RX630 Group

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

R01AN1271EJ0100
Rev.1.00
Jan. 15, 2014

Abstract

This application note describes the processing to transfer the erasure block number, write data size, and write data using asynchronous serial communication.

When using this application note, see also the "On-chip Flash Memory Reprogramming in Single-chip Mode via an UART Interface (Slave)" (R01AN1251EJ) RX630 Group application note for details on erasing and programming the internal flash memory (the user MAT) in the slave.

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

Erasing/programming the on-chip flash memory:
RX600 Series: Simple Flash API for RX600, rev. 2.20 (R01AN0544EU0220)

Products

- 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

The master sends the erasure block number, the write data size, and the write data to the slave using asynchronous serial communication and the slave programs/erases its own 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. When the master device determines that the slave device is in the ready state by confirming that the input level on the general port P15 pin (internal pull-up enabled) has switched to low-level, it indicates the slave ready state by means of LEDs.
2. Handshaking is used to control communication between the master and slave. In particular, after a transmission to the slave, the master waits until an [ACCEPTABLE] (55h) command is returned. After receiving an [ACCEPTABLE] (55h) command, the master starts the next communication.
3. After the LED indication of the slave ready state, the master waits for the switch connected to its external interrupt pin (IRQ2-DS) to be pressed. When the switch is pressed, the master turns off all the LEDs, then starts serial communication and controls programming/erasing of the slave's user MAT.
4. Using communication commands, the master tells the slave to erase one of its user MAT erasure blocks (EB00 to EB69). (If there are blocks in the slave's user MAT that cannot be erased/programmed, they should be identified by the slave.) In the sample program of this application note, the slave is told to erase the EB08 erasure block.
5. After the slave completes erasing EB08, the master transmits the write data size (4 bytes) and the write data (8 KB) to the slave.
6. When the slave has successfully erased/programmed the user MAT, the master indicates the normal end with the LEDs. Also, if an error occurs during communication with the slave, the error is indicated with LEDs.

Table 1.1 lists the Peripheral Function and Its Application and Figure 1.1 shows the Usage Example.

Table 1.1 Peripheral Function and Its Application

Peripheral Function	Application
Serial communication interface (channel 0)	Used for asynchronous serial communication with the slave device.

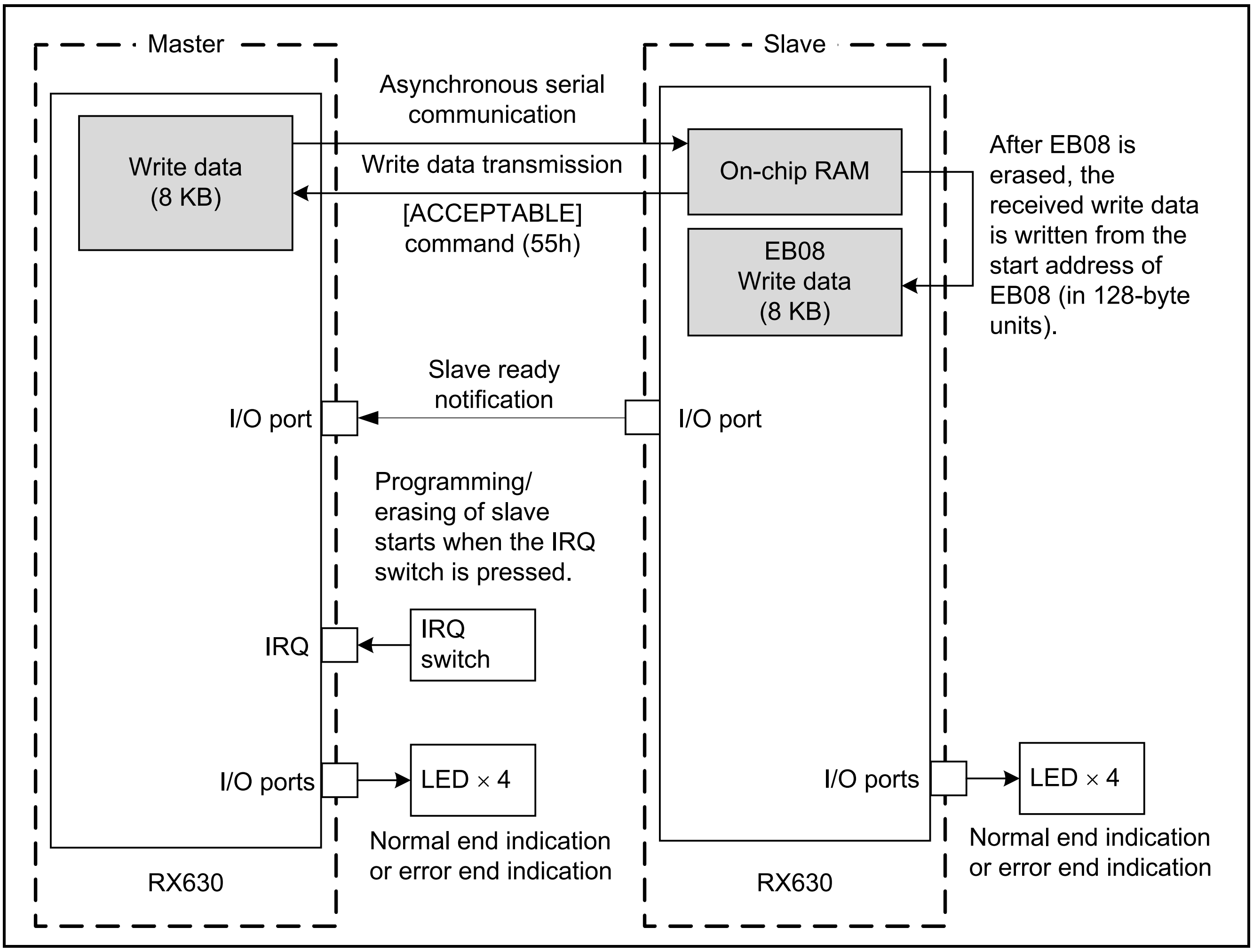


Figure 1.1 Usage Example

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	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 4.09.01.007
C compiler	Renesas Electronics RX Standard Toolchain Version 1.2.1.0
	Compiler options (The integrated development environment default settings are used.)
iodefine.h version	1.5
Endian	Little endian
Operating mode	Single-chip mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit for RX630 (Product No.: R0K505630C000BE)

3. Reference Application Notes

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

- RX630 Group Initial Setting, rev. 1.00 (R01AN1004EJ0100)
- RX600 Series Simple Flash API for RX600, rev. 2.20 (R01AN0544EU0220)

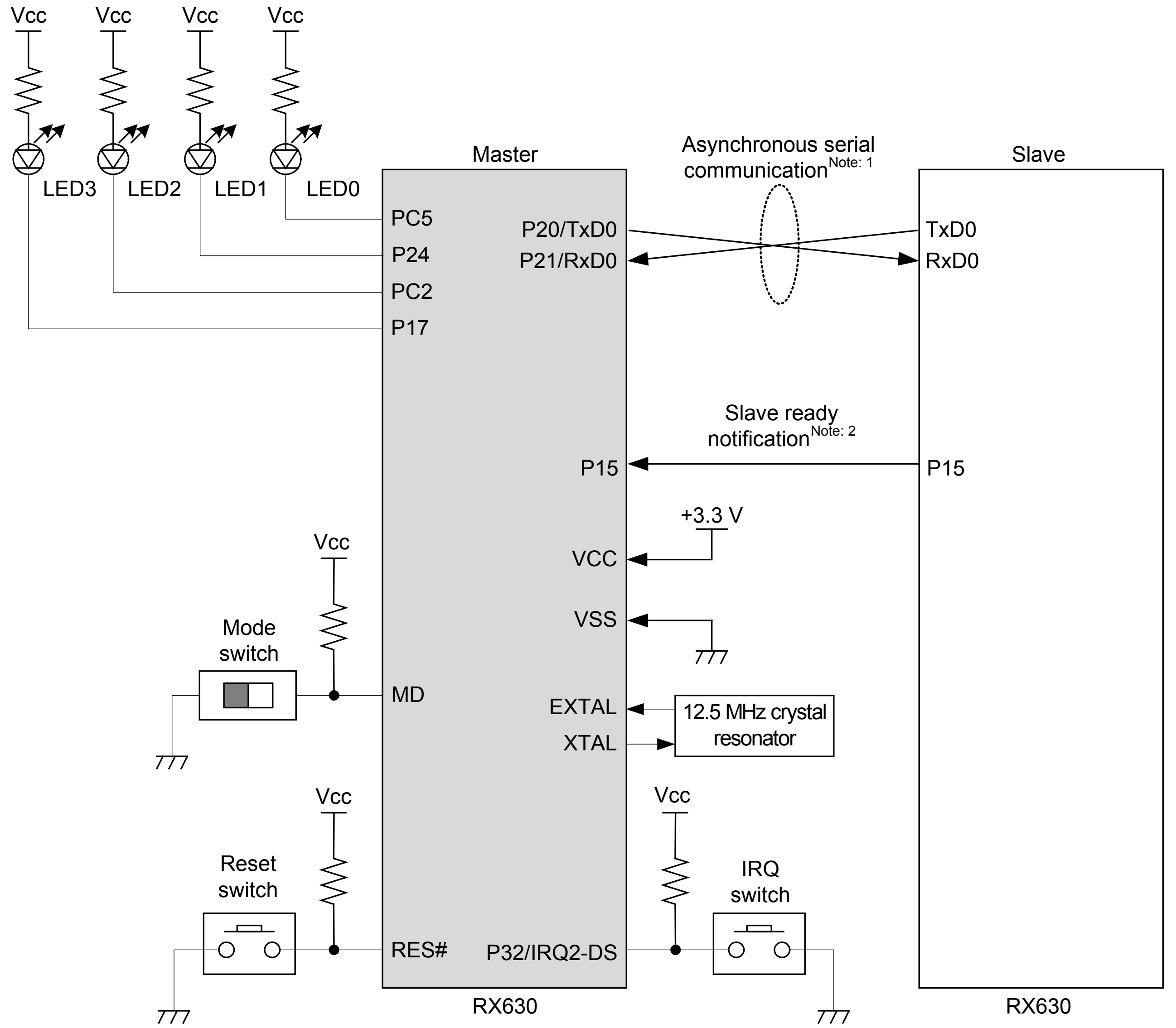
The functions from the above application notes are used in the sample code of this application note. The revision numbers indicated are current as of the time this application note was made.

