RX63N Group, RX631 Group

Using the RIIC to Access the EEPROM

R01AN1925EJ0100
Rev. 1.00
May 7, 2014

Abstract

This application note describes using the I^2C bus interface (RIIC) in the RX63N Group, RX631 Group to access the EEPROM.

Products

RX63N Group, 176-Pin and 177-Pin Packages, ROM Capacities: 768 Kbytes to 2 Mbytes

RX63N Group, 144-Pin and 145-Pin Packages, ROM Capacities: 768 Kbytes to 2 Mbytes

RX63N Group, 100-Pin Package, ROM Capacities: 768 Kbytes to 2 Mbytes

RX631 Group, 176-Pin and 177-Pin Packages, ROM Capacities: 256 Kbytes to 2 Mbytes

RX631 Group, 144-Pin and 145-Pin Packages, ROM Capacities: 256 Kbytes to 2 Mbytes

RX631 Group, 100-Pin Package, ROM Capacities: 256 Kbytes to 2 Mbytes

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 document describes writing to and reading the EEPROM using the RIIC's single-master communication.

After release from the reset state, data (three patterns) to be displayed on the Debug LCD (hereinafter referred to as LCD) is written to the EEPROM, and the MCU waits for switch 3 to be pushed.

Each time switch 3 (SW3) is pushed, data for each pattern to be displayed on the LCD is read from the EEPROM, and displayed on the LCD.

- Bit rate: 100 kbps
- Data transfer size: 1 page units (16 bytes)
- Address format: 7-bit address format
- Operating modes: Master transmit mode, master receive mode

For details on the I^2C bus communication format, refer to the RX63N Group, RX631 Group User's Manual: Hardware, or the I^2C bus specification.

Table 1.1 lists the Peripheral Functions and Their Applications, and Figure 1.1 shows the Operation Overview.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
RIIC	Writes data to and reads data from the EEPROM
External pin interrupt (IRQ)	Switch to obtain data to be displayed on the LCD
I/O ports	I/O for displaying data on the LCD

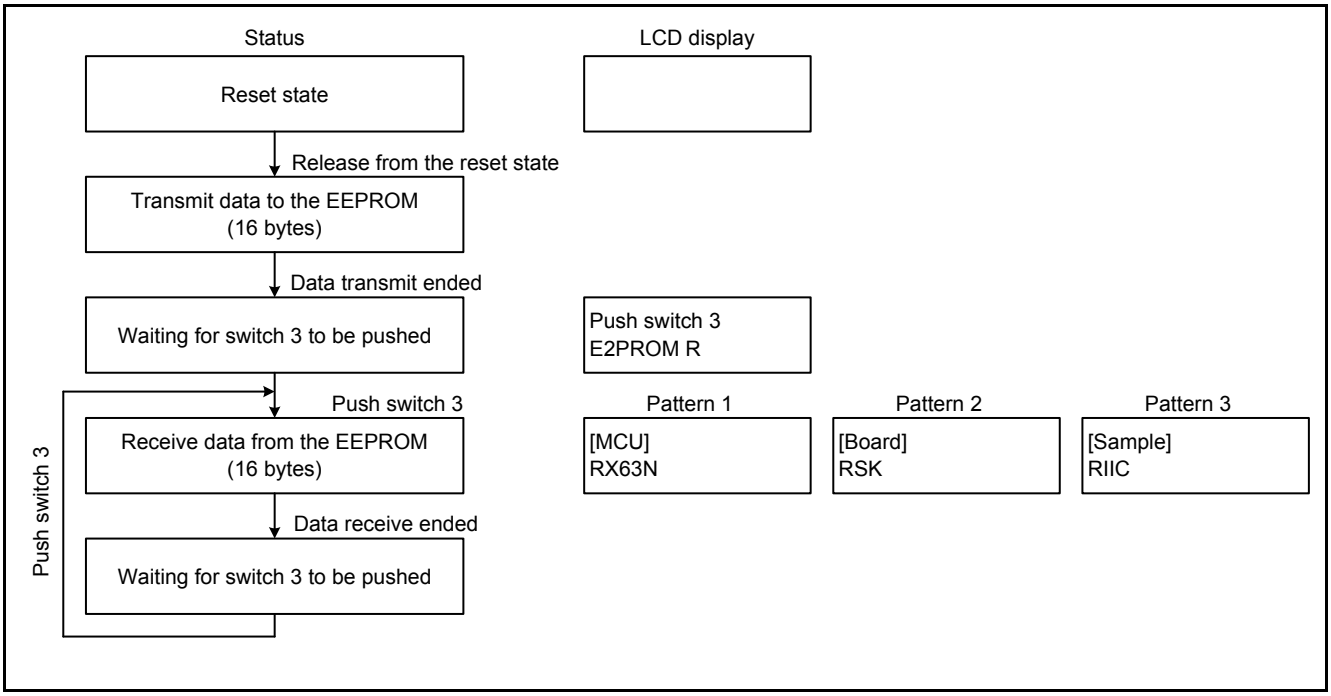


Figure 1.1 Operation Overview

2. Confirmed Operating Conditions

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

Table 2.1 Confirmed Operating Conditions

Item	Contents
MCU used	R5F563NBDDFC (RX63N Group)
Operating frequencies	• Main clock: 12 MHz • PLL clock: 192 MHz (main clock divided by 1 and multiplied by 16) • System clock (ICLK): 96 MHz (PLL clock divided by 2) • Peripheral module clock B (PCLKB): 48 MHz (PLL clock divided by 4)
Operating voltage	3.3 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=rx600 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo The integrated development environment default settings are used.
iodefine.h version	Version 1.6A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit+ for RX63N (product part number: R0K50563NC000BE)
EEPROM	Renesas R1EX24016ASAS0A

3. Reference Application Notes

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

- RX63N Group, RX631 Group Initial Setting Rev. 1.10 (R01AN1245EJ0110)
- RX63N Renesas Starter Kit Sample Code for Hi-performance Embedded Workshop (R01AN1395EG0100)

The initial setting functions and LCD output functions in the application notes above are used in the sample code in this application note. The revision number of the reference application note is current as of the issue date of this application note. However, the latest version is always recommended. Visit the Renesas Electronics Corporation website to download the latest version.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example.

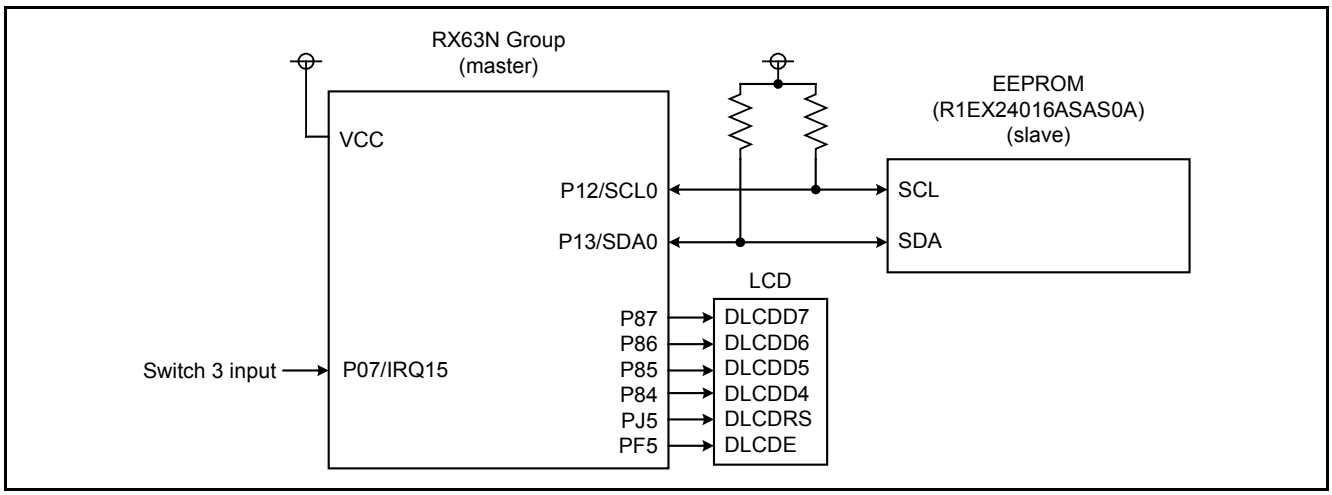


Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

This application note assumes the 176-pin package is used. When using packages with less than 176 pins, select the pins appropriate to the package used.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P12/SCL0	I/O	I/O pin for the serial clock
P13/SDA0	I/O	I/O pin for serial data
P87	Output	Output pin for data bit 7 of the LCD
P86	Output	Output pin for data bit 6 of the LCD
P85	Output	Output pin for data bit 5 of the LCD
P84	Output	Output pin for data bit 4 of the LCD
PJ5	Output	Output pin to enable the LCD
PF5	Output	Output pin to select the LCD register
P07/IRQ15	Input	Switch input to execute master reception

5. Software

After release from the reset state, the RIIC is initialized.

After the RIIC is initialized, master transmission is used to write 1 page (16 bytes) of data to the EEPROM. When the transmit processing is ended, the callback function is called. In the callback function, the RIIC status is obtained to see if master transmission was completed successfully. If master transmission is completed successfully, the next data is prepared to be transmitted. If a NACK response is received or if transmission was completed with an arbitration lost detected, the same data is prepared to be written. If a timeout occurs, "[ERROR] TIMEOUT" is displayed on the LCD, and loop processing is executed.

After 3 pages of data are written, "Push SW3" is displayed on the LCD.

When switch 3 is pushed, master reception is used to read 1 page of data from the EEPROM. When the receive processing is ended, the callback function is called. In the callback function, the RIIC status is obtained to see if master reception was completed successfully. If master reception was completed successfully, the data read from the EEPROM is displayed on the LCD. If an error occurred during master reception, information about the error is displayed on the LCD.

Settings for the peripheral functions are listed below.

RIIC

- Operating modes: Master transmit mode, master receive mode
- Address format: 7-bit address format
- Internal reference clock: PCLK/8
- Communication speed: Approx. 100 kbps [*1]
- Interrupts used: Transmit data empty interrupt (ICTXI)
  Transmit end interrupt (ICTEI)
  Receive data full interrupt (ICRXI)
  Stop condition detection interrupt (SPI)
  Arbitration-lost interrupt (ALI)
  Timeout interrupt (TMOI)

Note 1. The ICBRL.BRL[4:0] bits are set to 23, and the ICBRH.BRH[4:0] bits are set to 28.
If the SCL line rising time (tr) is 1000 ns, and the SCL line falling time (tf) is 300 ns, then the actual communication speed is calculated as follows:
= 1 ÷ { [(ICBRH.BRH[4:0] + 1) + (ICBRL.BRL[4:0] + 1)] ÷ Internal reference clock + tr + tf }
= 1 ÷ { [28 + 1) + (23 + 1)] ÷ (40 MHz ÷ 8) + 1000 (ns) + 300 (ns) }
= 98684.2 bps

5.1 Operation Overview

5.1.1 Writing Data to the EEPROM

(1) Starting master transmission
After confirming that the ICCR2.BBSY flag is 0 (bus free state), RIIC_BUSY (busy state) is set to the RIIC status variable (riic_status), the ICIER.TIE bit is set to 1 (ICTXI0 interrupt is enabled), and the ICCR2.ST bit is set to 1 (requests to issue a start condition).

(2) Issuing a start condition
When a start condition is issued, the ICCR2.BBSY flag becomes 1 (bus busy state). In addition, an ICTXI0 interrupt request is generated. In the ICTXI0 interrupt handling, the EEPROM storage address and W bit (0) are written to the ICDRT register.

(3) Transmitting data
When data is transferred from the ICDRT register to the ICDRS register, the ICTXI0 interrupt request is generated again. In the ICTXI0 interrupt handling, the write address for the EEPROM is written to the ICDRT register.

(4) Receiving an ACK or NACK
At the rising edge of the ninth bit on the SCL, the EEPROM outputs an ACK or NACK signal.
If an ACK signal is received, RIIC communication continues, and the ICTXI0 interrupt request is generated. In the ICTXI0 interrupt handling, data written to the EEPROM is written to the ICDRT register from the first byte.
If a NACK signal is received, RIIC communication is suspended, and the ICEEI0 interrupt request is generated. In the ICEEI0 interrupt handling, RIIC_NACK (NACK received) is set to the RIIC status variable, and the ICCR2.SP bit is set to 1 (requests to issue a stop condition).

(5) Writing the last data
When 1 page (16 bytes) of data written to the EEPROM has been written to the ICDRT register, the ICIER.TIE bit is set to 0 (ICTXI0 interrupt is disabled) and the ICIER.TEIE bit is set to 1 (ICTEI0 interrupt is enabled).

(6) End of data transmission
When the last data has been transmitted, the ICTEI0 interrupt is generated. In the ICTEI0 interrupt handling, the ICIER.TEIE bit is set to 0 (ICTEI0 interrupt is disabled) and the ICCR2.SP bit is set to 1.

