.9 Initial Setting

5.14.2 MCU Initial Setting

Figure 5.10 and Figure 5.11 show the MCU Initial Setting.

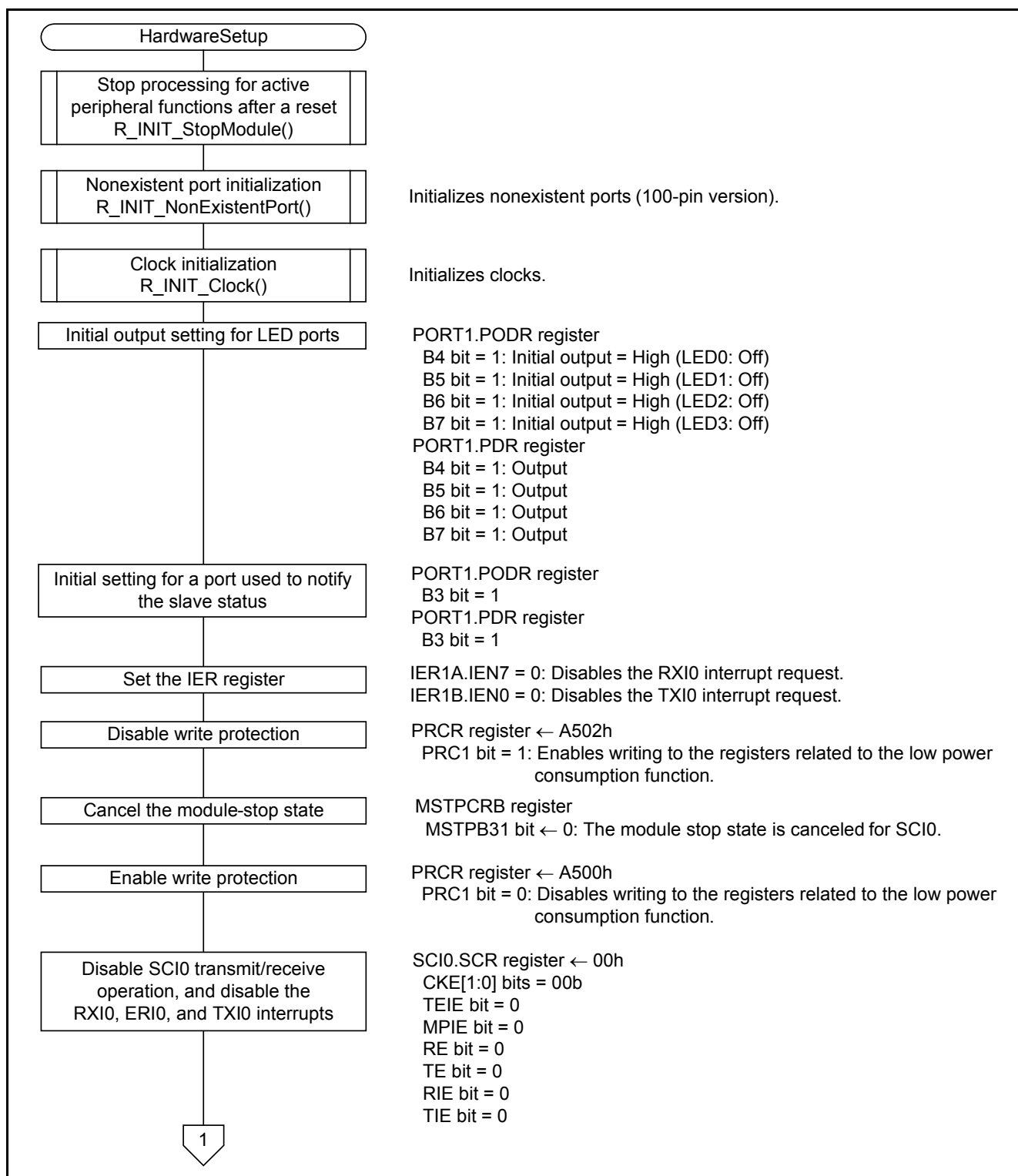


Figure 5.10 MCU Initial Setting (1/2)

