RX220 Group, RX21A Group

Communication with EEPROM Using the Renesas I2C Bus Module (RIIC)

R01AN1955EJ0100
Rev.1.00
May 07, 2014

Introduction

This application note describes single-master communication with EEPROM using the RIIC (I²C bus interface) provided by the Renesas RX220 Group and RX21A Group microcontrollers.

Target Devices

RX220 Group and RX21A Group

When this application note is used with other microcontrollers, modifications according to the specifications of the microcontroller used and thorough testing and evaluation are required.

Contents

1. Specifications

This sample program communicates with the EEPROM to write 8 bytes of data and then read the written data back. Between the write and read operations, it uses acknowledge polling to verify that the EEPROM write has completed.

Table 1.1 lists the peripheral functions used and their uses, table 1.2 lists the RIIC settings, table 1.3 lists the EEPROM specifications used with the RX220 Group microcontrollers, and table 1.4 lists the EEPROM specifications used with the RX21A Group microcontrollers. Figure 1.1 shows the circuit diagram for the RX220 and figure 1.2 shows the circuit diagram for the RX21A.

Table 1.1 Peripheral Functions Used and Their Uses

Peripheral function	Use
RIIC	Master transmission, master reception

Table 1.2 RIIC Settings

Item	Settings
Operating frequencies	Internal reference clock (IICϕ): 20 MHz
Master/slave	Single master
Address format	7-bit address format
Transfer speed	400 Kbps
Timeout detection	• The detection function counts while the SCLn line is low. • Long mode (16-bit counter (IICϕ): about 3.277 ms)

Table 1.3 EEPROM Specifications Used with the RX220

Item	Settings
Catalog number	R1EX24016ASAS0A
Capacity	16K (2-kword × 8-bit)
Slave address	Slave address: 1010xxxx Bit 0 is the R/W bit and bits 1 to 3 indicate the page in EEPROM. See the specifications of the EEPROM for further details.
Write protection	Always cleared • WP pin level: low level

Table 1.4 EEPROM Specifications Used with the RX21A

Item	Settings
Catalog number	R1EX24512ASAS0A
Capacity	512K (64-kword × 8-bit)
Slave address	Slave address: 1010xxxx Bit 0 is the R/W bit and bits 1 and 2 indicate the page in EEPROM. See the specifications of the EEPROM for further details.
Write protection	Always cleared • WP pin level: low level

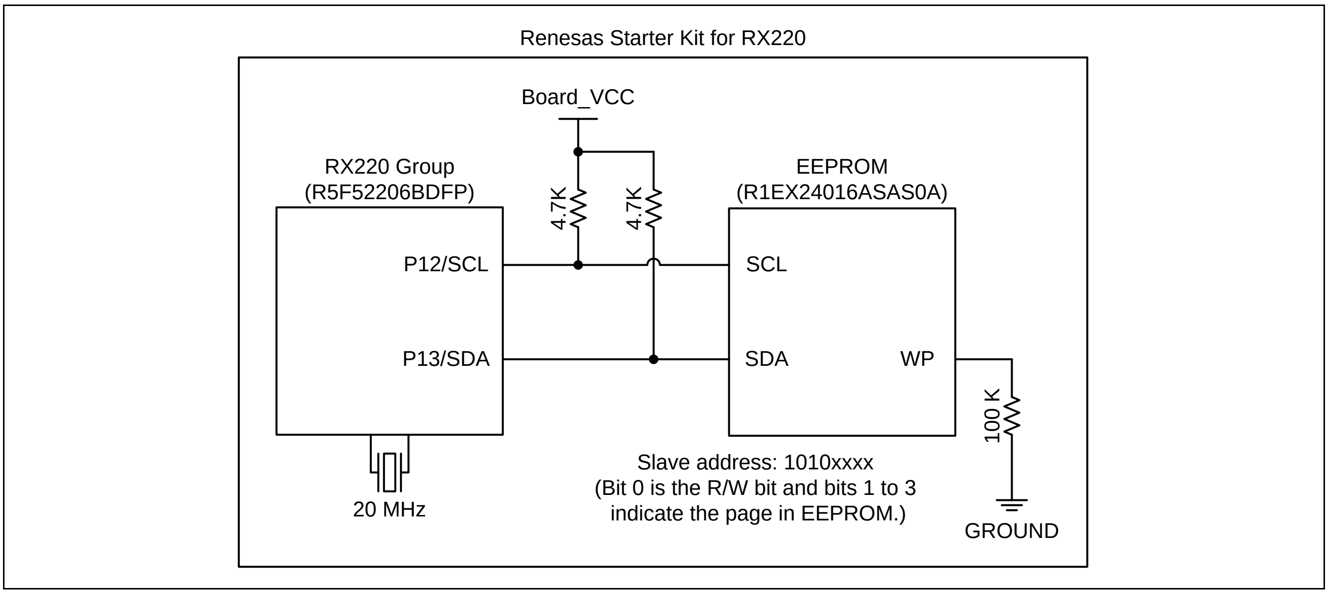

Figure 1.1 RX220 Circuit Diagram

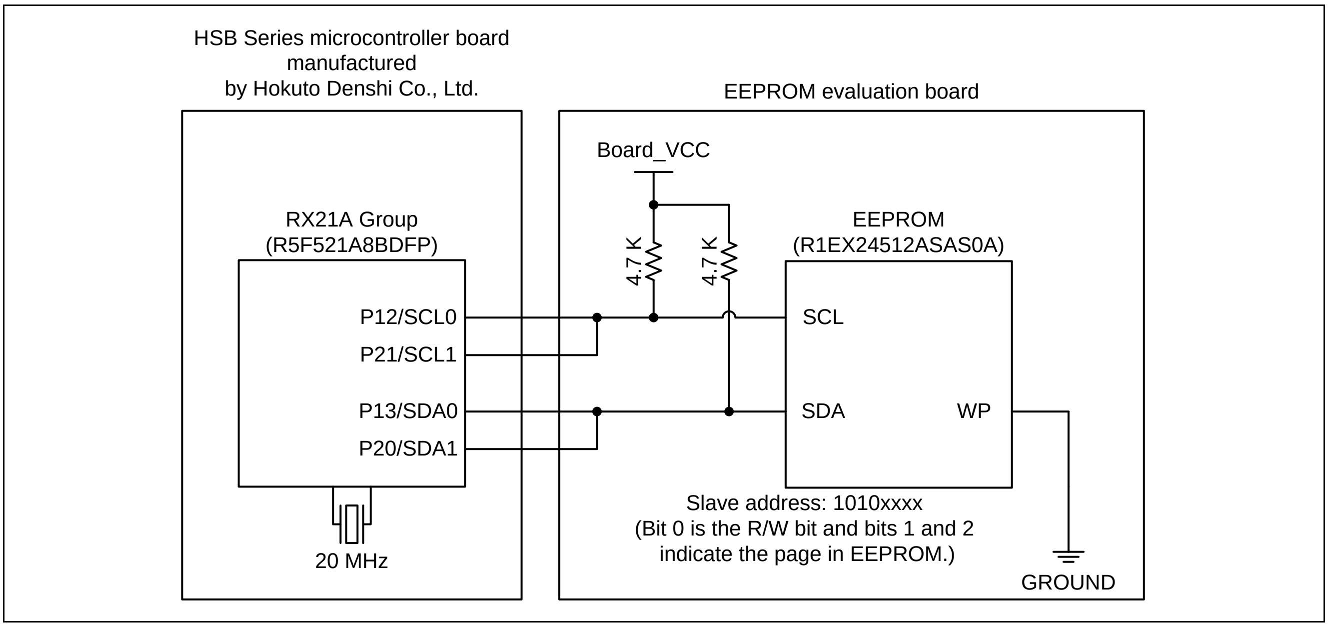

Figure 1.2 RX21A Circuit Diagram

2. Confirmed Operating Condition

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

Table 2.1 Conditions Under which Operation has been Confirmed - RX220

Item	Description
MCU used	R5F52206BDFP (RX220 Group)
Operating frequency	Main clock: 20.0 MHz System clock (ICLK): 20 MHz (main clock divided by 1) Peripheral module clock B (PCLKB): 20 MHz (main clock divided by 1) External bus clock (BCLK): 20 MHz (main clock divided by 1)
Operating voltage	5.0 V: Supplied from the E1 emulator
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
	Compiler option -cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug –nologo (The integrated development environment default settings are used.)
iodefine.h version	Version 1.0A
Endian order	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit for RX220 (Product number: R0K505220S000BE)

Table 2.1 Conditions Under which Operation has been Confirmed - RX21A

Item	Description
MCU used	R5F521A8BDFP (RX21A Group)
Operating frequency	Main clock: 20.0 MHz System clock (ICLK): 20 MHz (main clock divided by 1) Peripheral module clock B (PCLKB): 20 MHz (main clock divided by 1) External bus clock (BCLK): 20 MHz (main clock divided by 1)
Operating voltage	3.3 V: Supplied from the E1 emulator
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
	Compiler option -cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug –nologo (The integrated development environment default settings are used.)
iodefine.h version	Version 1.00
Endian order	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	HSB Series microcontroller board (Hokuto Denshi Co., Ltd.) (Catalog number: HSBRX21AP-B)
	EEPROM evaluation board (No EEPROM is mounted on the Hokuto Denshi Co., Ltd. HSB Series microcontroller boards. To verify operation, EEPROM must be provided separately.)

3. Reference Application Notes

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

- RX220 Group Initial Setting Rev.1.00 (R01AN1494EJ0100_RX220)
- RX21A Group Initial Setting Rev.1.00 (R01AN1486EJ0100_RX21A)

The sample code with this application note uses the initialization from the application note noted above.

The sample code in this application note changes the RX21A clock settings from their default values. (The system clock is switched from the PLL to the main clock.)

The revision number is the one that was current when this application note was written. If there is a more recent version, you should replace this code with the most recent version. The most recent version can be downloaded from the Renesas Electronics Corporation web site.

4. Hardware

4.1 Pins

Tables 4.1 and 4.2 list the pins used and their functions for the RX220 and RX21A, respectively.

Table 4.1 RX220 Pins and Functions

Pin Name	I/O	Function
P12/SCL	Input/output	RIIC0 serial clock input and output
P13/SDA	Input/output	RIIC0 serial data input and output

Table 4.1 RX21A Pins and Functions

Pin Name	I/O	Function
P12/SCL0	Input/output	RIIC0 serial clock input and output
P13/SDA0	Input/output	RIIC0 serial data input and output
P21/SCL1	Input/output	RIIC1 serial clock input and output
P20/SDA1	Input/output	RIIC1 serial data input and output

5. Operation

5.1 Writing to the EEPROM

This sample program uses master transmission for writing to an external EEPROM device. The RIIC module issues a start condition (S) and then sends the EEPROM's slave address. Since the eighth bit at this time is the R/W bit, a 0 must be sent at write time (master transmission). After that, the memory address is sent as two 8-bit bytes, and then the data to be written is sent to the EEPROM in order. The 2-byte memory address transmitted at this time indicates the address for the write operation in EEPROM. After the transmission of all the data has completed, the RIIC module issues a stop condition (P) and releases the bus. Note that the write address in memory used in this application note is 0000h.

Figure 5.1 shows an example of the signals used when writing the EEPROM.

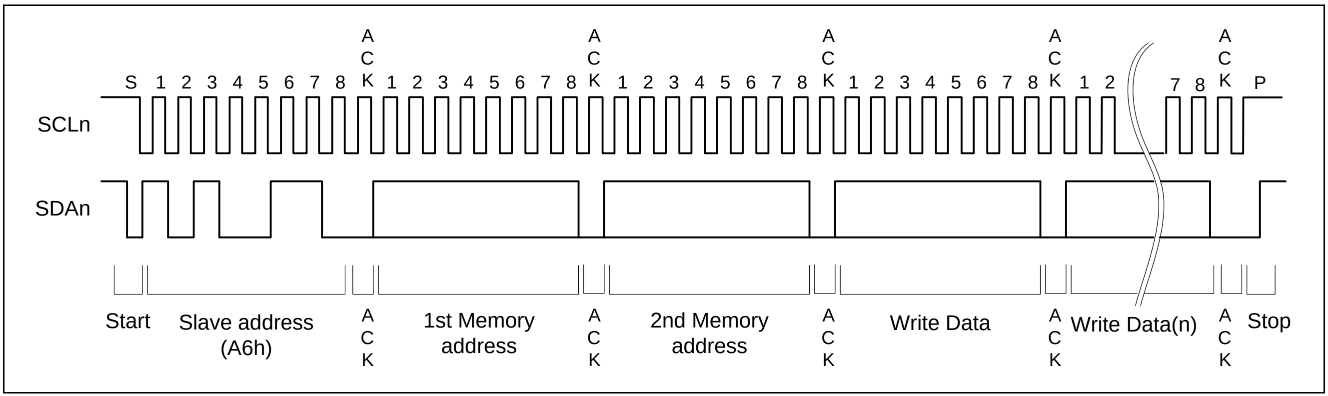

Figure 5.1 Signals when Writing to EEPROM

5.2 Reading from EEPROM

A compound format consisting of master transmission and master reception is used for reading data from EEPROM. First, the RIIC module issues a start condition (S) and then it transmits the EEPROM slave address and then a two byte (2 × 8 bits) memory address. At this time, the RIIC module sends 0 as the R/W bit in the EEPROM slave address transmission (master transmission). After that, it issues a restart condition (Sr) and sends the EEPROM slave address again. At this time, it transmits 1 as the R/W bit in the transmission to the EEPROM (master reception). After the EEPROM slave address has been sent, the data is read out from the EEPROM by the generation of the next clock cycle. During the read operation, the RIIC module transmits an ACK each time it receives a single byte. For the last data, however, it returns a NACK. After that, it generates a stop condition (P). Note that the memory address read by this sample program is 0000h.

Figure 5.2 shows an example of the signals used when reading the EEPROM.

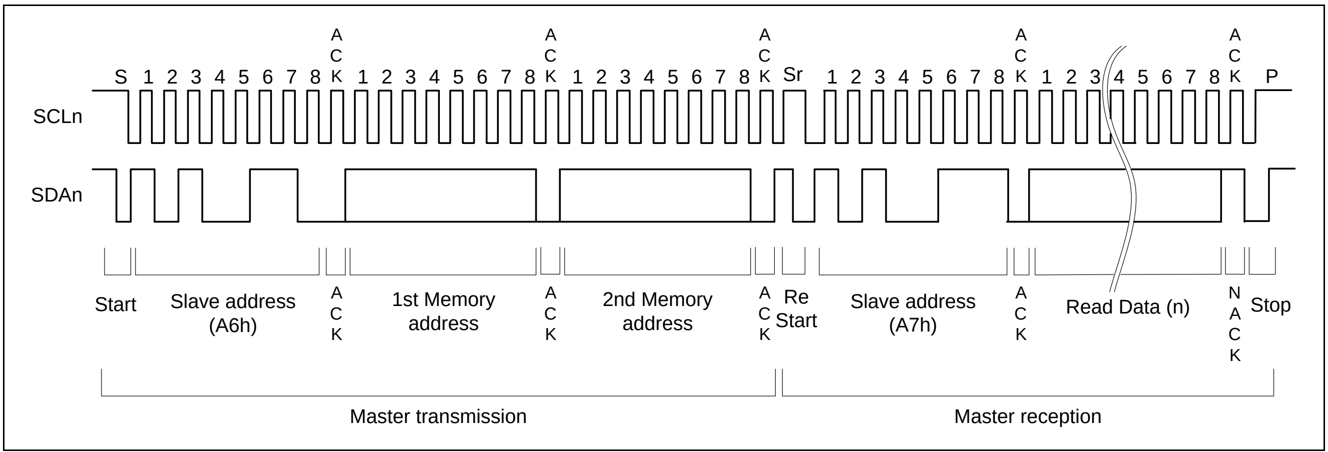

Figure 5.2 Signals when Reading from EEPROM

5.3 Acknowledge Polling

Acknowledge polling is used as the method for determining whether or not the EEPROM is in the write in progress state. To perform acknowledge polling, the sample program issues a start condition and then sends the EEPROM slave address and then a stop condition. At this time, if the EEPROM is writing, it will return a 1 on the ACK clock (NACK). Inversely, if the write has completed, it will return 0 (ACK). This allows the sample program to determine whether or not a write is in progress.

Figure 5.3 shows the acknowledge polling signals.

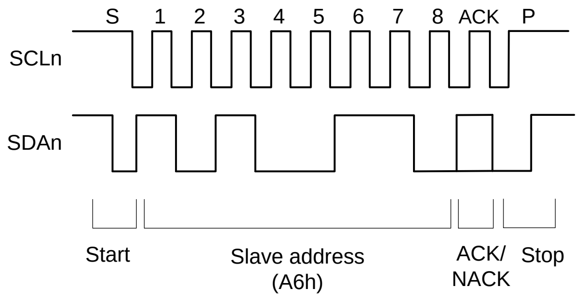

Figure 5.3 Acknowledge Polling Signals

5.4 File Structure

Table 5.1 lists the files in the sample code. Note that this list does not include files generated automatically by the integrated development environment.

Table 5.1 Sample Code Files

Target Device	File Name	Overview	Remarks
Common	main.c	Main processing	
	iic_eeprom.c	Single master transmission and reception	
	iic_eeprom.h	Header file for single master transmission and reception	
	iic_eeprom_cfg.h	Configuration header file for single master transmission and reception	
RX220	r_init_stop_module.c	Functions to stop peripheral functions operating after a reset	
	r_init_stop_module.h	Header file for r_init_stop_module.c	
	r_init_non_existent_port.c	Initialization of nonexistent ports	
	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	
RX21A	r_init_stop_module.c	Functions to stop peripheral functions operating after a reset	
	r_init_stop_module.h	Header file for r_init_stop_module.c	
	r_init_non_existent_port.c	Initialization of nonexistent ports	
	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	

5.5 Option Settings Memory

Table 5.2 lists the states of the option settings memory used by the sample code. These should be set to values appropriate for the actual user system as needed.