(7) Issuing a stop condition
When a stop condition is issued, the ICEEI0 interrupt request is generated. In the ICEEI0 interrupt handling, if transmission was completed successfully, set RIIC_RDY (ready state) to the RIIC status variable, and call the callback function.
When a stop condition is issued after receiving a NACK (when the RIIC status variable is RIIC_NACK), do not set the RIIC status variable, and call the callback function.

Figure 5.1 shows the Timing Diagram When Writing Data to the EEPROM.

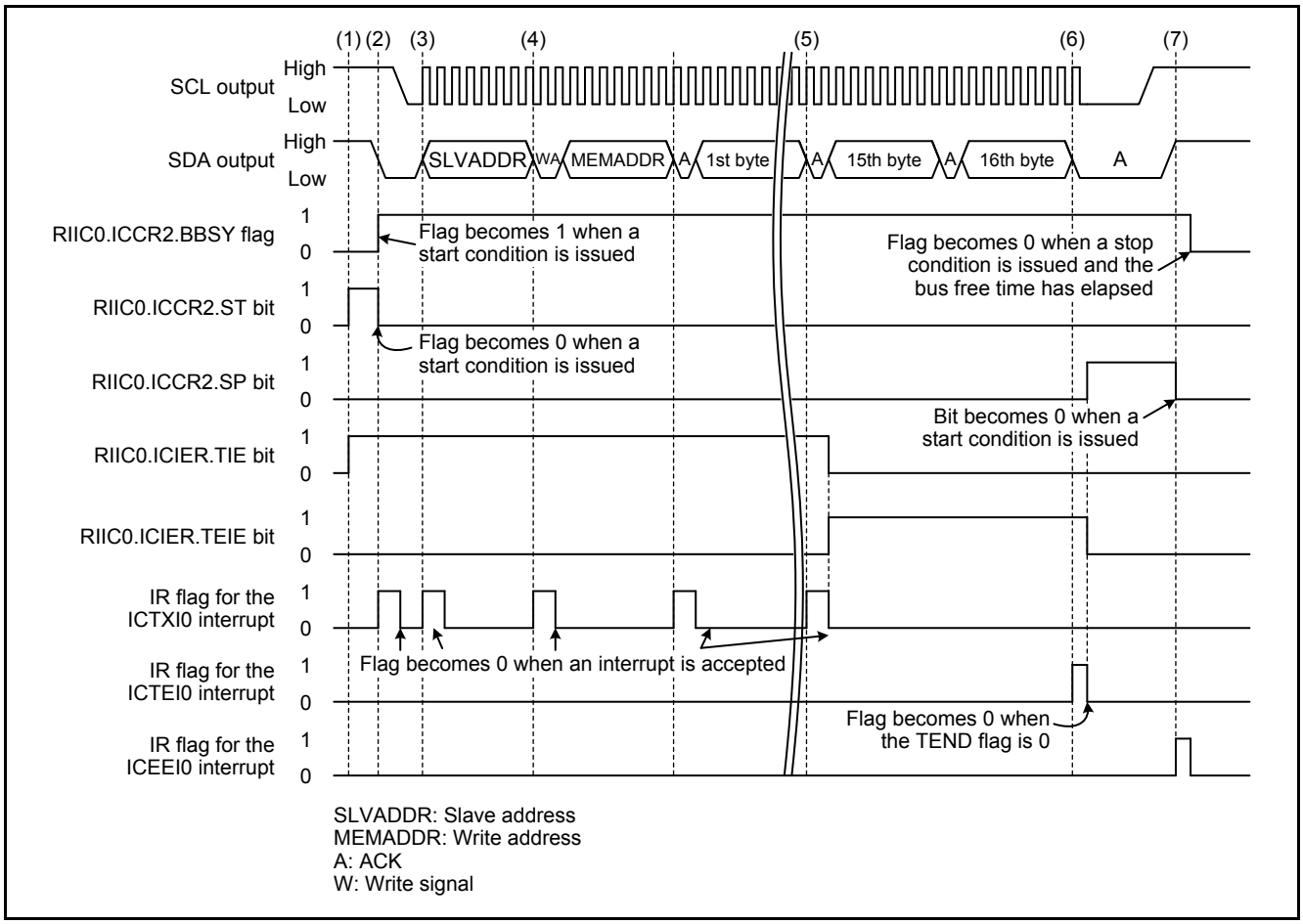


Figure 5.1 Timing Diagram When Writing Data to the EEPROM

5.1.2 Reading Data From the EEPROM

(1) Starting master transmission
When a falling edge is detected on the IRQ15 pin, if the ICCR2.BBSY flag is confirmed to be 0 (bus free state), RIIC_BUSY (busy state) is set to the RIIC status variable (riic_status), the ICIER.TIE bit is set to 1 (ICTXI0 interrupt is enabled), the ICIER.RIE bit is set to 1 (ICRXI0 interrupt is enabled), and the ICCR2.ST bit is set to 1 (requests to issue a start condition).

(2) Issuing a start condition
When a start condition is issued, the ICCR2.BBSY flag becomes 1 (bus busy state). In addition, the ICTXI0 interrupt request is generated. In the ICTXI0 interrupt handling, the slave address for the EEPROM and the W bit (0) are written to the ICDRT register.

(3) Transmitting the address
When data is transferred from the ICDRT register to the ICDRS register, the ICTXI0 interrupt request is generated again. In the ICTXI0 interrupt handling, the read address for the EEPROM is written to the ICDRT register, the ICIER.TIE bit is set to 0 (ICTXI0 interrupt is disabled), and the ICIER.TEIE bit is set to 1 (ICTEI0 interrupt is enabled).

(4) Receiving an ACK or NACK
On the rising edge of the ninth bit of the SCL, the EEPROM outputs an ACK signal or NACK signal.
If an ACK signal is received, RIIC communication continues.
If a NACK signal is received, RIIC communication is suspended, and the ICEEI0 interrupt request is generated. In the ICEEI0 interrupt handling, RIIC_NACK (NACK received) is set to the RIIC status variable, and the ICCR2.SP bit is set to 1 (requests to issue a stop condition).

(5) Starting master reception
When the read address been transmitted, the ICTEI0 interrupt request is generated. In the ICTEI0 interrupt handling, the ICIER.TEIE bit is set to 0 (ICTEI0 interrupt is disabled), the ICIER.TIE bit is set to 1 (ICTXI0 interrupt is enabled), and the ICCR2.RS bit is set to 1 (requests to issue a restart condition).

(6) Issuing a restart condition
When a restart condition is issued, the ICTXI0 interrupt request is generated. In the ICTXI0 interrupt handling, the slave address for the EEPROM and the R bit (1) are written to the ICDRT register.

(7) Receiving an ACK or NACK
On the rising edge of the ninth bit of the SCL, the EEPROM outputs an ACK signal or NACK signal.
If an ACK signal is received, the ICCR2.TRS bit becomes 0 (receive mode), and the ICRXI0 interrupt request is generated. In the ICRXI0 interrupt handling, a dummy read is performed on the ICDRR register.
If a NACK signal is received, RIIC communication is suspended, and the ICEEI0 interrupt request is generated. In the ICEEI0 interrupt handling, RIIC_NACK (NACK received) is set to the RIIC status variable, and the ICCR2.SP bit is set to 1 (requests to issue a stop condition).

(8) End of data reception
When all data has been received, the ICRXI0 interrupt request is generated. In addition, at the ninth bit of the SCL, the ICMR3.ACKBT bit value is output. In the ICRXI0 interrupt handling, the received data in the ICDDR register is read.

(9) Receiving the last data
When the last data has been received, after the ICCR2.SP bit is set to 1 (requests to issue a stop condition), the received data in the ICDRR register is read.

(10) Issuing a stop condition
When a stop condition is issued, the ICEEI0 interrupt request is generated. In the ICEEI0 interrupt handling, if data reception is completed successfully, RIIC_RDY (ready state) is set to the RIIC status variable, and the callback function is called.
When a stop condition is issued (when the RIIC status variable is RIIC_NACK) after a NACK is received, the callback function is called without the RIIC status variable being set.

Figure 5.2 shows the Timing Diagram for Master Reception.

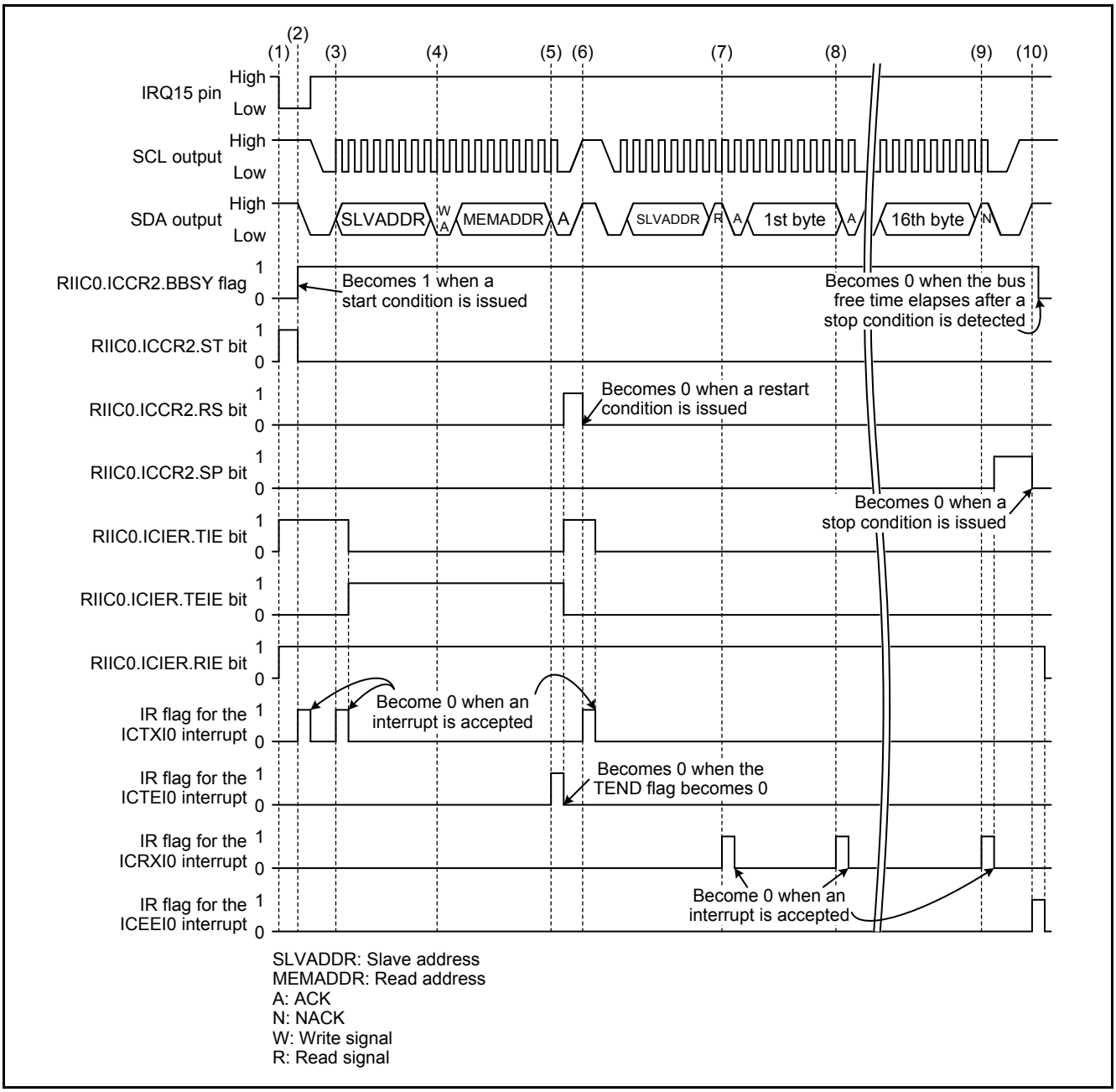


Figure 5.2 Timing Diagram for Master Reception

5.2 File Composition

Table 5.1 lists the Files Used in the Sample Code, Table 5.2 lists the Standard Include Files, and Table 5.3 lists the Functions and Setting Values for the Reference Application Notes. Files generated by the integrated development environment are not included in this table.

Table 5.1 Files Used in the Sample Code

File Name	Outline
main.c	Main processing
riic.c	Processing for writing to and reading the EEPROM
riic.h	Header file for riic.c

Table 5.2 Standard Include Files

File Name	Outline
stdbool.h	This file defines the macros associated with the Boolean and its value.
stdint.h	This file defines the macros declaring the integer type with the specified width.
machine.h	This file defines the types of intrinsic functions for the RX Family.

Table 5.3 Functions and Setting Values for the Reference Application Notes

File Name	Function	Setting Value
r_init_stop_module.c	R_INIT_StopModule()	—
r_init_stop_module.h	—	The module-stop state is canceled for the DMAC/DTC, EXDMAC, RAM0, and RAM1.
r_init_non_existent_port.c	R_INIT_NonExistentPort()	—
r_init_non_existent_port.h	—	Set to 176-pin package
r_init_clock.c	R_INIT_Clock()	—
r_init_clock.h	—	Sub-clock not used
lcd.c	Init_LCD() Display_LCD()	—
lcd.h	—	—
rskrx63ndef.h	—	—

5.3 Option-Setting Memory

Table 5.4 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.4 Option-Setting Memory Configured in the Sample Code

Symbol	Addresses	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	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.4 Constants

Table 5.5 to Table 5.7 list the constants used in the sample code.