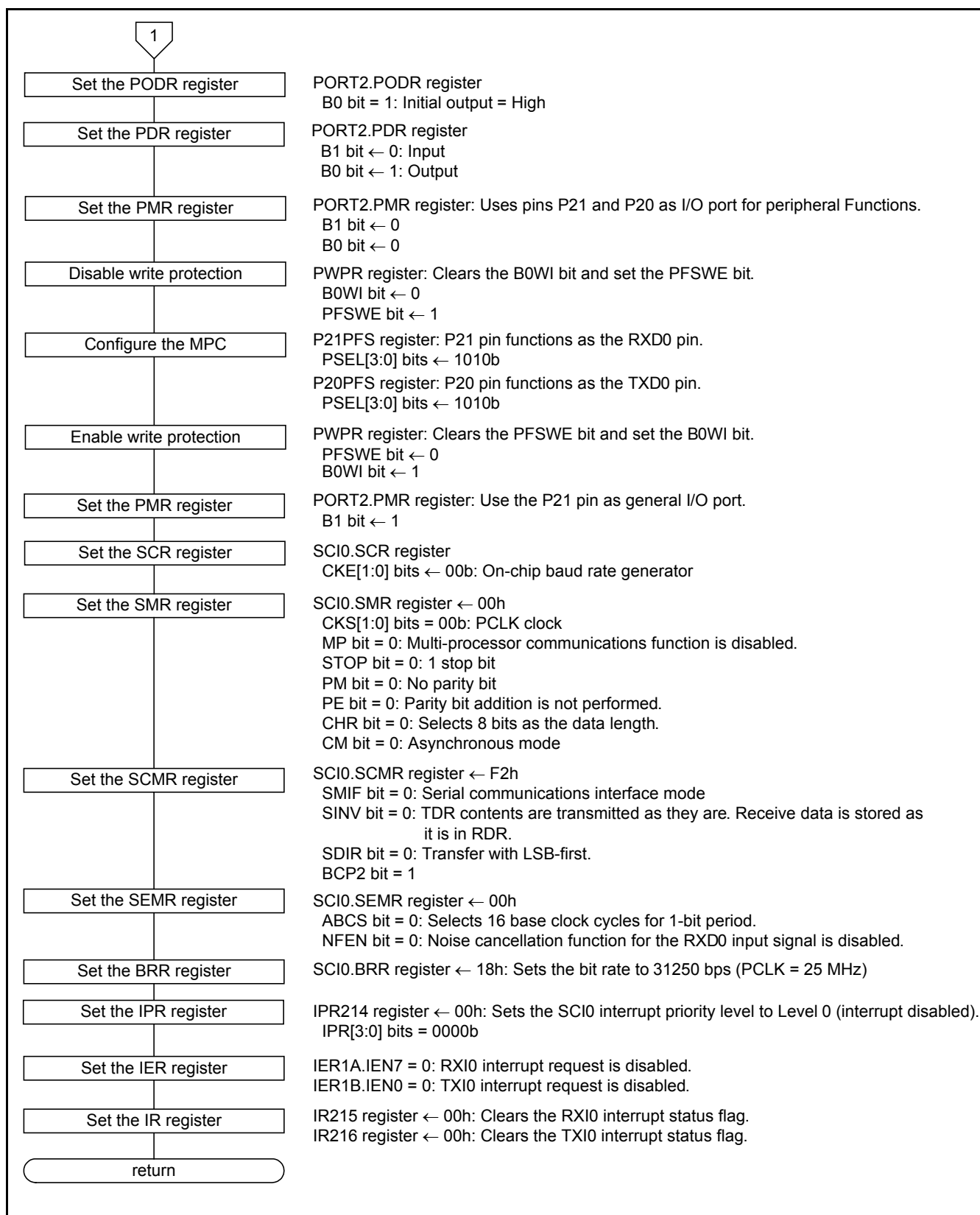


Figure 5.11 MCU Initial Setting (2/2)

5.14.3 Main Processing

Figure 5.12 shows the Main Processing.

