RX610 Group

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

R01AN0270EJ0100
Rev.1.00
Jan 18, 2011

Introduction

This application note presents a sample program that communicates with an EEPROM (in single master mode) using the Renesas MCU I²C bus interface module.

Target Device

The RX610 Group products

Other members of the RX Family that have the same I/O registers (peripheral unit control registers) as the RX610 Group products can also use the code from this application note. Note, however, that since certain aspects of the functions used may be changed in other devices due to function additions or other differences, the documentation for the device used must be checked carefully before using this code. When using this code in an end product or other application, its operation must be tested and evaluated thoroughly.

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.

1.1 Connection Diagram

Figure 1 shows the connections in the application example presented in this application note.

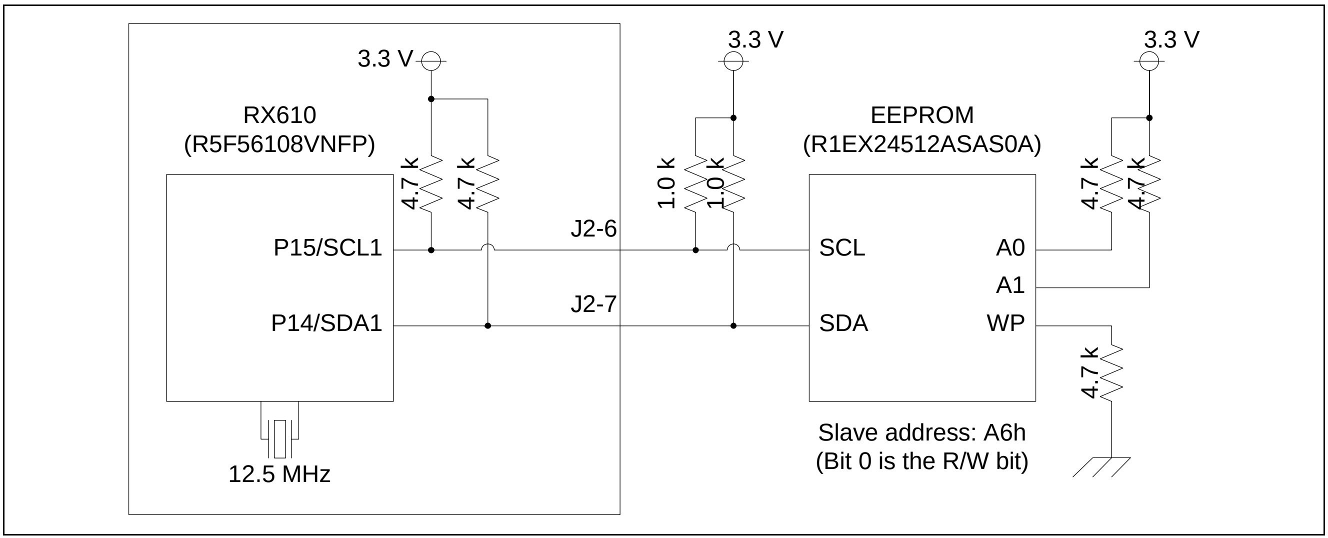

Figure 1 Connection Diagram

1.2 RIIC Settings

Table 1 lists the RIIC settings described in this application note.

Table 1 RIIC Settings

Item	Settings
Operating frequencies	• Input clock (EXTAL): 12.5 MHz • System clock (ICLK): 100 MHz • Peripheral module clock (PCLK): 50 MHz • External bus clock (BCLK): 25 MHz • Internal reference clock (IICϕ): 50 MHz
Master/slave	Single master
Address format	7-bit address format
Transfer speed	1 Mbps
Timeout detection	• The detection function counts while the SCLn line is low. • Long mode (16-bit counter (IICϕ): about 1.31 ms)

1.3 EEPROM

Table 2 lists the specifications of the EEPROM used in the application example described in this application note.

Table 2 EEROM Specifications

Item	Description
Catalog number	R1EX24512ASAS0A
Capacity	512k (64-kword × 8-bit)
Slave address	Slave address: A6h Bit 0 is the R/W bit. Bits 1 and 2 depend on the values of the A0 and A1 pins, respectively. • A0 pin: High • A1 pin: High
Write protection	Always released. • WP pin: low

2. Operation Confirmation Environment

Table 3 lists the environment used for confirming the operation of this application example.

Table 3 Operation Confirmation Environment

Item	Description
Device	RX610 (R5F56108VNFP)
Board	Evaluation board
Power supply voltage	3.3 V (Supplied from E1)
Input clock	12.5 MHz (ICLK = 100 MHz, PCLK = 50 MHz, BCLK = 25 MHz)
Operating temperature	Room temperature
HEW	Version 4.07.01.004
Toolchain	RX Standard Toolchain (V.1.0.0.0)
Debugger/Emulator	E1 emulator
Debugger component	RX E1/E20 SYSTEM V.1.00.00.000

3. Operation

3.1 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 2 shows an example of the signals used when writing the EEPROM.

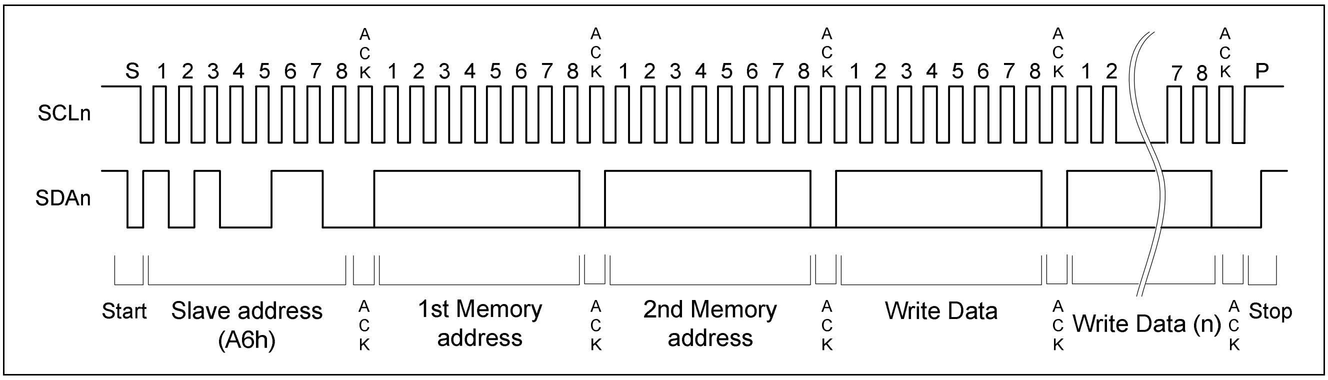

Figure 2 Signals when Writing to EEPROM

3.2 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 3 shows an example of the signals used when reading the EEPROM.