Table 5.2 Sample Code Option Settings Memory

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	Stops IWDT after a reset Stops WDT after a reset
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	Disables voltage monitoring resets after a reset Disables HOCOC oscillation after a reset
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.6 Constants

Tables 5.3, 5.4 and 5.5 lists the constants used in the sample code.

Table 5.3 Constants Used in the Sample Code(main.c)

Constant	Set value	Description
TARGET_SALVE_ADDRESS	0xA6	EEPROM address
EEPROM_ADDRESS_LENGTH	2u	EEPROM address length

Table 5.4 Constants Used in the Sample Code
(When the RIIC0 channel is selected in iic_eeprom_cfg.h)

Constant	Set value	Description
RIICn	RIIC0	RIIC channel: RIIC0
MSTP_RIICn	MSTP(RIIC0)	RIIC0 module stop setting bit
IEN_RIICn_EEIn	IEN(RIIC0,EEI0)	RIIC0.EEI0 interrupt request enable bit
IEN_RIICn_RXIn	IEN(RIIC0,RXI0)	RIIC0.RXI0 interrupt request enable bit
IEN_RIICn_TXIn	IEN(RIIC0,TXI0)	RIIC0.TXI0 interrupt request enable bit
IEN_RIICn_TEIn	IEN(RIIC0,TEI0)	RIIC0.TEI0 interrupt request enable bit
IPR_RIICn_EEIn	IPR(RIIC0,EEI0)	RIIC0.EEI0 interrupt priority level setting
IPR_RIICn_RXIn	IPR(RIIC0,RXI0)	RIIC0.RXI0 interrupt priority level setting
IPR_RIICn_TXIn	IPR(RIIC0,TXI0)	RIIC0.TXI0 interrupt priority level setting
IPR_RIICn_TEIn	IPR(RIIC0,TEI0)	RIIC0.TEI0 interrupt priority level setting
IR_RIICn_EEIn	IR(RIIC0,EEI0)	RIIC0.EEI0 interrupt status flag
IR_RIICn_RXIn	IR(RIIC0,RXI0)	RIIC0.RXI0 interrupt status flag
IR_RIICn_TXIn	IR(RIIC0,TXI0)	RIIC0.TXI0 interrupt status flag
IR_RIICn_TEIn	IR(RIIC0,TEI0)	RIIC0.TEI0 interrupt status flag
SCLn_RIICn_PDR	PORT1.PDR.BIT.B2	P12 pin direction control bit
SDAn_RIICn_PDR	PORT1.PDR.BIT.B3	P13 pin direction control bit
SCLn_RIICn_PMR	PORT1.PMR.BIT.B2	P12 pin mode control bit
SDAn_RIICn_PMR	PORT1.PMR.BIT.B3	P13 pin mode control bit
SCLn_RIICn_PFS	MPC.P12PFS.BIT.PSEL	P12 pin function control bit
SDAn_RIICn_PFS	MPC.P13PFS.BIT.PSEL	P13 pin function control bit
PSEL_SETTING	(0x0F)	Pin function selection bit setting value: SCL0, SDA0

Table 5.5 Constants Used in the Sample Code (When the RIIC1 channel is selected in iic_eeprom_cfg.h)

Constant	Set value	Description
RIICn	RIIC1	RIIC channel: RIIC1
MSTP_RIICn	MSTP(RIIC1)	RIIC1 module stop setting bit
IEN_RIICn_EEIn	IEN(RIIC1,EEI1)	RIIC1.EEI1 interrupt request enable bit
IEN_RIICn_RXIn	IEN(RIIC1,RXI1)	RIIC1.RXI1 interrupt request enable bit
IEN_RIICn_TXIn	IEN(RIIC1,TXI1)	RIIC1.TXI1 interrupt request enable bit
IEN_RIICn_TEIn	IEN(RIIC1,TEI1)	RIIC1.TEI1 interrupt request enable bit
IPR_RIICn_EEIn	IPR(RIIC1,EEI1)	RIIC1.EEI1 interrupt priority level setting
IPR_RIICn_RXIn	IPR(RIIC1,RXI1)	RIIC1.RXI1 interrupt priority level setting
IPR_RIICn_TXIn	IPR(RIIC1,TXI1)	RIIC1.TXI1 interrupt priority level setting
IPR_RIICn_TEIn	IPR(RIIC1,TEI1)	RIIC1.TEI1 interrupt priority level setting
IR_RIICn_EEIn	IR(RIIC1,EEI1)	RIIC1.EEI1 interrupt status flag
IR_RIICn_RXIn	IR(RIIC1,RXI1)	RIIC1.RXI1 interrupt status flag
IR_RIICn_TXIn	IR(RIIC1,TXI1)	RIIC1.TXI1 interrupt status flag
IR_RIICn_TEIn	IR(RIIC1,TEI1)	RIIC1.TEI1 interrupt status flag
SCLn_RIICn_PDR	PORT2.PDR.BIT.B1	P21 pin direction control bit
SDAn_RIICn_PDR	PORT2.PDR.BIT.B0	P20 pin direction control bit
SCLn_RIICn_PMR	PORT2.PMR.BIT.B1	P21 pin mode control bit
SDAn_RIICn_PMR	PORT2.PMR.BIT.B0	P20 pin mode control bit
SCLn_RIICn_PFS	MPC.P21PFS.BIT.PSEL	P21 pin function control bit
SDAn_RIICn_PFS	MPC.P20PFS.BIT.PSEL	P20 pin function control bit
PSEL_SETTING	(0x0F)	Pin function selection bit setting value: SCL1, SDA1

5.7 Structures and Unions

Figure 5.4 shows the structure used as an argument to the IIC_EeWrite() and IIC_RandomRead() functions. Table 5.6 lists the members of the iic_api_t structure.

```
struct str_iic_api_t
{
    uint8_t     SlvAdr;      /* Slave Address, Don't set bit0. It's a Read/Write bit */
    uint16_t    PreCnt;      /* Number of Predata */
    uint8_t     *pPreData;   /* Pointer for PreData (Memory Addr of EEPROM) */
    uint32_t    RWCnt;       /* Number of Data */
    uint8_t     *pRWData;    /* Pointer for Data buffer */
};
typedef struct str_iic_api_t iic_api_t;
```

Figure 5 Structure Uses as an Argument to IIC_EepWrite() and IIC_RandomRead()

Table 6 Members of the Structure IIC_API_T

Structure Member	Range of Values	**Description**
SlvAdr	00h to FEh	Slave address Since the low-order bit is the R/W bit, it should always beset to 0.
PreCnt	00h to FFh	Memory address counter This is always set to 2 in this sample program.
*pPreData	—	Memory address storage buffer pointer On write: The address in EEPROM to write data to (write destination) On read: The address in EEPROM to read data from (write source)
RWCnt	0000 0000h to FFFF FFFFh	Data counter On write: Number of data items to write to EEPROM On read: Number of data items to read from EEPROM
*pRWData	—	Data storage buffer pointer On write: Storage source for data to write to EEPROM. On read: Storage destination for data read from EEPROM.

5.8 Variables

Table 5.7 lists the static Variables.

Table 5.7 Static Variables

Type	Variable Name	Contents	Function Used
static uint8_t	trm_eeprom_adr[EEPROM_ ADDRESS_LENGTH]	EEPROM slave address storage buffer (for write)	SampleEepromWrite
static unit8_t	rcv_eeprom_adr[EEPROM_ ADDRESS_LENGTH]	EEPROM slave address storage buffer (for read)	SampleEepromRead
static uint8_t	trm_buff[256]	Transmit data buffer	SampleEepromWrite
static uint8_t	rcv_buff[256]	Receive data buffer	SampleEepromRead
static iic_api_t	iic_buff_prm[2]	Structure used as the argument to the functions IIC_EepWrite() and IIC_RandomRead()	SampleEepromWrite SampleEepromRead
static iic_api_t	iic_buff	Structure used as the argument to the functions IIC_EepWrite() and IIC_RandomRead() (Used by both IIC_EepWrite() and IIC_RandomRead())	IIC_EeWrite IIC_RandomRead iic_eei_int_sp iic_eei_int_st iic_rxi_int_eeread iic_txi_int_eewrite iic_txi_int_eeread
static enum riic_internal_mode_t	iic_mode	Internal mode	IIC_Create IIC_EeWrite IIC_RandomRead iic_eei_int_sp IIC_RXI_interrupt IIC_TXI_interrupt IIC_TEI_interrupt
static enum riic_status_t	iic_status	IIC status	IIC_Create IIC_EeWrite IIC_RandomRead IIC_GetStatus iic_eei_int_sp iic_eei_int_nack
static unit32_t	iic_trm_cnt	Internal IIC transmit counter	IIC_Create IIC_EeWrite IIC_RandomRead iic_eei_int_sp iic_txi_int_eewrite iic_txi_int_eeread
static unit32_t	iic_rcv_cnt	Internal IIC receive counter	IIC_Create IIC_RandomRead iic_eei_int_sp iic_rxi_int_eeread

5.9 Enumerations

The IIC status, the IIC bus status, the internal mode, and the return value from the functions IIC_EepWrite() and IIC_RandomRead() are all declared as enumerations. The IIC status values are listed in table 5.8 and their state transition diagram are shown in figure 5.5. Also, table 5.9 lists the IIC bus status values, table 5.10 lists the internal modes, and table 5.11 lists the return values of the functions IIC_EepWrite() and IIC_RandomRead().

The IIC status is stored at the address given by its first argument when the function IIC_GetStatus() is called. The internal mode is only used in the IIC-related functions in this sample program.

Table 5.8 IIC Status Values (enum RiicStatus_t)

Defined Name	Description
RIIC_STATUS_IDLE	The idle state The status transitions to this state after initialization in the function IIC_Create(). The status also transitions to this state after either an EEPROM write or an EEPROM read completes normally (after a stop condition is detected).
RIIC_STATUS_ON_COMMUNICATION	Communication in progress The status transitions to this state when communication is initiated by either IIC_EepWrite() or IIC_RandomRead().
RIIC_STATUS_NACK	NACK received The status transitions to this state when a NACK is received.
RIIC_STATUS_FAILED	Communication failure The status transitions to this state when a stop condition is detected before either an EEPROM write or an EEPROM read completes. In this sample program, since a stop condition is generated on either a timeout or an arbitration lost, the status will transition to this state on either of those events as well.

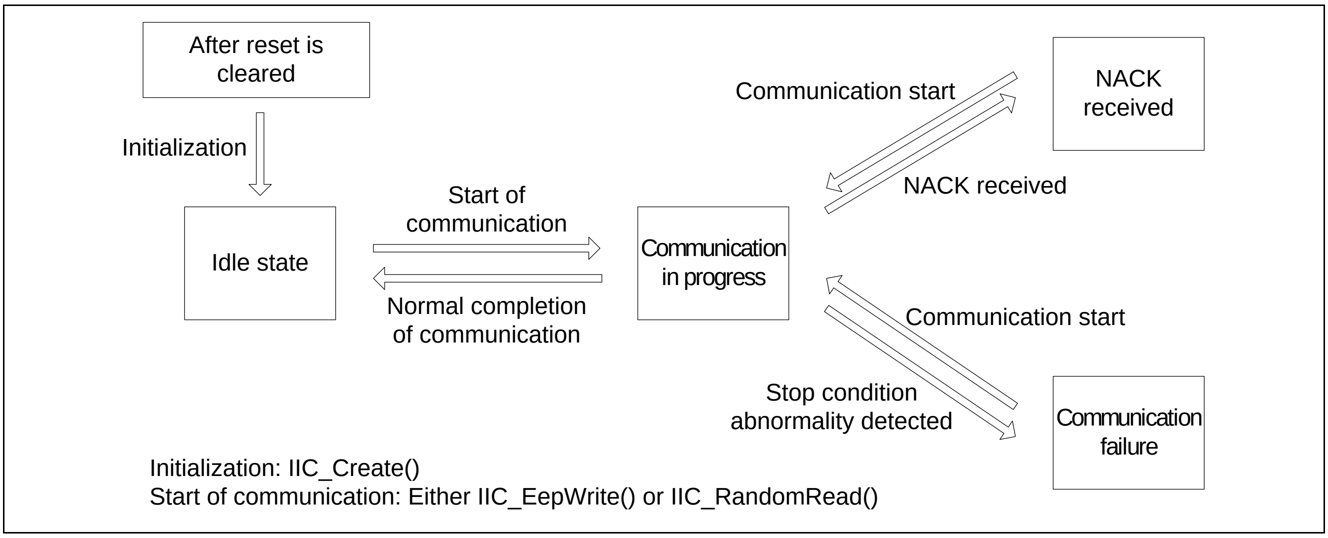

Figure 5.5 IIC Status State Transition Diagram

Table 5.9 IIC Bus Status (enum riic_bus_status_t)

Defined Name	Description
RIIC_BUS_STATUS_FREE	IIC bus busy
RIIC_BUS_STATUS_BBSY	IIC bus free

Table 5.10 Internal Modes (enum riic_internal_mode_t)

Defined Name	Description
IIC_MODE_IDLE	Idle mode The internal mode transitions to idle mode on initialization by IIC_Create() or when a stop condition is detected.
IIC_MODE_EE_READ	EEPROM read mode The internal mode transitions to this mode at the start of communication due to IIC_RandomRead().
IIC_MODE_EE_WRITE	EEPROM write mode The internal mode transitions to this mode at the start of communication due to IIC_EeWrite().

Table 5.11 IIC_EepWrite() and IIC_RandomRead() Return Value (enum riic_ee_fnc_t)

Defined Name	Description
RIIC_OK	This value is returned when communication starts up normally.
RIIC_BUS_BUSY	This value is returned when the I2C bus is busy.
RIIC_MODE_ERROR	This value is returned when the RIIC module has a communication operation in progress.
RIIC_PRM_ERROR	This value is returned when an illegal argument value is passed. (Only the function IIC_RandomRead() uses this value.)

5.10 Functions

Table 5.12 lists the Functions.

Table 5.12 Function(s)

Function	Description
main	Main processing
R_INIT_StopModule	Functions to stop peripheral functions operating after a reset
R_INIT_NonExistentPort	Initialization of nonexistent ports
R_INIT_Clock	Clock initialization
port_init	Port initialization
peripheral_init	Peripheral initialization
CpuIntCreate	CPU interrupt setting
SampleEepromWrite	EEPROM write processing example
SampleEepromRead	EEPROM read processing example
IICAckPolling	Acknowledge Polling
IIC_Create	IIC processing setup
IIC_Destroy	IIC termination processing
IIC_EeWrite	EEPROM write start processing
IIC_RandomRead	EEPROM read start processing
IIC_GetStatus	IIC status check
IIC_EEI_interrupt	Communication error or event interrupt
iic_eei_int_timeout	Timeout detection interrupt
iic_eei_int_al	Arbitration lost detected interrupt
iic_eei_int_sp	Stop condition detected interrupt
iic_eei_int_st	Start condition detected interrupt
iic_eei_int_nack	NACK detected interrupt
IIC_RXI_interrupt	Receive data full interrupt
iic_rxi_int_eeread	EEPROM read processing (master reception section)
IIC_TXI_interrupt	Transmit data empty interrupt
iic_txi_int_eewrite	EEPROM write processing
iic_txi_int_eeread	EEPROM read processing (master transmission section)
IIC_TEI_interrupt	Transmission complete interrupt
iic_tei_int_eewrite	Transmission end processing used after an EEPROM write
iic_tei_int_eeread	Transmission end processing used after an EEPROM read
iic_gen_clk_sp	Stop condition generation used when an error occurs
iic_error	Error handling
Excep_RIICn_EEIn	RIICn.EEIn interrupt handler
Excep_RIICn_RXIn	RIICn.RXIn interrupt handler
Excep_RIICn_TXIn	RIICn.TXIn interrupt handler
Excep_RIICn_TEIn	RIICn.TEIn interrupt handler

5.11 Function Specifications

The following tables lists the sample code function specifications.

main	
Overview	Main processing
Header	None
Declaration	void main(void)
Description	This function performs initialization, port initialization, and peripheral function initialization. After that it calls the functions that perform sample EEPROM write processing, acknowledge polling, EEPROM read processing, and IIC processing termination.
Arguments	None
Return values	None

R_INIT_StopModule	
Overview	Functions to stop peripheral functions operating after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Performs the settings that transition to module stop state.
Arguments	None
Return values	None
Remarks	Transitioning to the module stop state is not performed in the sample code. See the two application notes RX220 Group Microcontroller initialization Rev.1.00 and RX21A Group Microcontroller Initialization Rev.1.00 for details on this function.

R_INIT_NonExistentPort	
Overview	Initialization of nonexistent ports
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initializes the port direction registers for port pins that do not exist in products provided in packages with less than 100 pins.
Arguments	None
Return values	None
Remarks	He sample code is set up to operate using 100-pin versions of the microcontrollers (PIN_SIZE = 100). After this function is called, applications should the port direction control bit to 1 and the port output data storage bit to 0 for ports that do not exist if they will write in byte units to PDR and PODR registers that include ports that do not exist. See the two application notes RX220 Group Microcontroller initialization Rev.1.00 and RX21A Group Microcontroller Initialization Rev.1.00 for details on this function.