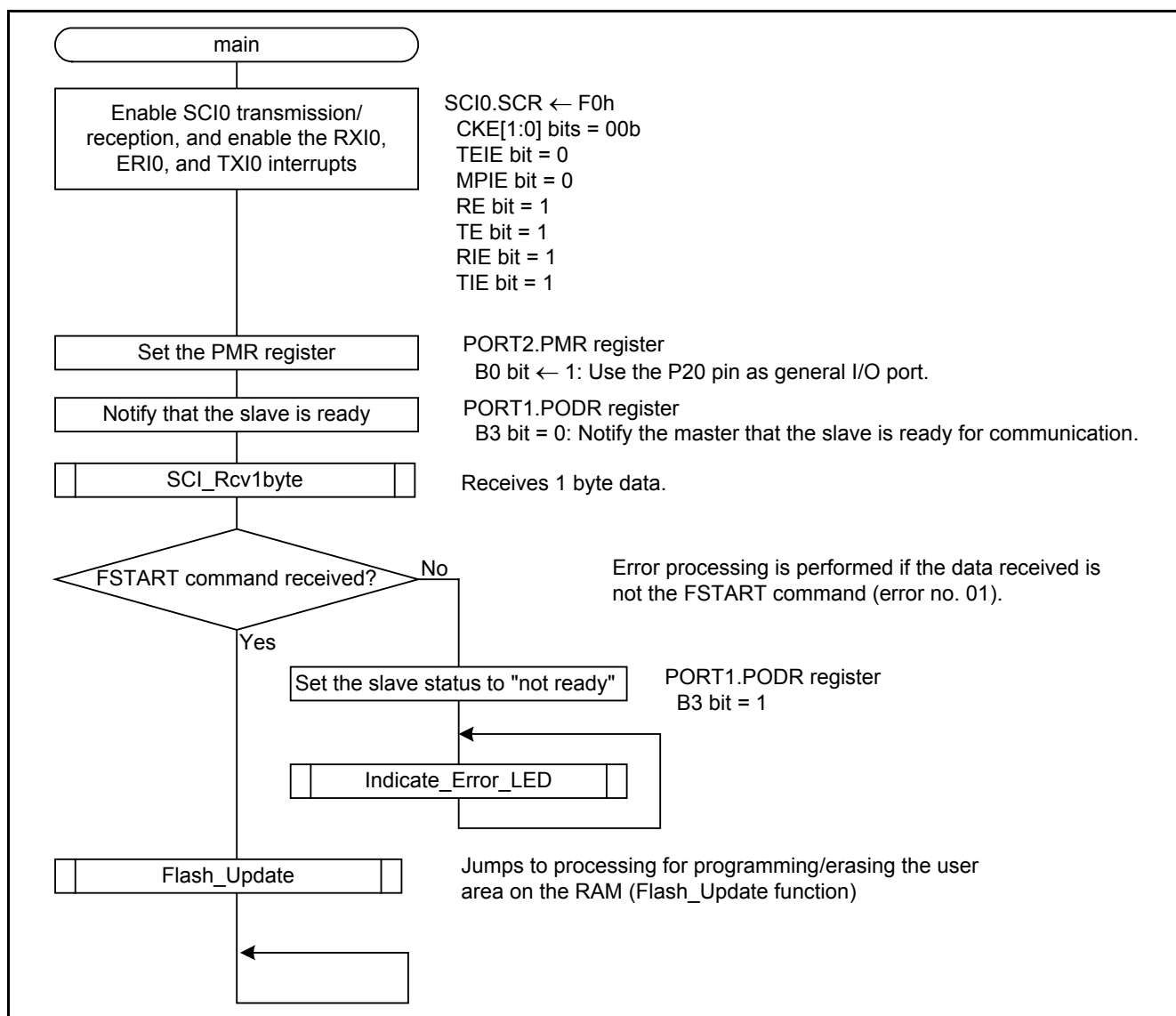


Figure 5.12 Main Processing

5.14.4 Program/Erase Operations

Figure 5.13 to Figure 5.15 show the Program/Erase Operations.

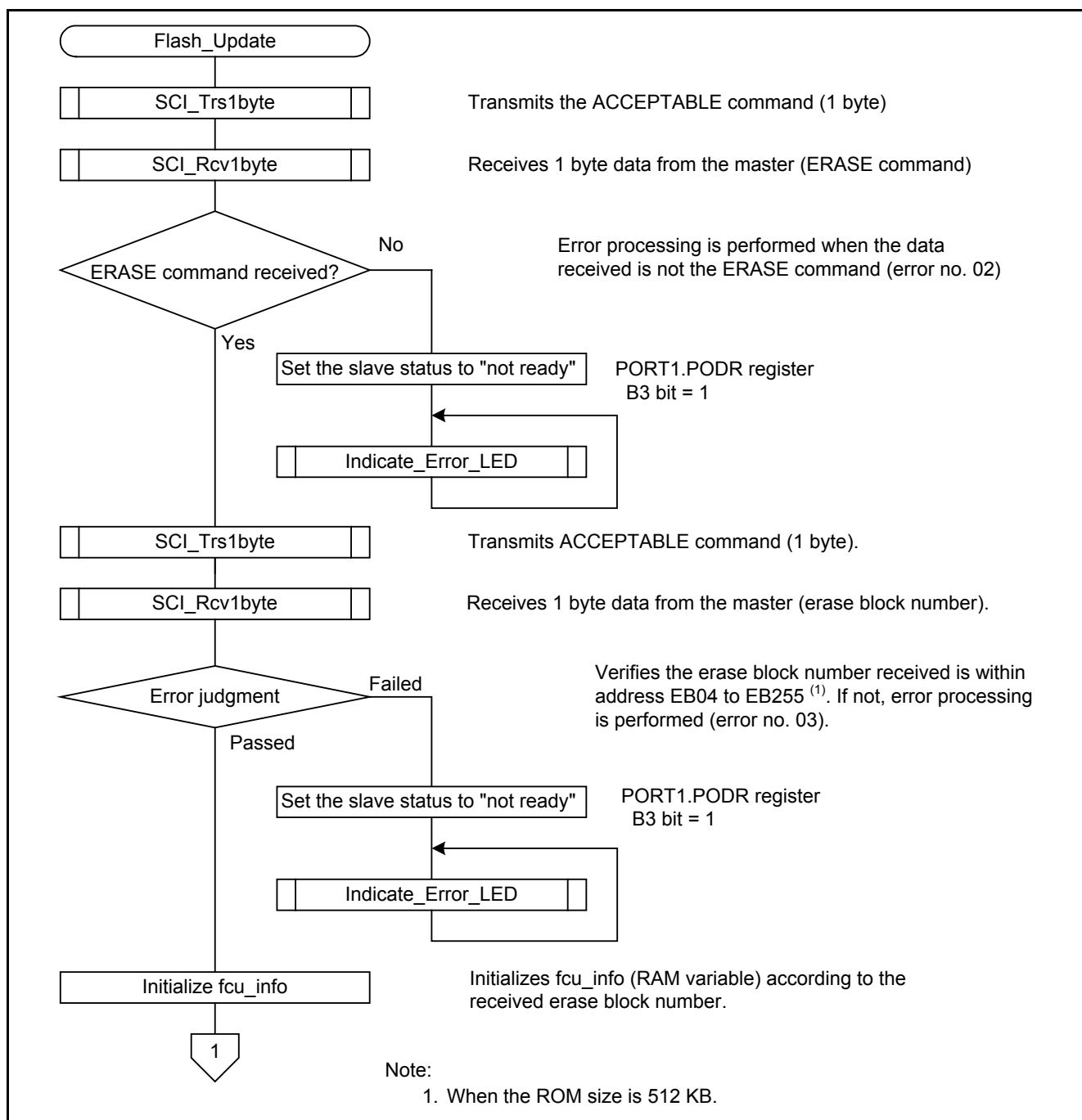


Figure 5.13 Program/Erase Operations (1/3)