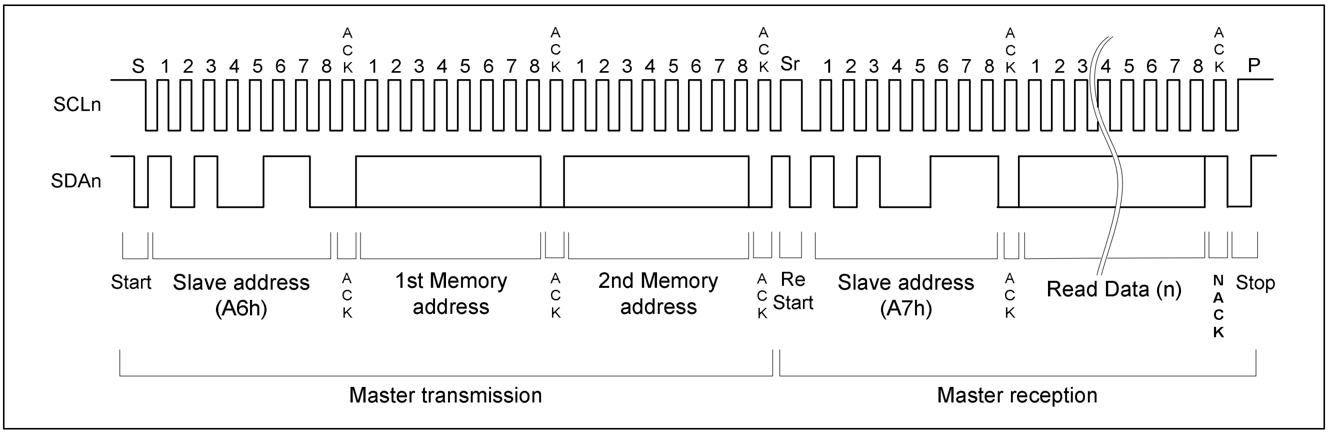

Figure 3 Signals when Reading from EEPROM

3.3 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 4 shows the acknowledge polling signals.

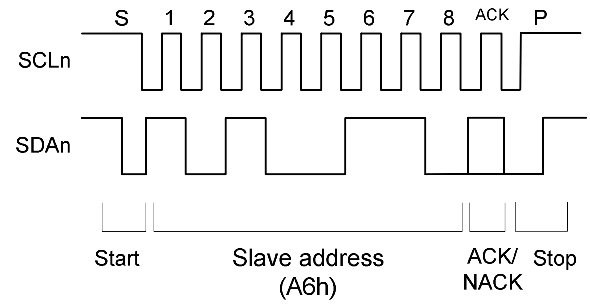

Figure 4 Acknowledge Polling Signals

4. Software

4.1 Functions

Tables 4 and 5 list the functions in this sample program. The functions that are not in bold are static functions.

Table 4 Functions in File main.c

Function Name	Operation	Notes
main	Main processing	Figure 7
SampleEepromWrite	EEPROM write processing example	Figure 11
SampleEepromRead	EEPROM read processing example	Figure 12
IICAckPolling	Acknowledge polling	Figure 13
CpuCreate	CPU initialization	Figure 8
CpuIntCreate	CPU interrupt setting	Figure 9
IICPortCreate	IIC port settings	Figure 10

Table 5 Functions in File iic.c

Function Name	Operation	Notes
IIC_Create	IIC initialization	Figure 14
IIC_Destroy	IIC termination processing	Figure 15
IIC_EepWrite	EEPROM write start processing	Figure 16
IIC_RandomRead	EEPROM read start processing	Figure 17
IIC_GetStatus	IIC status check	Figure 18
IIC_EEI_Int	Communication error or event interrupt	Figure 19
IIC_EEI_IntTimeOut	Timeout detection interrupt Called from within IIC_EEI_Int()	Figure 20
IIC_EEI_IntAL	Arbitration lost detected interrupt Called from within IIC_EEI_Int()	Figure 21
IIC_EEI_IntSP	Stop condition detected interrupt Called from within IIC_EEI_Int()	Figure 22
IIC_EEI_IntST	Start condition detected interrupt Called from within IIC_EEI_Int()	Figure 23
IIC_EEI_IntNack	NACK detected interrupt Called from within IIC_EEI_Int()	Figure 24
IIC_RXI_Int	Receive data full interrupt	Figure 25
IIC_RXI_IntEepRead	EEPROM read processing (master reception section) Called from within IIC_RXI_Int()	Figure 26
IIC_TXI_Int	Transmit data empty interrupt	Figure 27
IIC_TXI_IntEepWrite	EEPROM write processing Called from within IIC_TXI_Int()	Figure 28
IIC_TXI_IntEepRead	EEPROM read processing (master transmission section) Called from within IIC_TXI_Int()	Figure 29
IIC_TEI_Int	Transmission complete interrupt	Figure 30
IIC_TEI_IntEepWrite	Transmission end processing used after an EEPROM write Called from within IIC_TEI_Int()	Figure 31
IIC_TEI_IntEepRead	Transmission end processing used after an EEPROM read Called from within IIC_TEI_Int()	Figure 32
IIC_GenClkSP	Stop condition generation used when an error occurs Called from within IIC_EEI_IntTimeOut() and IIC_EEI_IntAL()	Figure 33
IIC_Error	Error handling	Figure 34

4.2 Variables

4.2.1 Structures

Figure 5 shows the structure used as the argument to the functions IIC_EepWrite() and IIC_RandomRead(). Also, table 6 lists the members of this 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 be set 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.

4.2.2 Functions

Tables 7 and 8 list the functions in this sample program.

Table 7 Functions in the File main.c

Function	Description
uint8_t trm_buff[256]	Transmit data buffer
uint8_t rcv_buff[256]	Receive data buffer
uint8_t trm_eeprom_adr[2]	EEPROM slave address storage buffer (for write)
uint8_t rcv_eeprom_adr[2]	EEPROM slave address storage buffer (for read)
IIC_API_T iic_buff_prm[2]	Structure used as the argument to the functions IIC_EepWrite() and IIC_RandomRead()

Table 8 Functions in the File iic.c

Function	Description
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())
static int8_t iic_mode	Internal mode
static int8_t iic_status	IIC status
static uint32_t iic_trm_cnt	Internal IIC transmit counter
static uint32_t iic_rcv_cnt	Internal IIC receive counter

4.2.3 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 9 and their state transition diagram are shown in figure 6. Also, table 10 lists the IIC bus status values, table 11 lists the internal modes, and table 12 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 9 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.