R_INIT_Clock	
Overview	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initializes the clocks.
Arguments	None
Return values	None
Remarks	The sample code selects processing in which the system clock is used as the main clock and subclocks are not used. See the two application notes RX220 Group Microcontroller initialization Rev.1.00 and RX21A Group Microcontroller Initialization Rev.1.00 for details on this function.

port_init	
Overview	Port initialization
Header	None
Declaration	static void port_init(void)
Description	Initializes the ports.
Arguments	None
Return values	None

peripheral_init	
Overview	Peripheral function initialization
Header	None
Declaration	static void peripheral_init(void)
Description	Initializes the used peripheral functions.
Arguments	None
Return values	None

CpuIntCreate	
Overview	CPU interrupt setting
Header	None
Declaration	void CpuIntCreate(void)
Description	Initializes the RIIC CPU interrupt.
Arguments	None
Return values	None

SampleEepromWrite	
Overview	EEPROM write processing example
Header	iic_eeprom.h
Declaration	static void SampleEepromWrite(void)
Description	Uses master transmission to write to the EEPROM.
Arguments	None
Return values	None

SampleEepromRead	
Overview	EEPROM read processing example
Header	iic_eeprom.h
Declaration	static void SampleEepromRead(void)
Description	Reads out data from EEPROM using master transmission and master reception (compound format).
Arguments	None
Return values	None

IICAckPolling	
Overview	Acknowledge Polling
Header	iic_eeprom.h
Declaration	bool IICAckPolling(uint8_t in_addr1, unit8_t in_num, uint32_t in_len)
Description	Determines whether or not the EEPROM is in the write busy state.
Arguments	uint8_t in_addr1 : Slave address (0x00 to 0xFE) uint8_t in_num : Acknowledge polling repeat count uint32_t in_len : Length of the interval time between acknowledge polling operations
Return values	true : ACK response received false: No ACK response

IIC_Create	
Overview	IIC processing setup
Header	iic_eeprom.h
Declaration	void IIC_Create(void)
Description	Initializes the RIIC module, sets the transfer bit rate, sets up interrupts, and sets up the timeout operation.
Arguments	None
Return values	None

IIC_Destroy	
Overview	IIC termination processing
Header	None
Declaration	void IIC_Destroy(void)
Description	Stops the RIIC module and clears all the RIIC module related registers.
Arguments	None
Return values	None

IIC_EeWrite	
Overview	EEPROM write start processing
Header	iic_eeprom.h
Declaration	enum riic_ee_fnc_t IIC_EeWrite(iic_api_t data1)
Description	Uses master transmission to write to the EEPROM. If the I^{2}C bus is busy or if the RIIC module is in the communication in progress state, it does not start master transmission.
Arguments	iic_api_t data1 : Slave address Memory address counter Memory address storage buffer pointer Data counter Data storage buffer pointer
Return values	RIIC_OK : If communication starts up normally RIIC_BUS_BUSY : If the I^{2}C bus is busy RIIC_MODE_ERROR: If the RIIC module is communicating

IIC_RandomRead	
Overview	EEPROM read start processing
Header	iic_eeprom.h
Declaration	enum riic_ee_fnc_t IIC_RandomRead(iic_api_t data1)
Description	Reads data from EEPROM using master transmission and master reception (compound format). If the I^{2}C bus is busy or the RIIC is already communicating, it does not start a master transmission.
Arguments	iic_api_t data1 : Slave address Memory address counter Memory address storage buffer pointer Data counter Data storage buffer pointer
Return values	RIIC_OK : If communication starts up normally RIIC_BUS_BUSY : If the I^{2}C bus is busy RIIC_MODE_ERROR: If the RIIC module is communicating RIIC_PRM_ERROR : If the argument value is illegal

IIC_GetStatus	
Overview	IIC status check
Header	iic_eeprom.h
Declaration	void IIC_GetStatus(enum riic_status_t *data1,enum riic_bus_status_t *data2)
Description	This function stores the IIC status in the area indicated by the first argument. It also stores the IIC bus state in the area indicated by the second argument.
Arguments	Enum riic_status_t *data1 : IIC status Enum riic_bus_status *data2 : IIC bus status
Return values	None

IIC_EEI_interrupt	
Overview	Communication error or event interrupt
Header	None
Declaration	void IIC_EEI_interrupt(void)
Description	Calls the handler for the detected interrupt request.
Arguments	None
Return values	None

iic_eei_int_timeout	
Overview	Timeout detection interrupt
Header	None
Declaration	static void iic_eei_int timeout(void)
Description	Calls the stop condition generation function when an abnormality occurs.
Arguments	None
Return values	None

iic_eei_int_al	
Overview	Arbitration lost detected interrupt
Header	None
Declaration	static void iic_eei_int_al(void)
Description	Calls the stop condition generation function when an abnormality occurs.
Arguments	None
Return values	None

iic_eei_int_sp	
Overview	Stop condition detected interrupt
Header	iic_eeprom.h
Declaration	static void iic_eei int_sp(void)
Description	Terminates communication and sets the internal mode to IDLE.
Arguments	None
Return values	None

iic_eei_int_st	
Overview	Start condition detected interrupt
Header	iic_eeprom.h
Declaration	static void iic_eei_int_st(void)
Description	Transmits the EEPROM slave address. This interrupt is only used when reading data from EEPROM.
Arguments	None
Return values	None

iic_eei_int_nack	
Overview	NACK detected interrupt
Header	iic_eeprom.h
Declaration	static void iic_eei_int_nack(void)
Description	Requests that a stop condition be issued.
Arguments	None
Return values	None

IIC_RXI_interrupt	
Overview	Receive data full interrupt
Header	iic_eeprom.h
Declaration	void IIC_RXI_interrupt(void)
Description	Checks the internal mode and calls a handler.
Arguments	None
Return values	None

iic_rxi_int_eeread	
Overview	EEPROM read processing (master reception section)
Header	iic_eeprom.h
Declaration	static void iic_rxi_int_eeread(void)
Description	Stores the receive data in a buffer.
Arguments	None
Return values	None

IIC_TXI_interrupt	
Overview	Transmit data empty interrupt
Header	iic_eeprom.h
Declaration	void IIC_TXI_interrupt(void)
Description	Checks the internal mode and calls a handler.
Arguments	None
Return values	None

iic_txi_int_eewrite	
Overview	EEPROM write processing
Header	iic_eeprom.h
Declaration	static void iic_txi_int eewrite(void)
Description	Transmits, in order, the slave address, the memory address, and the write data.
Arguments	None
Return values	None

iic_txi_int_eeread	
Overview	EEPROM read processing (master transmission section)
Header	iic_eeprom.h
Declaration	static void iic_txi_int_eeread(void)
Description	Transmits, in order, the slave address and the memory address.
Arguments	None
Return values	None

IIC_TEI_interrupt	
Overview	Transmission complete interrupt
Header	iic_eeprom.h
Declaration	void IIC_TEI_interrupt(void)
Description	Checks the internal mode and calls a handler.
Arguments	None
Return values	None

iic_tei_int_eewrite	
Overview	Transmission end processing used after an EEPROM write
Header	None
Declaration	static void iic_tei_int_eewrite(void)
Description	Requests that a stop condition be issued.
Arguments	None
Return values	None

iic_tei_int_eeread	
Overview	Transmission end processing used after an EEPROM read
Header	None
Declaration	static void iic_tei_int_eeread(void)
Description	Requests that a restart condition be issued.
Arguments	None
Return values	None

iic_gen_clk_sp	
Overview	Stop condition generation used when an error occurs
Header	None
Declaration	static void iic_gen_clk_sp(void)
Description	When an abnormality occurs, requests that a stop condition be issued.
Arguments	None
Return values	None

iic_error	
Overview	Error handling
Header	None
Declaration	static void iic_error (enum riic_err_code_t error_code)
Description	Normally, this function will not be called. If called, it executes an infinite loop.
Arguments	enum riic_err_code_t error_code : IIC error code
Return values	None

Excep_RIICn_EEIn	
Overview	RIICn.EEIn interrupt handler (level detection interrupt)
Header	None
Declaration	static void Excep_RIICn_EEIn(void)
Description	Calls the communication error/event generation interrupt handler.
Arguments	None
Return values	None

Excep_RIICn_RXIn	
Overview	RIICn.RXIn interrupt handler (edge detection interrupt)
Header	None
Declaration	static void Excep_RIICn_RXIn(void)
Description	Calls the receive data full interrupt handler.
Arguments	None
Return values	None

Excep_RIICn_TXIn	
Overview	RIICn.TXIn interrupt handler (edge detection interrupt)
Header	None
Declaration	static void Excep_RIICn_TXIn(void)
Description	Calls the transmit data empty interrupt handler.
Arguments	None
Return values	None

Excep_RIICn_TEIn	
Overview	RIICn.TEIn interrupt handler (level detection interrupt)
Header	None
Declaration	static void Excep_RIICn_TEIn(void)
Description	Calls the transmission complete interrupt handler.
Arguments	None
Return values	None

5.12 Flowcharts

The flowchart for this application show here is for the case when the RIIC0 channel is selected. Since the RX220 has only one RIIC channel (RIIC0), the channel cannot be selected. The RX21A has two channels, RIIC0 and RIIC1, from which the channel can be selected.

5.12.1 Main Processing

Figure 5.6 shows the flowchart for the main processing.

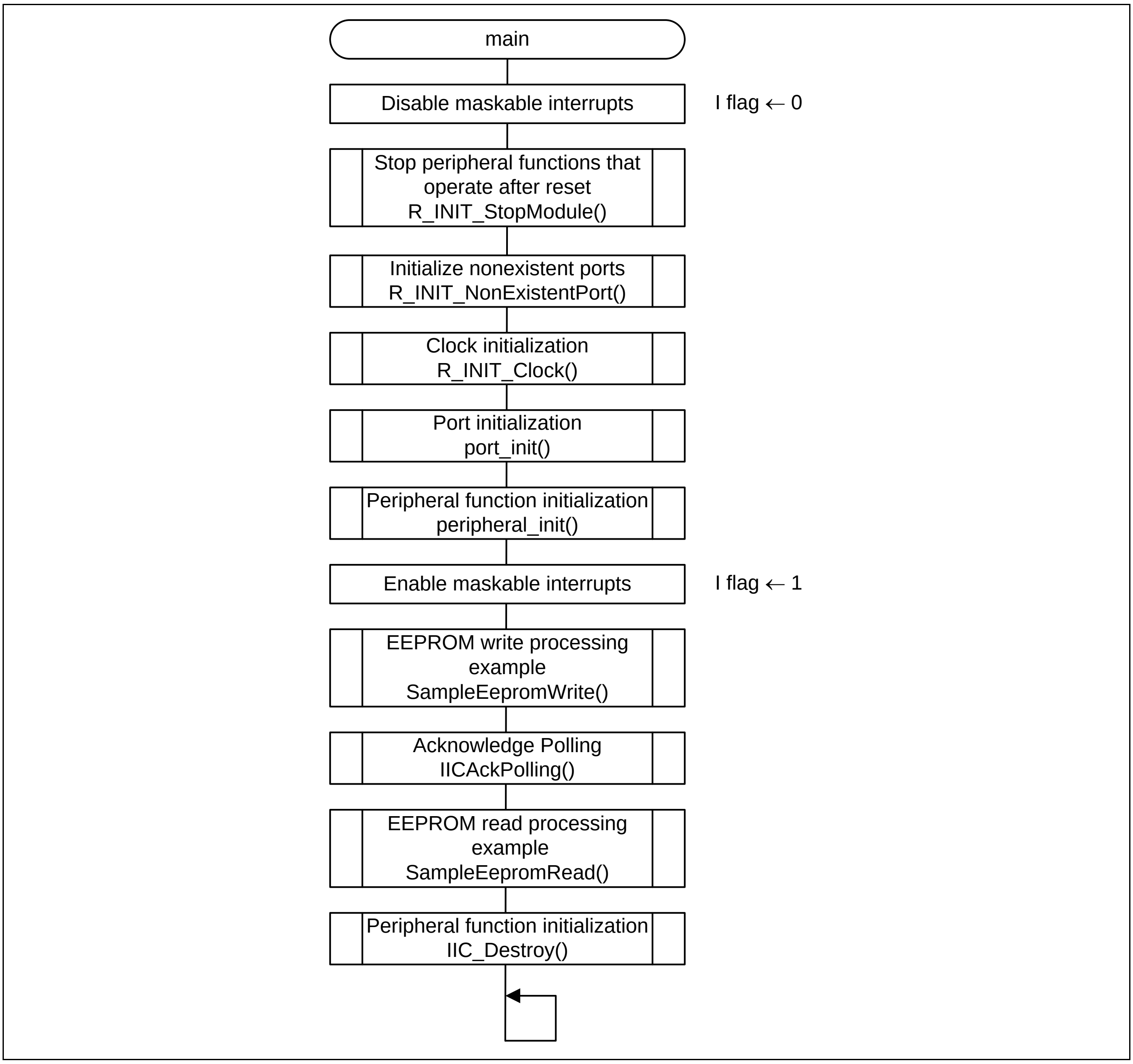

Figure 5.6 Main Processing

5.12.2 Port Initialization

Figure 5.7 shows the flowchart for port initialization.

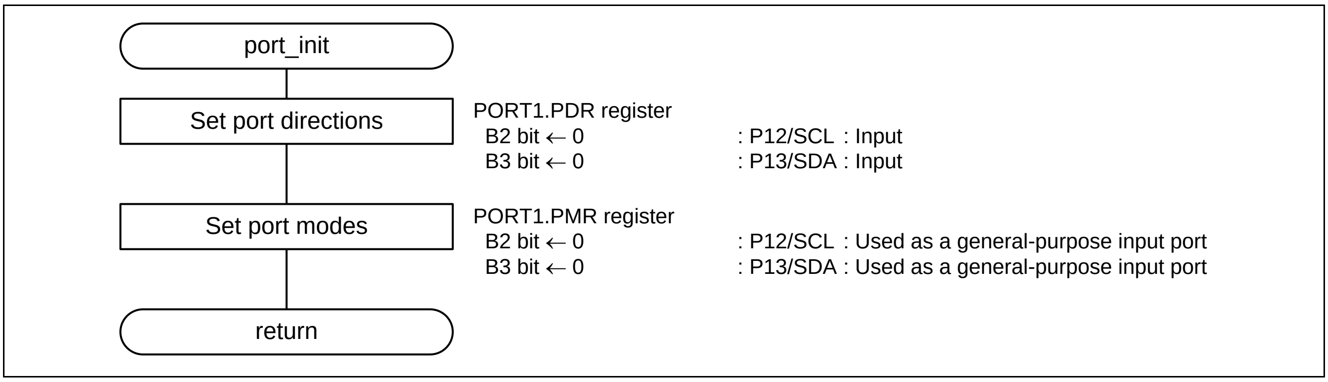

Figure 5.7 Port Initialization

5.12.3 Peripheral Function Initialization

Figure 5.8 shows the flowchart for peripheral function initialization.

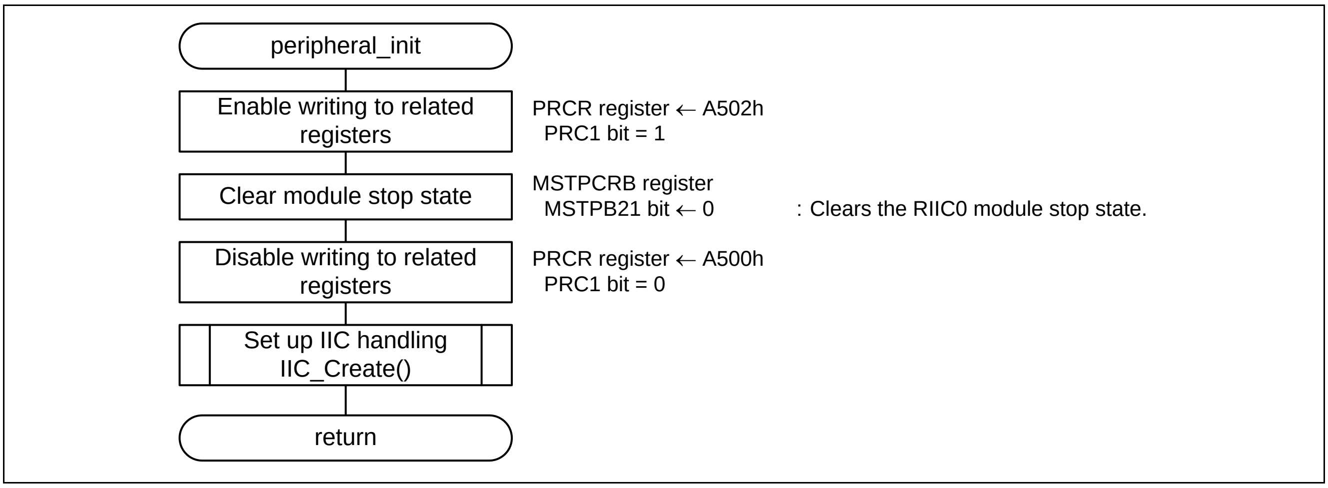

Figure 5.8 Peripheral Function Initialization

5.12.4 CPU Interrupt Settings

Figure 5.9 and Figure 5.10 shows the flowchart for CPU interrupt settings.