Table 5.5 Constants Used in the Sample Code (main.c)

Constant Name	Setting Value	Contents
EEPROM_ADDR	0x50	EEPROM device address
EEPROM_PAGE_SIZE	16	EEPROM page size
EEPROM_RW_SIZE	16	Size for reading from and writing to the EEPROM
DISP_PTN_NUM	3	Number of patterns for displayed text
LCD_LINE_SIZE	8	Number of characters on one line of the LCD
MSG_TIMEOUT_ERROR	"[ERROR] TIMEOUT "	Message when a timeout error occurs
MSG_AL_ERROR	"[ERROR] AL "	Message when arbitration lost occurs
MSG_NACK_ERROR	"[ERROR] NACK "	Message when a NACK is received
SW3_REQ	IR(ICU, IRQ15)	IR flag for switch 3 (IRQ15)
SW_ON	1	Switch 3 is detected as being pushed
SW_OFF	0	Switch 3 is not detected as being pushed

Table 5.6 Constants Used in the Sample Code (riic.h)

Constant Name	Setting Value	Contents
RIIC_RDY	0x00	RIIC status (ready state)
RIIC_BUSY	0x01	RIIC status (busy state)
RIIC_NACK	0x02	RIIC status (NACK received)
RIIC_AL	0x03	RIIC status (arbitration lost occurred)
RIIC_TIMEOUT	0x04	RIIC status (timeout occurred)
RIIC_OK	0x00	RIIC processing completed successfully
RIIC_FAIL	0x01	RIIC processing failed
RIIC_PARAM_ERR	0x02	Parameter error
NULL	0x00000000	NULL is specified

Table 5.7 Constants Used in the Sample Code (riic.c)

Constant Name	Setting Value	Contents
RIIC_W_BIT	0xFE	Masked data for the write bit setting
RIIC_R_BIT	0x01	Data for the read bit setting
RIIC_WRITE_SLVADDR	1	Slave address size (in bytes)
RIIC_WRITE_MEMADDR	1	Memory address size (in bytes)
RIIC_WRITE_HEADDER	RIIC_WRITE_SLVADDR + RIIC_WRITE_MEMADDR	Specified address size
RIIC_MD_INIT	0	RIIC operating mode (initialization)
RIIC_MD_WRITE	1	RIIC operating mode (data write mode)
RIIC_MD_READ_ADDR	2	RIIC operating mode (read address setting mode)
RIIC_MD_READ_DATA	3	RIIC operating mode (data read mode)
RIIC_CNT_SLVADDR	0	Determine the slave address transmission
RIIC_CNT_MEMADDR	1	Determine the memory address transmission

5.5 Variables

Table 5.8 and Table 5.9 list the static variables, and Table 5.10 list the const Variable.

Table 5.8 static Variables (main.c)

Type	Variable Name	Contents	Functions Used
uint8_t	read_buf[16]	Storage buffer for data read from the EEPROM	main cb_read_end
uint8_t	wpt	Write pointer	Main cb_write_end
uint8_t	rpt	Read pointer	Main cb_read_end

Table 5.9 static Variables (riic.c)

Type	Variable Name	Contents	Functions Used
uint8_t	riic_status	RIIC status	RIIC_Init RIIC_Write RIIC_Read Excep_RIIC0_EEI0 RIIC_GetStatus
uint8_t	riic_slv_addr	Storage area for the EEPROM slave address	RIIC_Write RIIC_Read Excep_RIIC0_TXI0 Excep_RIIC0_TEI0
uint8_t	riic_mem_addr	Storage area for the EEPROM memory address	RIIC_Write RIIC_Read Excep_RIIC0_TXI0
uint8_t	p_riic_rx_buf	Pointer for the storage buffer for data read from the EEPROM	RIIC_Read Excep_RIIC0_RXI0
uint8_t	p_riic_tx_buf	Pointer for the storage buffer for data written to the EEPROM	RIIC_Write Excep_RIIC0_TXI0
uint8_t	riic_tx_cnt	RIIC transmit counter	RIIC_Write RIIC_Read Excep_RIIC0_TXI0 Excep_RIIC0_TEI0
uint8_t	riic_tx_size	RIIC transmit size	RIIC_Write RIIC_Read Excep_RIIC0_TXI0 Excep_RIIC0_TEI0
uint8_t	riic_rx_cnt	RIIC receive counter	RIIC_Write RIIC_Read Excep_RIIC0_RXI0
uint8_t	riic_rx_size	RIIC receive size	RIIC_Write RIIC_Read Excep_RIIC0_RXI0
uint8_t	riic_rw_mode	RIIC R/W mode	RIIC_Init RIIC_Write RIIC_Read Excep_RIIC0_TXI0 Excep_RIIC0_TEI0 Excep_RIIC0_EEI0
CallBackFunc	pcb_riic_end	Storage area for the callback function pointer	RIIC_Write RIIC_Read Excep_RIIC0_EEI0

Table 5.10 const Variable (main.c)

Type	Variable Name	Contents	Function Used
static const uint8_t	write_buf[DISP_PTN_NUM][16]	Example of text written to the EEPROM Pattern 1: "[MCU] RX63N " Pattern 2: "[Board] RSK " Pattern 3: "[Sample]RIIC "	main

5.6 Functions

Table 5.11 and Table 5.12 list the functions.

Table 5.11 Functions in main.c

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
IRQ_Init	IRQ initialization
cb_write_end	Callback function after writing data to the EEPROM
cb_read_end	Callback function after reading data from the EEPROM
Display_LCD_All	Processing to display two lines of text on the LCD
Init_LCD	LCD initialization
Display_LCD	Processing to display text on the LCD

Table 5.12 Functions in riic.c

Function Name	Outline
RIIC_Init	RIIC initialization
RIIC_Write	Processing to start writing data to the EEPROM
RIIC_Read	Processing to start reading data from the EEPROM
Excep_RIIC0_TXI0	ICTXI0 interrupt handling
Excep_RIIC0_RXI0	ICRXI0 interrupt handling
Excep_RIIC0_TEI0	ICTEI0 interrupt handling
Excep_RIIC0_EEI0	ICEEI0 interrupt handling
RIIC_GetStatus	Obtain RIIC status
RIIC_proc_end	Processing when writing data to or reading data from the EEPROM has ended

5.7 Function Specifications

The following tables list the specifications for the functions in the sample code.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After the initial settings, data to be displayed on the LCD is written to the EEPROM. After the data is written, when a falling edge is detected on the IRQ15 pin, data to be displayed on the LCD is read from the EEPROM.
Arguments	None
Return Value	None

port_init	
Outline	Port initialization
Header	None
Declaration	static void port_init(void)
Description	This function initializes the ports.
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	This function configures settings to enter the module-stop state.
Arguments	None
Return Value	None
Remark	Transition to the module-stop state is not performed in the sample code. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	This function initializes port direction registers for ports that do not exist in products with less than 176 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR and 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 more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	This function initializes the clocks.
Arguments	None
Return Value	None
Remark	In the sample code, the PLL clock is selected as the system clock, and the sub-clock is not used. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.

peripheral_init	
Outline	Peripheral function initialization
Header	None
Declaration	static void peripheral_init(void)
Description	This function initializes the peripheral functions being used.
Arguments	None
Return Value	None

IRQ_Init	
Outline	IRQ initialization
Header	None
Declaration	static void IRQ_Init(void)
Description	This function initializes IRQ15.
Arguments	None
Return Value	None

cb_write_end	
Outline	Callback function after writing data to the EEPROM
Header	None
Declaration	static void cb_write_end (void)
Description	After obtaining the RIIC status, if data has been written to the EEPROM, the write pointer is updated. If a timeout occurred, a timeout error message is displayed on the LCD.
Arguments	None
Return Value	None

cb_read_end	
Outline	Callback function after reading data to the EEPROM
Header	None
Declaration	static void cb_read_end (void)
Description	After obtaining the RIIC status, if data has been read from the EEPROM, the data read from the EEPROM is displayed on the LCD, and the read pointer is updated. If an error occurred, an error message is displayed on the LCD.
Arguments	None
Return Value	None

Display_LCD_All	
Outline	Processing to display two lines of text on the LCD
Header	None
Declaration	static void Display_LCD_All(uint8_t *p_disp)
Description	This function displays the specified text on the LCD.
Arguments	uint8_t* p_disp: Text displayed
Return Value	None

Init_LCD	
Outline	LCD initialization
Header	lcd.h
Declaration	void Init_LCD(void)
Description	This function initializes the LCD.
Arguments	None
Return Value	None
Remarks	This function uses the Renesas StarterKit sample code for the RX63N High-performance Embedded Workshop.

Display_LCD	
Outline	Processing for displaying the LCD
Header	lcd.h
Declaration	void Display_LCD(uint8_t position, uint8_t* string)
Description	This function displays the specified text on the LCD.
Arguments	uint8_t position: Position to display text uint8_t* string: Text to be displayed
Return Value	None
Remarks	This function uses the Renesas StarterKit sample code for the RX63N High-performance Embedded Workshop.

RIIC_Init

Outline	RIIC initialization
Header	riic.h
Declaration	void RIIC_Init (void)
Description	This function initializes the RIIC.
Arguments	None
Return Value	None

RIIC_Write

Outline	Processing to start writing data to the EEPROM
Header	riic.h
Declaration	uint8_t RIIC_Write (uint8_t slv_addr, uint8_t mem_addr, uint8_t * pbuf, uint8_t num, CallBackFunc pcb)
Description	After setting information for writing to the EEPROM, a start condition is output and the data starts being written to the EEPROM.
Arguments	uint8_t slv_addr: EEPROM slave address uint8_t mem_addr: EEPROM write address uint8_t* pbuf: Storage pointer for data to be written to the EEPROM uint8_t num: Size of data to be written to the EEPROM CallBackFunc pcb: Callback function when the write operation has ended
Return Value	RIIC_OK: Processing to start writing data to the EEPROM was successful. RIIC_FAIL: Processing to start writing data to the EEPROM failed.

RIIC_Read

Outline	Processing to start reading data from the EEPROM
Header	riic.h
Declaration	uint8_t RIIC_Read (uint8_t slv_addr, uint8_t mem_addr, uint8_t * pbuf, uint8_t num, CallBackFunc pcb)
Description	After setting information for reading from the EEPROM is set, a start condition is output and the data starts being read from the EEPROM.
Arguments	uint8_t slv_addr: EEPROM slave address uint8_t mem_addr: EEPROM read address uint8_t* pbuf: Storage pointer for data to be read from the EEPROM uint8_t num: Size of data to be read from the EEPROM CallBackFunc pcb: Callback function when the read operation has ended
Return Value	RIIC_OK: Processing to start reading data from the EEPROM was successful. RIIC_FAIL: Processing to start reading data from the EEPROM failed. RIIC_PARAM_ERR: The argument value is undefined.

Excep_RIIC0_TXI0

Outline	ICTXI0 interrupt handling
Header	None
Declaration	static void Excep_RIIC0_TXI0(void)
Description	This function performs ICTXI0 interrupt handling. Transmit data is written to the ICDRT register. After the last data is written, the ICTXI0 interrupt request is disabled. In addition, the ICTEI0 interrupt request is enabled when master transmission is performed.
Arguments	None
Return Value	None

Excep_RIIC0_RXI0	
Outline	ICRXI0 interrupt handling
Header	None
Declaration	static void Excep_RIIC0_RXI0(void)
Description	This function performs ICRXI0 interrupt handling. Receive data is read from the ICDRR register. When the last data is received, a stop condition is output, and then the receive data is read.
Arguments	None
Return Value	None

Excep_RIIC0_TEI0	
Outline	ICTEI0 interrupt handling
Header	None
Declaration	static void Excep_RIIC0_TEI0(void)
Description	This function performs ICTEI0 interrupt handling. When ICTEI0 interrupt handling is performed in the EEPROM write processing, after the ICTEI0 interrupt request is disabled, a stop condition is output. When ICTEI0 interrupt handling is performed in the EEPROM read processing, after the ICTEI0 interrupt request is disabled, a restart condition is output.
Arguments	None
Return Value	None

Excep_RIIC0_EEI0	
Outline	ICEEI0 interrupt handling
Header	None
Declaration	static void Excep_RIIC0_EEI0(void)
Description	This function performs ICEEI0 interrupt handling. When the source of the interrupt request is stop condition detection without NACK reception, RIIC_RDY is set as the RIIC status variable, and the callback function is called. When the sources of the interrupt request are stop condition detection and NACK reception, the RIIC status variable is not changed, and the callback function is called. When the source of the interrupt request is arbitration lost occurring, RIIC_AL is set as the RIIC status variable, and the callback function is called. When the source of the interrupt request is a timeout, RIIC_TIMEOUT is set as the RIIC status variable, and the callback function is called. When the source of the interrupt request is NACK reception, RIIC_NACK is set as the RIIC status variable, and the stop condition is output.
Arguments	None
Return Value	None

RIIC_GetStatus	
Outline	Obtain RIIC status
Header	riic.h
Declaration	uint8_t RIIC_GetStatus (void)
Description	This function returns the RIIC status.
Arguments	None
Return Value	RIIC_RDY: Ready state RIIC_BUSY: Busy state RIIC_NACK: NACK received RIIC_AL: Arbitration lost occurred RIIC_TIMEOUT: Timeout occurred

RIIC_proc_end	
Outline	Processing when writing data to or reading data from the EEPROM has ended
Header	None
Declaration	static void RIIC_proc_end(void)
Description	All RIIC0 interrupt requests are disabled, and bits ICMR3.RDRFS, ICMR3.ACKBT, and ICMR3.WAIT are initialized.
Arguments	None
Return Value	None

5.8 Flowcharts

5.8.1 Main Processing

Figure 5.3 shows the Main Processing.