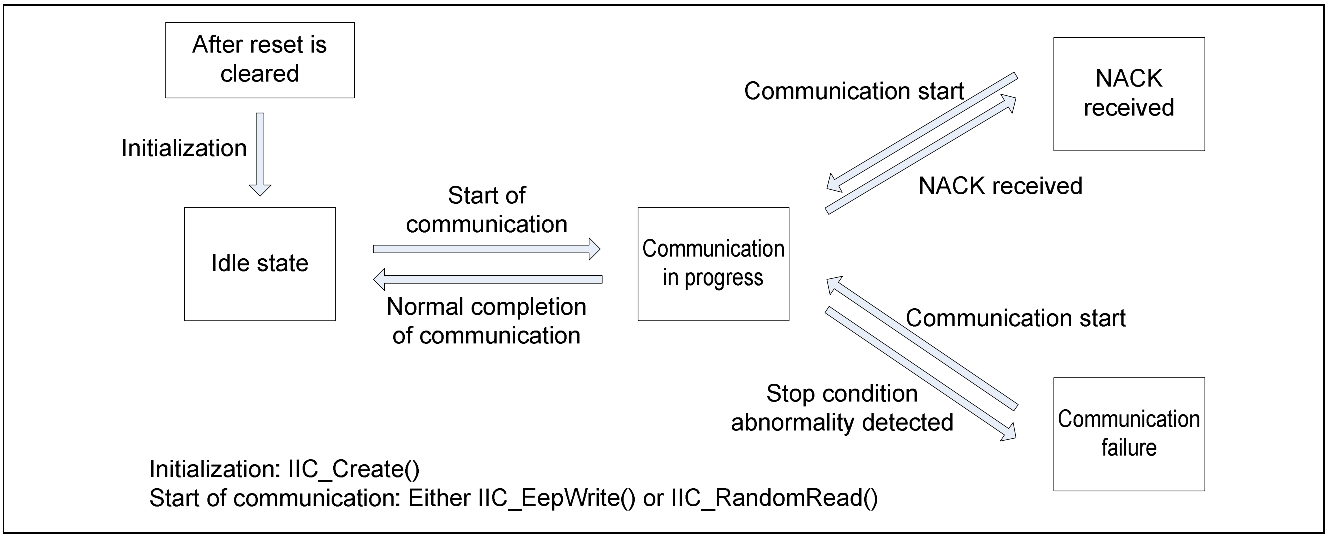

Figure 6 IIC Status State Transition Diagram

Table 10 IIC Bus Status (enum RiicBusStatus_t)

Defined Name	Description
RIIC_BUS_STATUS_FREE	IIC bus busy
RIIC_BUS_STATUS_BBSY	IIC bus free

Table 11 Internal Modes (enum RiicInternalMode_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_EEP_READ	EEPROM read mode The internal mode transitions to this mode at the start of communication due to IIC_RandomRead().
IIC_MODE_EEP_WRITE	EEPROM write mode The internal mode transitions to this mode at the start of communication due to IIC_EepWrite().

Table 12 IIC_EepWrite() and IIC_RandomRead() Return Value (enum RiicEepFnc_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.)

4.3 Function Specifications

This section presents the specifications of the sample code functions that control the RIIC module.

IIC_Create	
Overview	Initializes the RIIC module.
Header	r_apn_iic.h
Declaration	void IIC_Create(void)
Description	Performs the following settings. • Transfer speed setting: 1 Mbps • Interrupt settings • Timeout settings
Arguments	None
Return value	None
Notes	

IIC_Destroy	
Overview	Stops the RIIC module.
Header	r_apn_iic.h
Declaration	void IIC_Destroy(void)
Description	Stops the RIIC module and clears all the RIIC module related registers.
Arguments	None
Return value	None
Notes	If this function is called during a communication operation, it forcibly stops the RIIC module.

IIC_EepWrite	
Overview	Starts a write to the EEPROM.
Header	r_apn_iic.h
Declaration	int8_t IIC_EepWrite(IIC_API_T)
Description	Uses master transmission to write to the EEPROM. If the I^2C 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
Return value	If communication starts up normally: RIIC_OK If the I^2C bus is busy: RIIC_BUS_BUSY If the RIIC module is communicating: RIIC_MODE_ERROR
Notes	See section 4.2.1, Structures, for details on the argument IIC_API_T data1. See section 4.2.3, Enumerations, for details on the return value. Bit 0 in the slave address (SlvAdr), which is a member of the argument structure, must be set to 0.

IIC_RandomRead	
Overview	Starts a read from the EEPROM.
Header	r_apn_iic.h
Declaration	int8_t IIC_RandomRead(IIC_API_T);
Description	This function reads data from the EEPROM using both master transmission and master reception. If the I^2C bus is busy or the RIIC is already communicating, it does not start a master transmission.
Arguments	IIC_API_T data1
Return value	If communication starts up normally: RIIC_OK If the I^2C bus is busy: RIIC_BUS_BUSY If the RIIC module is communicating: RIIC_MODE_ERROR If the argument value is illegal: RIIC_PRM_ERROR
Notes	See section 4.2.1, Structures, for details on the argument IIC_API_T data1. See section 4.2.3, Enumerations, for details on the return value. The argument is recognized as illegal if both the memory address counter and the data counter are 0. Bit 0 in the slave address (SlvAdr), which is a member of the argument structure, must be set to 0.

IIC_GetStatus	
Overview	Acquires the status of the RIIC module.
Header	r_apn_iic.h
Declaration	void IIC_GetStatus(enum RiicStatus_t*, enum RiicBusStatus_t*);
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 RiicStatus_t *data1 enum RiicBusStatus_t *data2
Return value	None
Notes	See section 4.2.3, Enumerations, for details on the arguments.

4.4 Flowchart

This section presents the flowcharts for the functions in this sample program.