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 master device as used in this application note.



Notes: 1. The TxD0 pin of the master 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 Master 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	Slave communication serial transmit pin
P21/RXD0	Input	Slave communication serial receive pin
P15	Input	Slave ready confirmation pin
P32/IRQ2-DS	Input	When the connected switch is pressed, programming/erasing of the slave’s user MAT starts.
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

The sample program of this application note uses asynchronous serial communication to transmit communication commands, the erasure block number, the write data size, and the write data from the master to the slave. The slave transmits the [ACCEPTABLE] command (55h) as a status command for handshaking. The TxD0 pin of SCI0 is pulled up externally, and the RxD0 pin is connected to the RS232C driver output pin and is high-level when in the normal state.

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.

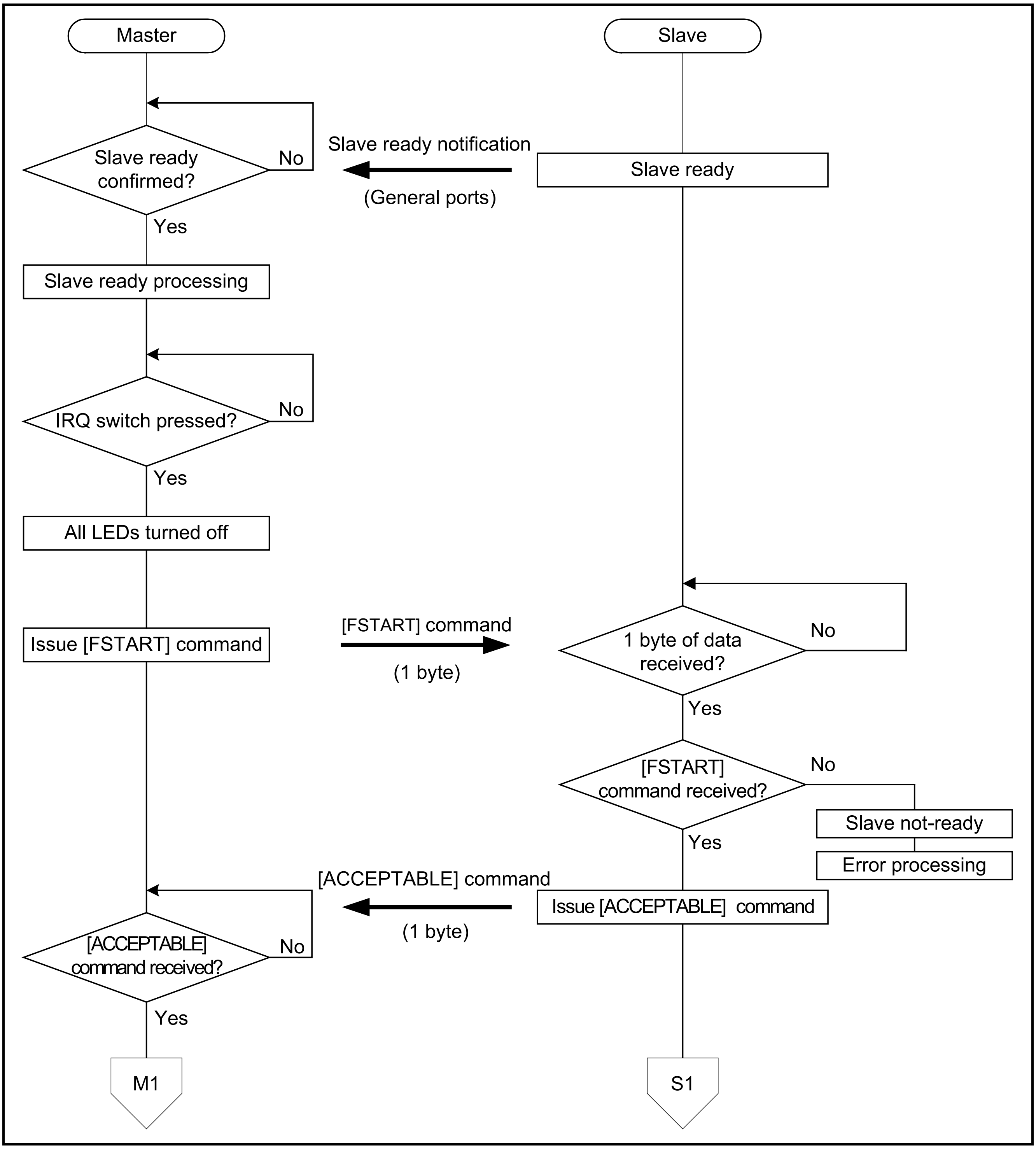


Figure 5.1 Communication Sequence (1)

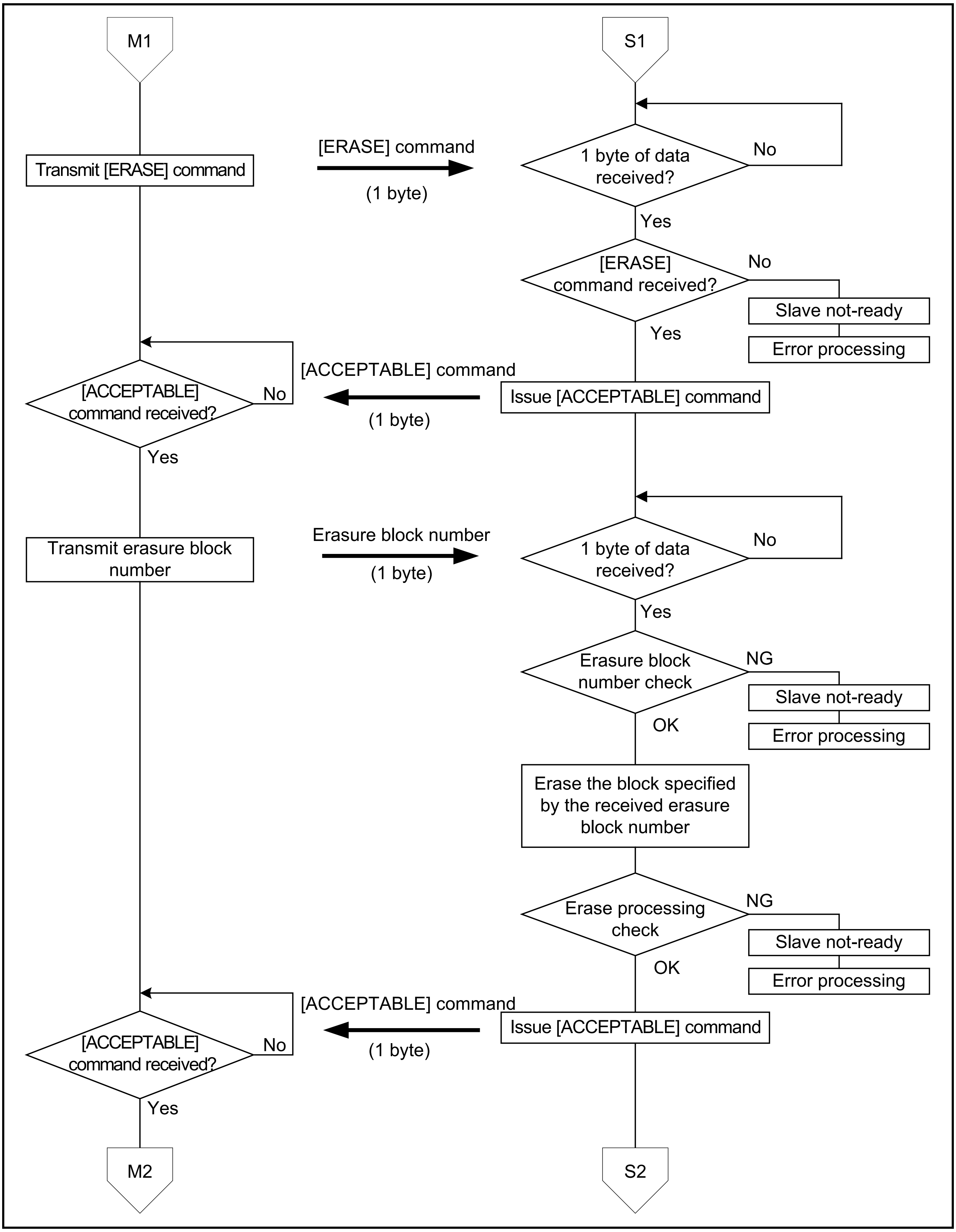


Figure 5.2 Communication Sequence (2)