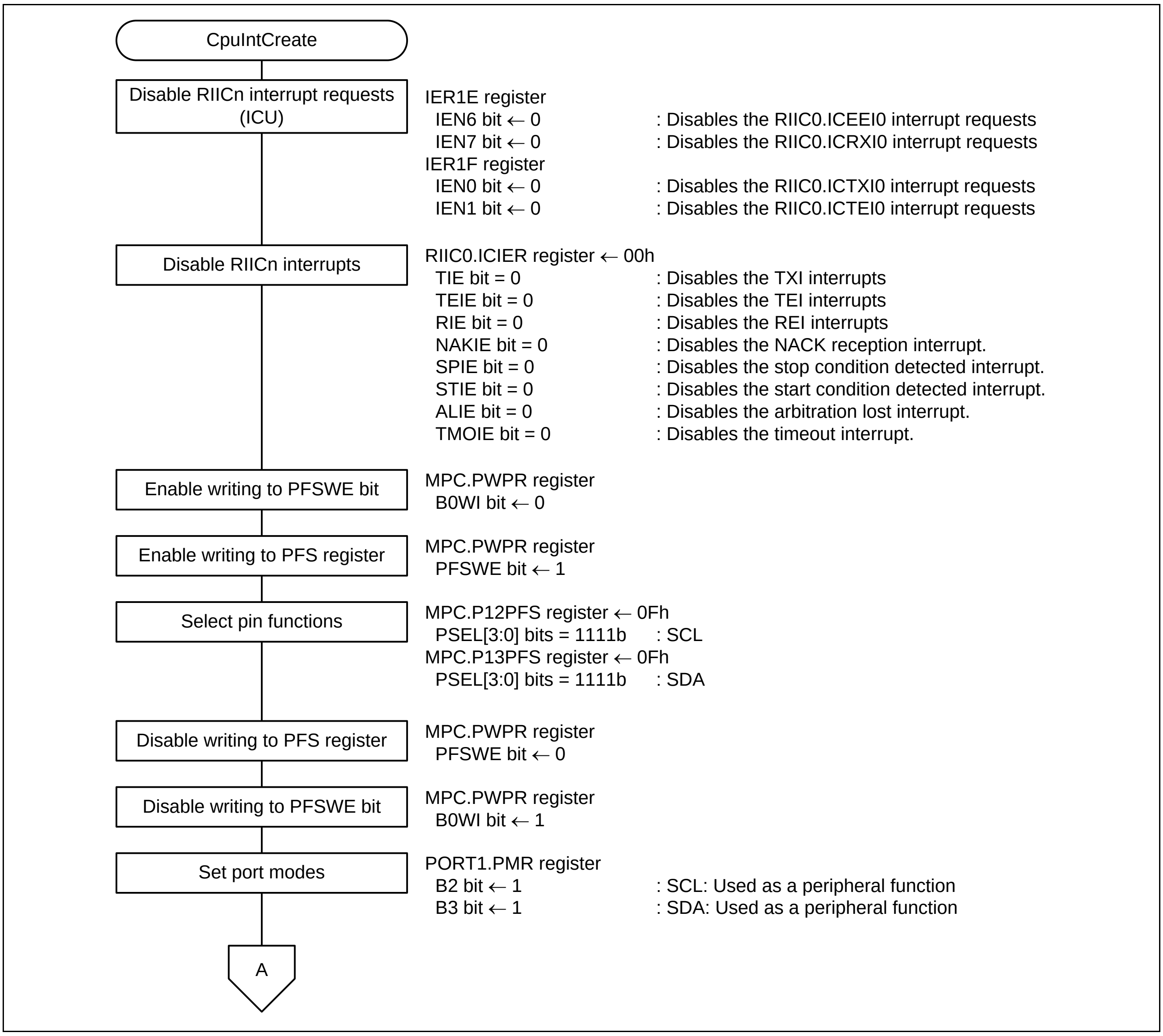

Figure 5.9 CPU Interrupt Settings (1/2)

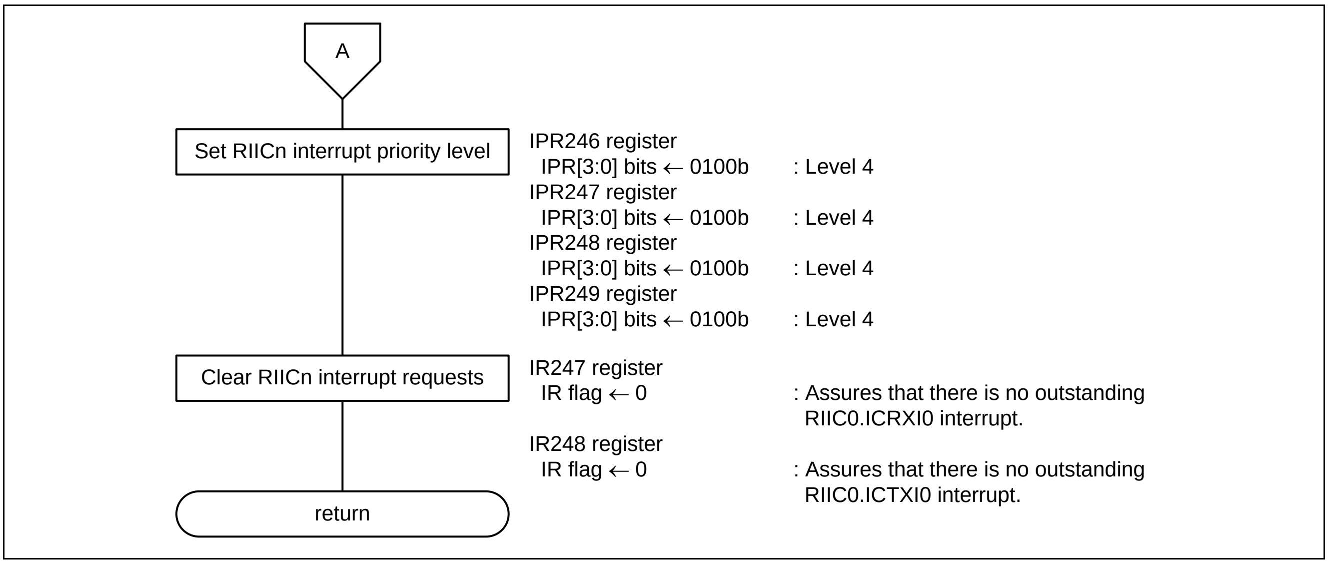

Figure 5.10 CPU Interrupt Settings (2/2)

5.12.5 Sample EEPROM Write Processing

Figure 5.11 shows the flowchart for sample EEPROM write processing.

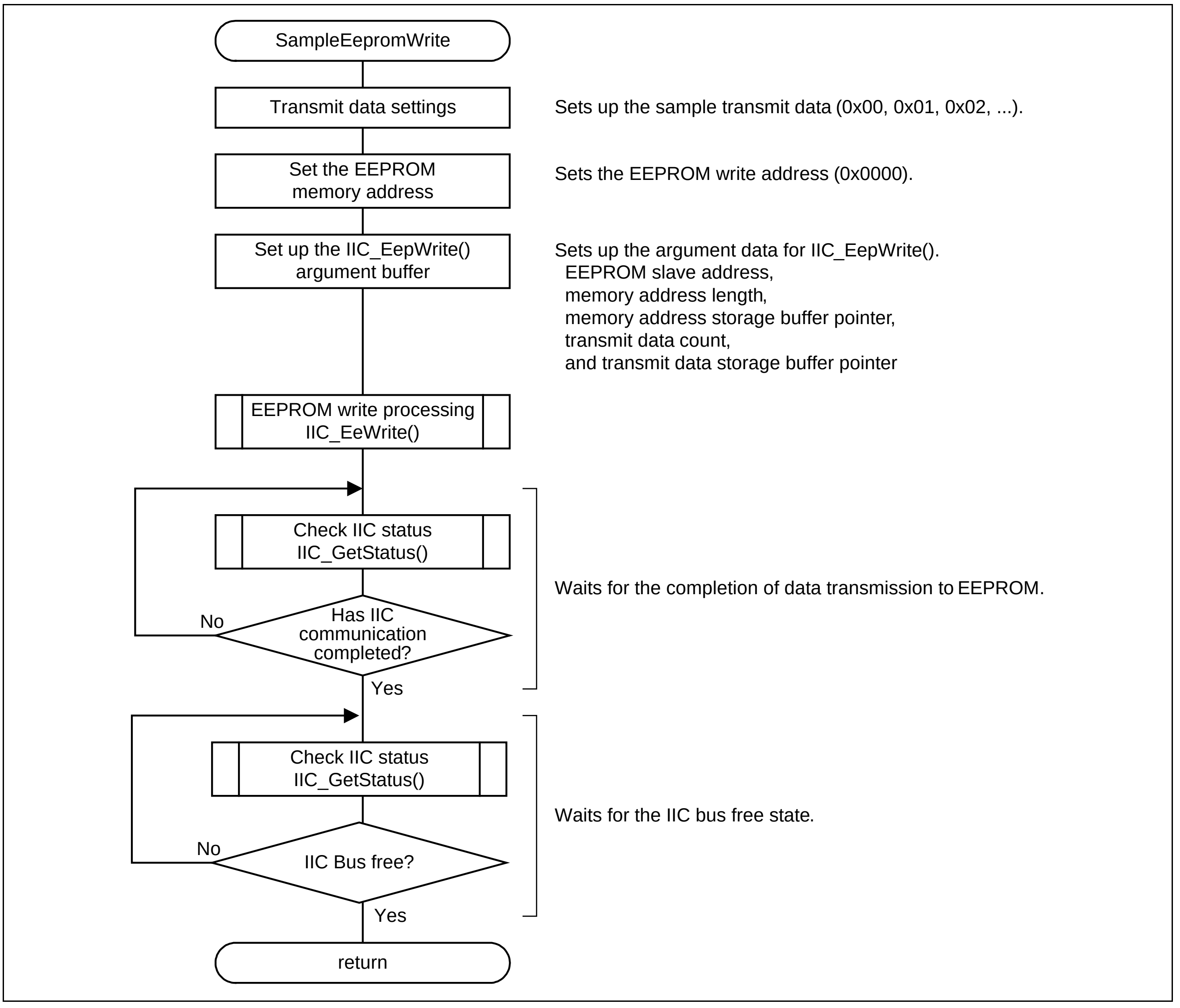

Figure 5.11 Sample EEPROM Write Processing

5.12.6 Sample EEPROM Read Processing

Figure 5.12 shows the flowchart for sample EEPROM read processing.

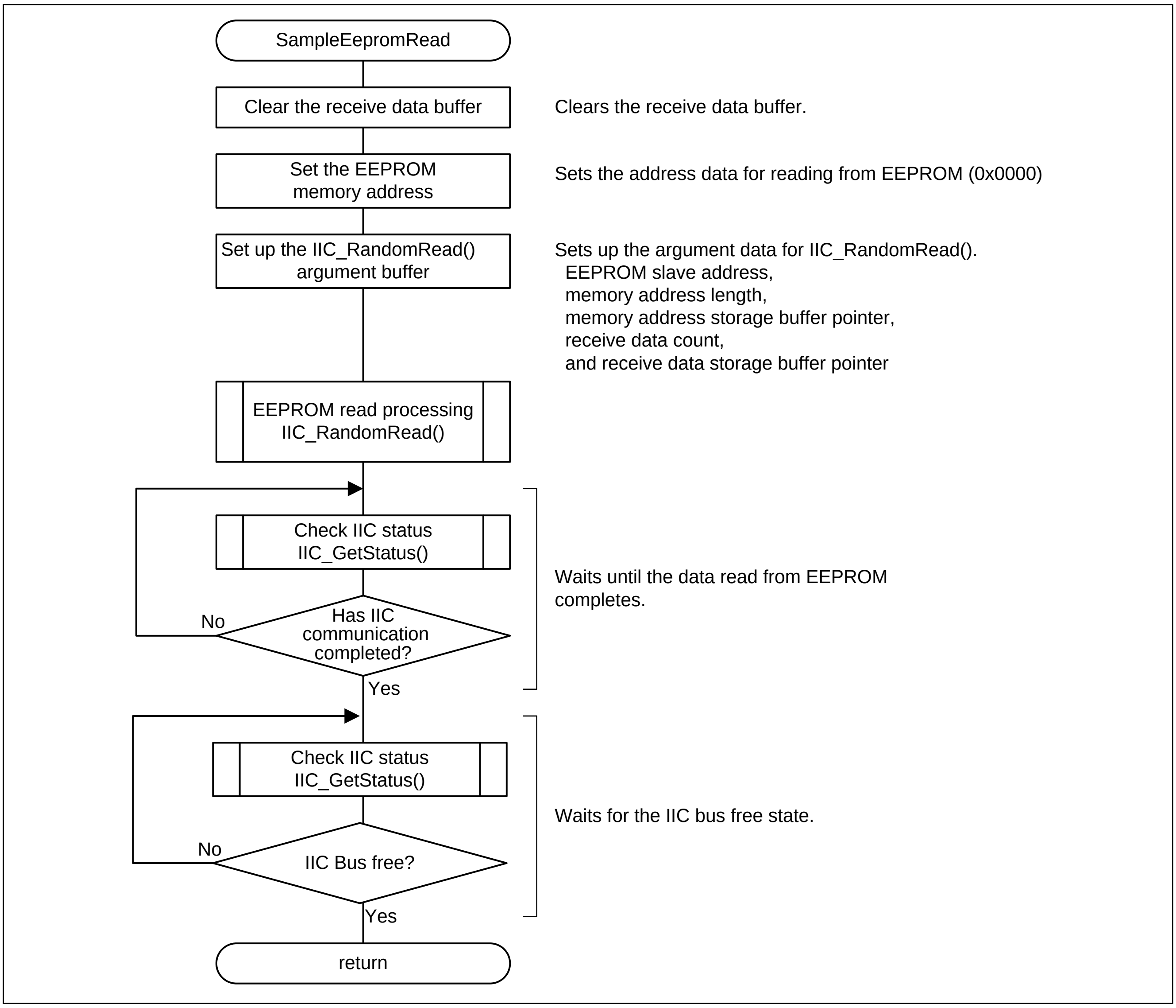

Figure 5.12 Sample EEPROM Read Processing

5.12.7 Acknowledge Polling

Figure 5.13 shows the flowchart for acknowledge polling.

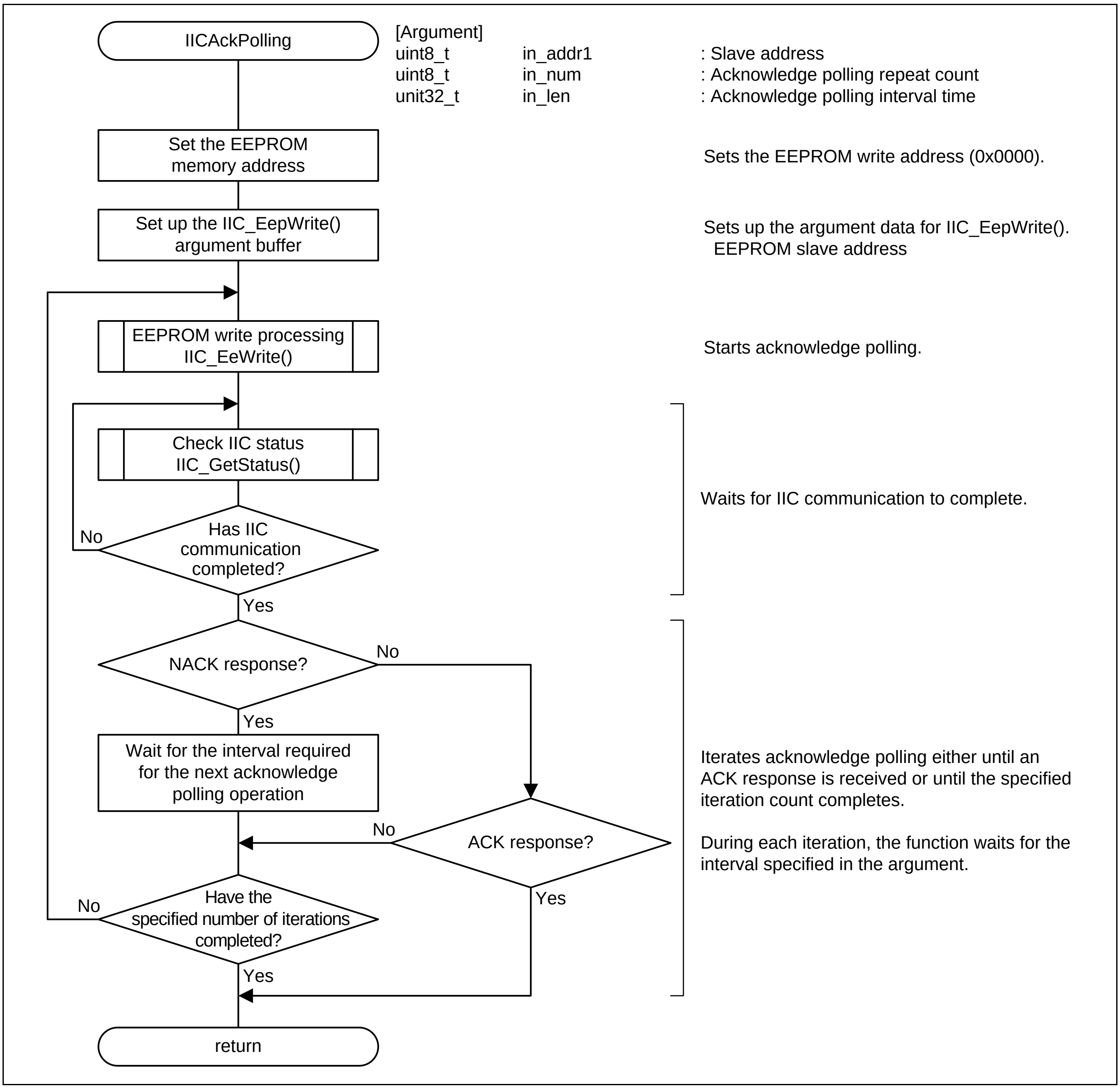

Figure 5.13 Acknowledge Polling

5.12.8 IIC Processing Setup

Figure 5.14 shows the flowchart for IIC processing setup.