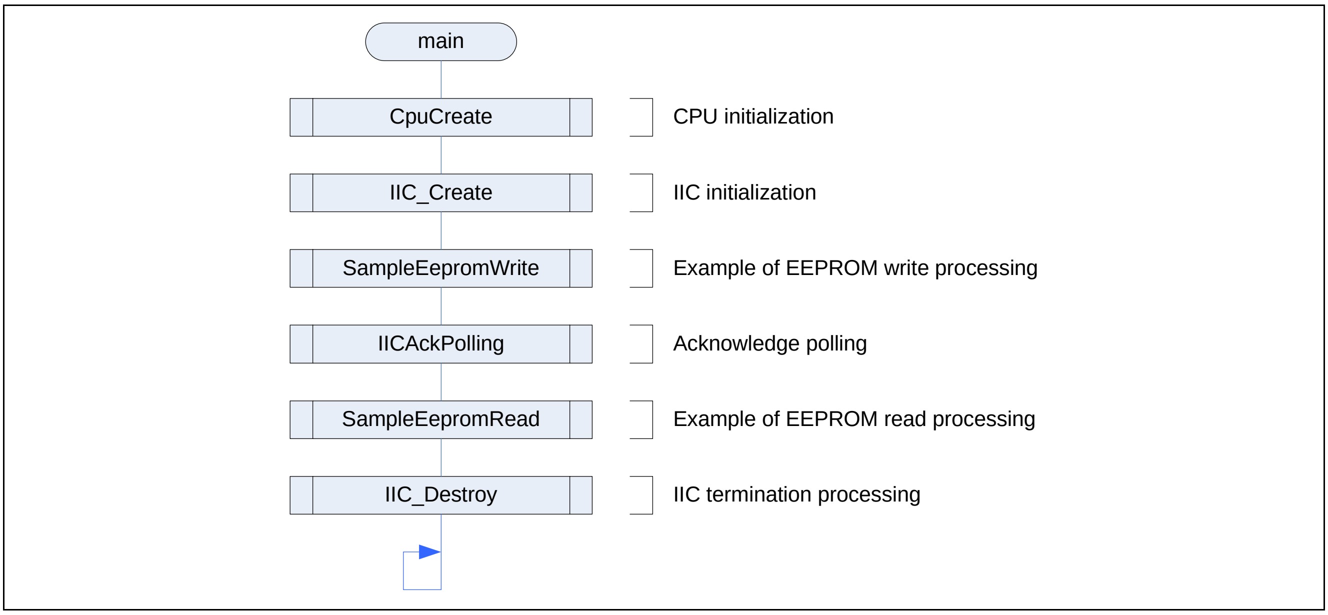

Figure 7 Main Processing

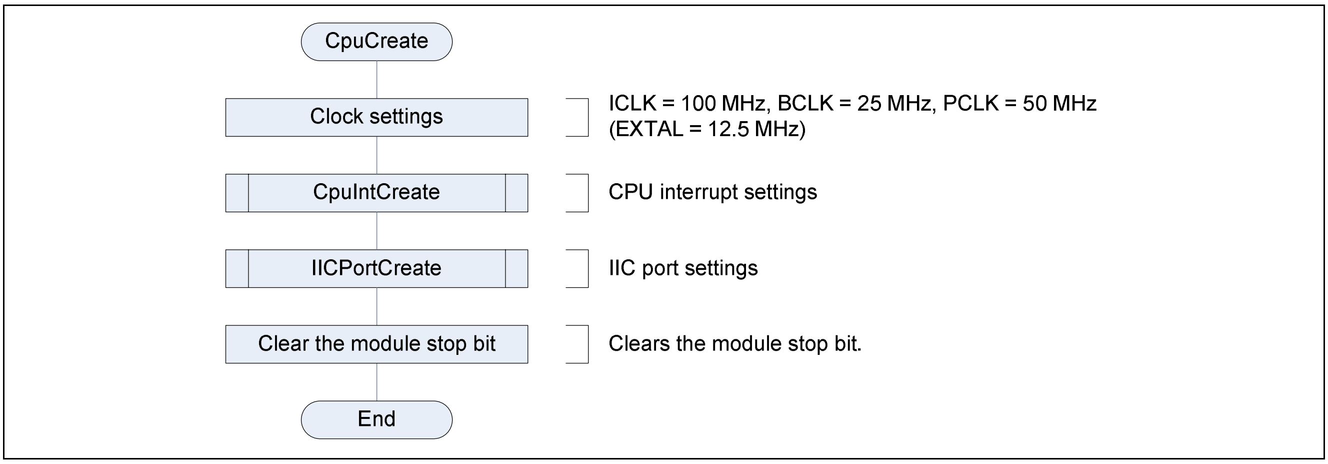

Figure 8 CPU Initialization

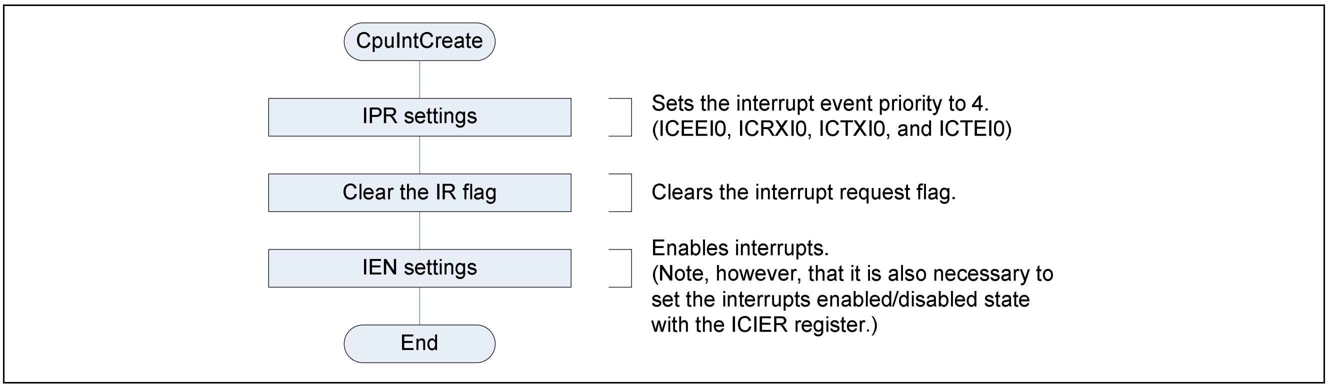

Figure 9 CPU Interrupt Settings

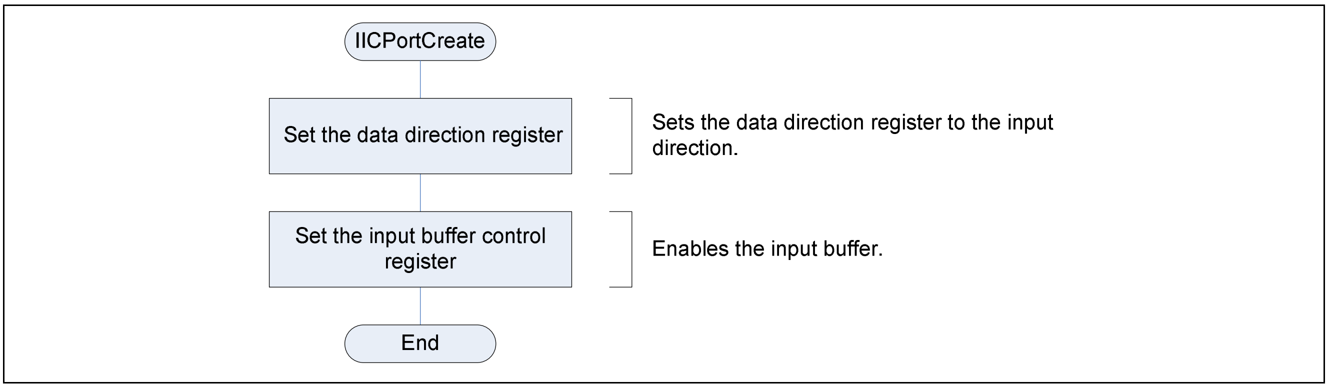