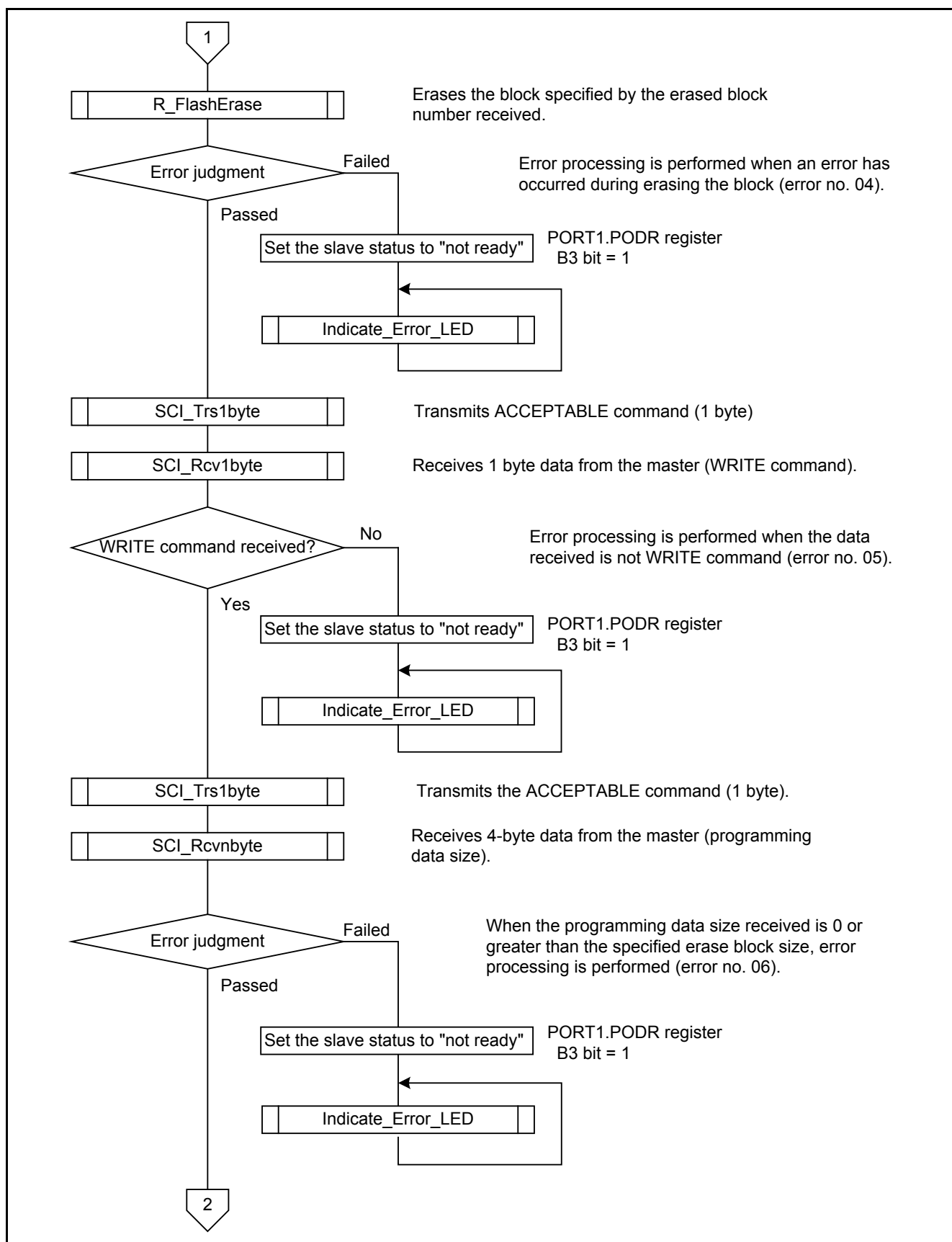


Figure 5.14 Program/Erase Operations (2/3)