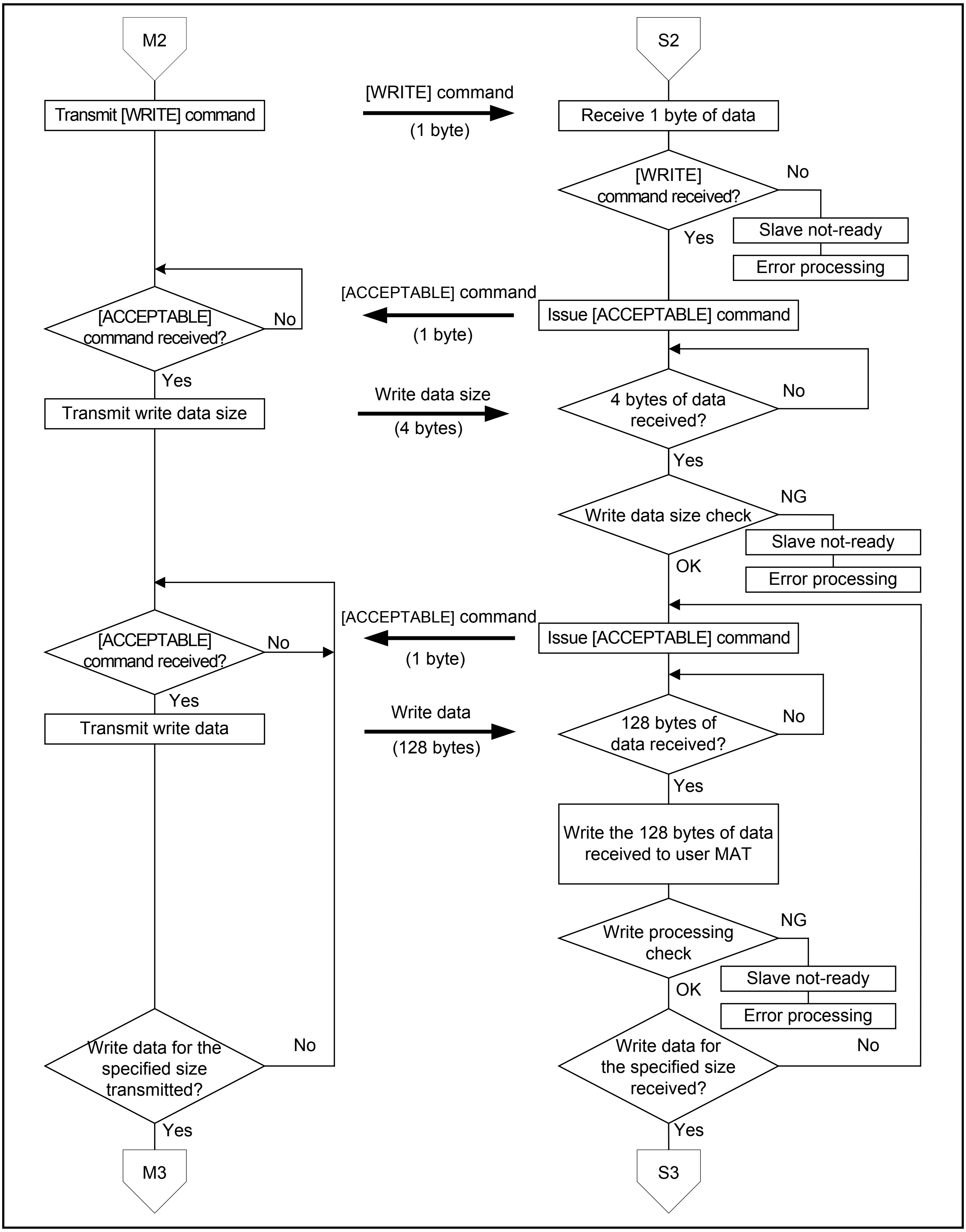


Figure 5.3 Communication Sequence (3)

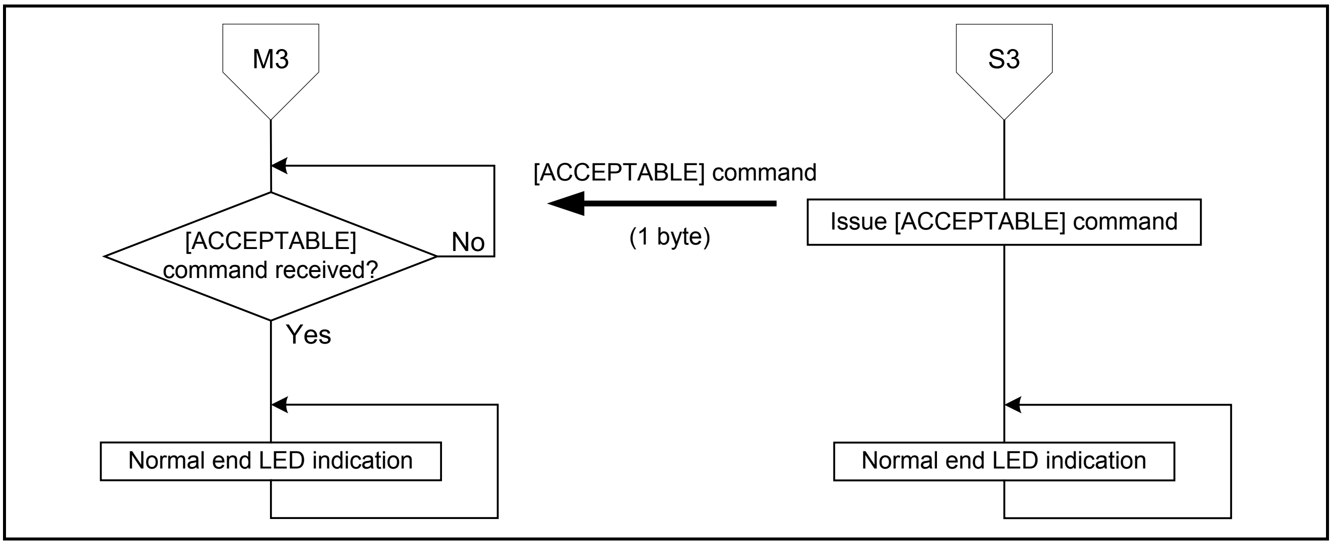


Figure 5.4 Communication Sequence (4)

5.1.4 Erasure Block Number

After transmitting an ERASE command, the master transmits the erasure block number (one byte of data defined by a symbolic constant). Note that since the slave uses the code in the "RX600 Simple Flash API" application note Rev.2.20 (R01AN0544EU0220), the constants defined in the r_flash_api_rx600.h header file are used for this erasure block number.

Table 5.3 lists the erasure block numbers defined in r_flash_api_rx600.h. Figure 5.5 shows the erasure block number specifications.

For details on the erasure block numbers, see the “RX600 Simple Flash API” application note listed in section 8, Reference Documents.

Table 5.3 Erasure Block Numbers from r_flash_api_rx600.h

Erasure Block Number		
Symbolic Constant	Value	Description
BLOCK_0	0	Specifies erasure block EB0 (size: 4 KB)
BLOCK_1	1	Specifies erasure block EB1 (size: 4 KB)
BLOCK_2	2	Specifies erasure block EB2 (size: 4 KB)
. . .	. . .	. . .
BLOCK_6	6	Specifies erasure block EB6 (size: 4 KB)
BLOCK_7	7	Specifies erasure block EB7 (size: 4 KB)
BLOCK_8	8	Specifies erasure block EB8 (size: 16 KB)
BLOCK_9	9	Specifies erasure block EB9 (size: 16 KB)
. . .	. . .	. . .
BLOCK_36	36	Specifies erasure block EB36 (size: 16 KB)
BLOCK_37	37	Specifies erasure block EB37 (size: 16 KB)
BLOCK_38	38	Specifies erasure block EB38 (size: 32 KB)
BLOCK_39	39	Specifies erasure block EB39 (size: 32 KB)
. . .	. . .	. . .
BLOCK_52	52	Specifies erasure block EB52 (size: 32 KB)
BLOCK_53	53	Specifies erasure block EB53 (size: 32 KB)
BLOCK_54	54	Specifies erasure block EB54 (size: 64 KB)
BLOCK_55	55	Specifies erasure block EB55 (size: 64 KB)
. . .	. . .	. . .
BLOCK_68	68	Specifies erasure block EB68 (size: 64 KB)
BLOCK_69	69	Specifies erasure block EB69 (size: 64 KB)

Erasure block number (uint8_t type)

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

The sample code in this application note specifies [BLOCK_8(8)] as the erasure block number to program and erase the erasure block EB08.