Figure 10 IIC Port Settings

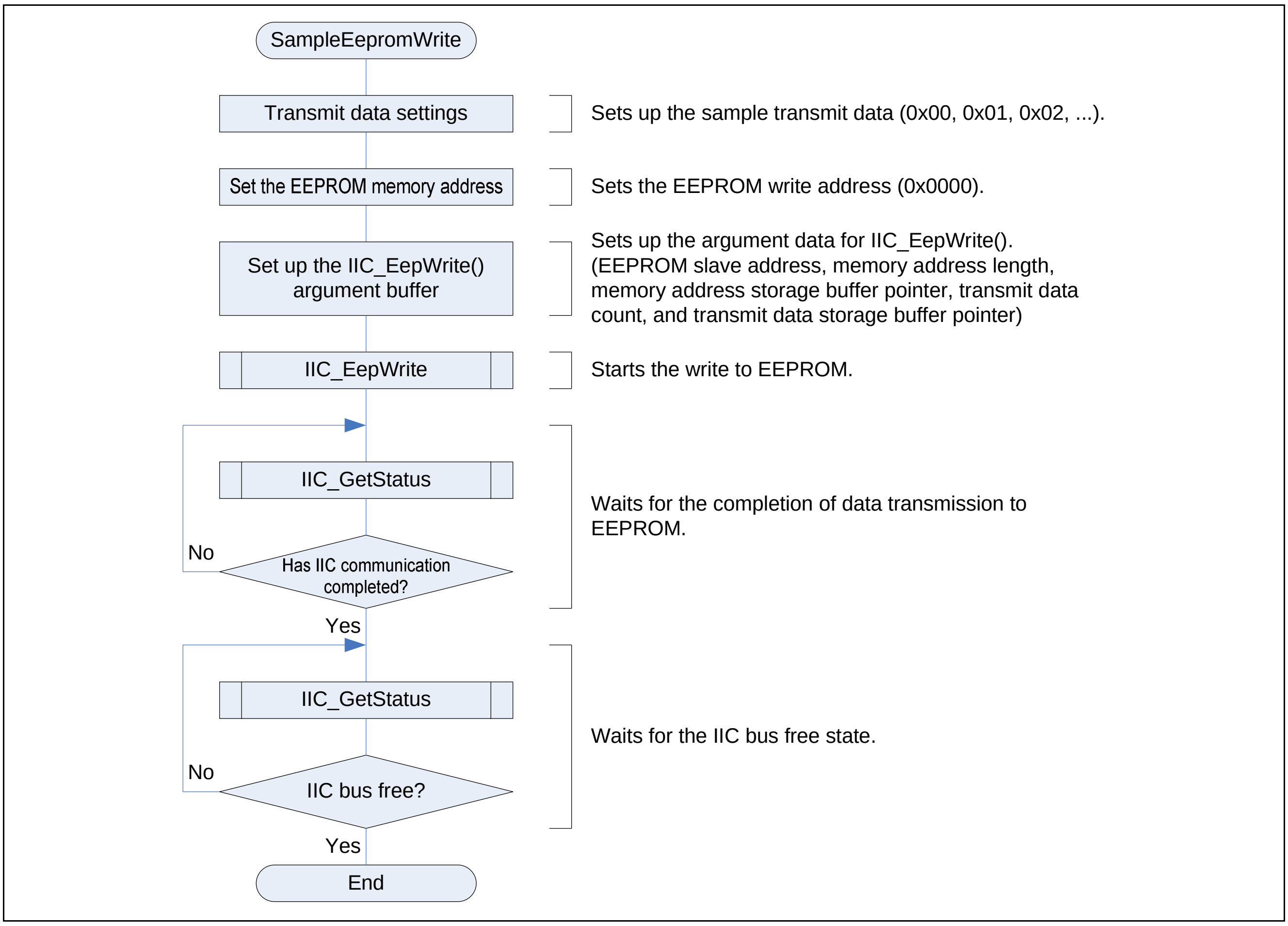

Figure 11 Sample EEPROM Write Processing

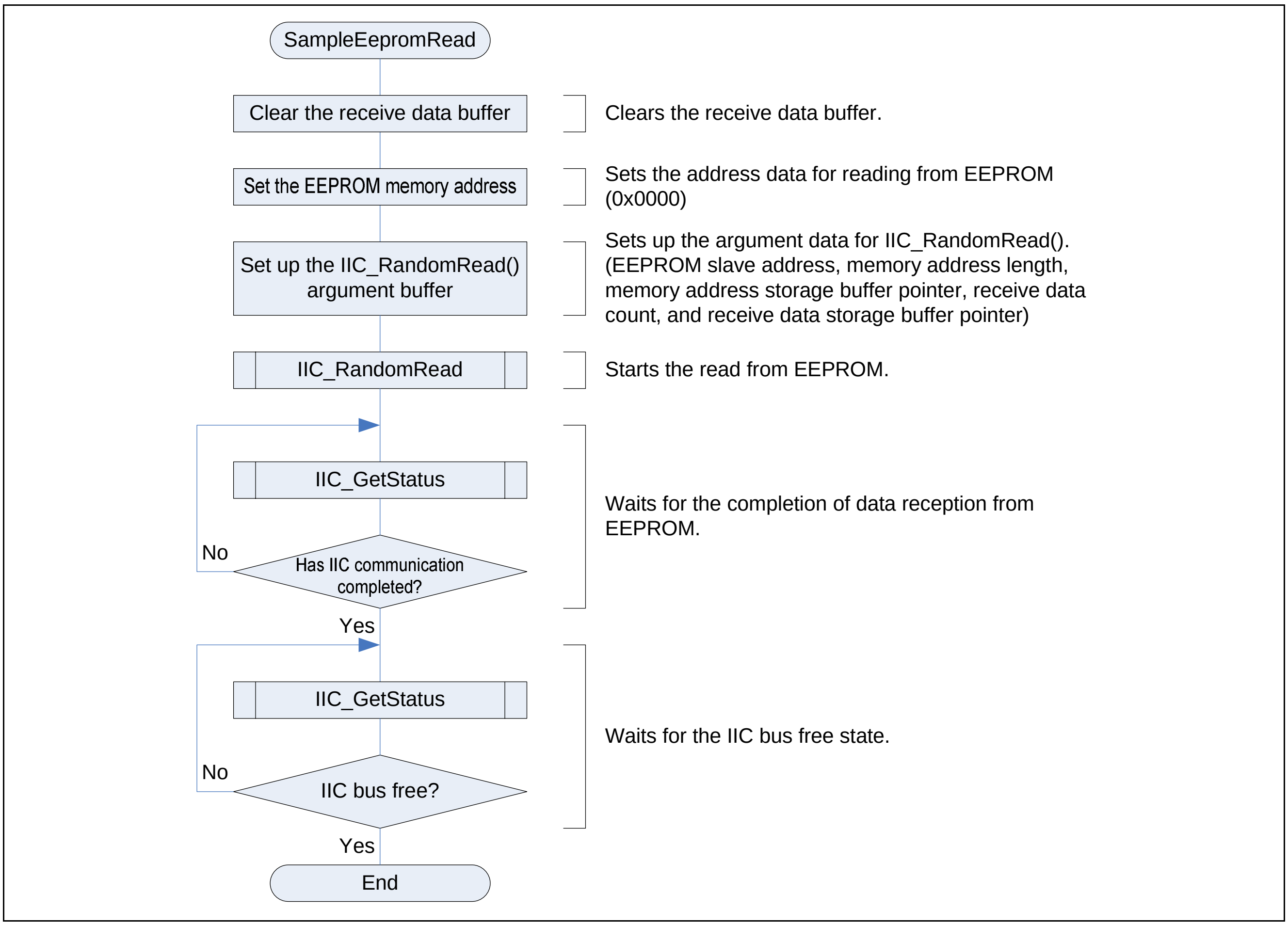

Figure 12 Sample EEPROM Read Processing

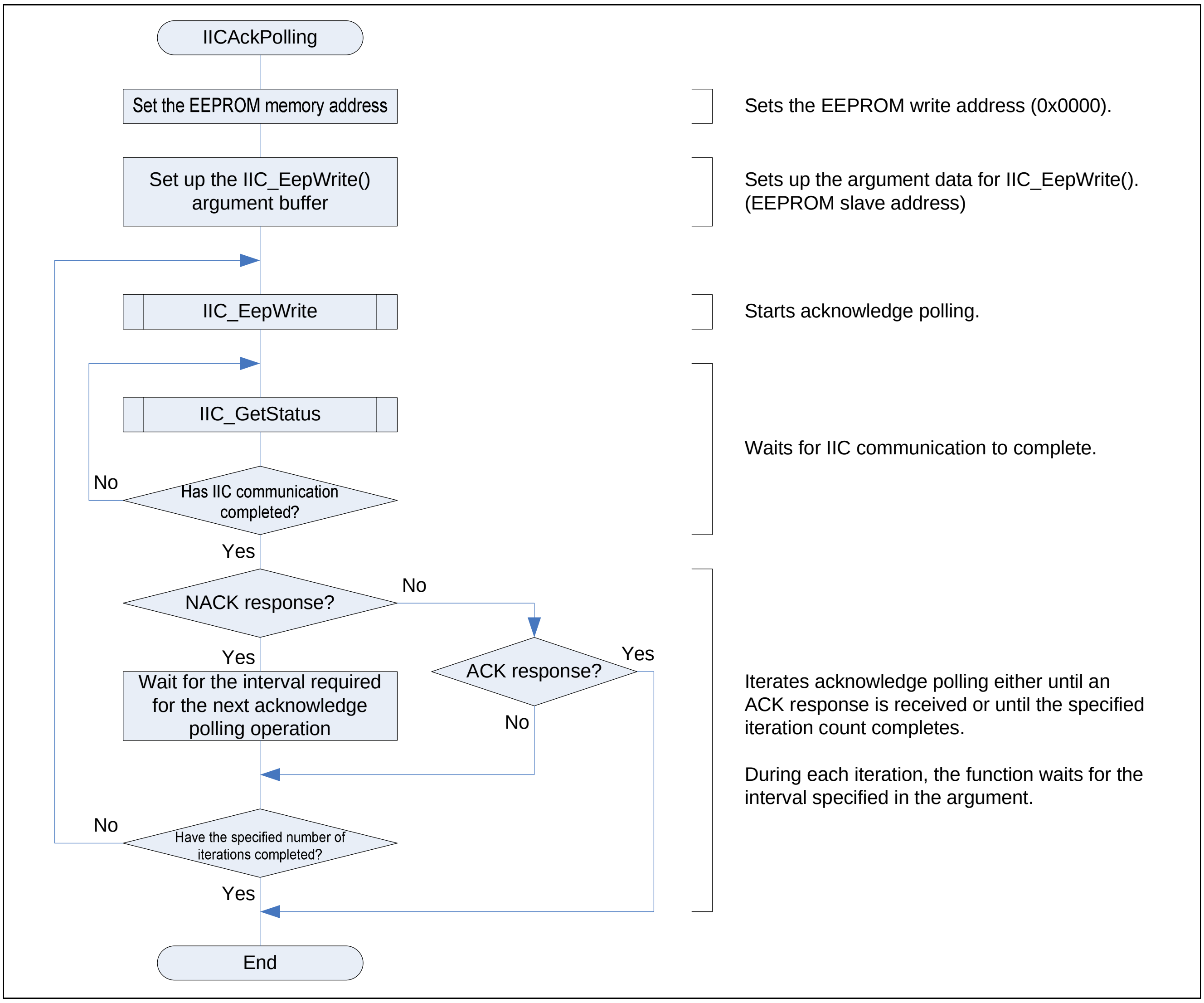