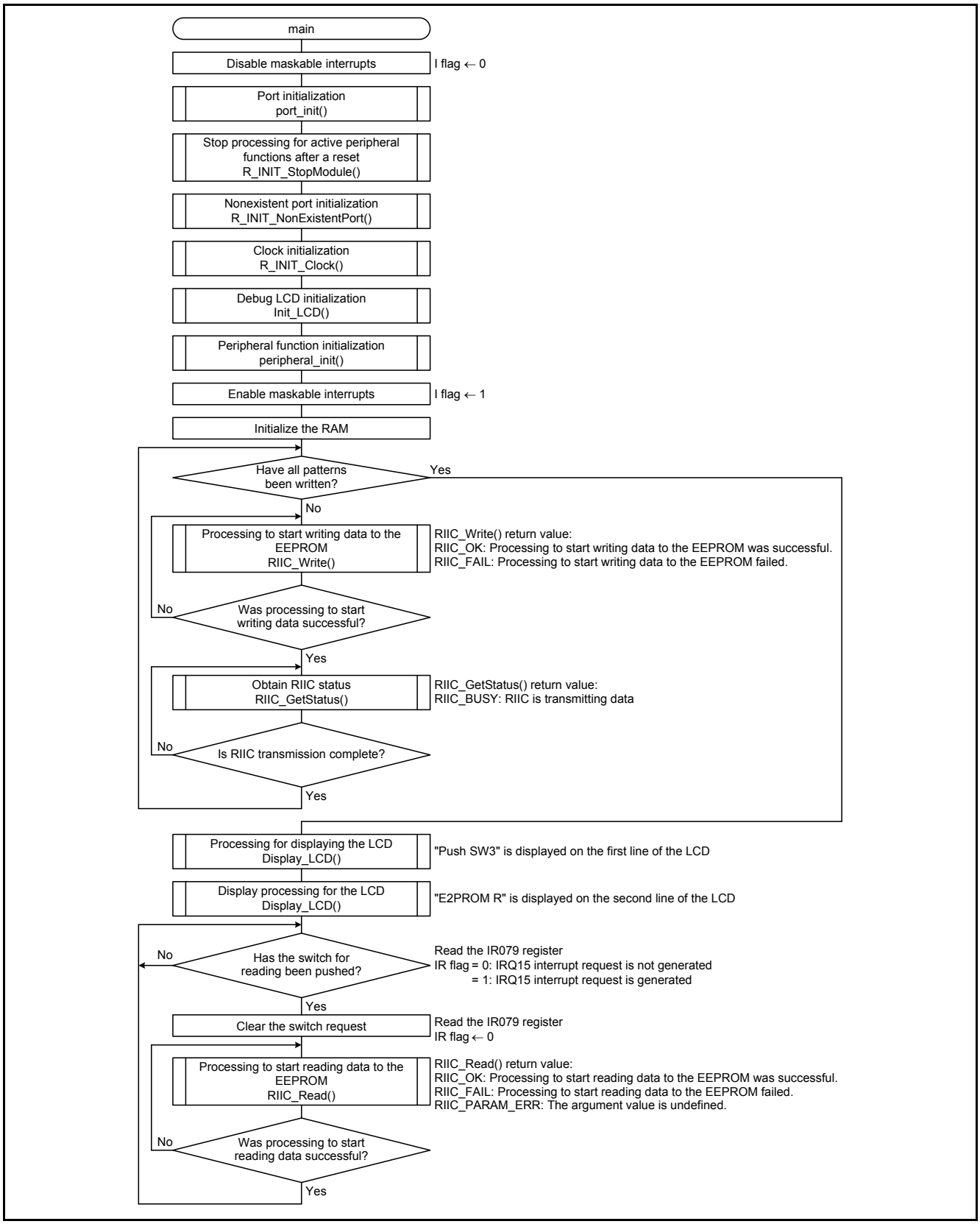


Figure 5.3 Main Processing

5.8.2 Port Initialization

Figure 5.4 shows the Port Initialization.

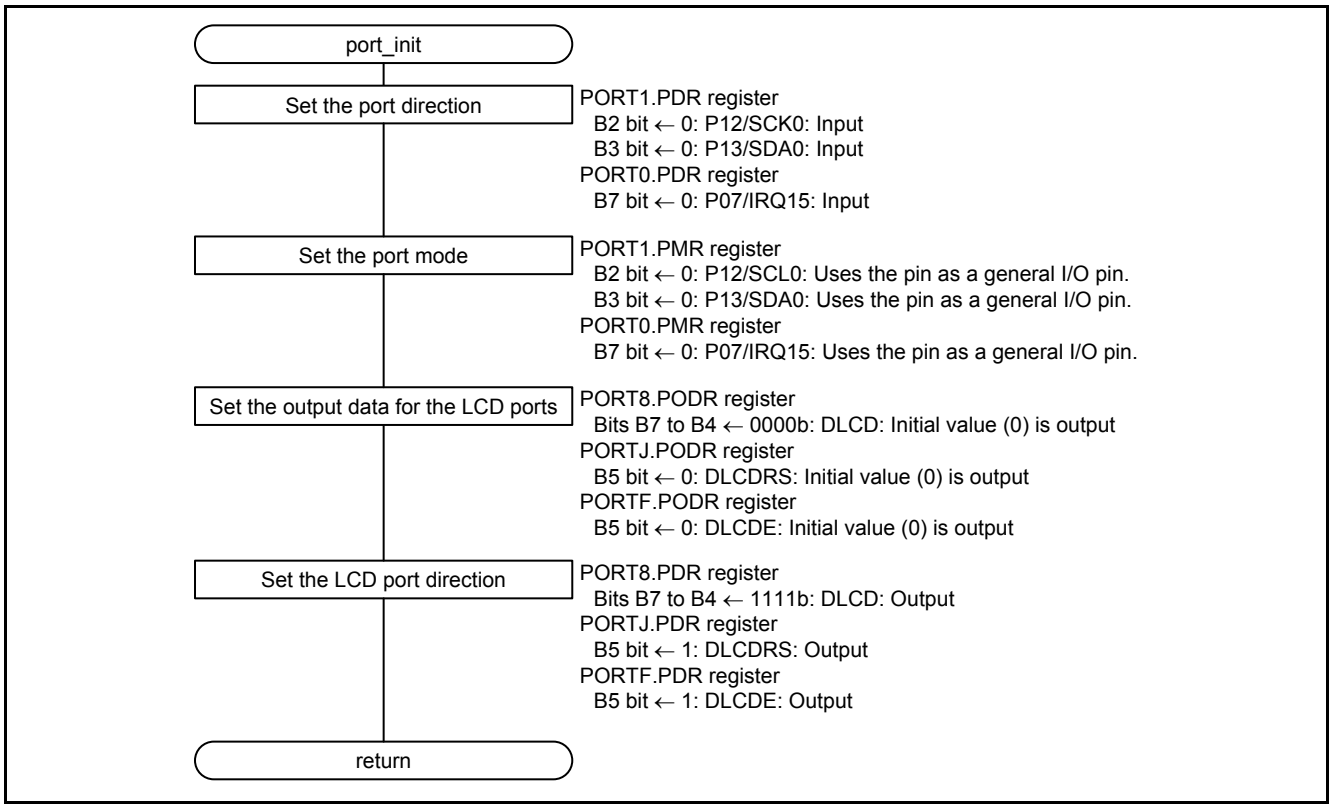


Figure 5.4 Port Initialization

5.8.3 Peripheral Function Initialization

Figure 5.5 shows the Peripheral Function Initialization.

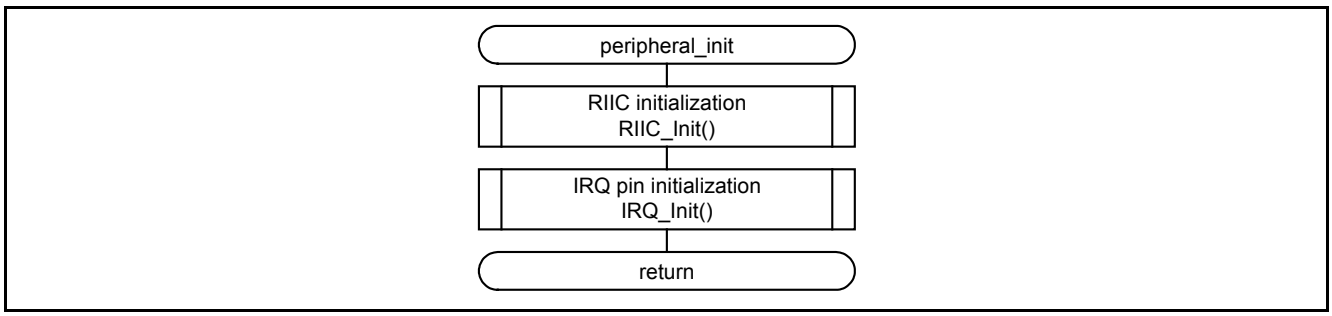


Figure 5.5 Peripheral Function Initialization

5.8.4 IRQ Initialization

Figure 5.6 shows the IRQ Initialization.

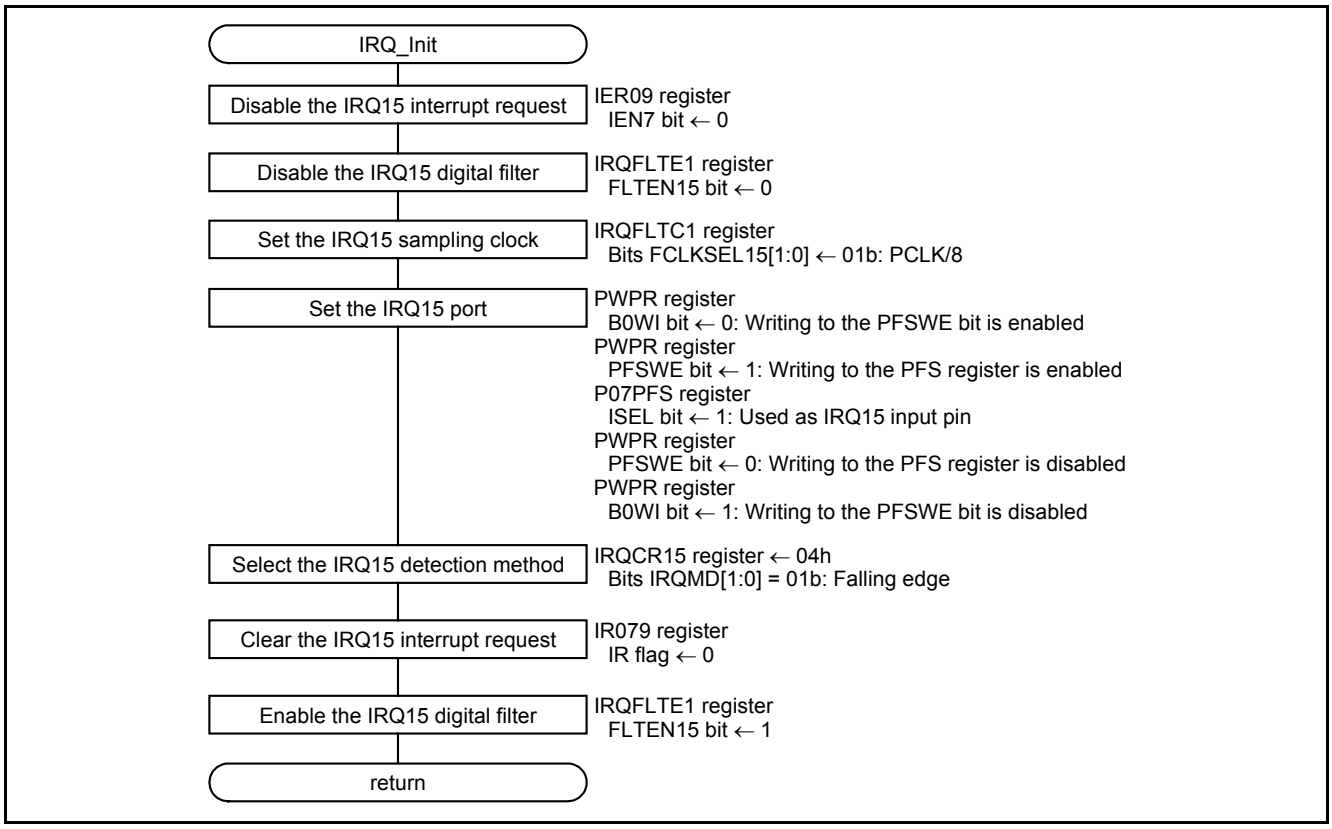


Figure 5.6 IRQ Initialization

5.8.5 Callback Function After Writing Data to the EEPROM

Figure 5.7 shows the Callback Function After Writing Data to the EEPROM.