Note: A value shown in table 5.3, [BLOCK_0(0)] to [BLOCK_69(69)], should be specified as the erasure block number. Make sure to check the specifications of the slave device as well.

Figure 5.5 Erasure Block Number Specifications

5.1.5 Write Data Size

After transmitting a [WRITE] command, the master transmits 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

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

Note: The size of write data transmission is fixed to 128 bytes, which is a unit to write the user MAT in the RX630 Group devices. Consequently, the master transmits write data in units of 128 bytes. When the write 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 transmitted to the slave.

Figure 5.6 Write Data Size Specifications

5.1.6 Overrun Error

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

5.1.7 Framing Error

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

5.2 LED Indications

Figure 5.7 lists the LED indications corresponding to the operating states of the sample code.

Error Number	Operating State	LED Indication				
		LED3	LED2	LED1	LED0	Sequence
	Slave ready	○	○	○	○	
	Normal end (Indication shifts at fixed intervals.)	●	●	●	○	↓
		●	●	○	●	↓
		●	○	●	●	↓
		○	●	●	●	↑ (repeat)
Error No.01	Overrun error	⊗	●	●	●	
Error No.02	Framing error	⊗	●	●	⊗	

On: ○ Flashing: ⊗ Off: ●

Figure 5.7 LED Indications

5.3 Handshaking Control

The master uses handshaking with the slave for communications control.

As handshaking control, the master waits after serial transmission until an [ACCEPTABLE] command (55h) is received from the slave. The master starts the next serial transmission after it has received an [ACCEPTABLE] command from the slave.

5.4 File Composition

Table 5.4 lists the files used in the sample code. Files automatically generated by the integrated development environment should not be listed in this table.

Table 5.4 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 slave, transmission 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.h	Include header file for the simple flash API (erasure block number references)	For details, see the application note RX600 Series: Simple Flash API for RX600.
r_flash_api_rx600_config.h	Include header file for the simple flash API to specify parameters	
r_flash_api_rx600_private.h	Include header file for the simple flash API to specify parameters (device-related information)	
mcu_info.h	Include header file of the RX600 Series: Simple Flash API to specify parameters for RX600	

5.5 Option-Setting Memory

Table 5.5 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 5.5 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*[1]	FFFF FF83h to FFFF FF80h	 FFFF FFFFh FFFF FFF8h	(Single-chip mode) Little-endian Big-endian

Note: 1. The sample code uses the little-endian setting. For information on changing the endianness, see 6.1, Endianness.

5.6 Constants

Table 5.6 lists the constants used in the sample code.

Table 5.6 Constants Used in the Sample Code

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 from 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
READY	0	Input level indicating slave ready
SLAVE_READY_PORT	PORT1.PIDR.BIT.B5	Port to confirm the slave-ready status
SLAVE_READY_PCR	PORT1.PCR.BIT.B5	Pull-up control for the port corresponding to the port to confirm the slave-ready status
FALL_EDGE	1	Falling edge setting
RISE_EDGE	2	Rising edge setting
SW_ON	1	START_SW_IR value when IRQ switch is on
SW_OFF	0	START_SW_IR value when IRQ switch is off
START_SW_IR	ICU.IR[IR_ICU_IRQ2].BIT.IR	IRQ switch state
START_SW_PMR	PORT3.PMR.BIT.B2	IRQ switch pin select
START_SW_IRQMD	ICU.IRQCR[2].BIT.IRQMD	IRQ switch detect setting
RxD0_PMR	PORT2.PMR.BIT.B1	RxD0 pin select
TxD0_PMR	PORT2.PMR.BIT.B0	TxD0 pin select
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*[1]	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.
TRS_SIZE	ROM_PROGRAM_SIZE	Write data transmit size
BUF_SIZE	8192	Write buffer size
WRITE_SIZE	BUF_SIZE	Write data storage area size

Note: 1. These values are valid when an RX630 group MCU is used.

5.7 Variables

Table 5.7 lists the const variables.

Table 5.7 const Variables

Type	Variable Name	Contents	Function Used
const uint8_t	SAMPLE_DATA [BUF_SIZE]	Write data (8,192 bytes) transmitted to the slave for programming the user MAT In this sample code, the SAMPLE_DATA[BUF_SIZE] is allocated in the CP_DATA_1 section and allocated in the erasure blocks EB03 to EB02 (FFFF C000h to FFFF DFFFh).	main

5.8 Functions

Table 5.8 lists the Functions.

Table 5.8 Functions

Function Name	Outline
HardwareSetup	Initialization processing
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
Indicate_Ending_LED	Normal end processing function
SCI_Rcv1byte	1 byte data reception function
SCI_Trs1byte	1 byte data transmission function
SCI_Trsnbyte	n byte data transmission function
mpc_init	MPC initial settings function
pmr_init	PMR initial settings function

5.9 Function Specifications

The following tables list the sample code function specifications.

HardwareSetup	
Outline	Initialization processing
Header	iodefine.h, r_init_clock.h, r_init_non_existent_port.h, r_init_stop_module.h
Declaration	void HardwareSetup(void)
Description	Performs initialization processing. • 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. • Pull-up control for the port corresponding to the port to confirm the slave-ready status. • Cancels the module stop state. • IRQ multifunction pin controller (MPC) • IRQ port mode register (PMR) settings. • Initial pin function settings for the I/O port (P32/IR2-DS) connected to the switch. • SCI0 initial setting.
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. 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. For details of this function, see the application note RX630 Group: Initial Setting, rev. 1.00.

main	
Outline	Main function
Header	iodefine.h
Declaration	void main(void)
Description	The main function performs the following processing: • Determines when the IRQ switch is pressed. • SCI multifunction pin controller (MPC) • Makes SCI port mode register (PMR) settings. • Controls transmission of communication commands to the slave. • Controls transmission of the erasure block number. • Controls transmission of the write data size. • Controls transmission of write data. • Controls reception of [ACCEPTABLE] commands from the slave. • At normal end, calls the Indicate_Ending_LED function.
Arguments	None
Return Value	None

Indicate_Ending_LED	
Outline	Normal end processing function
Header	None
Declaration	static void Indicate_Ending_LED(void)
Description	When programming/erasing the slave's user MAT finishes successfully, the Indicate_Ending_LED function indicates a normal end by means of LED0 to LED3. The normal end indication is made with LED0 to LED3 illuminating one after another, in sequence.
Arguments	None
Return Value	None

SCI_Rcv1byte	
Outline	1 byte data receive function
Header	None
Declaration	static uint8_t SCI_Rcv1byte(void)
Description	The SCI_Rcv1byte function controls reception of one byte data with SCI0 asynchronous serial communication.
Arguments	None
Return Value	One byte of received data with SCI0 asynchronous serial communication

SCI_Trs1byte		
Outline	1 byte data transmit function	
Header	None	
Declaration	static void SCI_Trs1byte(uint8_t data)	
Description	The SCI_Trs1byte function controls transmission of one byte data with SCI0 asynchronous serial communication.	
Arguments	uint8_t data	: One byte of transmit data using the SIC0 asynchronous serial communication
Return Value	None	

SCI_Trsnbyte		
Outline	*n* byte data transmit function	
Header	None	
Declaration	static void SCI_Trsnbyte(uint16_t size, uint8_t *trs_buffer)	
Description	The SCI_Trsnbyte function controls transmission of n-byte data with SCI0 asynchronous serial communication (*n* is the first argument in uint16_t type).	
Arguments	uint16_t size	: Transmit data size
	uint8_t *trs_buffer	: Pointer to buffer for storing transmit data
Return Value	None	

mpc_init

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

pmr_init

Outline	PMR initial settings function
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

5.10 Flowcharts

5.10.1 Initial Settings Function

Figure 5.8 is a flowchart of the initial settings function.