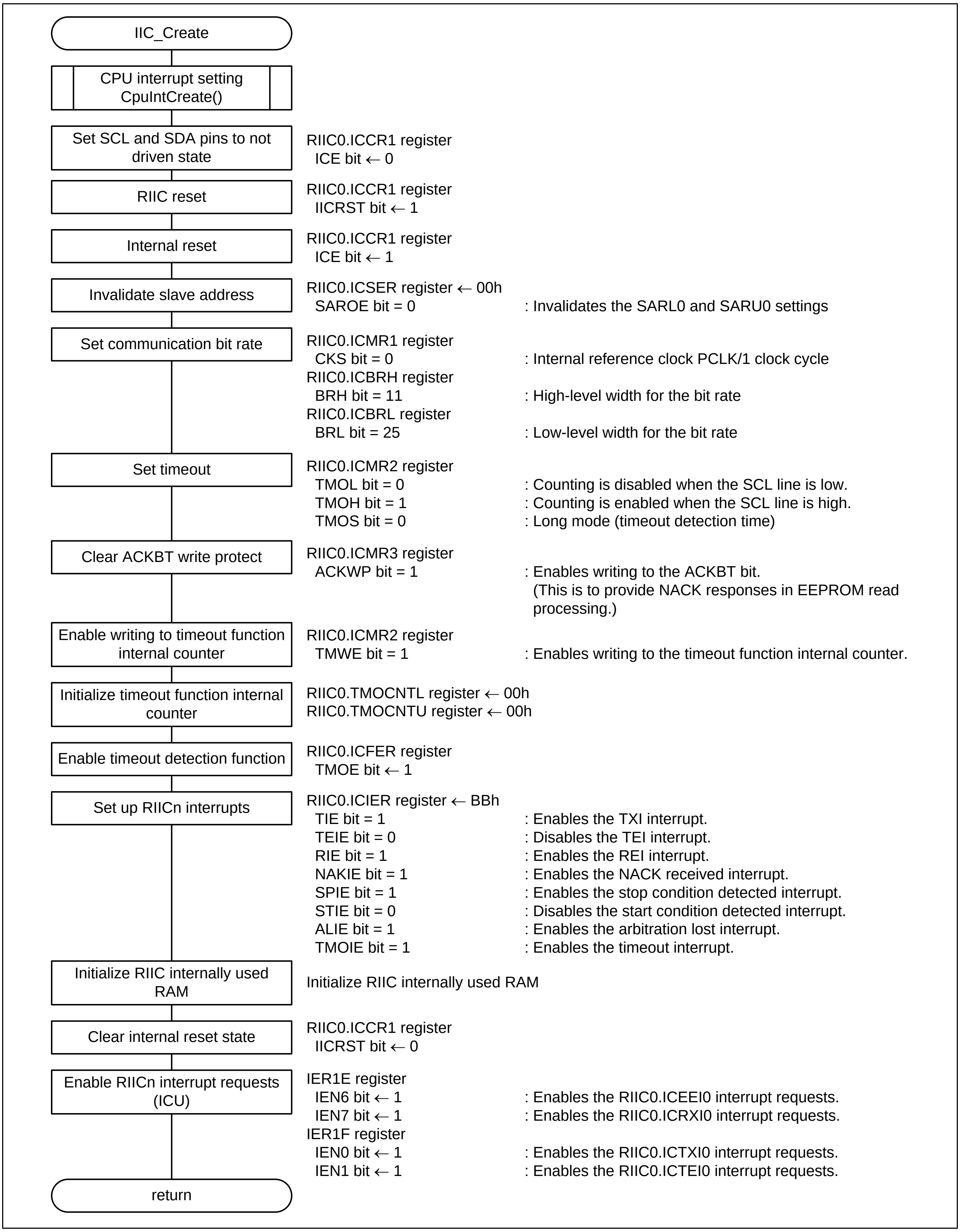

Figure 5.14 IIC Processing Setup

5.12.9 IIC Termination Processing

Figure 5.15 shows the flowchart for IIC termination processing.

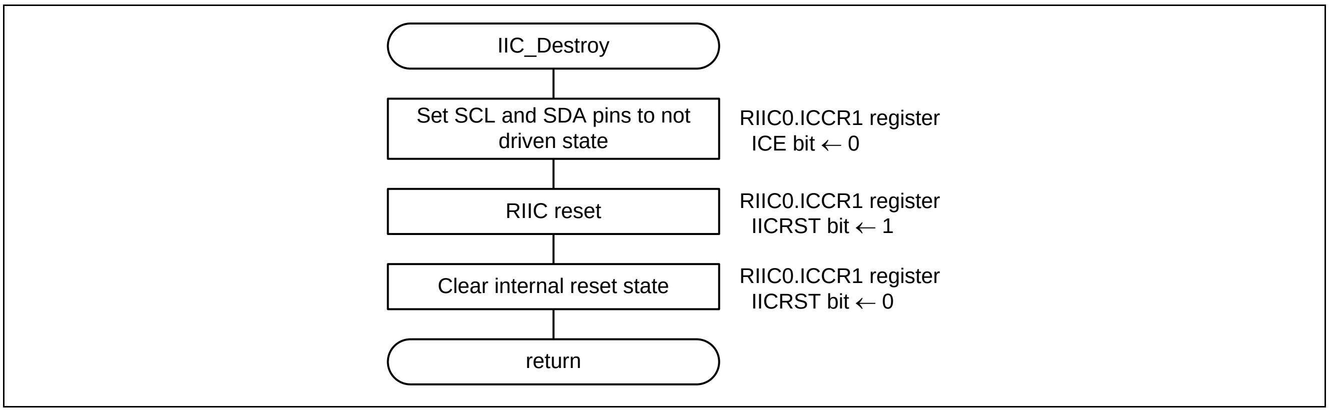

Figure 5.15 IIC Termination Processing

5.12.10 EEPROM Write Start Processing

Figure 5.16 shows the flowchart for EEPROM write start processing.

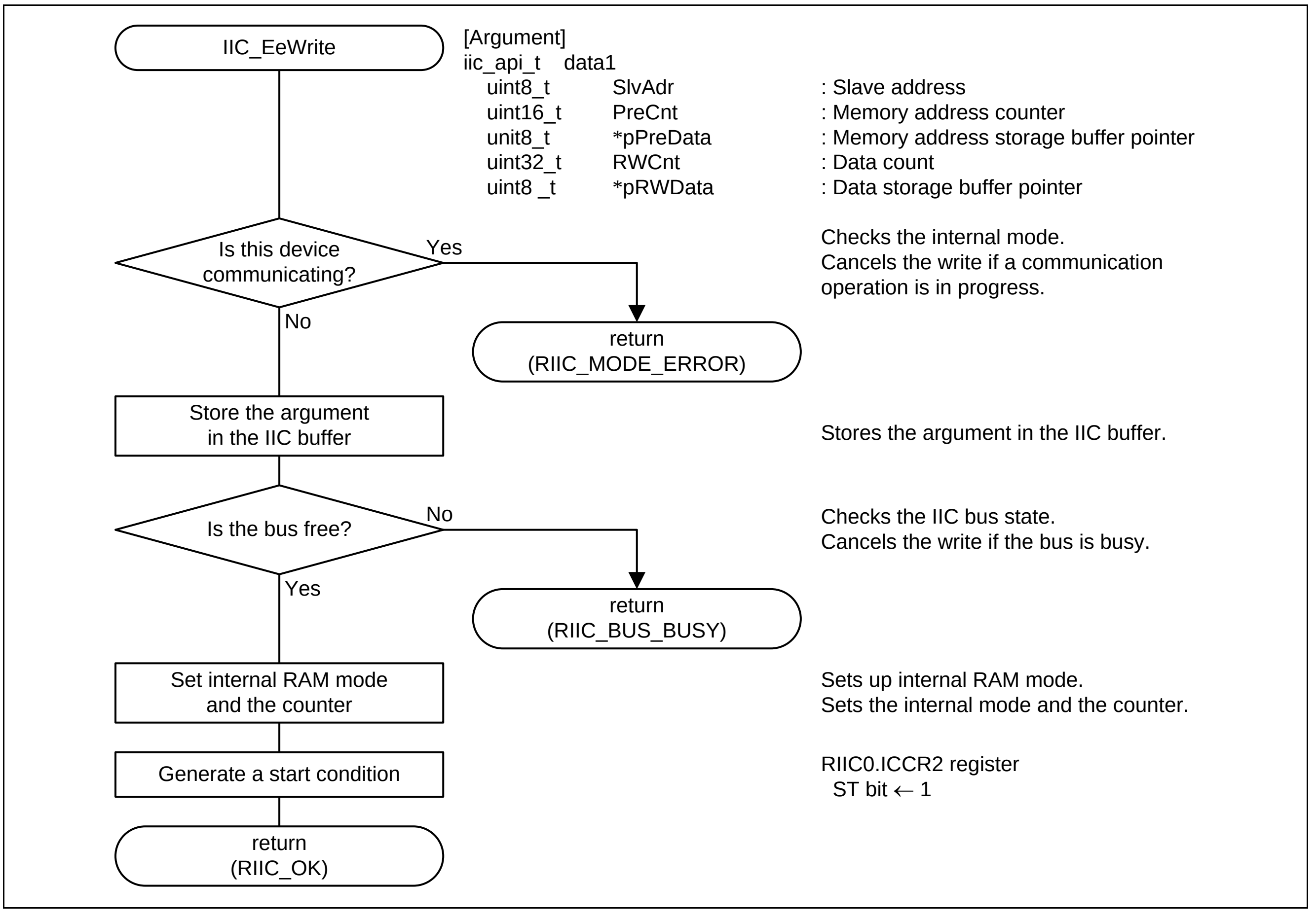

Figure 5.16 EEPROM Write Start Processing

5.12.11 EEPROM Read Start Processing

Figure 5.17 shows the flowchart for EEPROM read start processing.

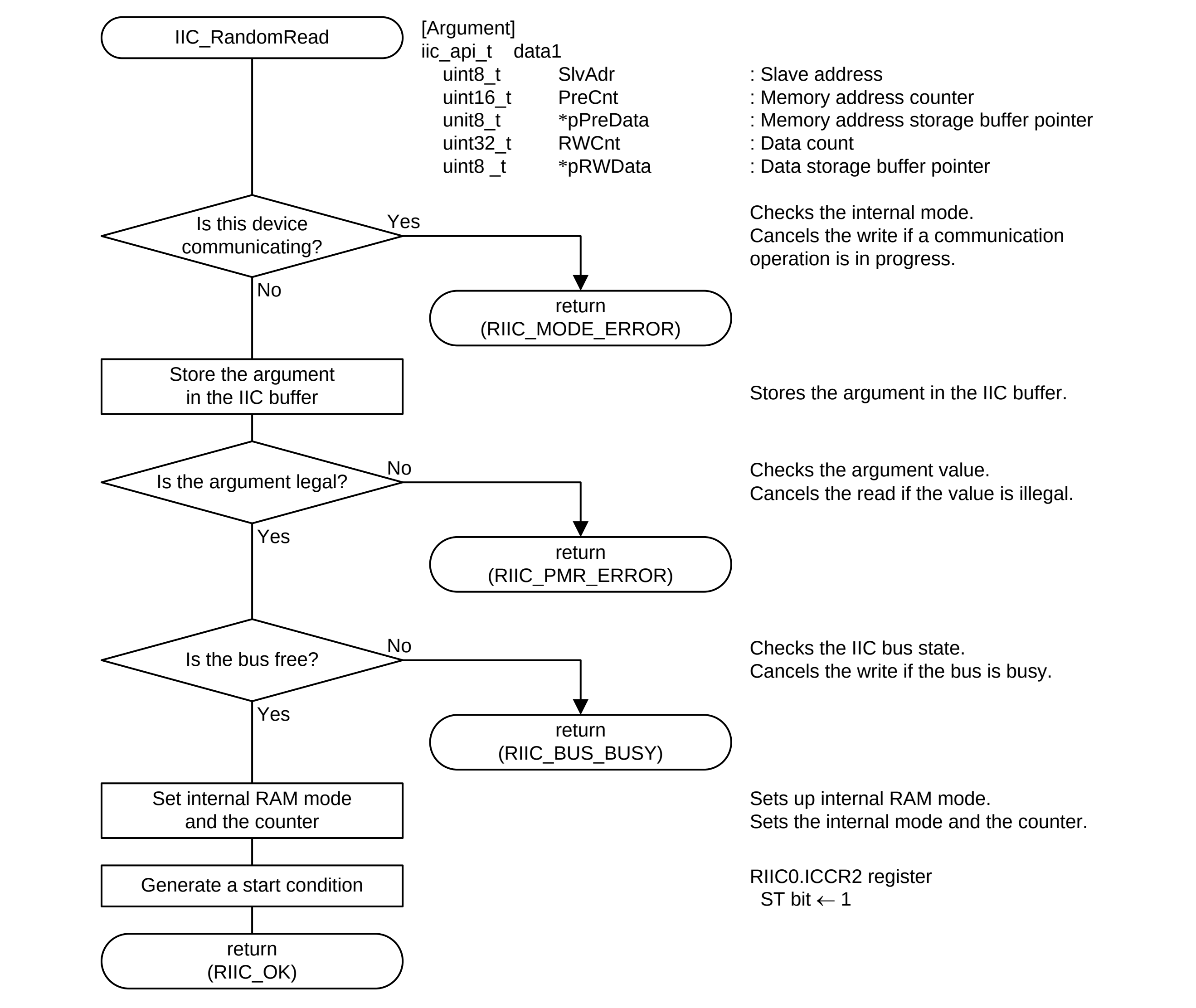

Figure 5.17 EEPROM Read Start Processing

5.12.12 IIC Status Check

Figure 5.18 shows the flowchart for IIC status check.

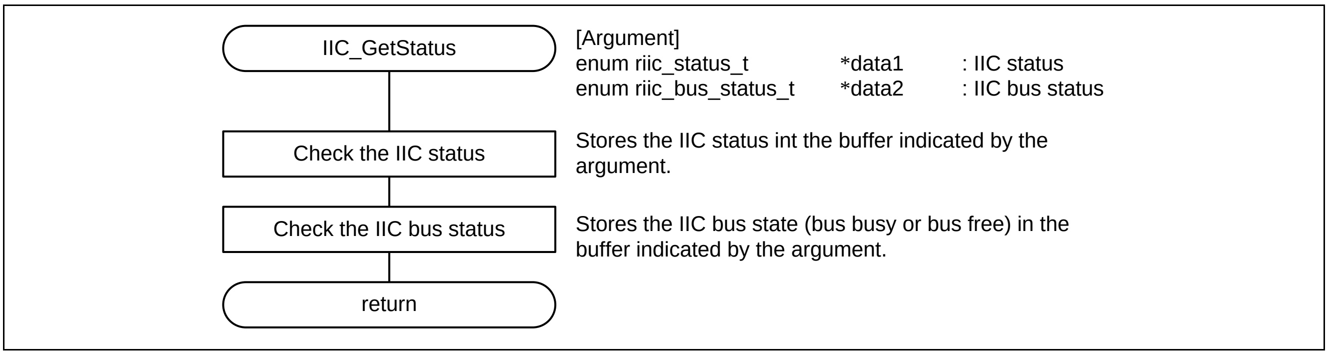

Figure 5.18 IIC Status Check

5.12.13 Communication Error or Event Interrupt

Figure 5.19 shows the flowchart for communication error or event interrupt.

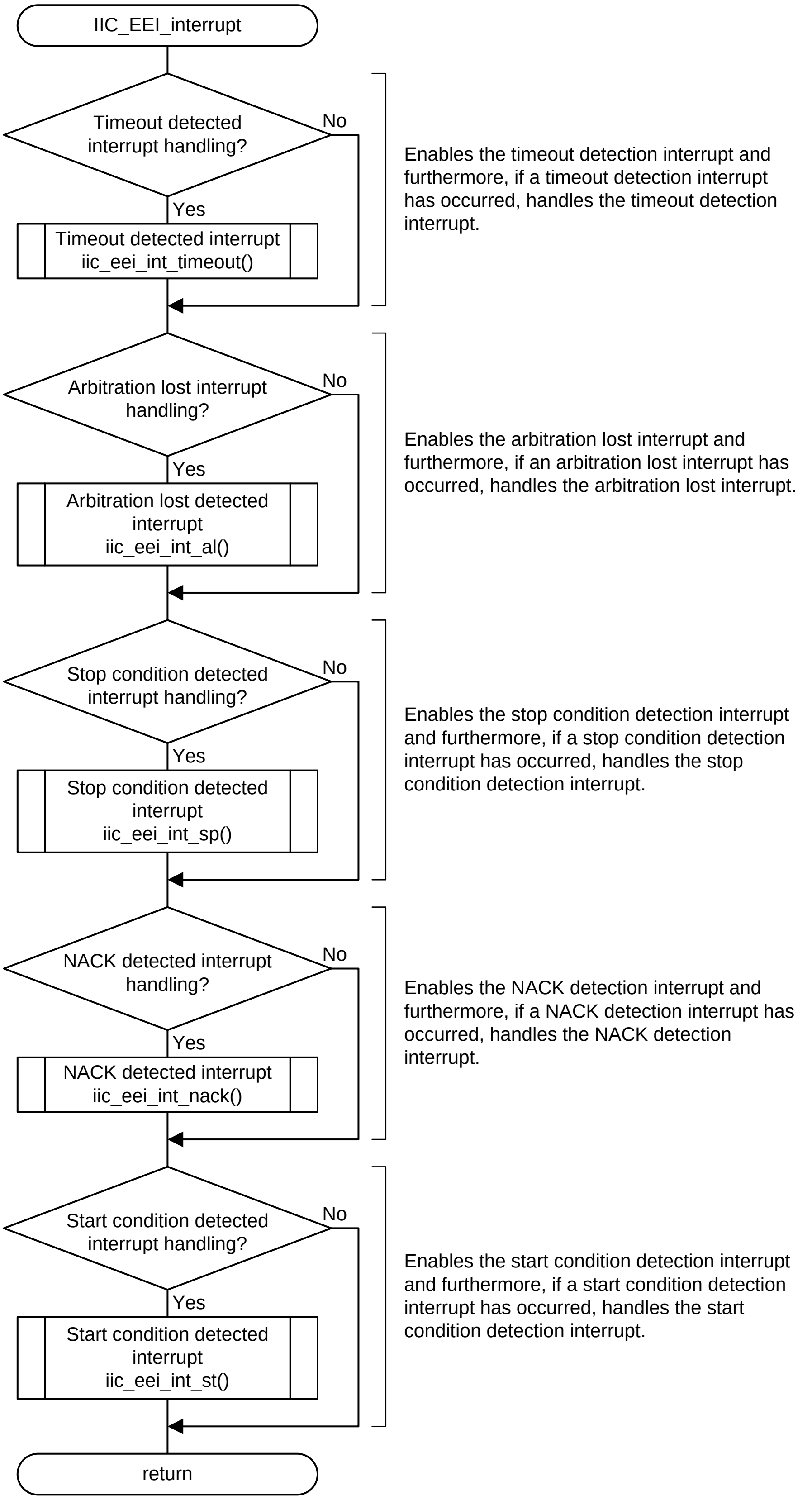

Figure 5.19 Communication Error or Event Interrupt

5.12.14 Timeout Detection Interrupt

Figure 5.20 shows the flowchart for timeout detection interrupt.