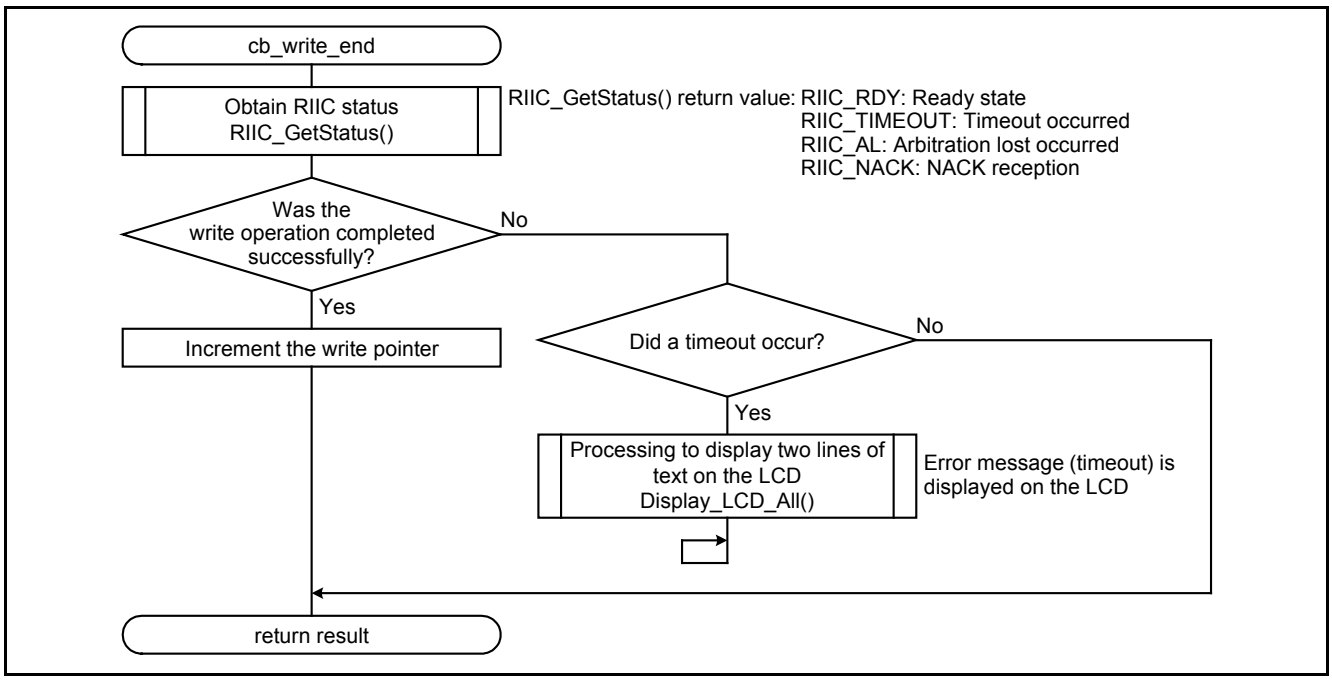


Figure 5.7 Callback Function After Writing Data to the EEPROM

5.8.6 Callback Function After Reading Data From the EEPROM

Figure 5.8 shows the Callback Function After Reading Data From the EEPROM.

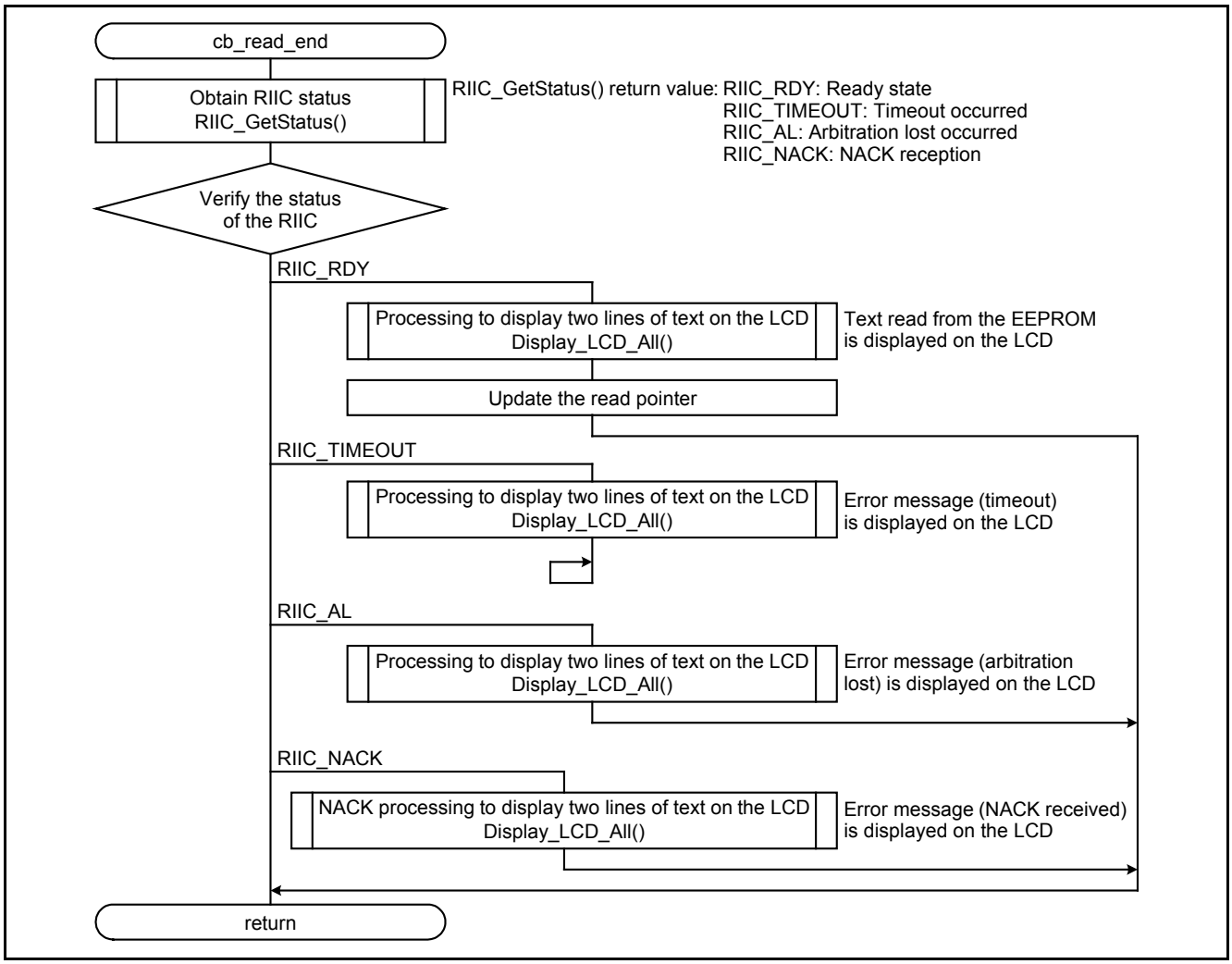


Figure 5.8 Callback Function After Reading Data From the EEPROM

5.8.7 Processing to Display Two Lines of Text on the LCD

Figure 5.9 shows Processing to Display Two Lines of Text on the LCD.

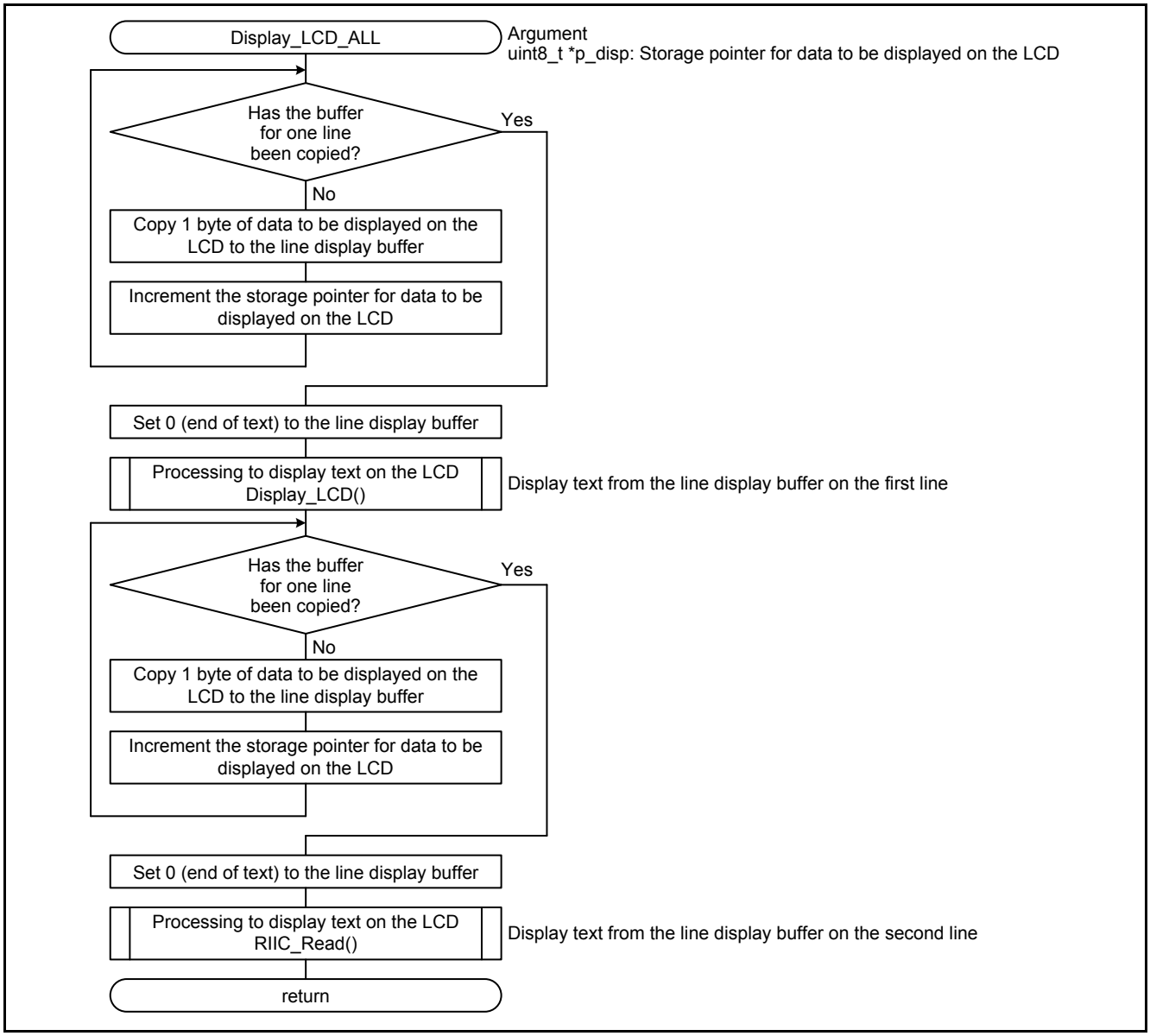
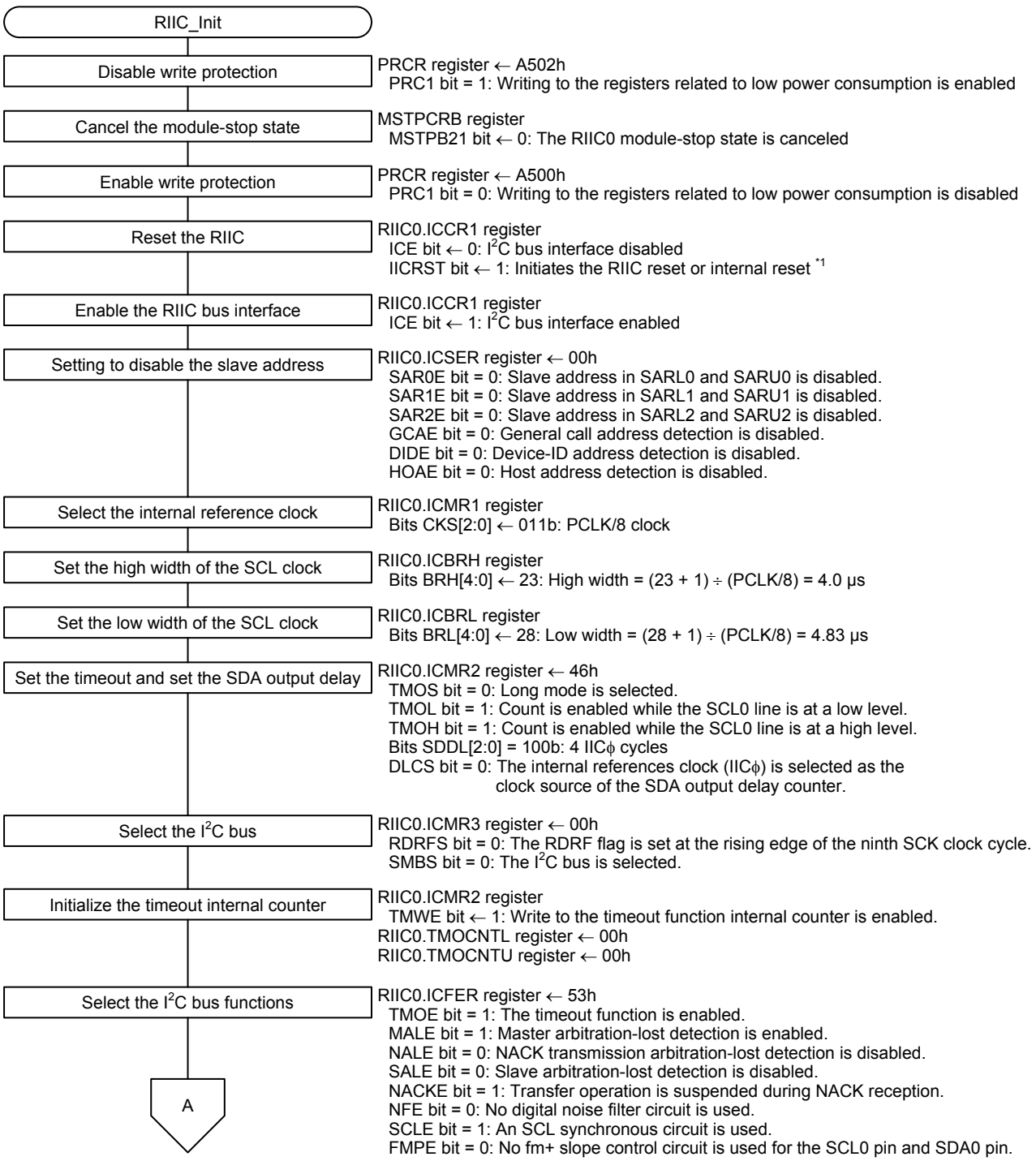