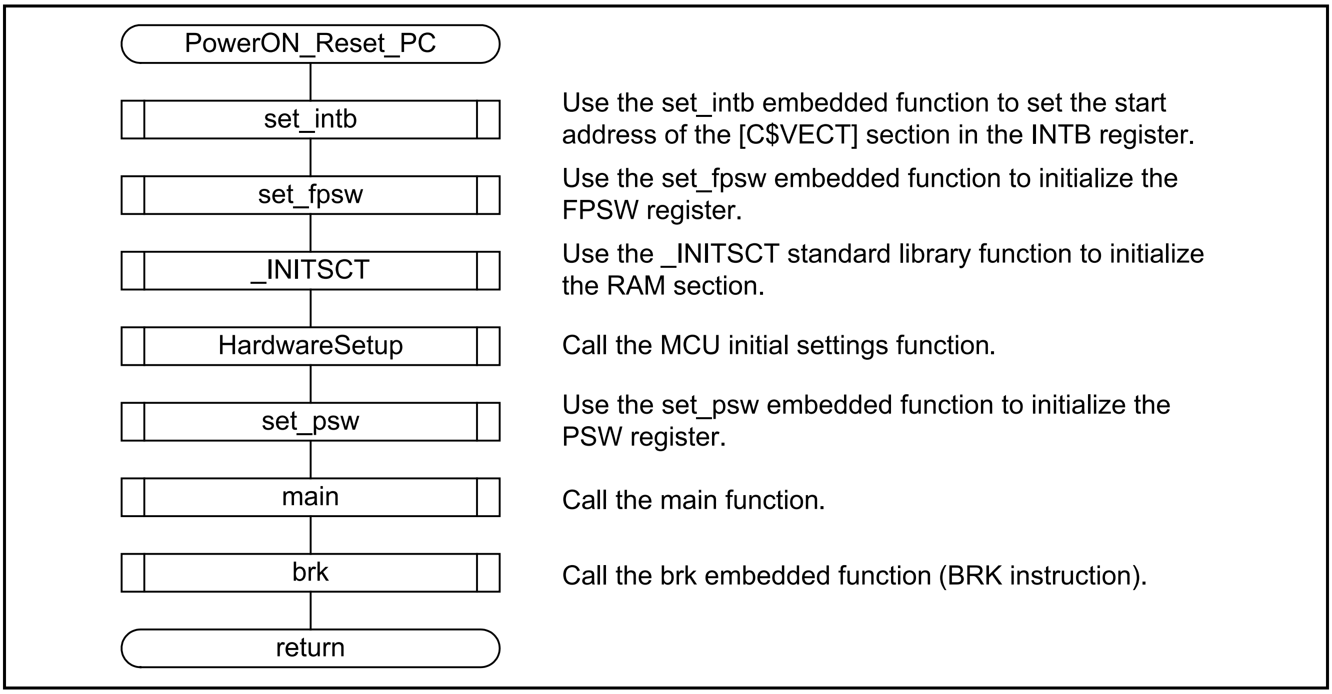


Figure 5.8 Initial Settings Function

5.10.2 Initialization Processing

Figures 5.9 and 5.10 are flowcharts of initialization processing 1 and initialization processing 2.

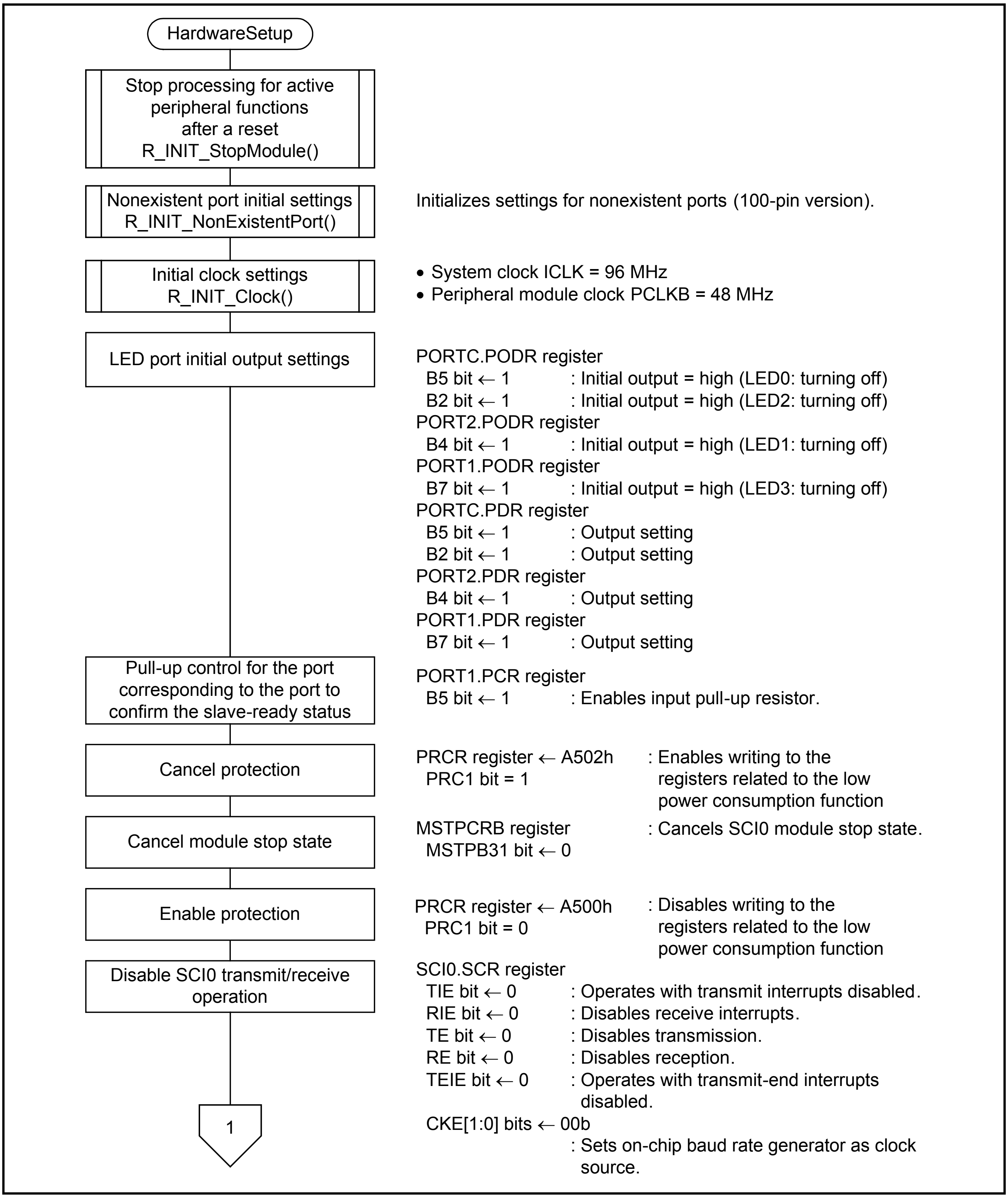
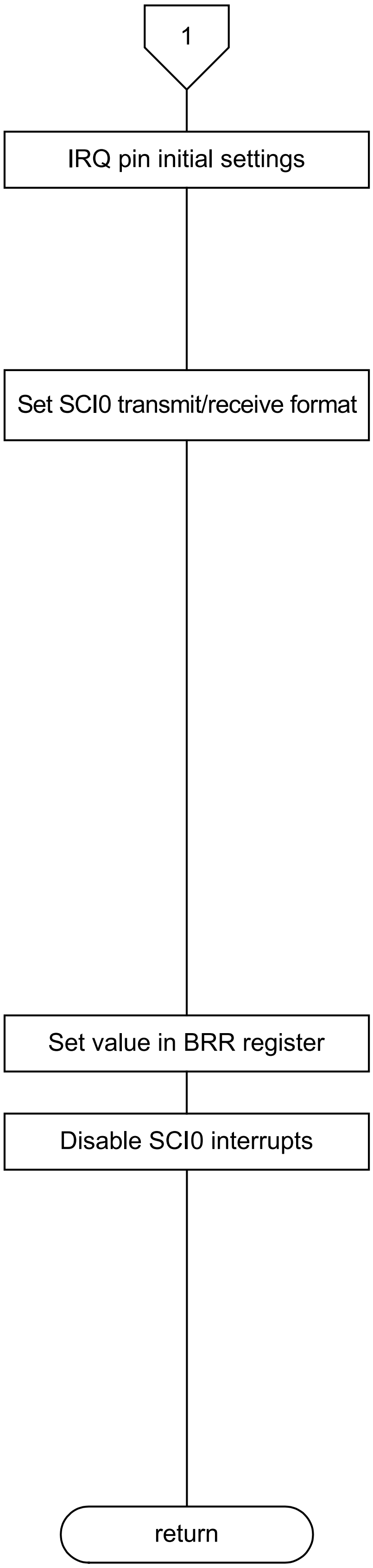


Figure 5.9 Initialization Processing 1



IRQCR2 register
IRQMD bit ← 01b : Sets IRQ2 detection edge to falling edge.
P32PFS register
ISEL bit ← 1 : Sets as IRQ2 pin.
PORT3.PMR register
B2 bit ← 1 : Sets P32 as IRQ2-DS input pin.

SCI0.SMR register ← 00h
CM bit = 0 : Operates in asynchronous mode.
CHR bit = 0 : Sets data length to 8 bits.
PE bit = 0 : Selects no parity.
STOP bit = 0 : Selects 1 stop bit.
MP bit = 0 : Disables multiprocessor communication function.
CKS[1:0] bits = 00b : Sets PCLK clock as clock source.

SCI0.SCMR register ← F2h
SDIR bit = 0 : Selects LSB-first transmit/receive operation.
SINV bit = 0 : Contents of TDR register are transmitted unmodified; received data is stored unmodified in RDR register.

SMIF bit = 0 : Selects serial communication interface mode.

SCI0.SEMR register ← 20h
NFEN bit = 1 : Enables noise elimination function for RXDn input signal.
ABCS bit = 0 : Sets period of 16 cycles of reference clock as 1-bit period transfer rate.

SCI0.BRR register ← 2Fh
: Sets bit rate to 31,250 bps (when PCLKB = 48 MHz).

IPR214 register
IPR[3:0] bits ← 0000b : Sets interrupt priority level of SCI0 to disabled.

IER1A register
IEN7 bit ← 0 : Disables RXI0 interrupt requests.
IEN6 bit ← 0 : Disables TXI0 interrupt requests.