Figure 13 Acknowledge Polling

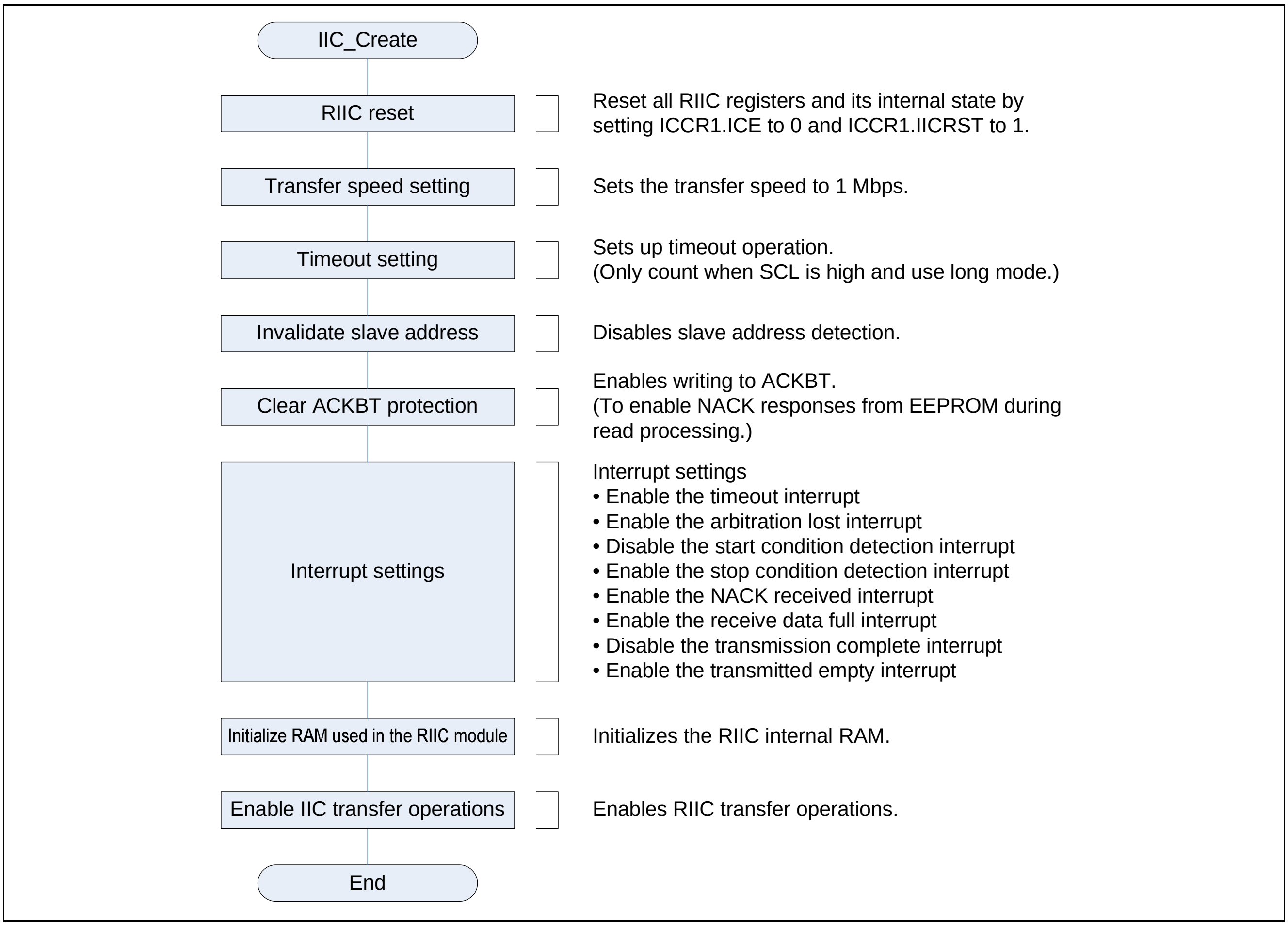

Figure 14 IIC Initialization

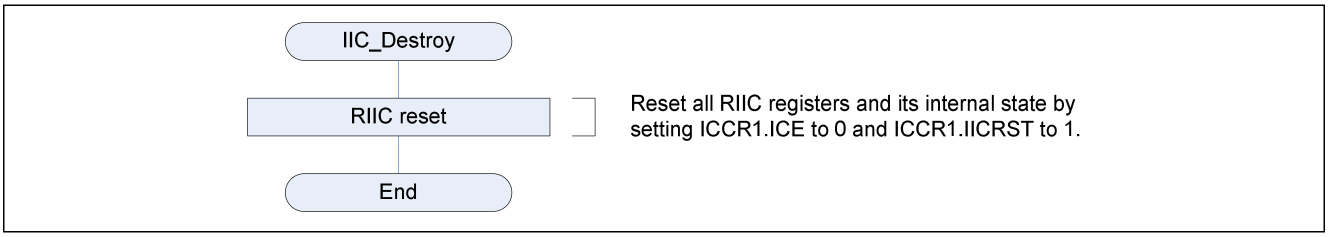

Figure 15 IIC Termination Processing

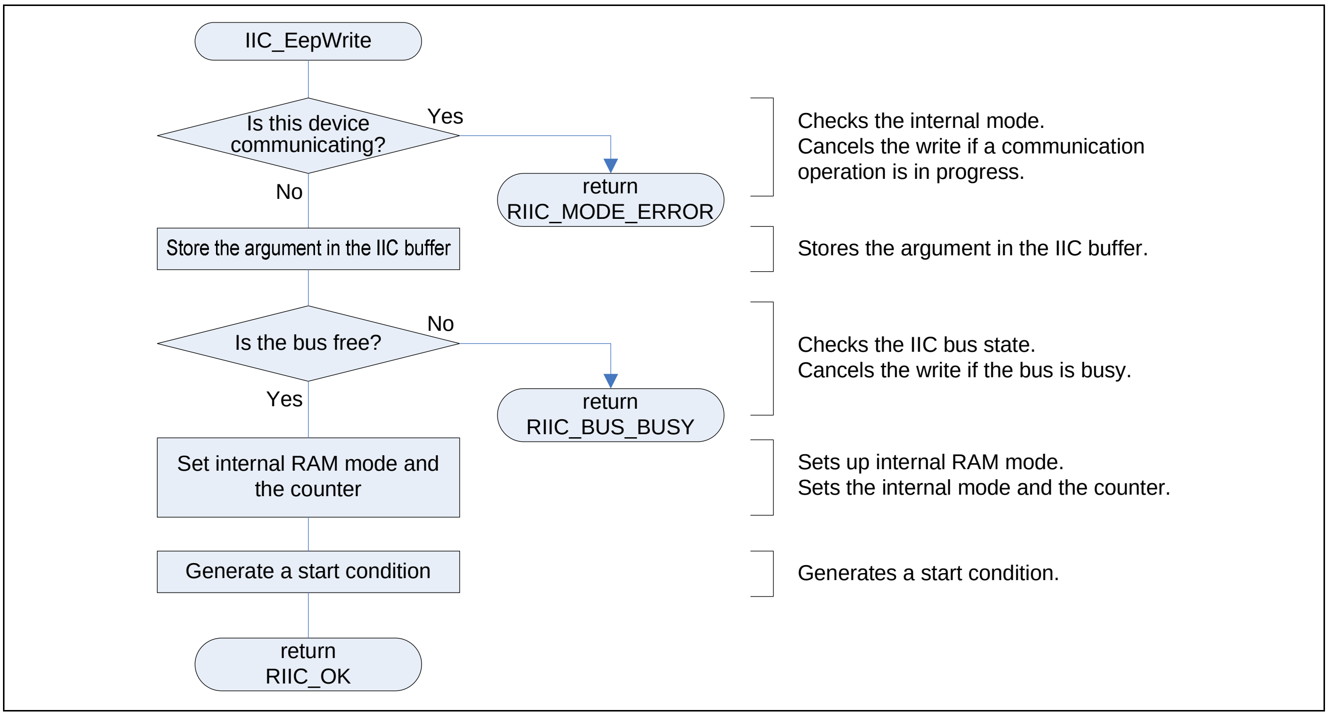