Figure 5.9 Processing to Display Two Lines of Text on the LCD

5.8.8 RIIC Initialization

Figure 5.10 and Figure 5.11 show RIIC initialization.



Note 1. When the ICCR1.IICRST bit is set to 1 while the ICCR1.ICE bit is 0, the RIIC enters the reset state.

Figure 5.10 RIIC Initialization (1/2)

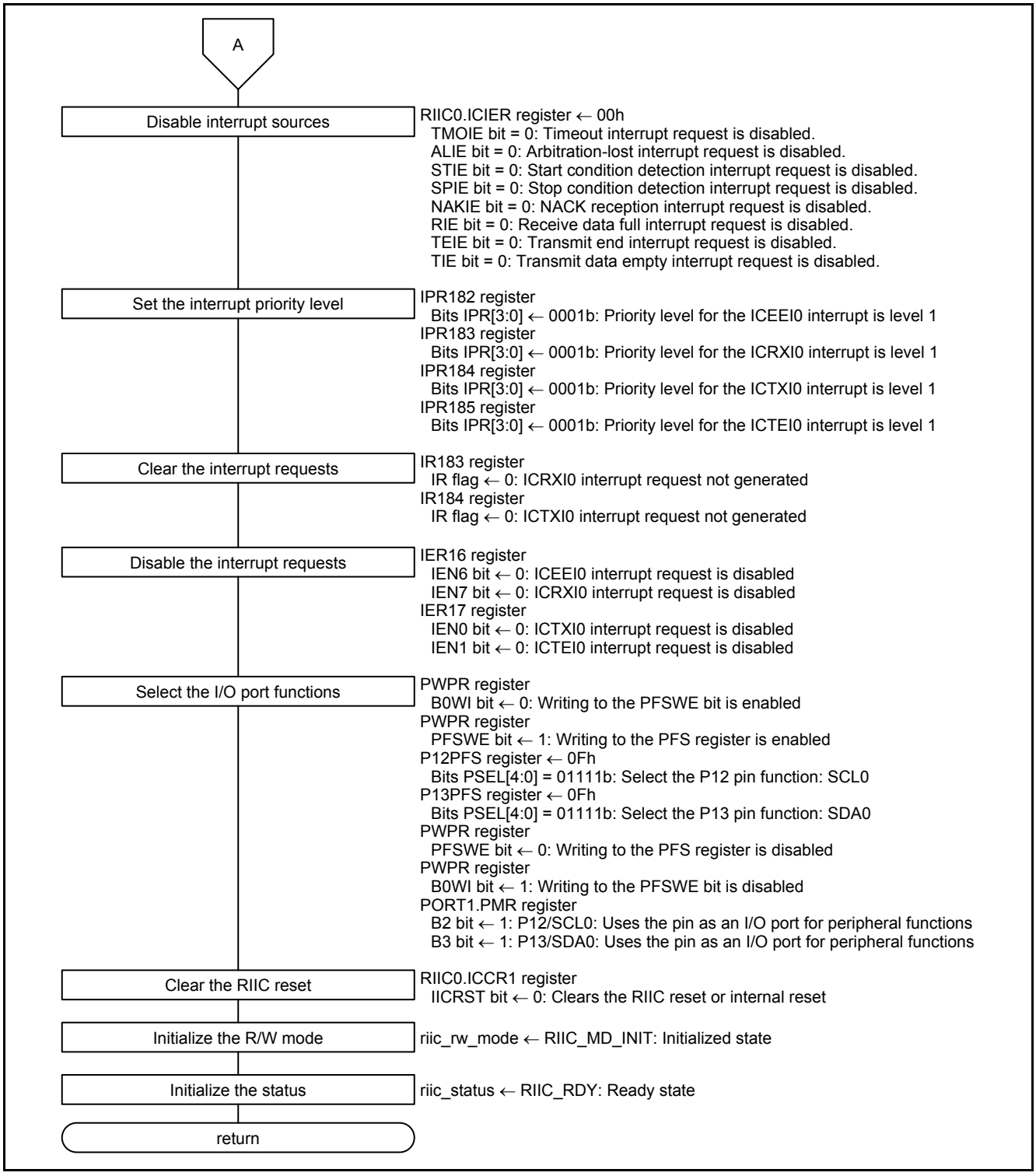


Figure 5.11 RIIC Initialization (2/2)

5.8.9 Processing to Start Writing Data to the EEPROM

Figure 5.12 shows Processing to Start Writing Data to the EEPROM.

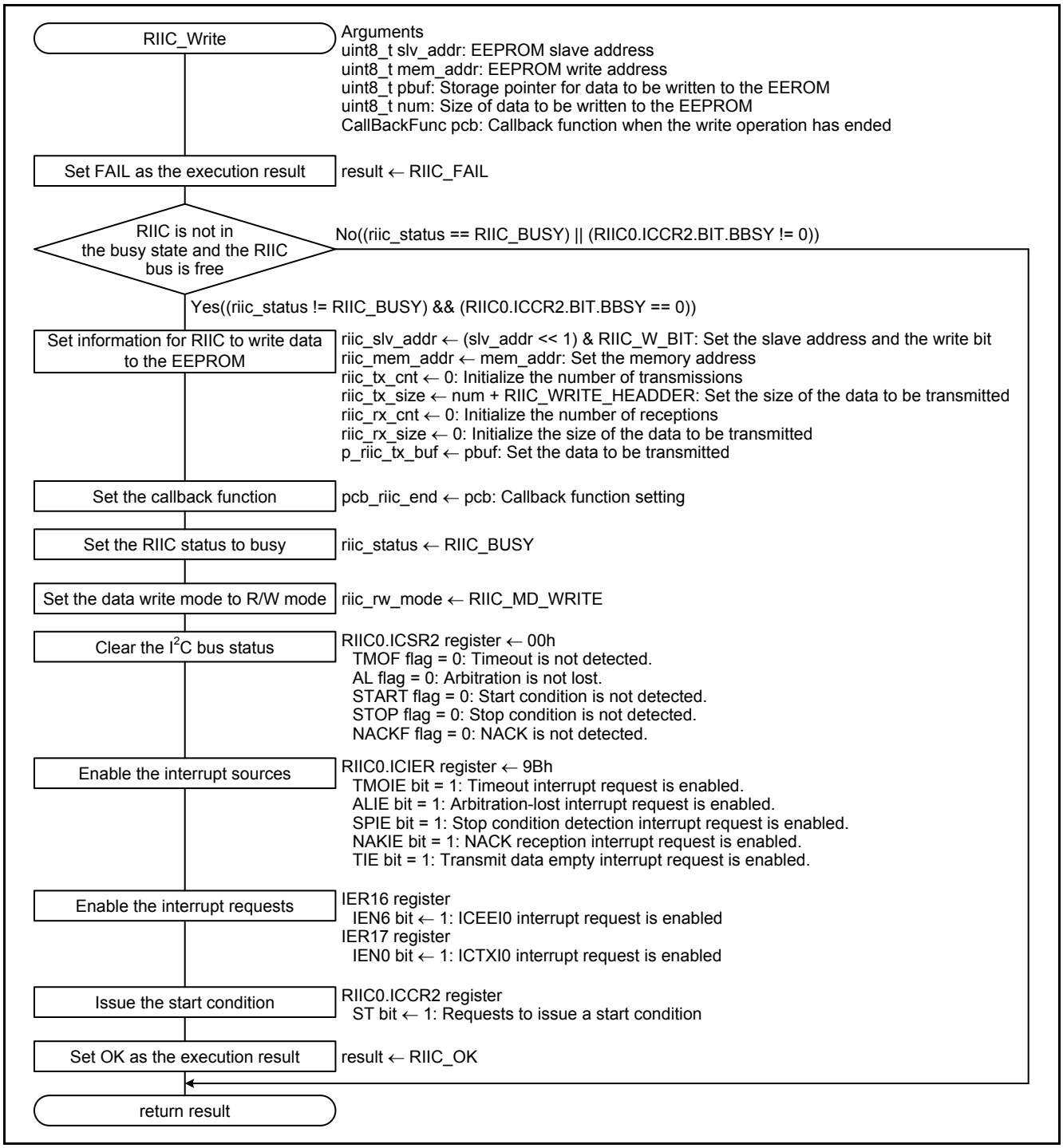


Figure 5.12 Processing to Start Writing Data to the EEPROM

5.8.10 Processing to Start Reading Data From the EEPROM

Figure 5.13 shows Processing to Start Reading Data From the EEPROM.

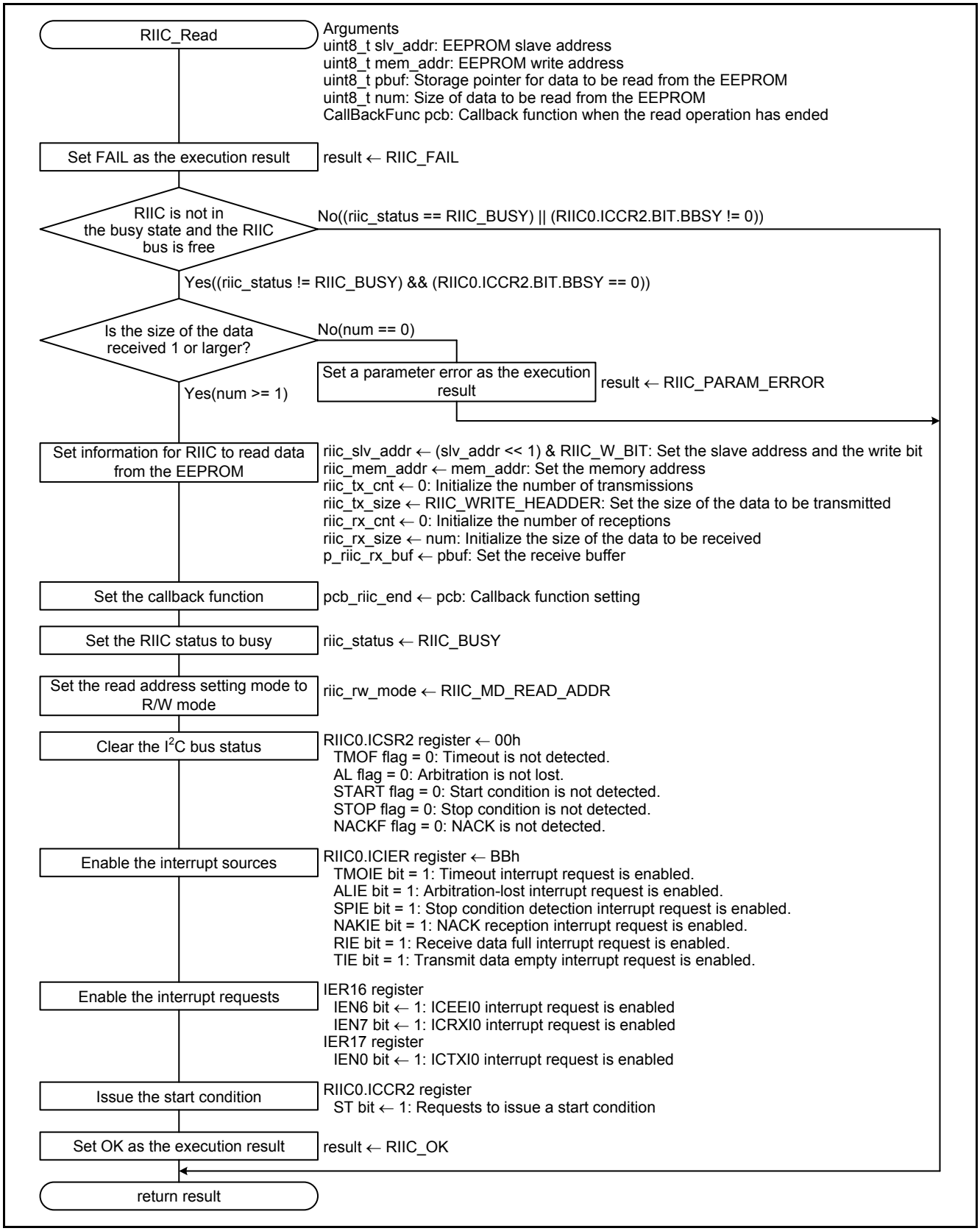


Figure 5.13 Processing to Start Reading Data From the EEPROM

5.8.11 ICTXI0 Interrupt Handling

Figure 5.14 shows ICTXI0 Interrupt Handling.