IR214 register
IR bit ← 0 : Clears RXI0 interrupt status flag.

IR215 register
IR bit ← 0 : Clears TXI0 interrupt status flag.

Figure 5.10 Initialization Processing 2

5.10.3 Main Function

Figures 5.11 and 5.12 are flowcharts of main function 1 and main function 2.

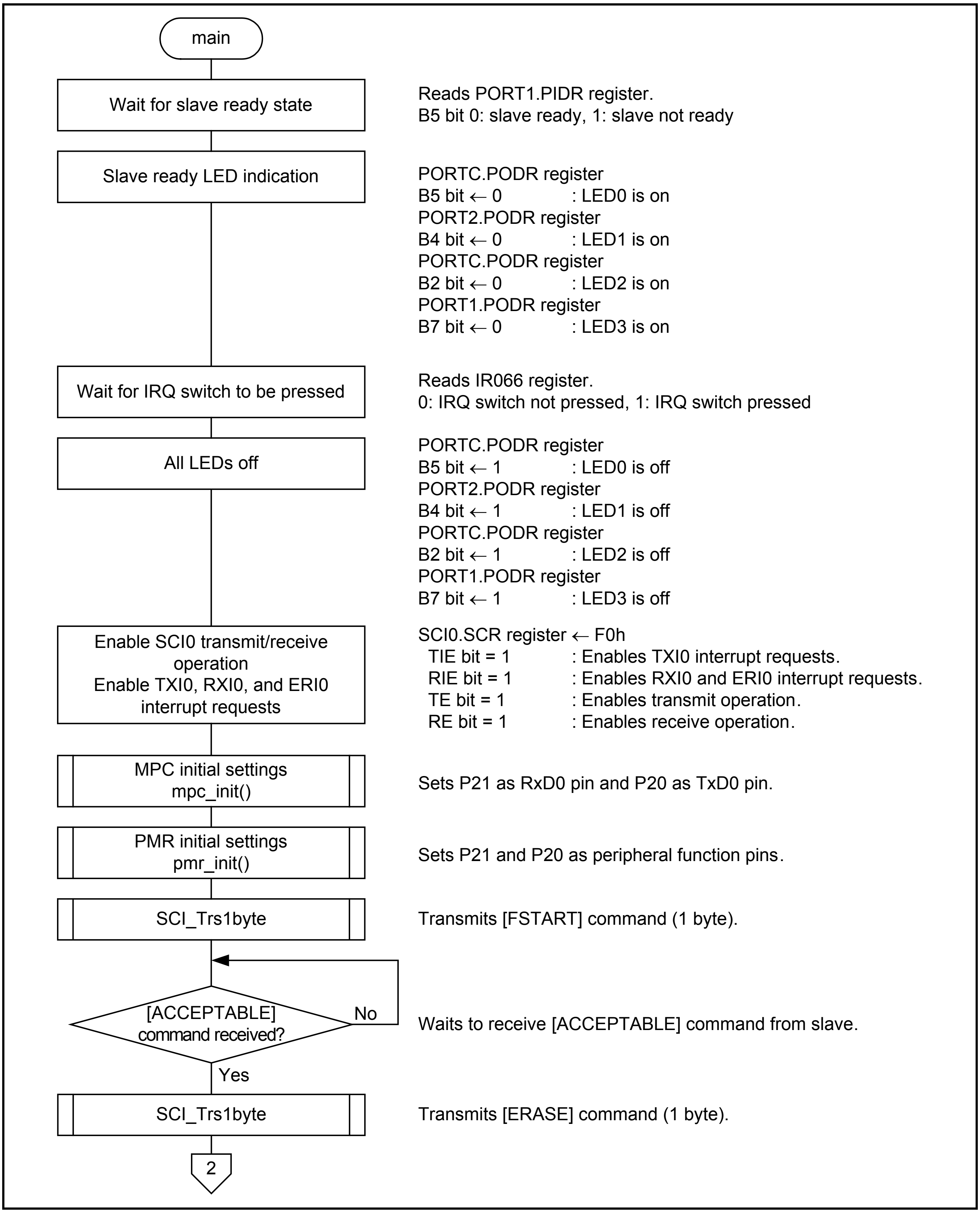


Figure 5.11 Main Function 1

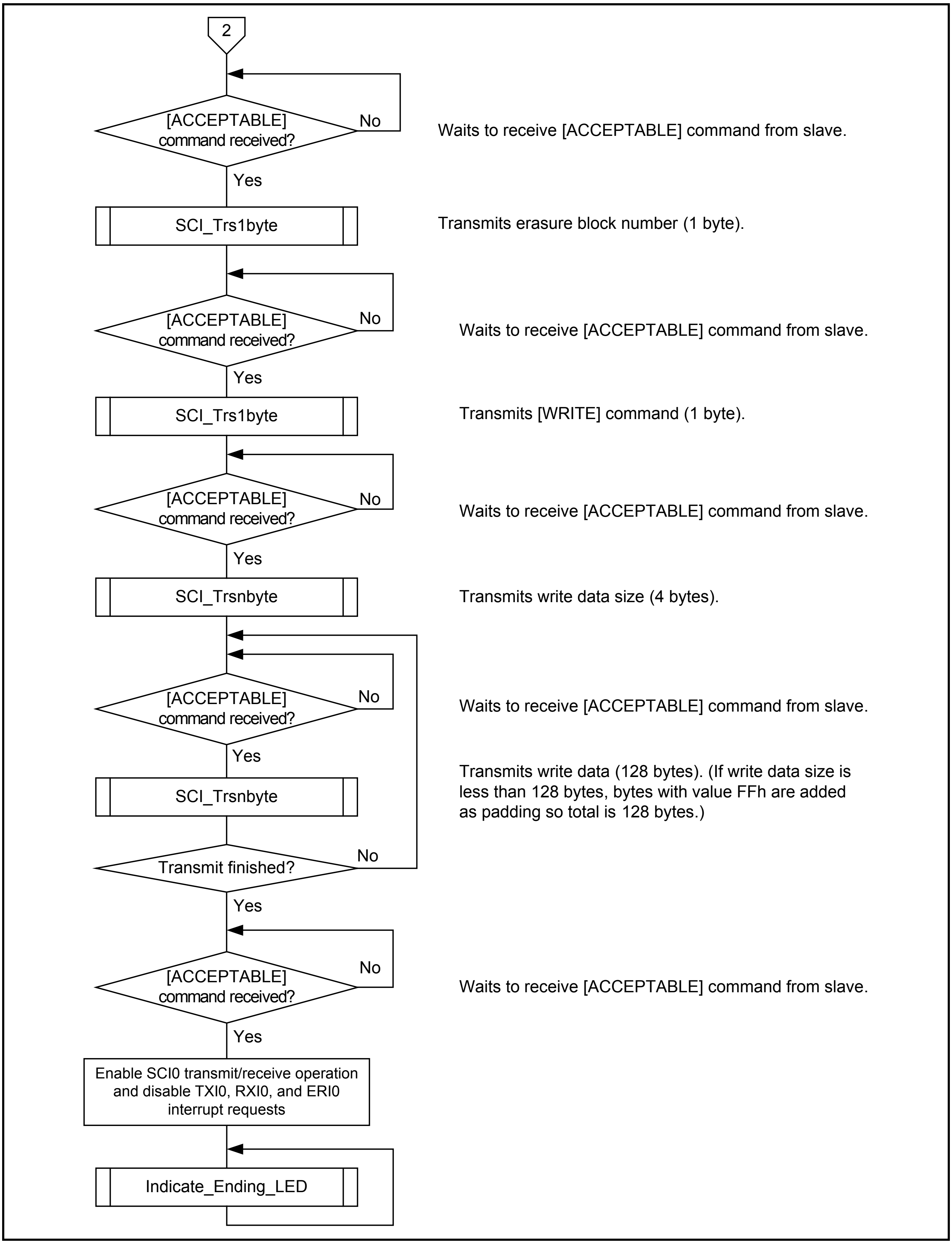


Figure 5.12 Main Function 2

5.10.4 Normal Processing Function

Figure 5.13 is a flowchart of the normal processing function.