Figure 16 EEPROM Write Start Processing

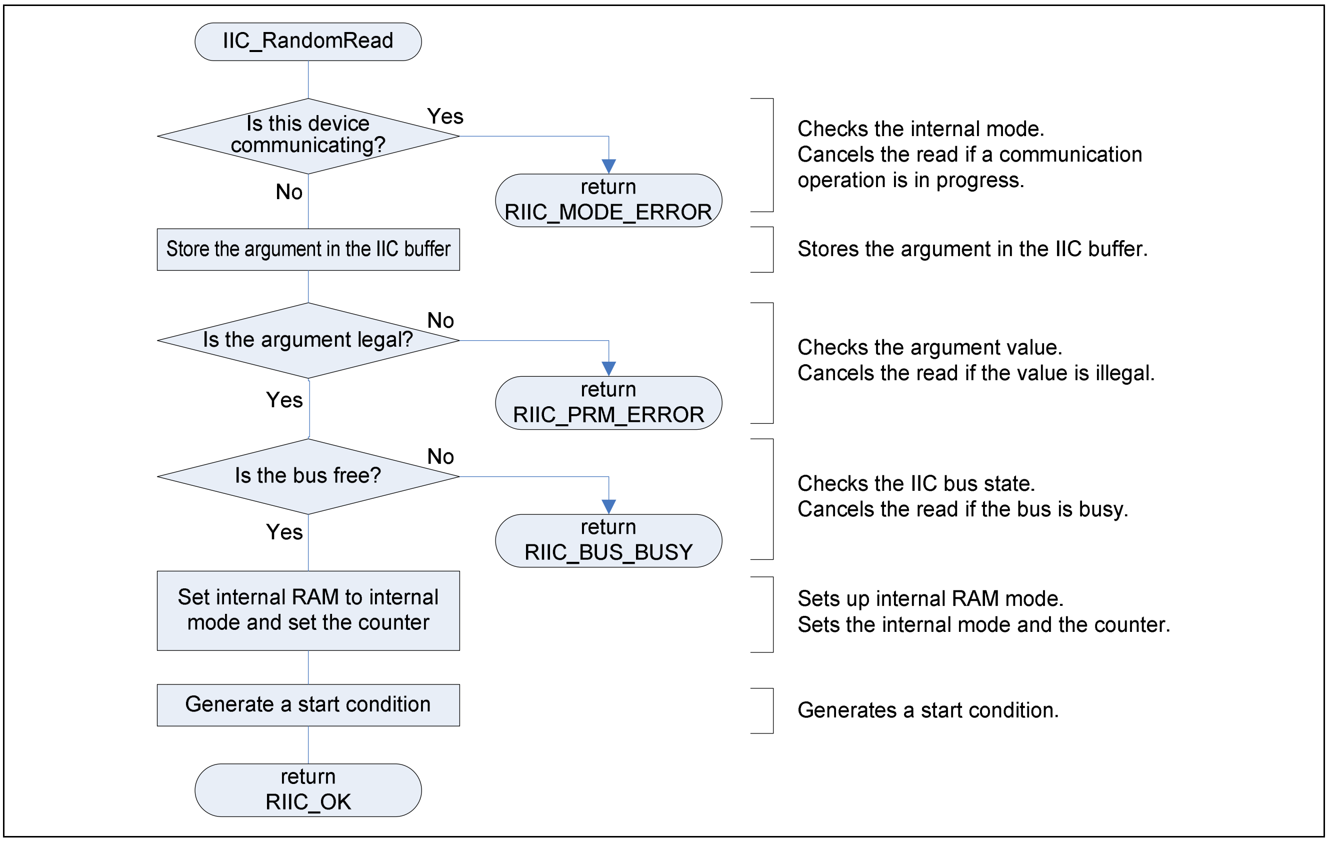

Figure 17 EEPROM Read Start Processing

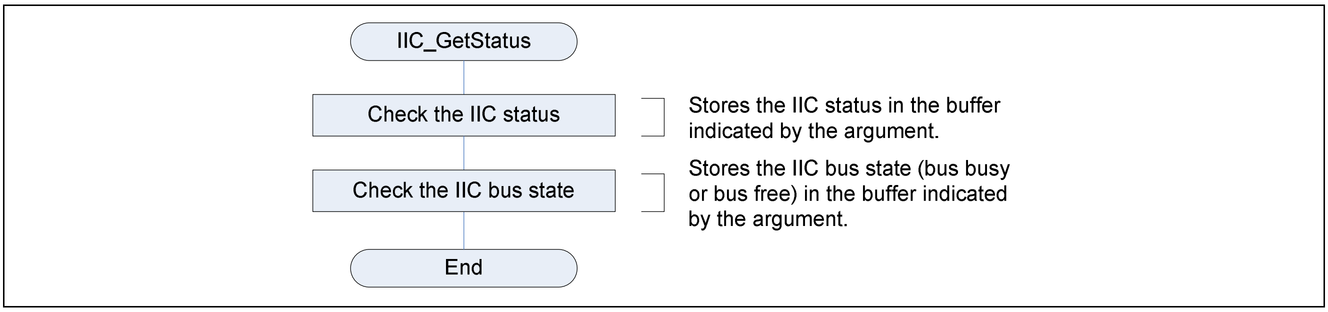

Figure 18 IIC State Verification Processing

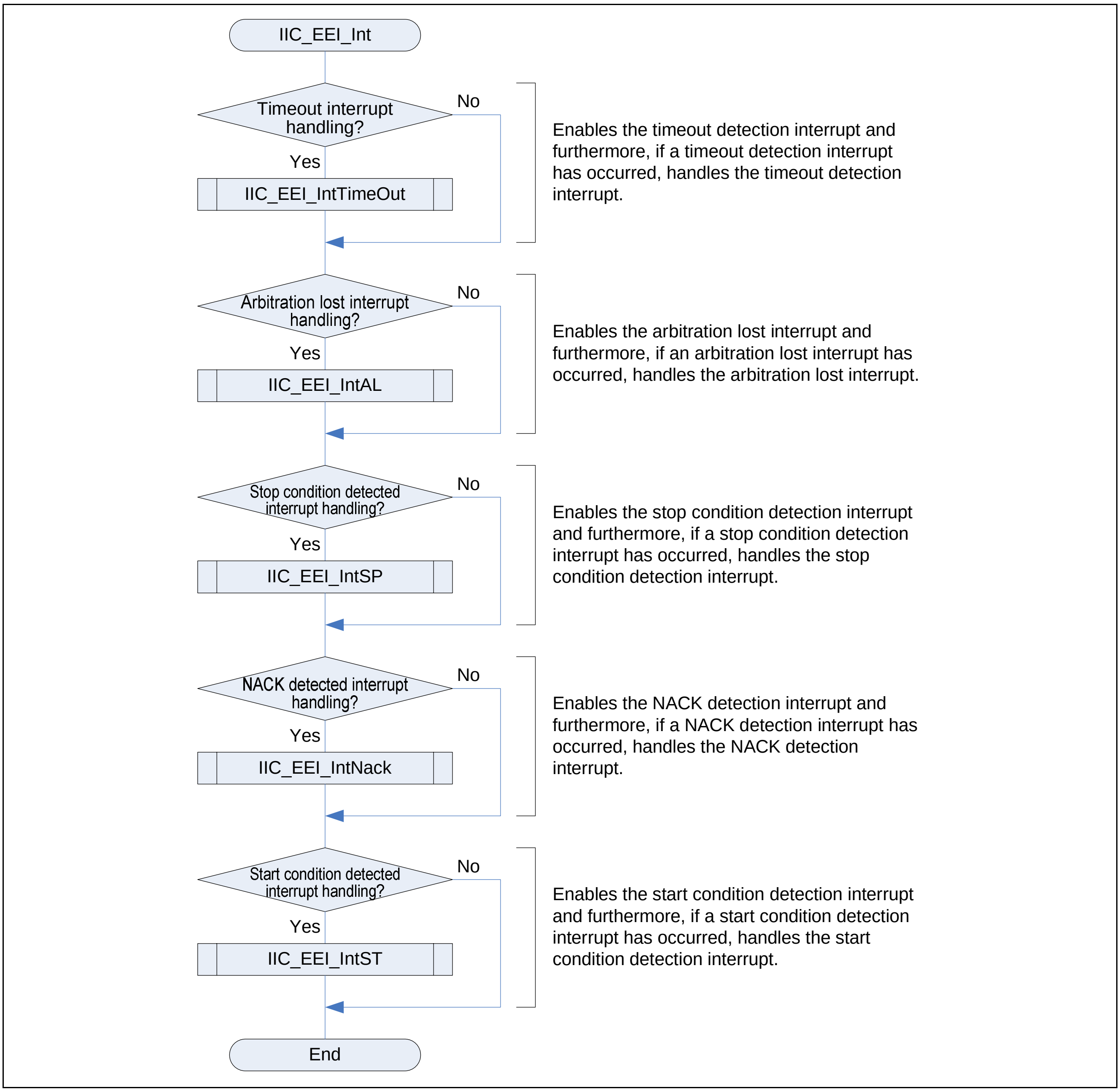