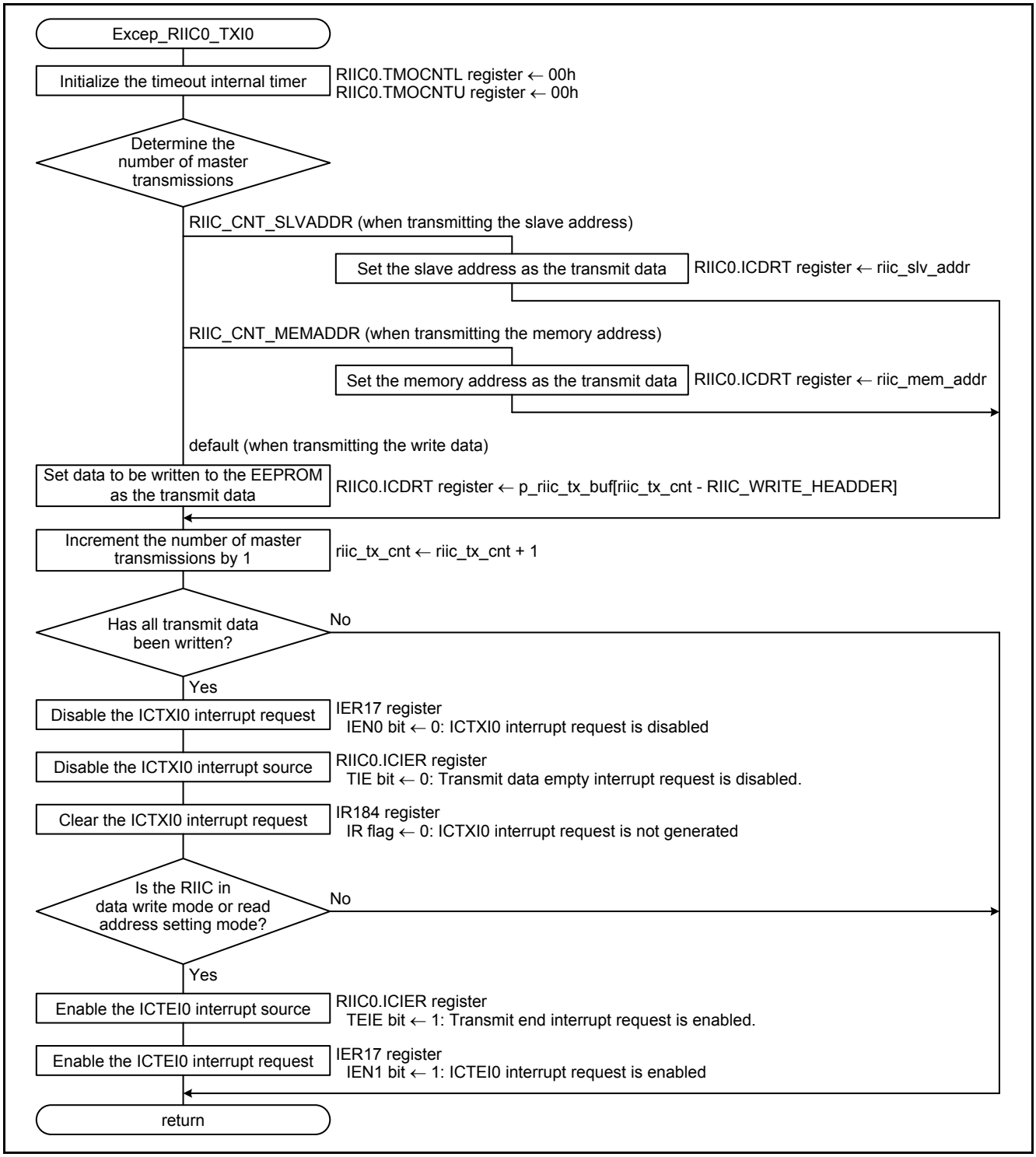


Figure 5.14 ICTXI0 Interrupt Handling

5.8.12 ICRXI0 Interrupt Handling

Figure 5.15 shows ICRXI0 Interrupt Handling.

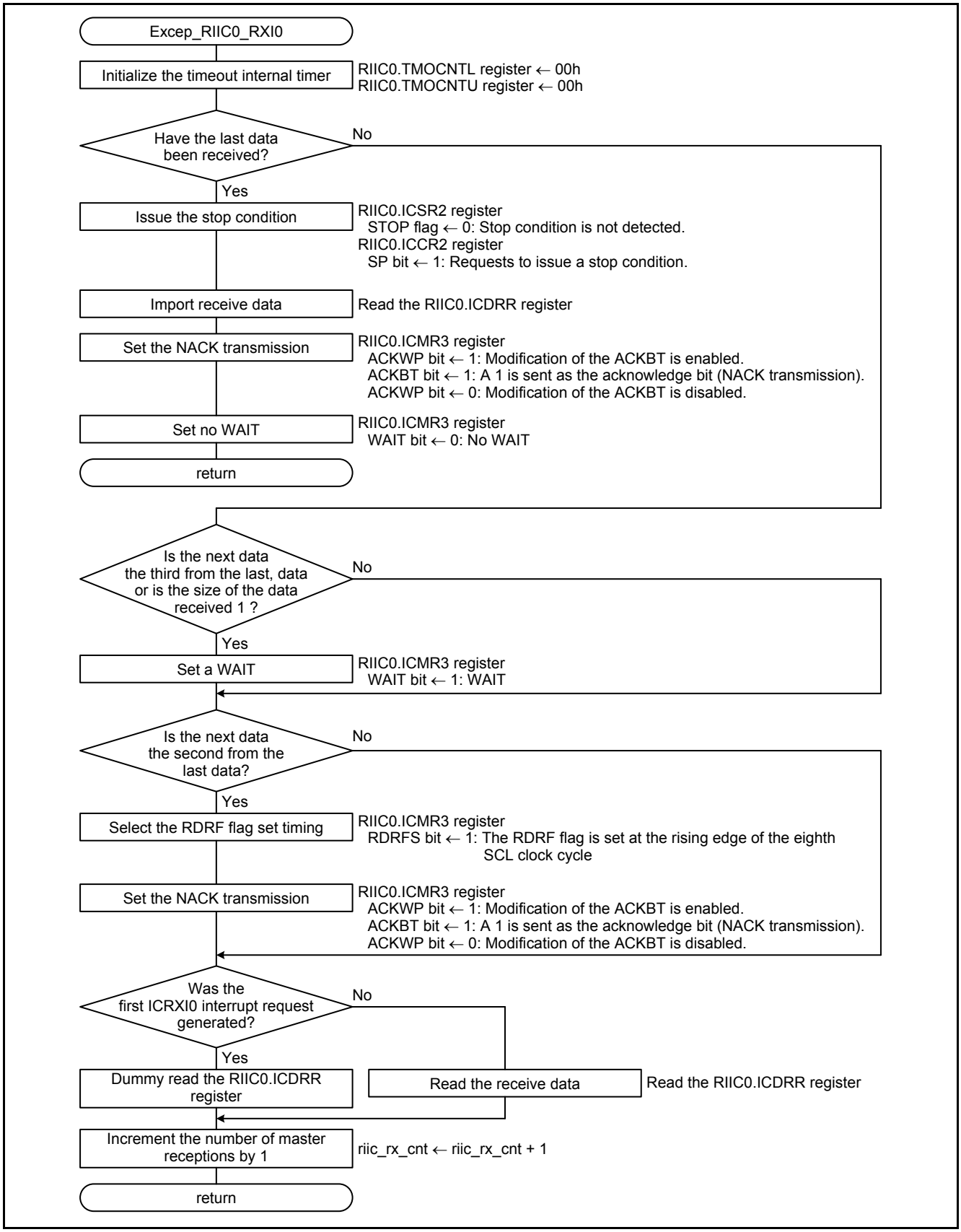


Figure 5.15 ICRXI0 Interrupt Handling

5.8.13 ICTEI0 Interrupt Handling

Figure 5.16 shows ICTEI0 Interrupt Handling.

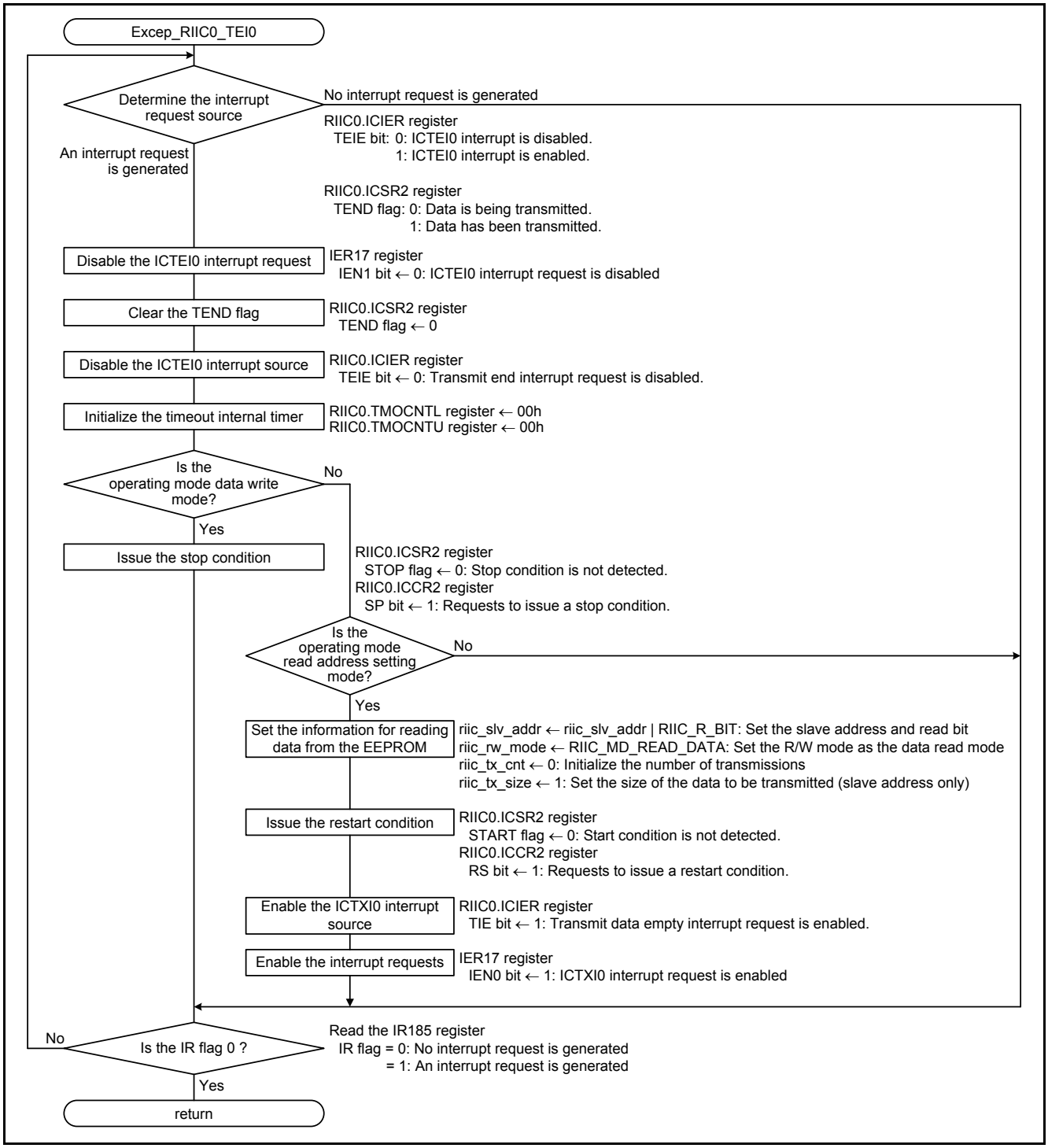


Figure 5.16 ICTEI0 Interrupt Handling

5.8.14 ICEEI0 Interrupt Handling

Figure 5.17 and Figure 5.18 show the ICEEI0 Interrupt Handling.