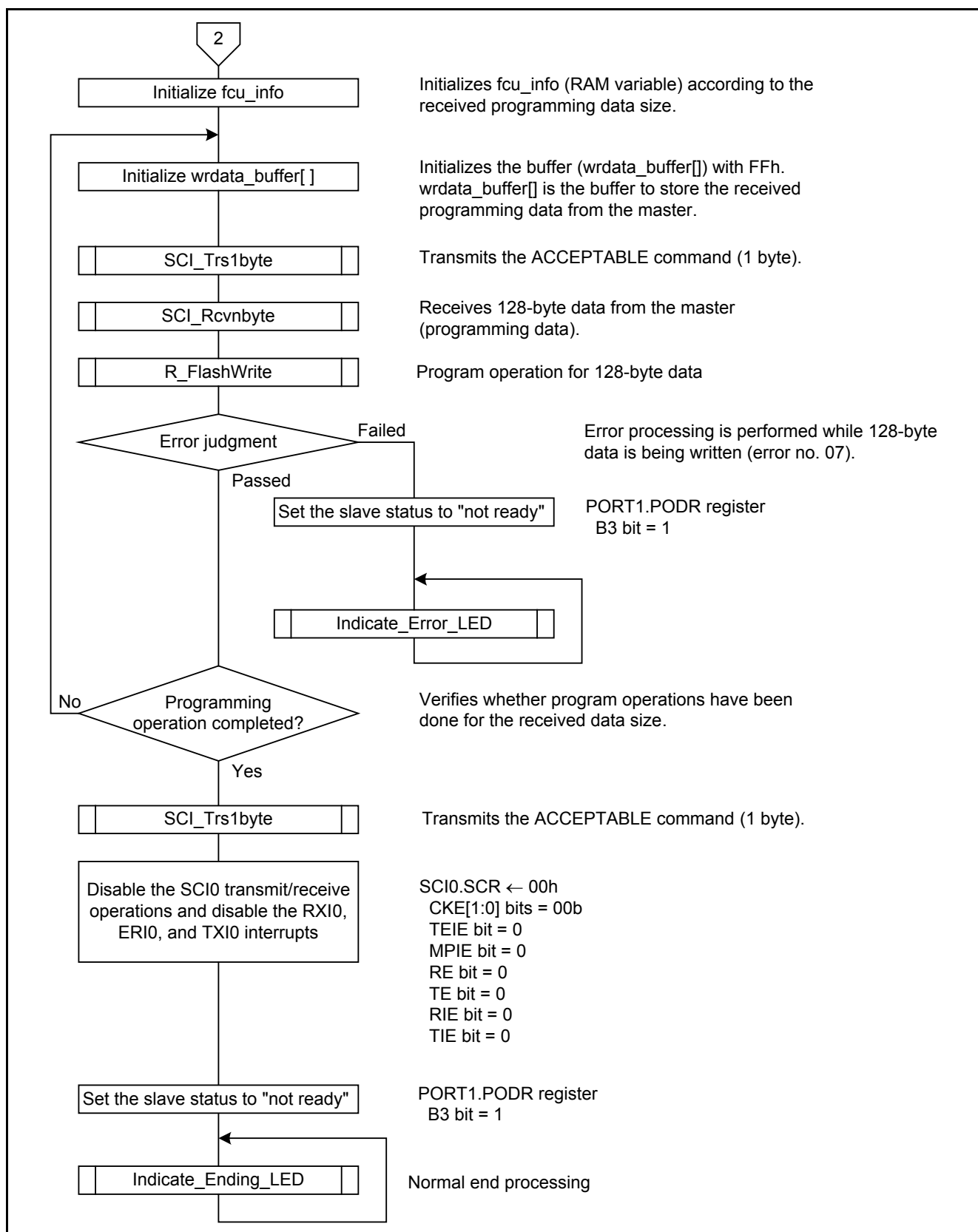


Figure 5.15 Program/Erase Operations (3/3)

5.14.5 Normal End Processing

Figure 5.16 shows the Normal End Processing.

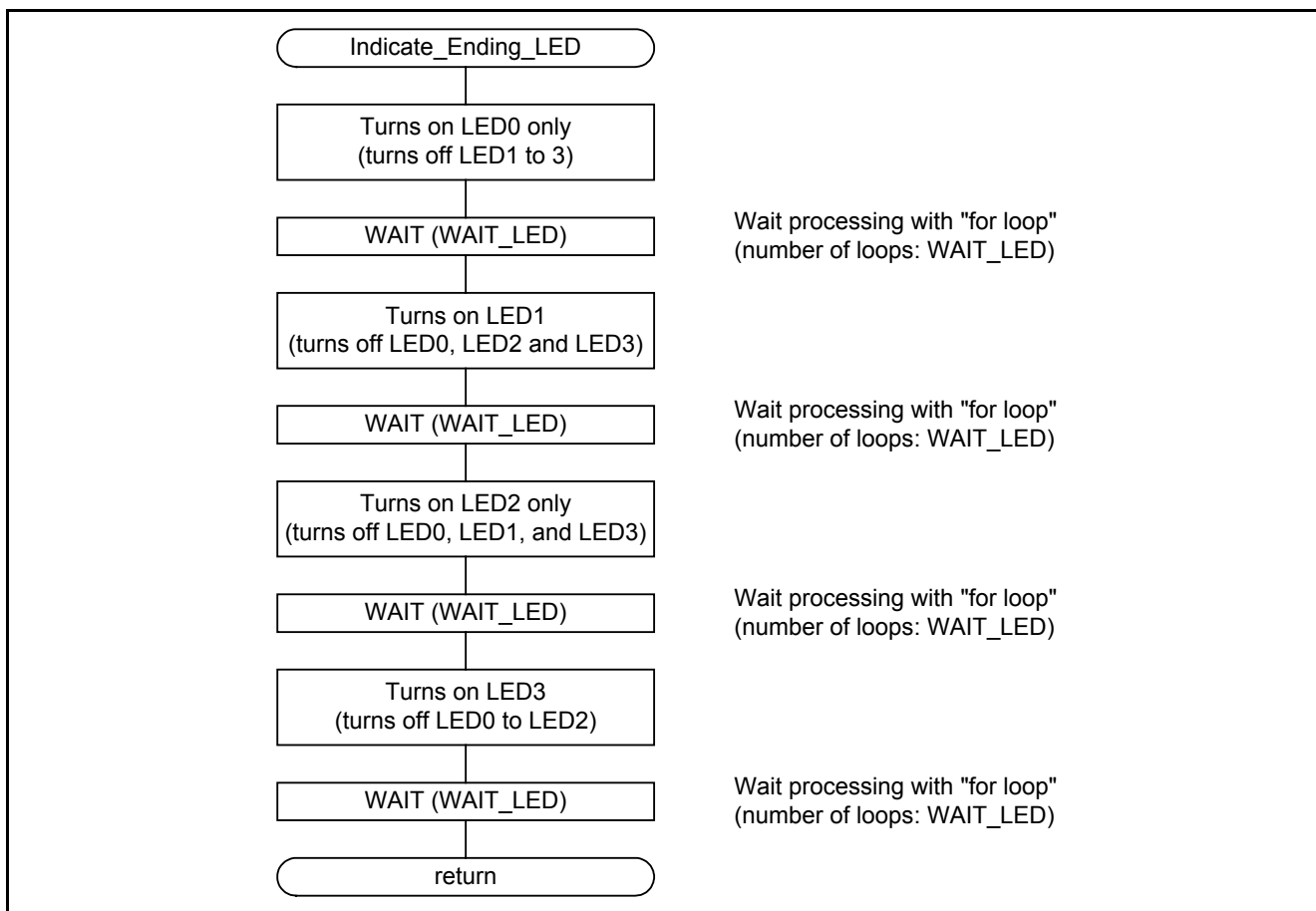


Figure 5.16 Normal End Processing

5.14.6 Error Processing

Figure 5.17 shows the Error Processing.