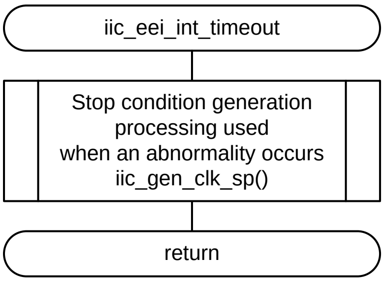

Figure 5.20 Timeout Detection Interrupt

5.12.15 Arbitration Lost Detected Interrupt

Figure 5.21 shows the flowchart for arbitration lost detected interrupt.

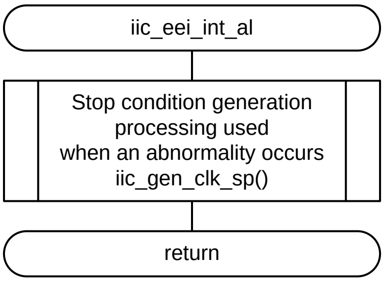

Figure 5.21 Arbitration Lost Detected Interrupt

5.12.16 Stop Condition Detected Interrupt

Figures 5.22 and 5.23 shows the flowchart for stop condition detected interrupt.

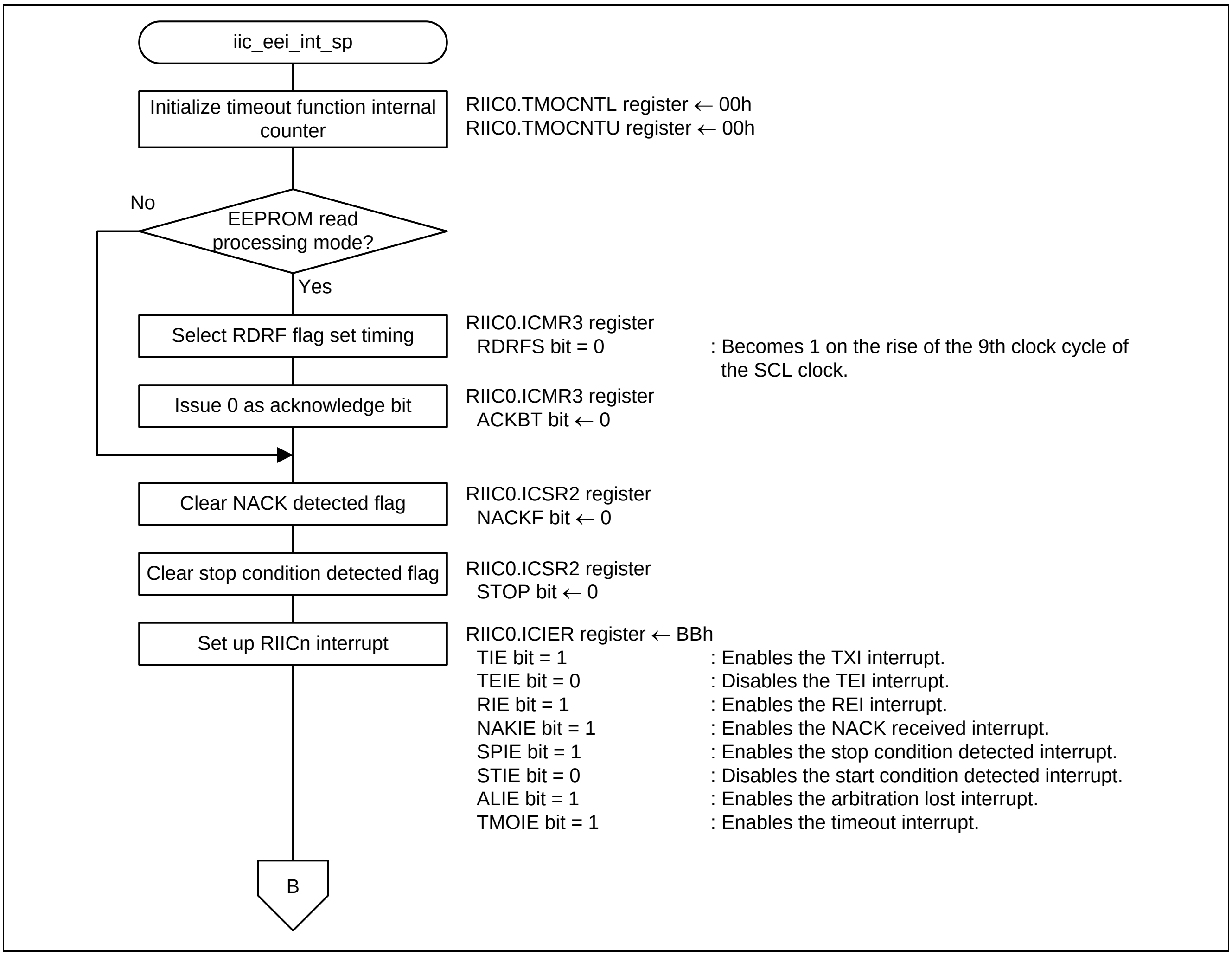

Figure 5.22 Stop Condition Detected Interrupt (1/2)

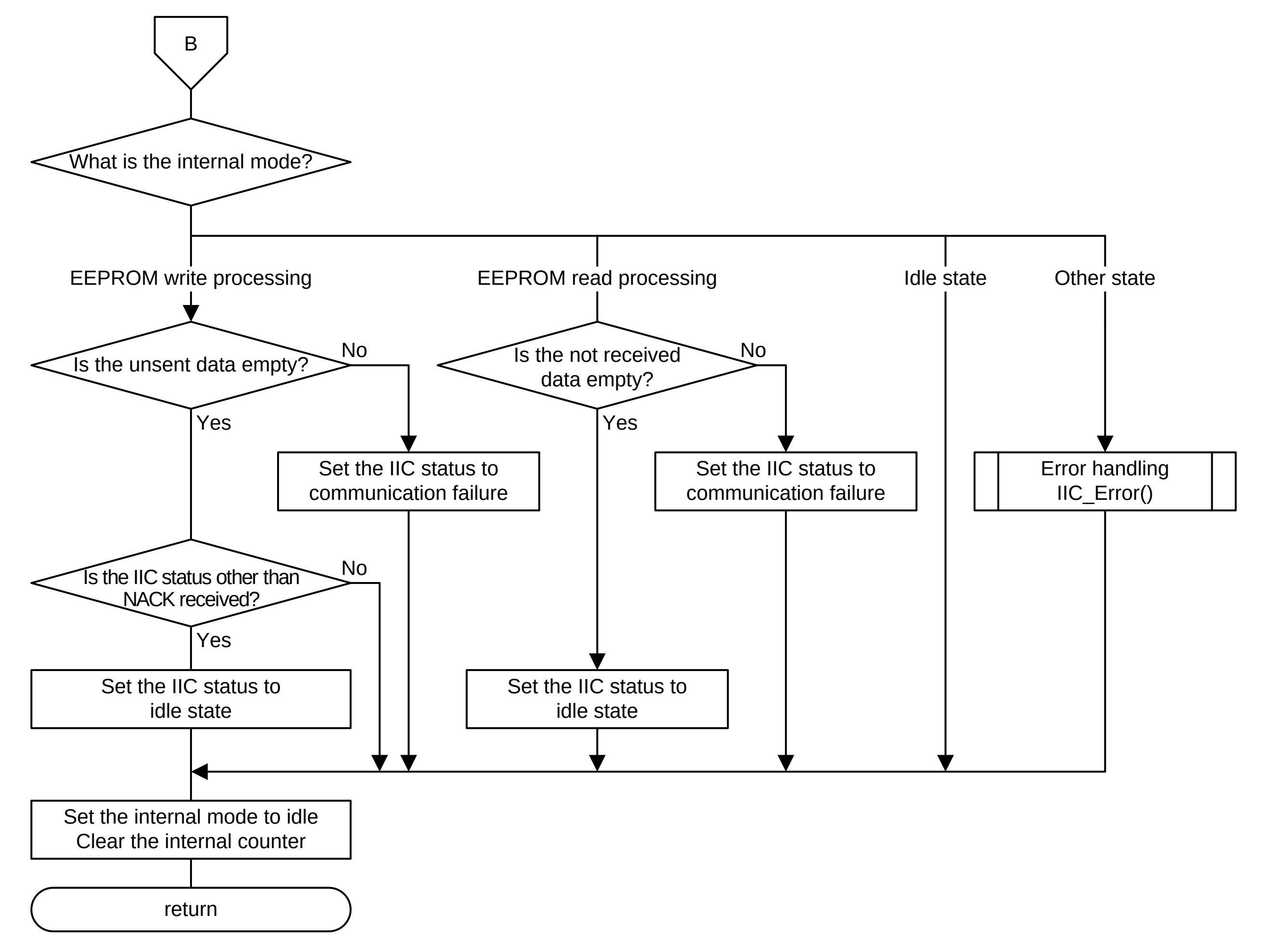

Figure 5.23 Stop Condition Detected Interrupt (2/2)

5.12.17 Start Condition Detected Interrupt

Figure 5.24 shows the flowchart for start condition detected interrupt.

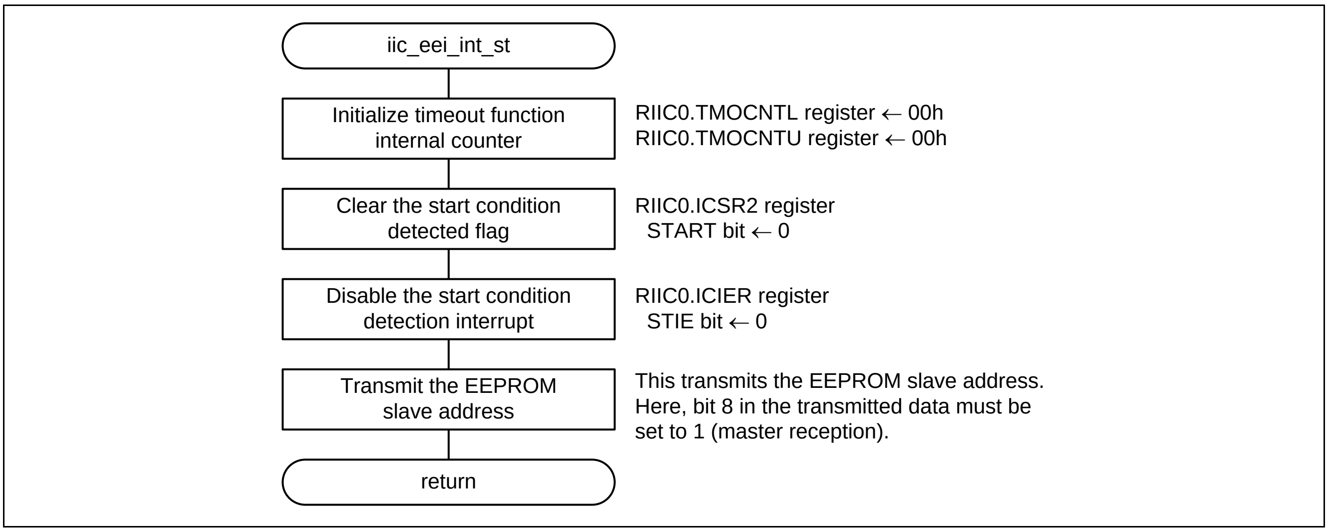

Figure 5.24 Start Condition Detected Interrupt

5.12.18 NACK Detected Interrupt

Figure 5.25 shows the flowchart for NACK detected interrupt.

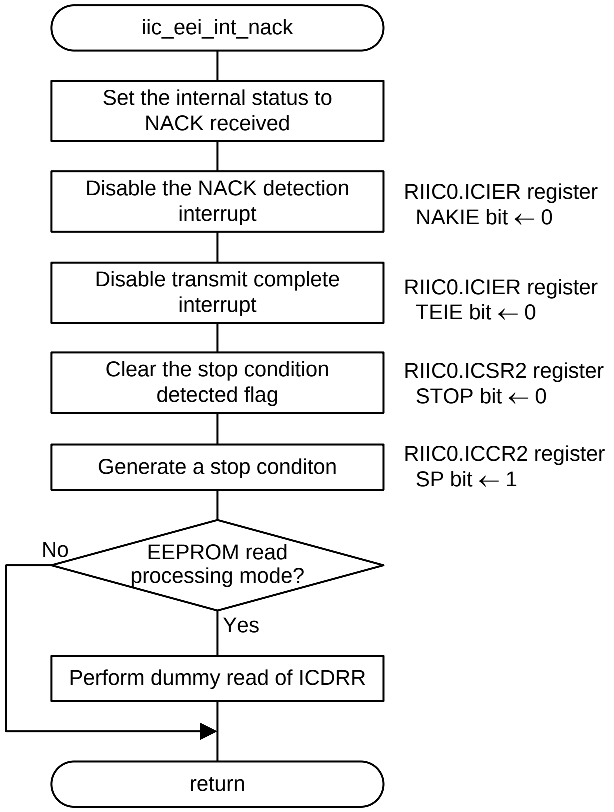

Figure 5.25 NACK Detected Interrupt

5.12.19 Receive Data Full Interrupt

Figure 5.26 shows the flowchart for receive data full interrupt.

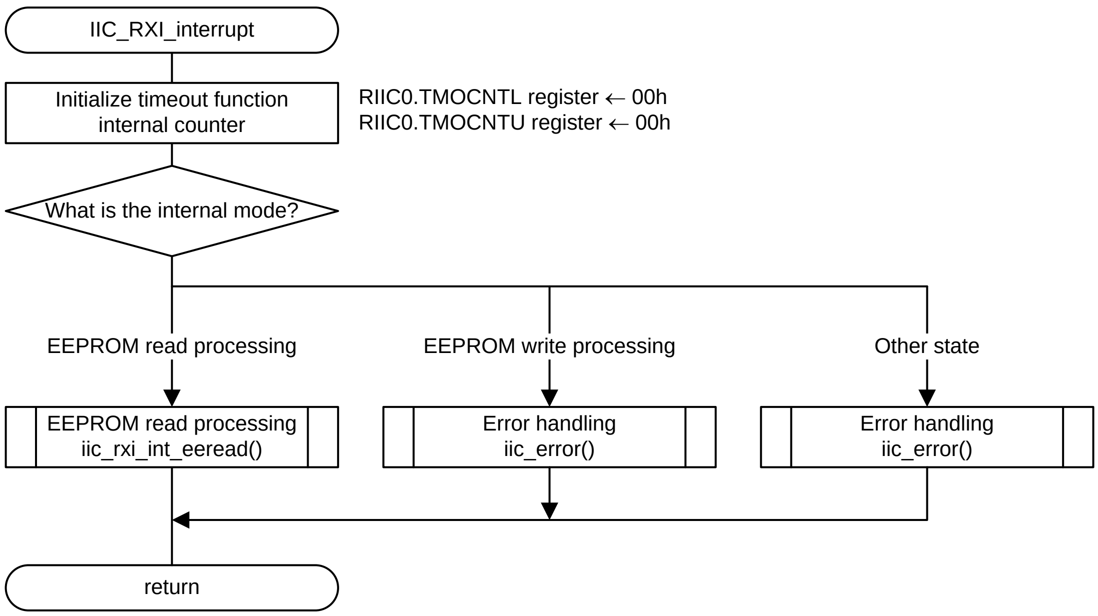

Figure 5.26 Receive Data Full Interrupt

5.12.20 EEPROM Read Processing (Master Reception Section)

Figure 5.27 shows the flowchart for EEPROM read processing (master reception section).