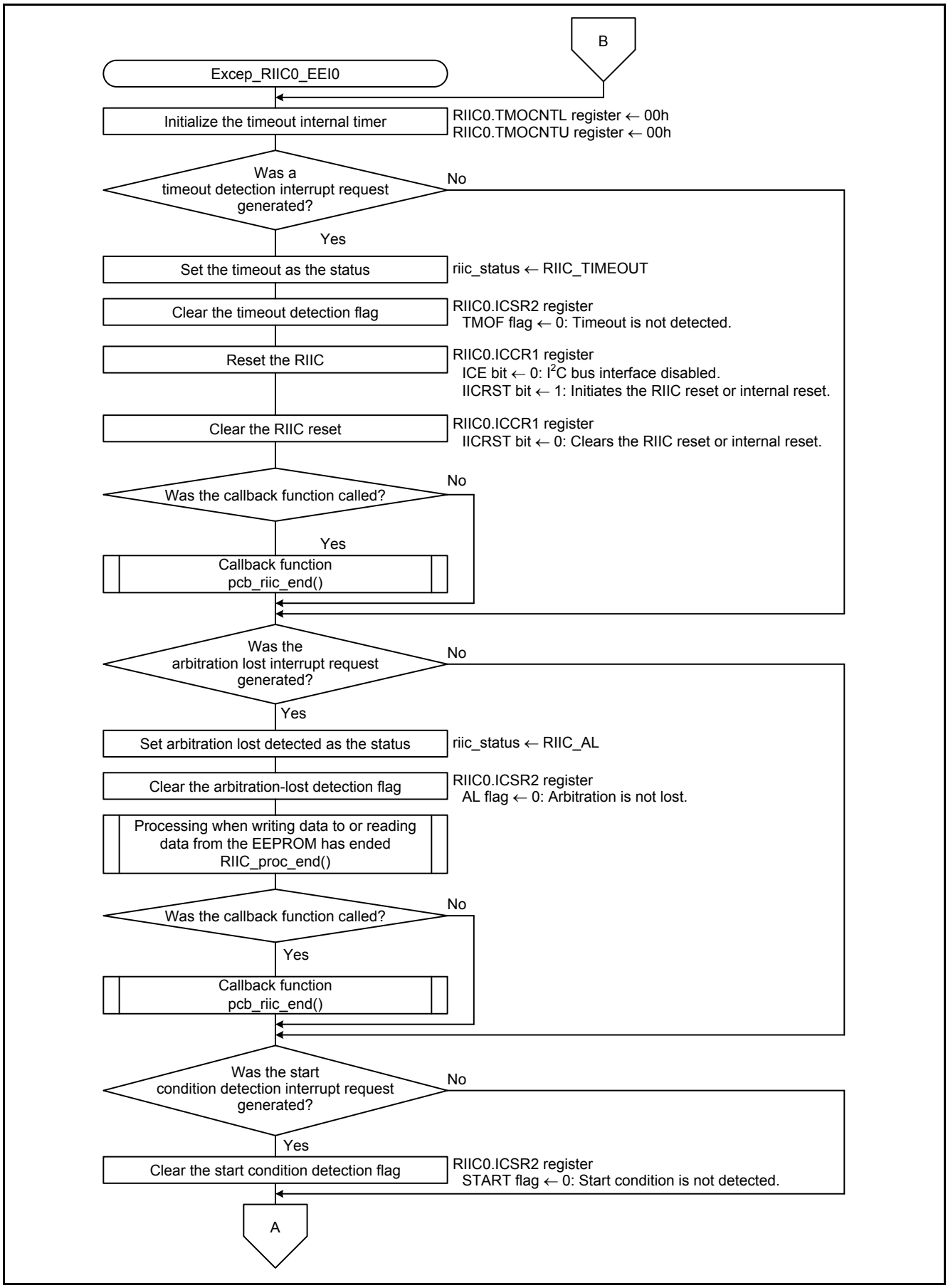


Figure 5.17 ICEEI0 Interrupt Handling (1/2)

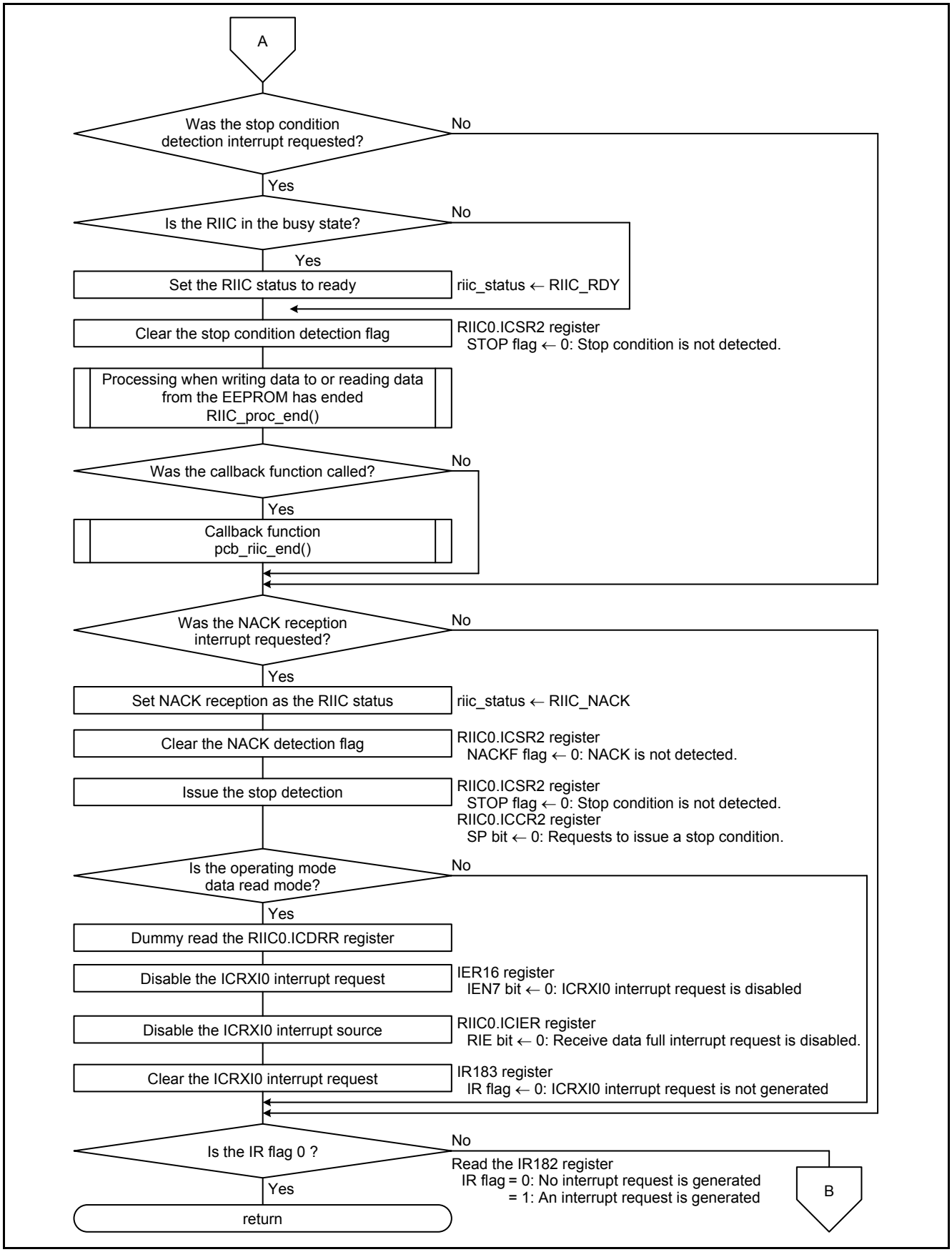


Figure 5.18 ICEEI0 Interrupt Handling (2/2)

5.8.15 Obtain RIIC Status

Figure 5.19 shows the Obtain RIIC Status.

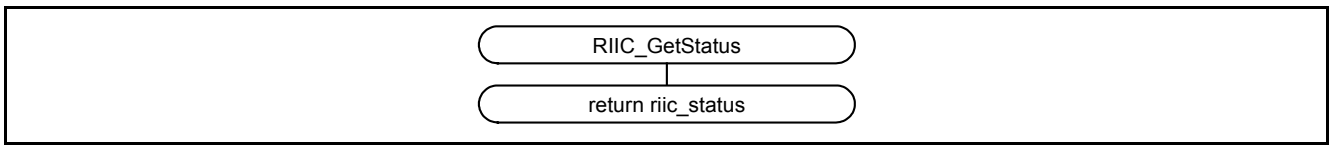
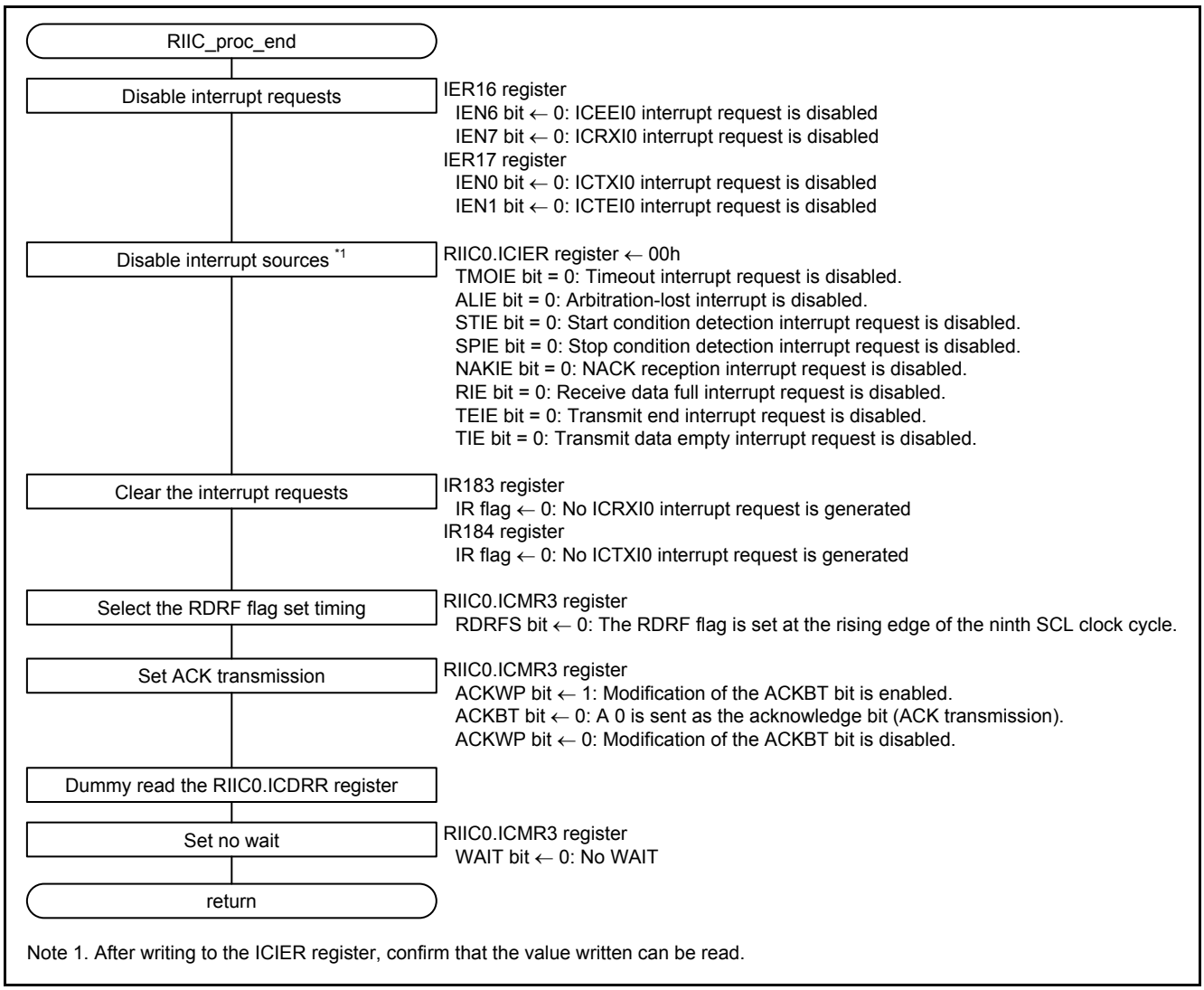


Figure 5.19 Obtain RIIC Status

5.8.16 Processing When Writing Data To or Reading Data From the EEPROM Has Ended

Figure 5.20 shows Processing When Writing Data To or Reading Data From the EEPROM Has Ended.



Note 1. After writing to the ICIER register, confirm that the value written can be read.

Figure 5.20 Processing When Writing Data To or Reading Data From the EEPROM Has Ended

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware

RX63N Group, RX631 Group User's Manual: Hardware Rev.1.70 (R01UH0041EJ)
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	RX63N Group, RX631 Group Application Note Using the RIIC to Access the EEPROM

Rev.	Date	Description	
		Page	Summary
1.00	May 7, 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.

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