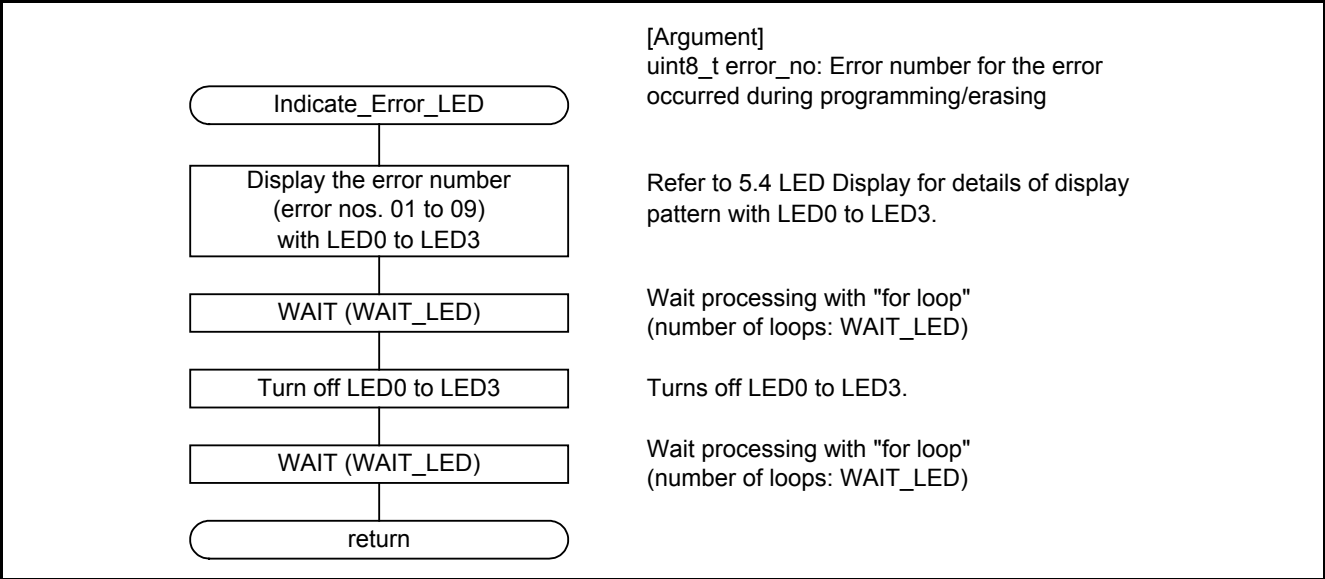


Figure 5.17 Error Processing

5.14.7 1-Byte Data Reception

Figure 5.18 shows the 1-Byte Data Reception.