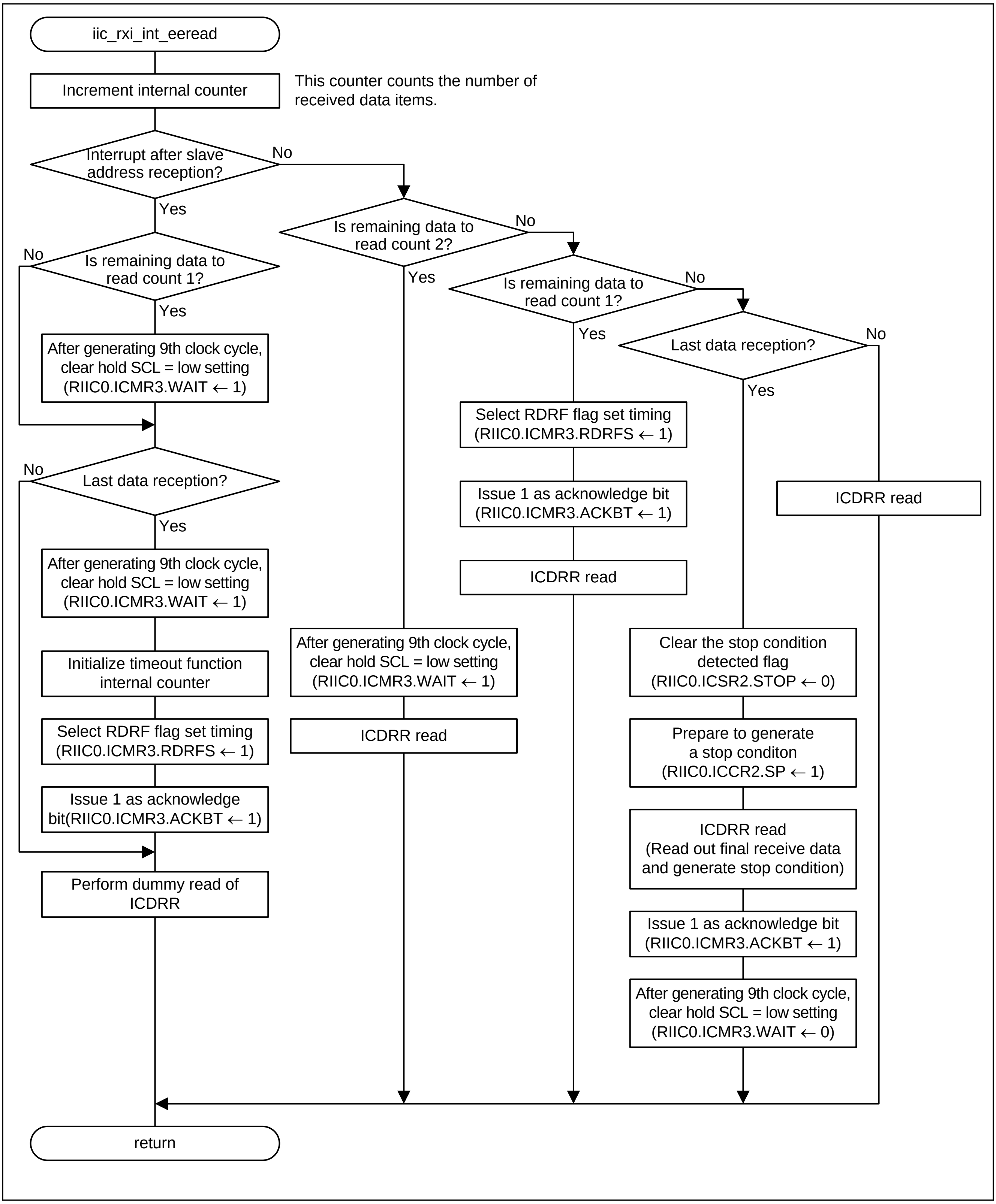

Figure 5.27 EEPROM Read Processing (Master Reception Section)

5.12.21 Transmit Data Empty Interrupt

Figure 5.28 shows the flowchart for transmit data empty interrupt.

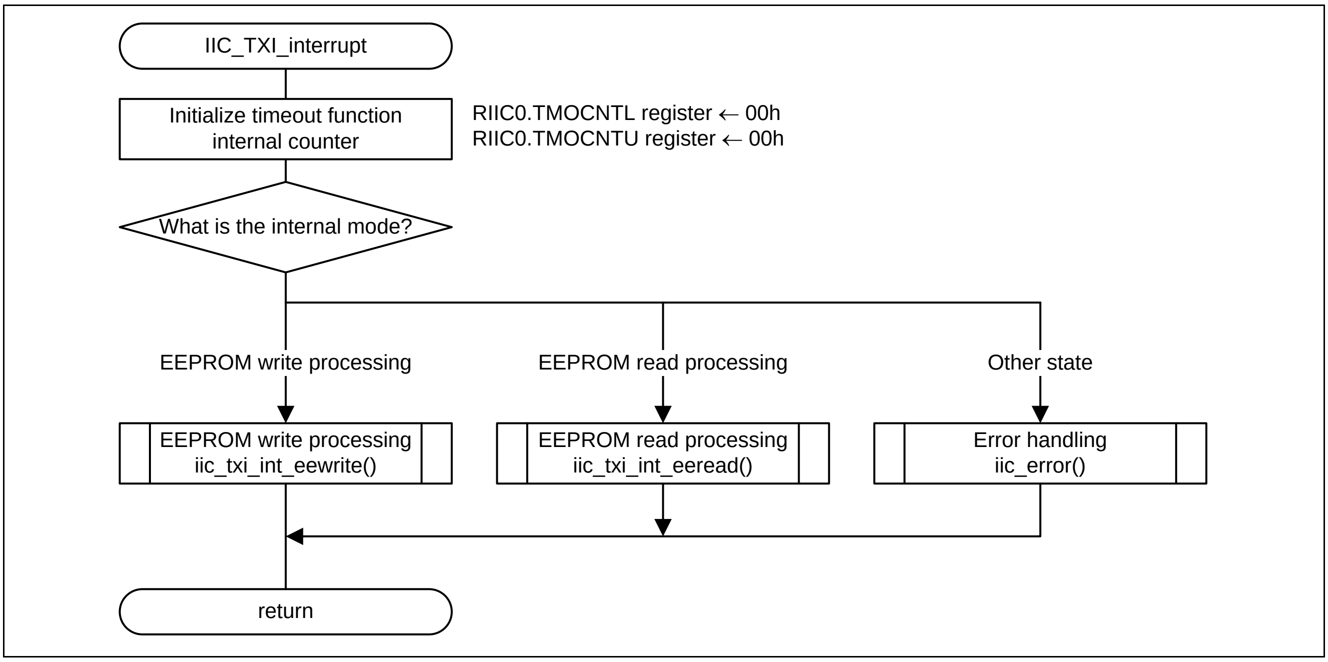

Figure 5.28 Transmit Data Empty Interrupt

5.12.22 EEPROM Write Processing

Figure 5.29 shows the flowchart for EEPROM write processing.

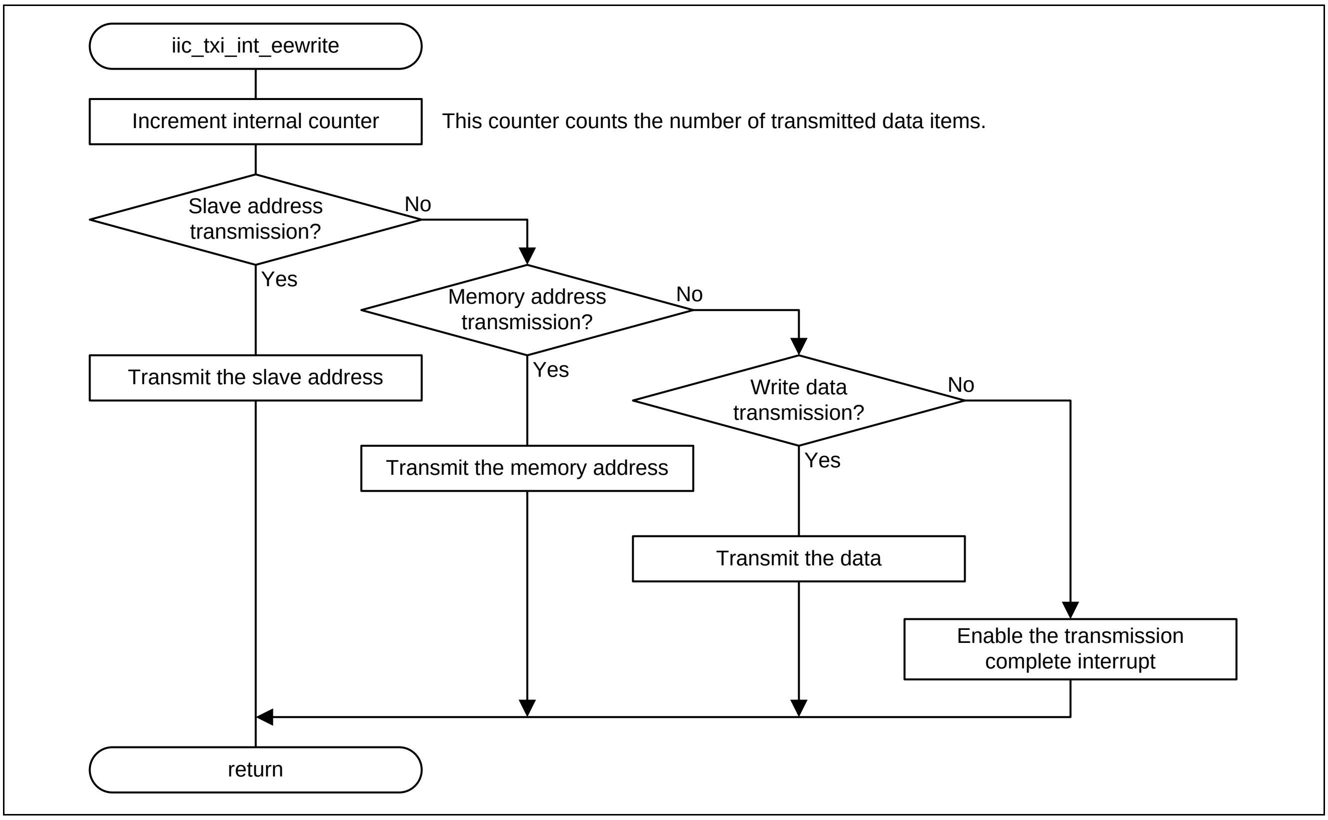

Figure 5.29 EEPROM Write Processing

5.12.23 EEPROM Read Processing (Master Transmission Section)

Figure 5.30 shows the flowchart for EEPROM read processing (master transmission section).

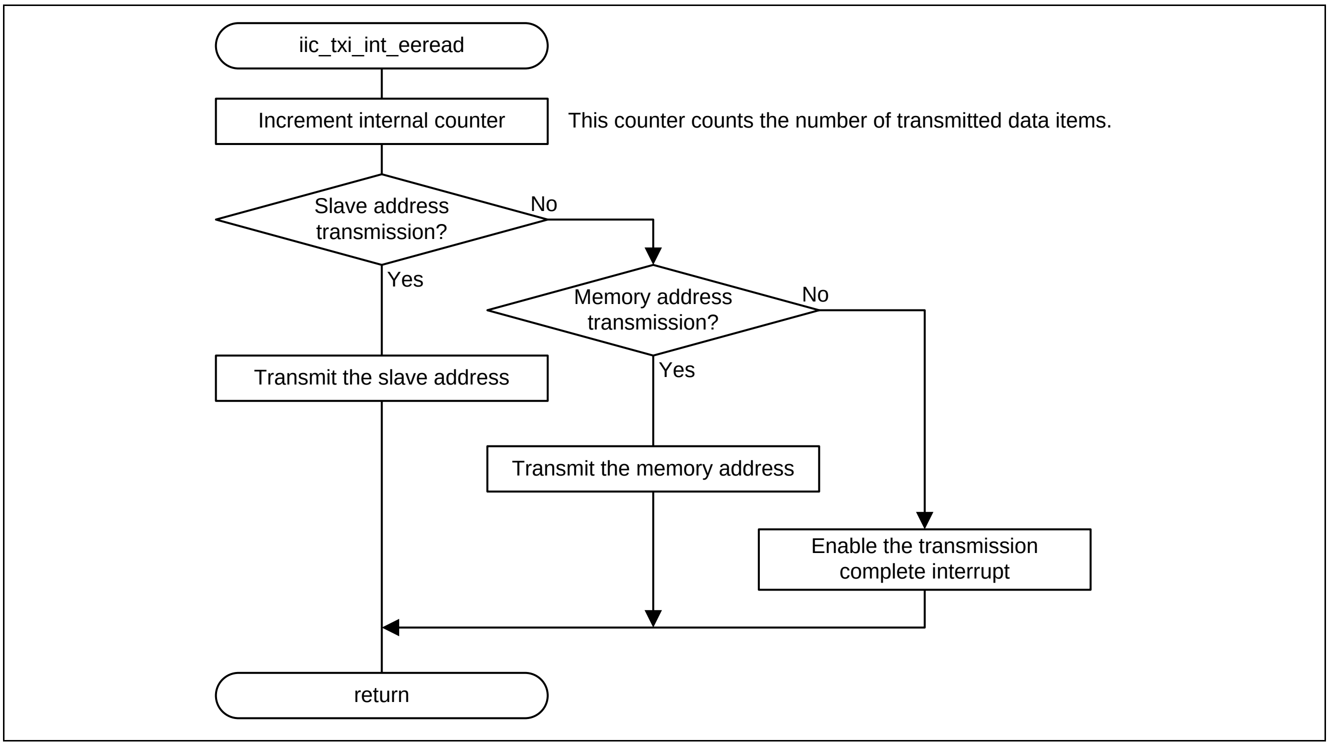

Figure 5.30 EEPROM Read Processing (Master Transmission Section)

5.12.24 Transmission Complete Interrupt

Figure 5.31 shows the flowchart for transmission complete interrupt.

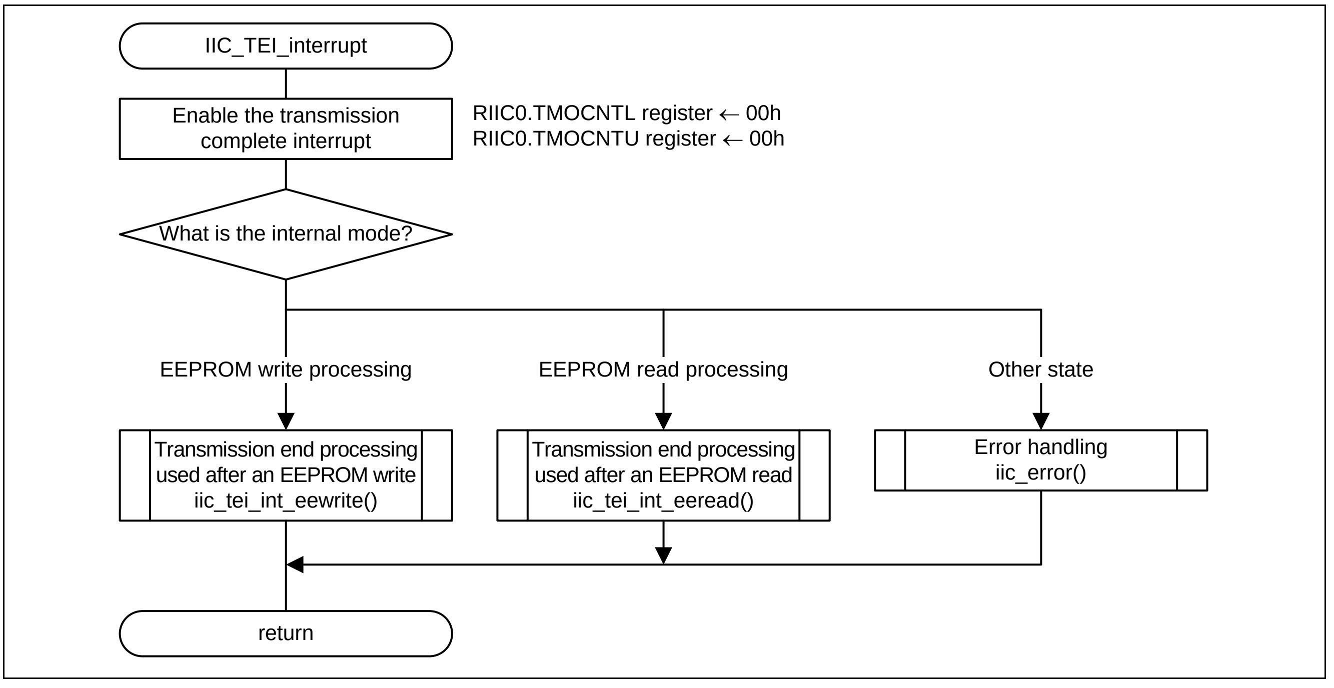

Figure 5.31 Transmission Complete Interrupt

5.12.25 Transmission End Processing Used After an EEPROM Write

Figure 5.32 shows the flowchart for transmission end processing used after an EEPROM write.

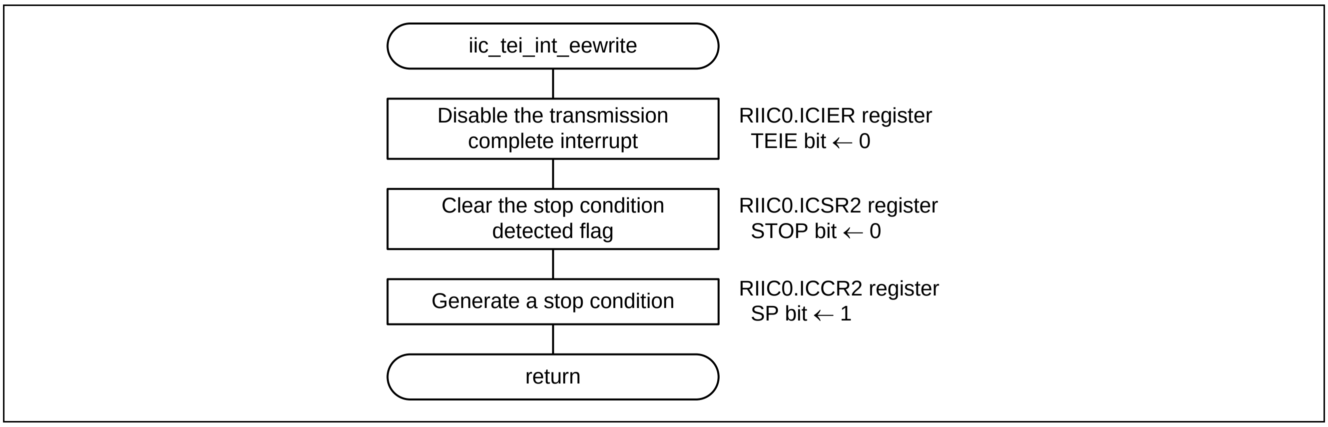

Figure 5.32 Transmission End Processing Used After an EEPROM Write

5.12.26 Transmission End Processing Used After an EEPROM Read

Figure 5.33 shows the flowchart for transmission end processing used after an EEPROM read.