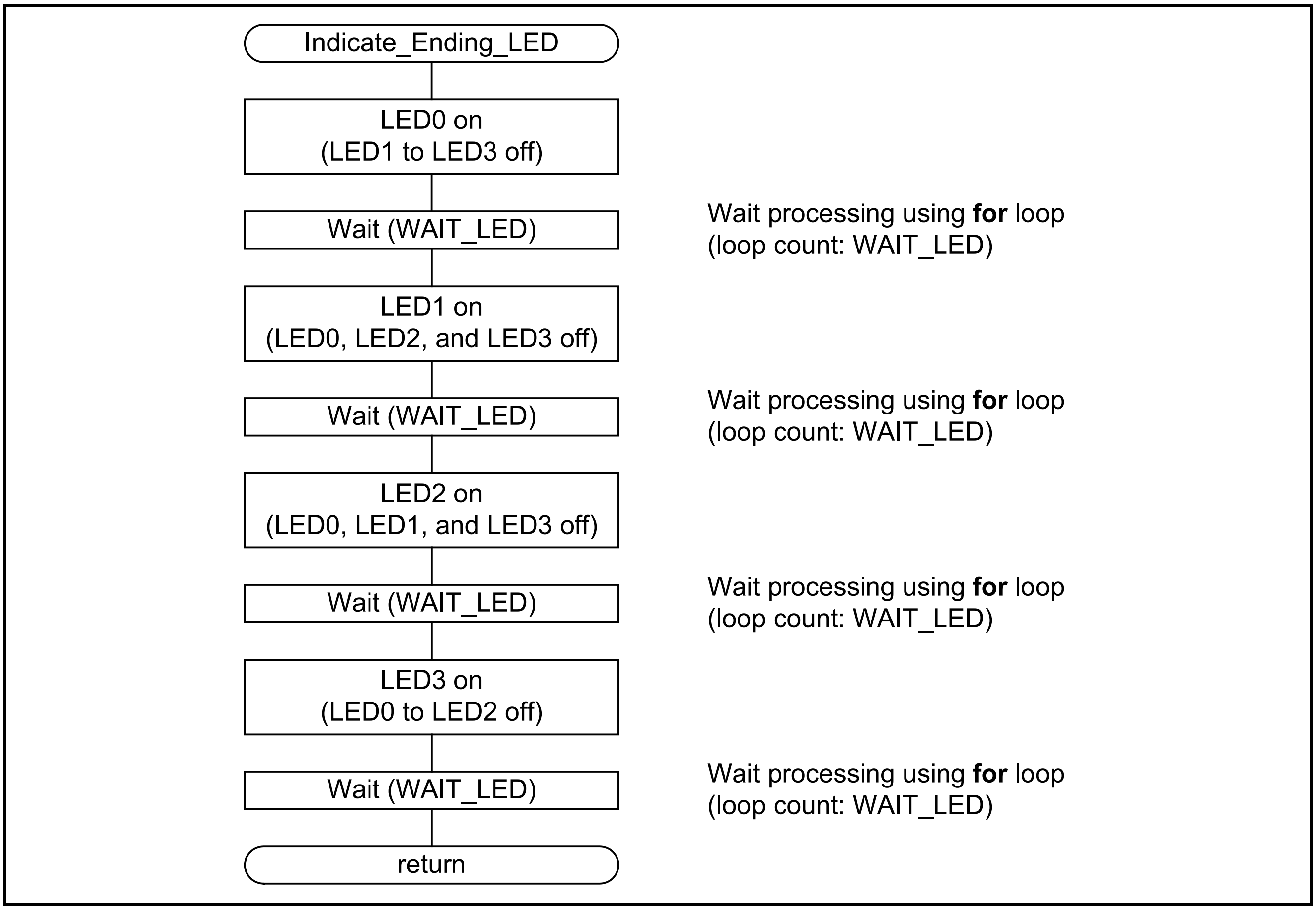


Figure 5.13 Normal Processing Function

5.10.5 1 Byte Data Receive Function

Figure 5.14 is a flowchart of the 1 byte data receive function.

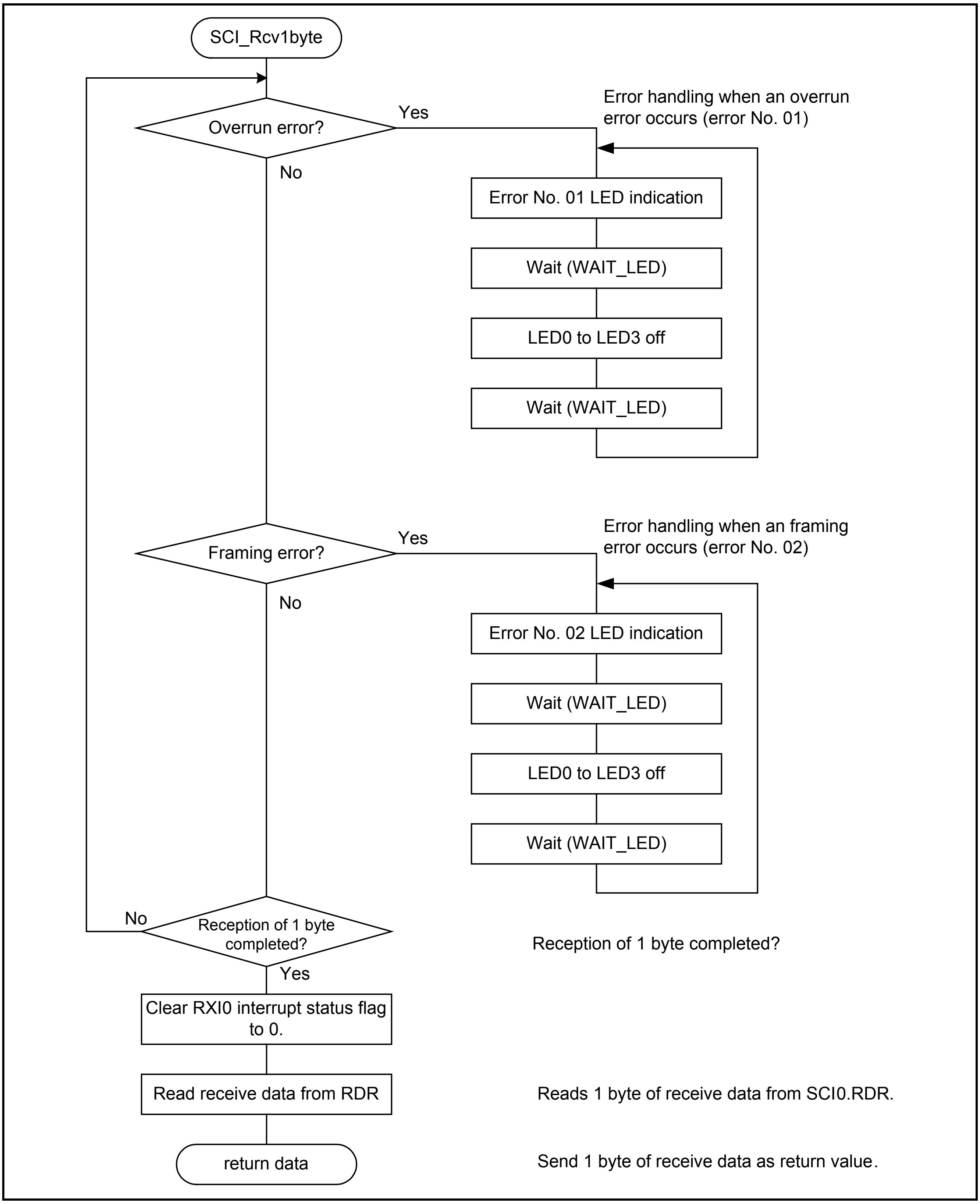


Figure 5.14 1 Byte Data Receive Function

5.10.6 1 Byte Data Transmit Function

Figure 5.15 is a flowchart of the 1 byte data transmit function.

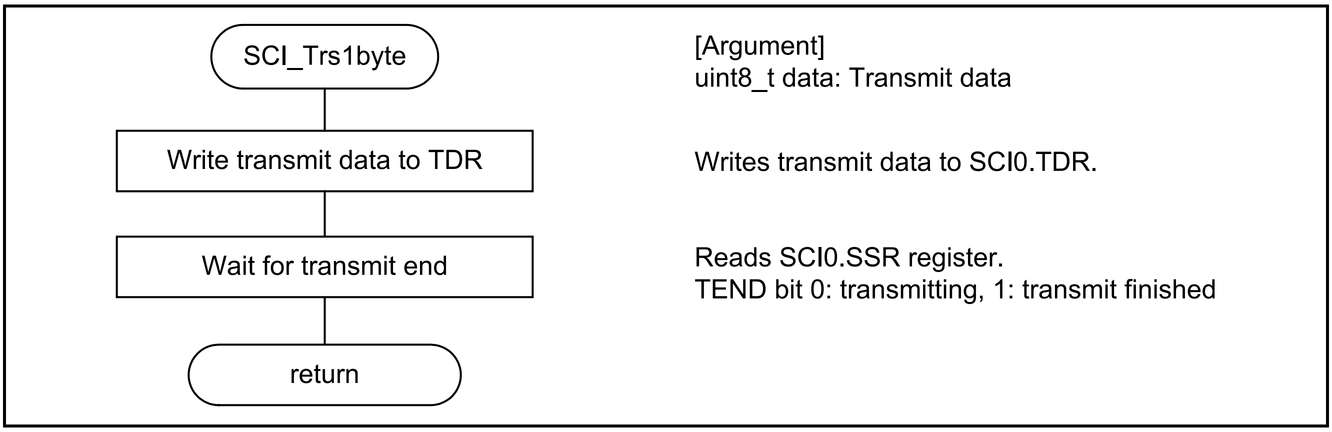


Figure 5.15 1 Byte Data Transmit Function

5.10.7 *n* Byte Data Transmit Function

Figure 5.16 is a flowchart of the *n* byte data transmit function.