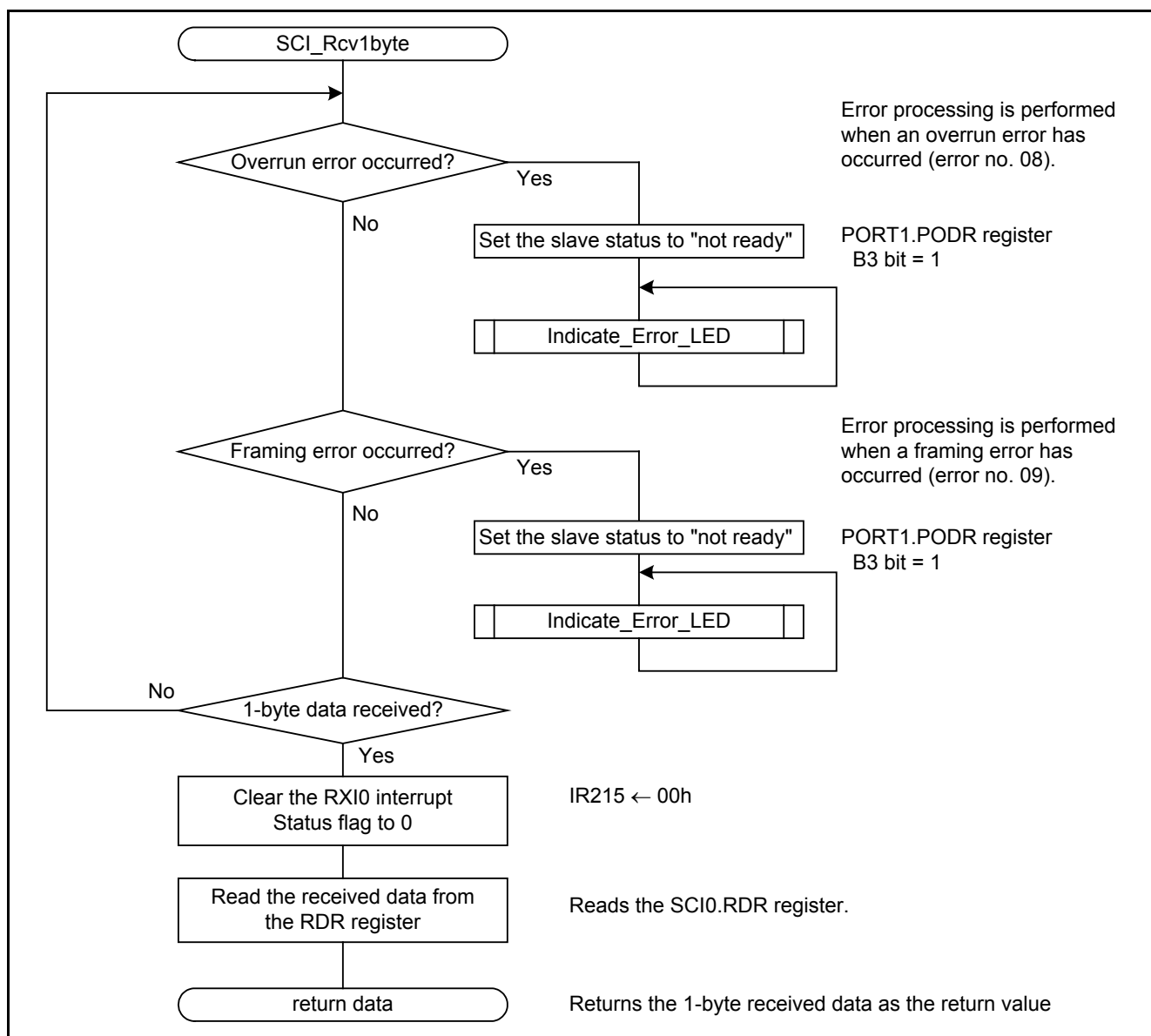


Figure 5.18 1-Byte Data Reception

5.14.8 n-Byte Data Reception

Figure 5.19 shows the n-Byte Data Reception.

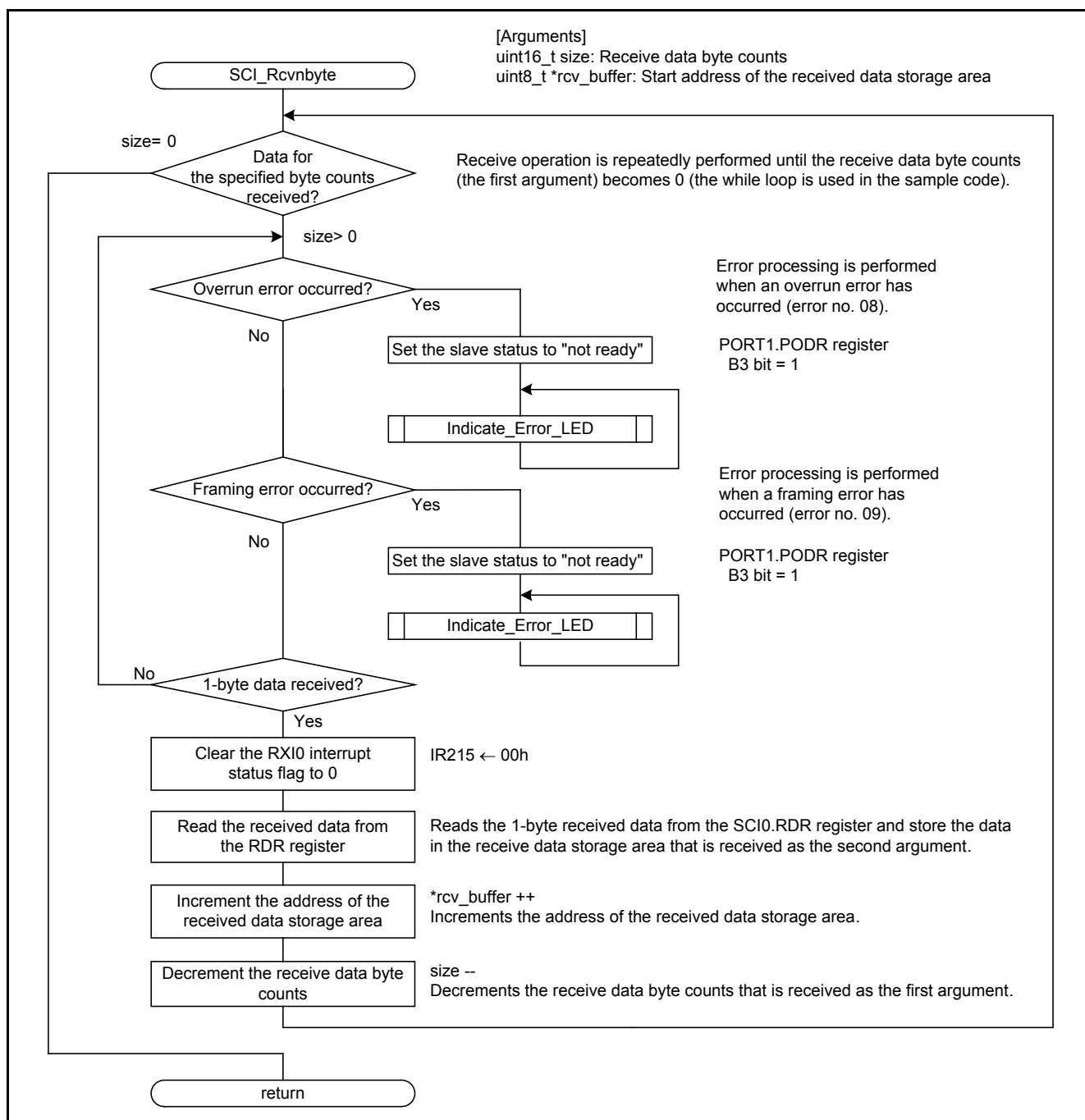


Figure 5.19 n-Byte Data Reception

5.14.9 1-Byte Data Transmission

Figure 5.20 shows the 1-Byte Data Transmission.