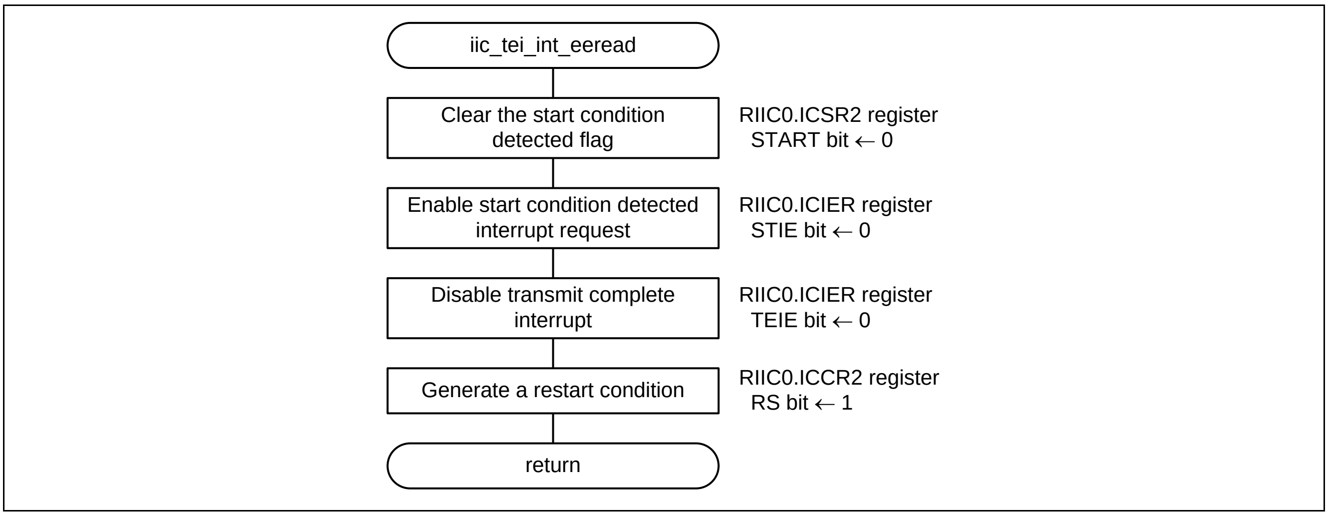

Figure 5.33 Transmission End Processing Used After an EEPROM Read

5.12.27 Stop Condition Generation Used When an Error Occurs

Figure 5.34 and Figure 5.35 shows the flowchart for stop condition generation used when an error occurs.

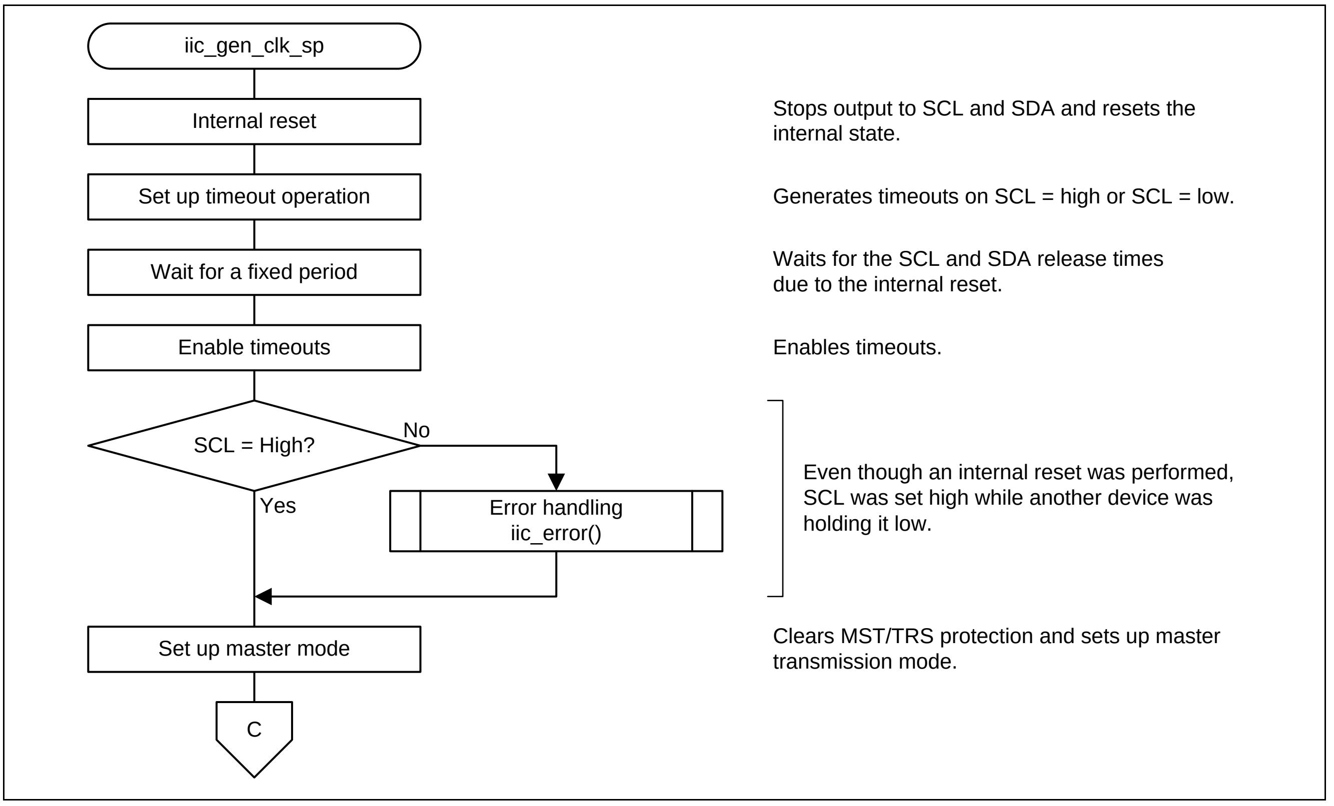

Figure 5.34 Stop Condition Generation Used When an Error Occurs (1/2)

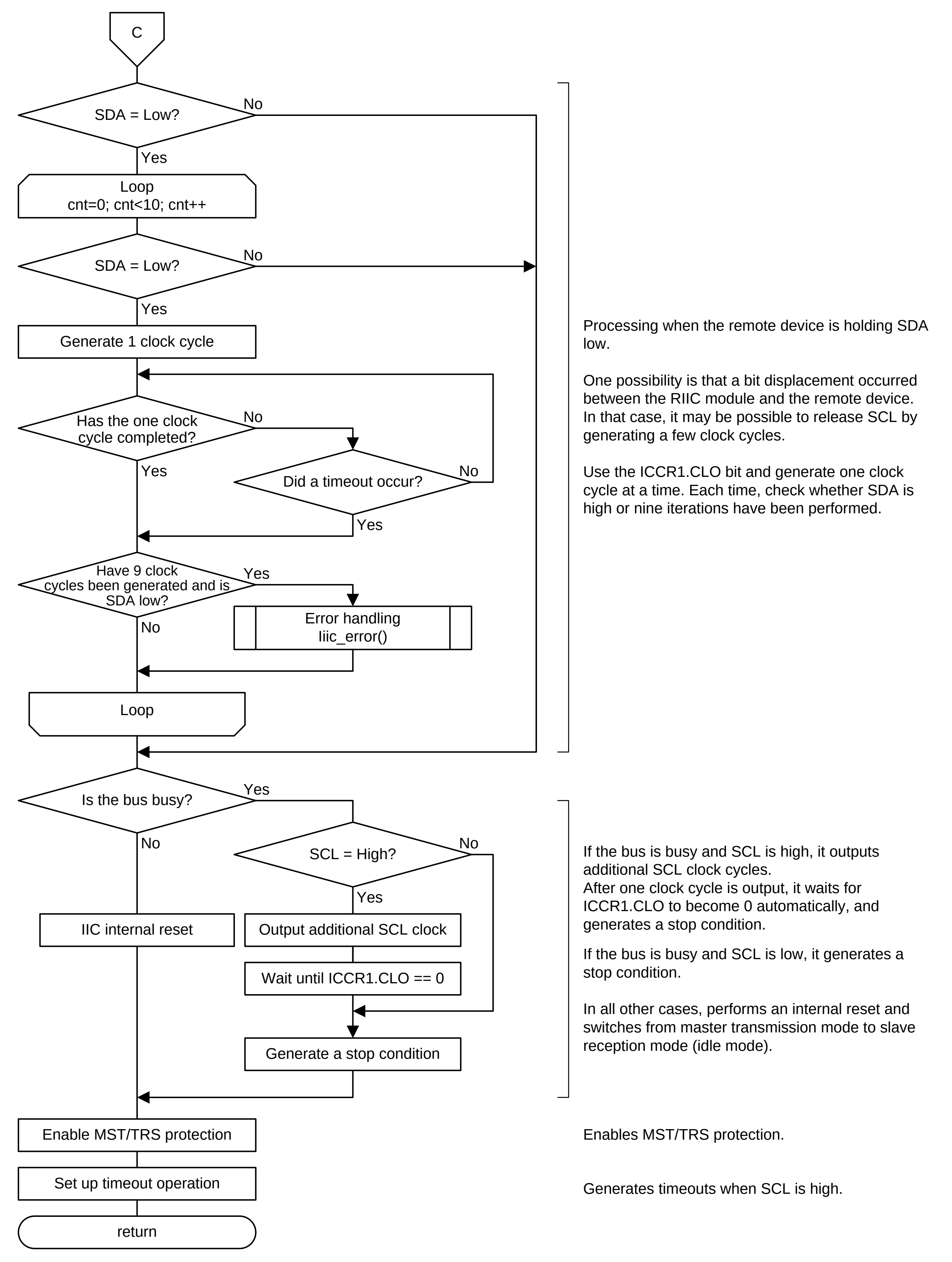

Figure 5.35 Stop Condition Generation Used When an Error Occurs (2/2)

5.12.28 Error Handling

Figure 5.36 shows the flowchart for error handling.

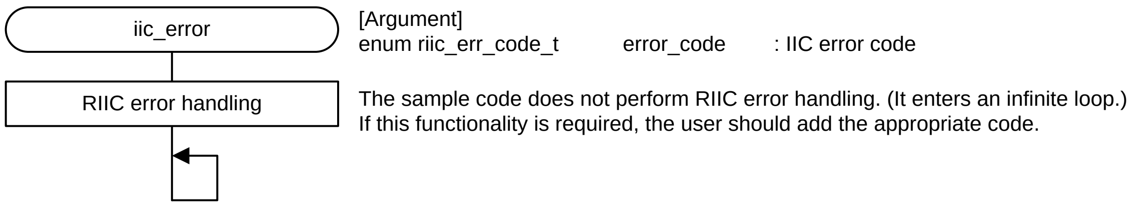

Figure 5.36 Error Handling

5.12.29 RIICn.EEIn Interrupt Handler (Level Detection Interrupt)

Figure 5.37 shows the flowchart for RIICn.EEIn interrupt handler (level detection interrupt).

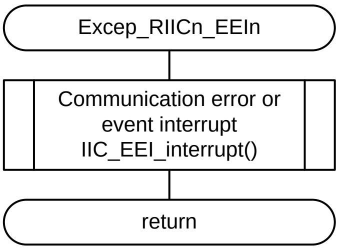

Figure 5.37 RIICn.EEIn Interrupt Handler (Level Detection Interrupt)

5.12.30 RIICn.RXIn Interrupt Handler (Edge Detection Interrupt)

Figure 5.38 shows the flowchart for RIICn.RXIn interrupt handler (edge detection interrupt).

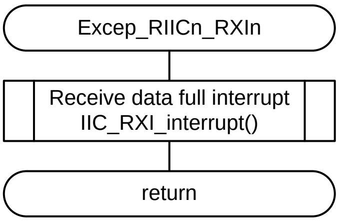

Figure 5.38 RIICn.RXIn Interrupt Handler (Edge Detection Interrupt)

5.12.31 RIICn.TXIn Interrupt Handler (Edge Detection Interrupt)

Figure 5.39 shows the flowchart for RIICn.TXIn interrupt handler (edge detection interrupt).

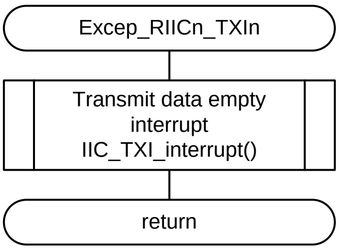

Figure 5.39 RIICn.TXIn Interrupt Handler (Edge Detection Interrupt)

5.12.32 RIICn.TEIn Interrupt Handler (Level Detection Interrupt)

Figure 5.40 shows the flowchart for RIICn.TEIn interrupt handler (level detection interrupt).

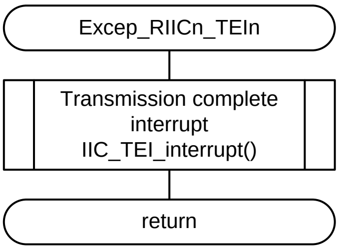

Figure 5.40 RIICn.TEIn Interrupt Handler (Level Detection Interrupt)

6. Sample Code

The sample code can be downloaded from the Renesas Electronics Corporation web site.

7. Usage Notes

7.1 Sample Code Usage Notes

Either an RX220 Group or an RX21A Group device can be selected in the sample code. Use the following settings when selecting the device.

1. In the project tab in the workspace window in the High-performance Embedded Workshop, set the project for the device to be used to be the active project.
   Refer to the latest version of the High-performance Embedded Workshop user's manual for the procedure for setting the active project.

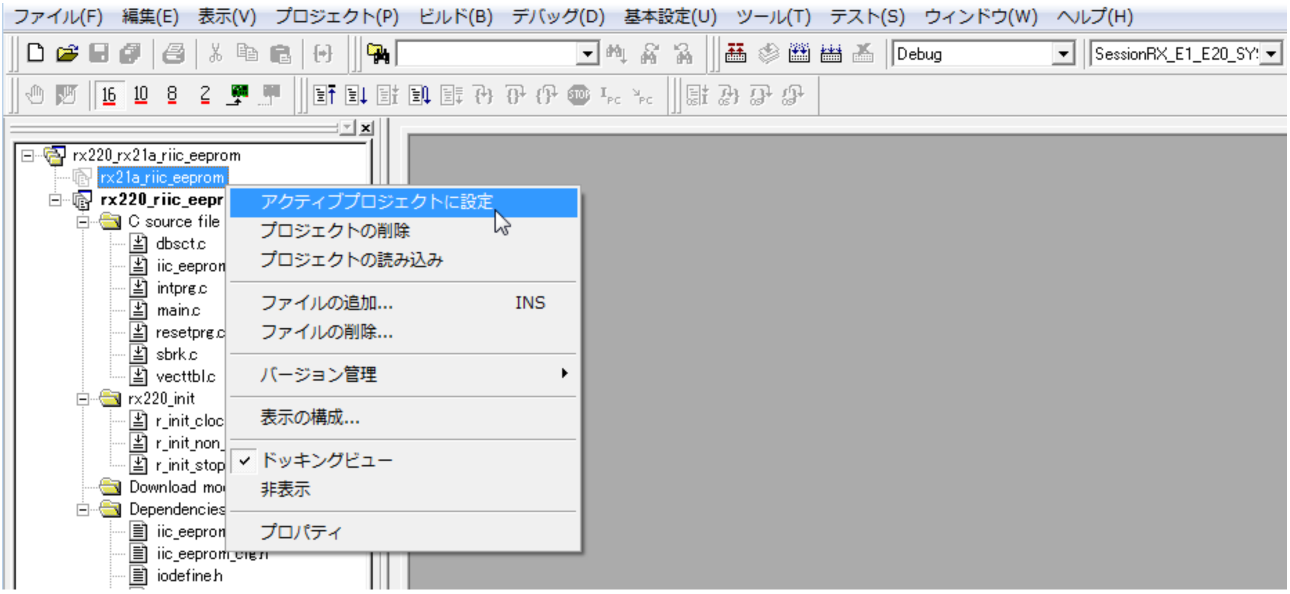

Figure 7.1 Setting the Active Project

2. Select the device used in the device configuration file (iic_eeprom_cfg.h). Uncomment the code for the device used and comment out the codes for unused devices.

7.2 Board Usage Notes

Keep the following points in mind when verifying the operation of the sample code for the board used as stipulated in this application note.

Board used: Hokuto Denshi Co., Ltd. HSB Series microcontroller boards (Product Part Number: HSBRX21AP-B)

No EEPROM is mounted on the Hokuto Denshi Co., Ltd. HSB Series microcontroller boards. To verify operation, EEPROM must be provided separately.

8. Reference Documents

User's Manual: Hardware

RX220 Group User's Manual: Hardware Rev.1.00 (R01UH0292EJ)
RX21A Group User's Manual: Hardware Rev.1.00 (R01UH0251EJ)
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.

High-performance Embedded Workshop V.4.09 User's Manual Rev.1.00 (R20UT0372EJ)
The latest version can be downloaded from the Renesas Electronics website.

Datasheets: EEPROM

R1EX24016ASAS0A Datasheet V.1.00 Rev.1.00 (rej03c270_r1ex24016axxs0abs)
R1EX24512ASAS0A Datasheet V.1.00 Rev.1.00 (rej03c249_r1ex24512axxs0abs)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

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

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	May 07, 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 accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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