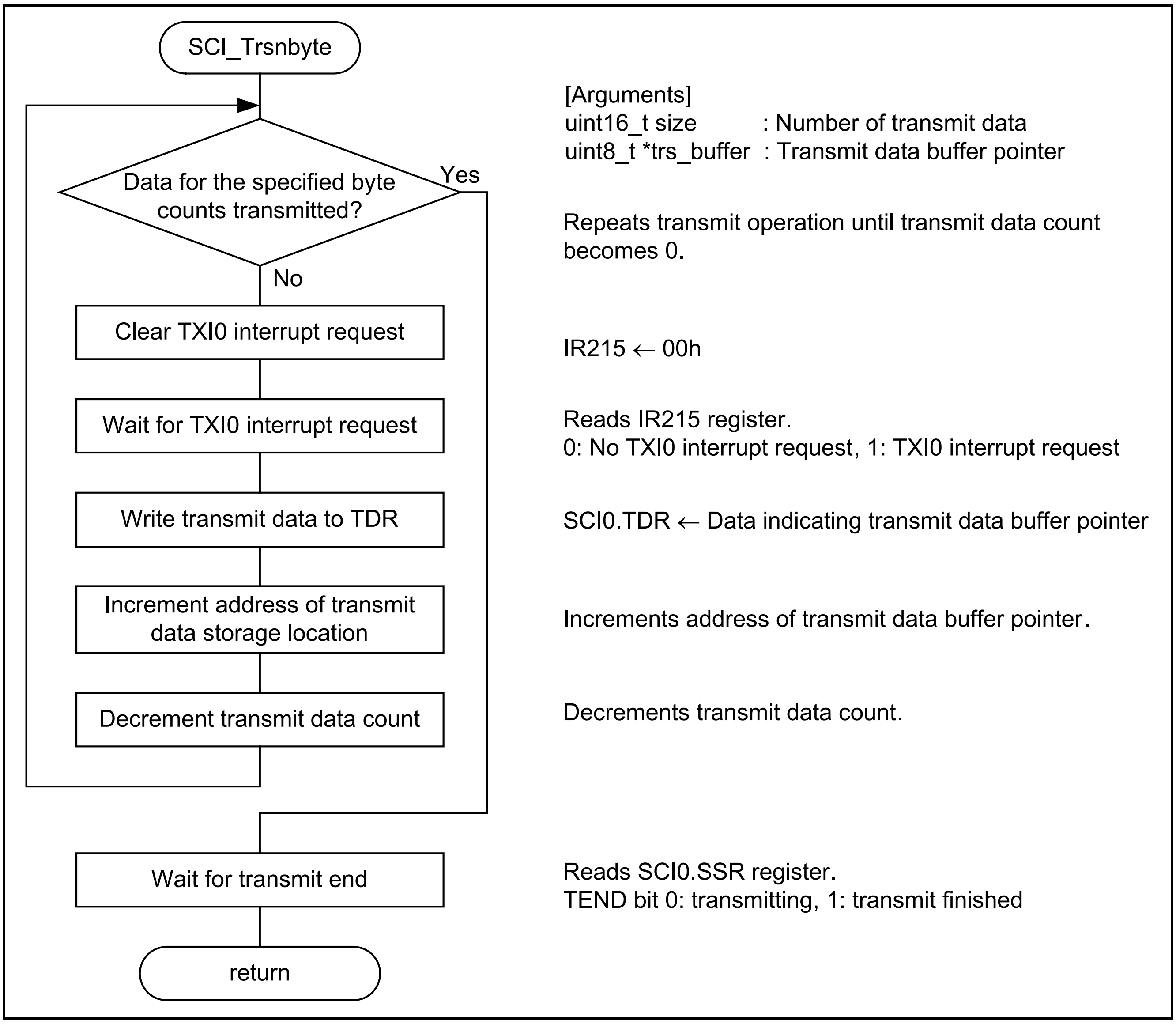


Figure 5.16 *n* Byte Data Transmit Function

5.10.8 MPC Initial Settings Function

Figure 5.17 is a flowchart of the MPC initial settings function.

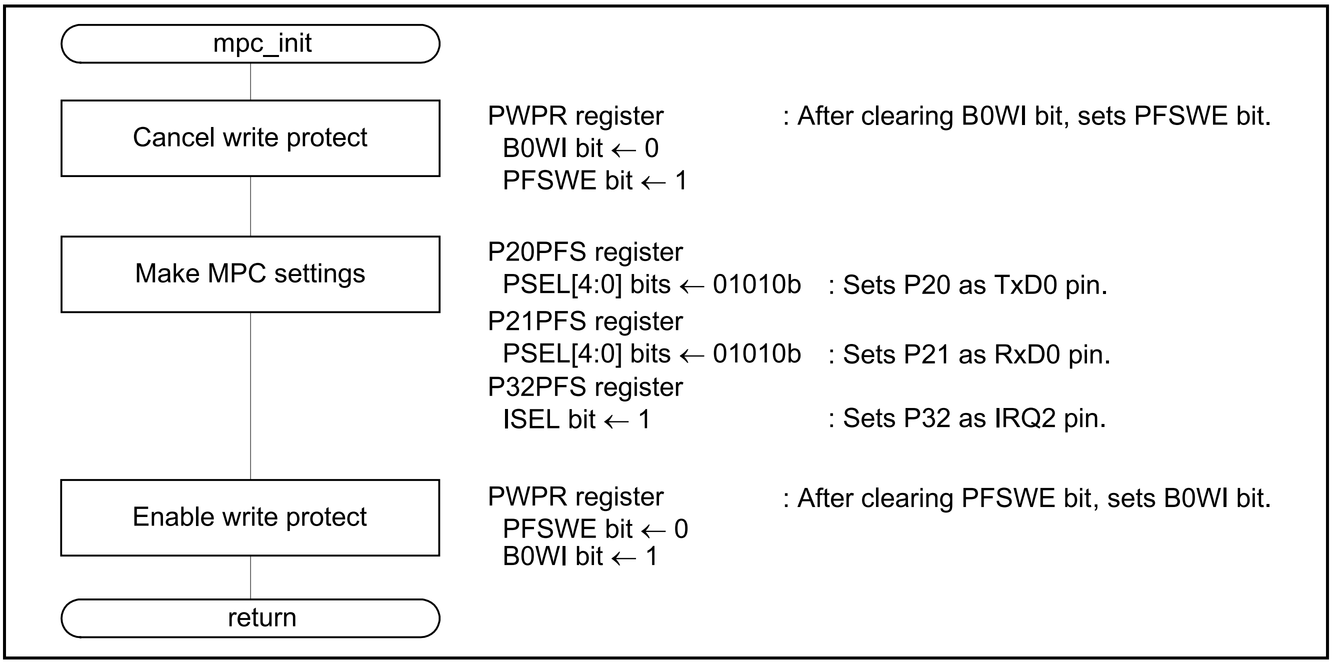


Figure 5.17 MPC Initial Settings Function

5.10.9 PMR Initial Settings Function

Figure 5.18 is a flowchart of the PMR initial settings function.

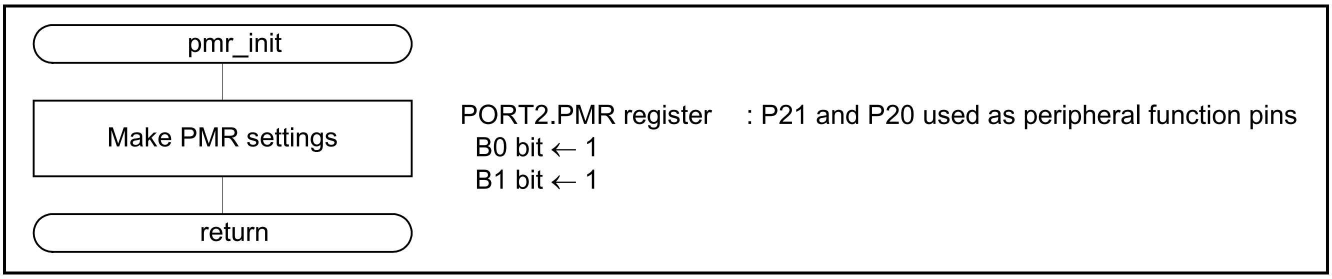


Figure 5.18 PMR Initial Settings Function

6. Usage Notes

6.1 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.

6.1.1 Using the Little-Endian Setting

For little-endian operation, select “little-endian data” for the endianness setting in the compiler options. Use the little-endian value for MDES shown in 5.5, Option Settings Memory.

6.1.2 Using the Big-Endian Setting

For big-endian operation, make settings as follows:

In the compiler options, select “big-endian data” for the endianness setting. Use the big-endian value for MDES shown in 5.5, Option Settings Memory.

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 (Slave) (R01AN1251EJ)
(The latest version can be downloaded from the Renesas Electronics website.)

RX630 Group Initial Setting Revision 1.00 (R01AN1004EJ0100)
(The latest version can be downloaded from the Renesas Electronics website.)

RX600 Series Simple Flash API for RX Revision 2.20 (R01AN0544EU)
(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	RX630 Group Application Note On-chip Flash Memory Reprogramming in Single-chip Mode via an UART Interface (Master)

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 15, 2014	—	First edition issued

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 accordance with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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.
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 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
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: +852-2886-9318, 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.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 3.0