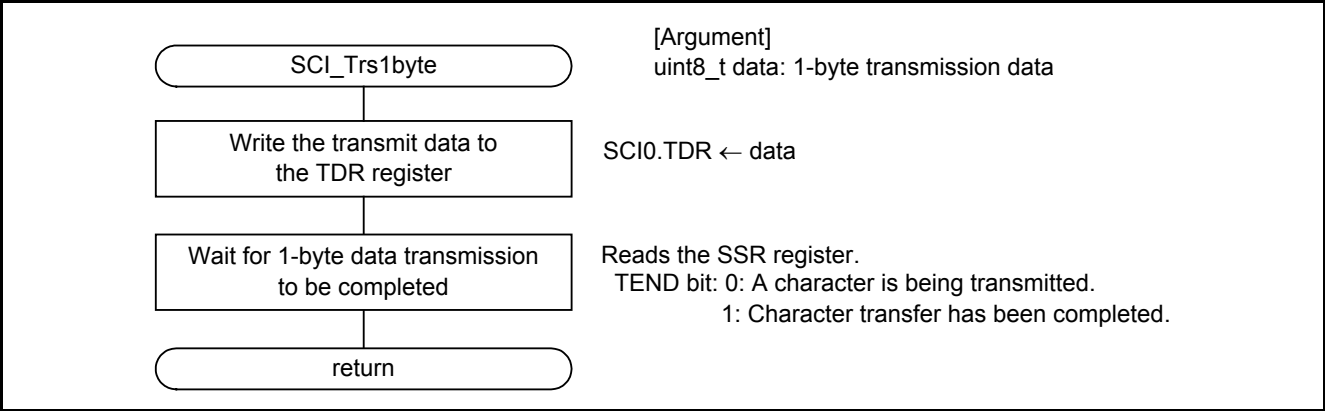


Figure 5.20 1-Byte Data Transmission

6. Usage Notes

6.1 Note on Reprogramming the Erase Block EB00

The fixed vector (FFFF FF80h to FFFF FFFFh), ID code protection (FFFF FFA0h to FFFF FFAFh), and other settings are located in the erase block EB00 (address for programming/erasing: 00FF F800h to 00FF FFFFh, address for reading: FFFF F800h to FFFF FFFFh).

When programming/erasing EB00, settings for the fixed vector and ID code protection are deleted. Therefore after erasing EB00, the fixed vector and ID code protection need to be configured again.

The ID code protection is the function to disable read, program and erase the host, and is determined using the control code and ID code on the ROM. Refer to the User's Manual: Hardware for details on this function.

6.2 Changing the ROM size

The ROM size used in the sample code is 512 KB. When using the MCU with 512 KB, 384 KB, 256 KB, or 128 KB of ROM, change the ROM_SIZE definition in r_Flash_API_RX200_UserConfig.h depending on the MCU used. For example when the ROM size is 512 KB, the definition is as follows: #define ROM_SIZE (524288)

Table 6.1 lists the ROM Sizes

Table 6.1 ROM Sizes

Product Part No.	ROM Size	Definition	Available Erase Block
R5F52108	512 KB	(524288)	EB04 to EB255
R5F52107	384 KB	(393216)	EB04 to EB191
R5F52106	256 KB	(262144)	EB04 to EB127
R5F52105	128 KB	(131072)	EB04 to EB63

6.3 Operation Mode Settings

The following operation mode settings are used in this application note.

- Mode pin: MD=High
- Operating mode: Single-chip mode
- SYSCR0.ROME bit ⁽¹⁾: 1 (the on-chip ROM is enabled)

Note:

1. The default setting of the SYSCR0.ROME bit is 1, thus the SYSCR0 register is not set in the sample code.

6.4 Endian Settings

This application note supports both big endian and little endian. When using the sample code, endian settings in the master and slave should be same.

6.4.1 When Using Little Endian

When using the little endian, specify "Little-endian data" in the endian setting of the compiler option. The setting value for MDES described in 5.8 Option-Setting Memory will be the value for the little endian.

6.4.2 When Using Big Endian

When using the big endian, specify "Big-endian data" in the endian setting of the compiler option. The setting value for MDES described in 5.8 Option-Setting Memory will be the value for the big endian.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware

RX210 Group User's Manual: Hardware Rev.1.20 (R01UH0037EJ)

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

Technical Update/Technical News

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

User's Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)

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

Website and Support

Renesas Electronics website

<http://www.renesas.com>

Inquiries

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

REVISION HISTORY	RX210 Group Application Note Reprogramming the On-Chip Flash Memory via UART in Single-Chip Mode
-------------------------	---

Rev.	Date	Description	
		Page	Summary
1.00	July 1, 2013	—	First edition issued

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

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 one with a different part number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

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.



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: +44-1628-651-700, Fax: +44-1628-651-804

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.
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.
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