Figure 19 Communication Error and Event Interrupts

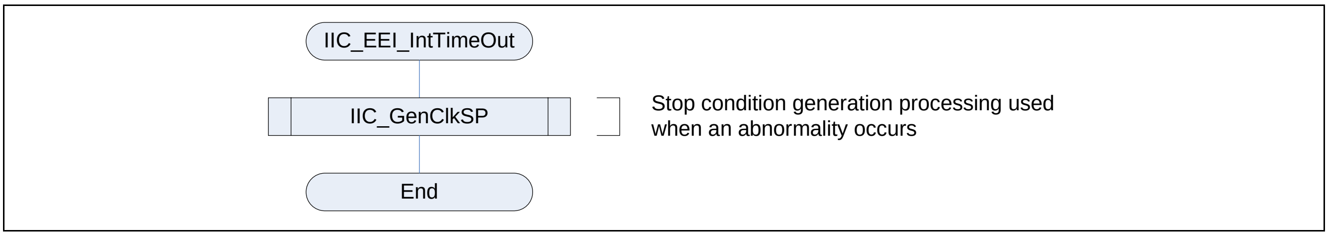

Figure 20 Timeout Detection Interrupt

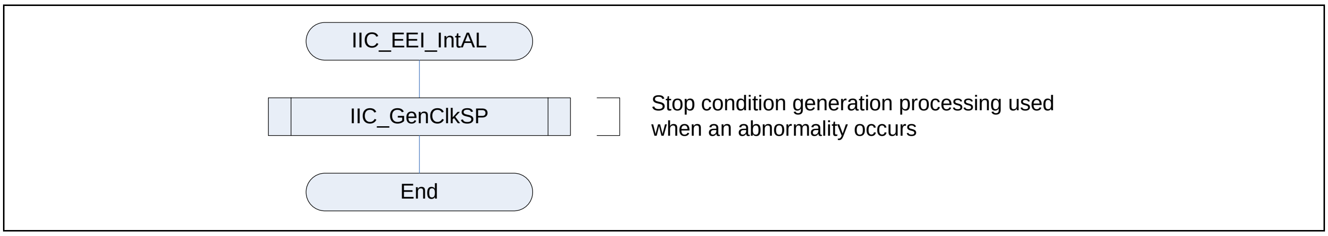

Figure 21 Arbitration Lost Detection Interrupt

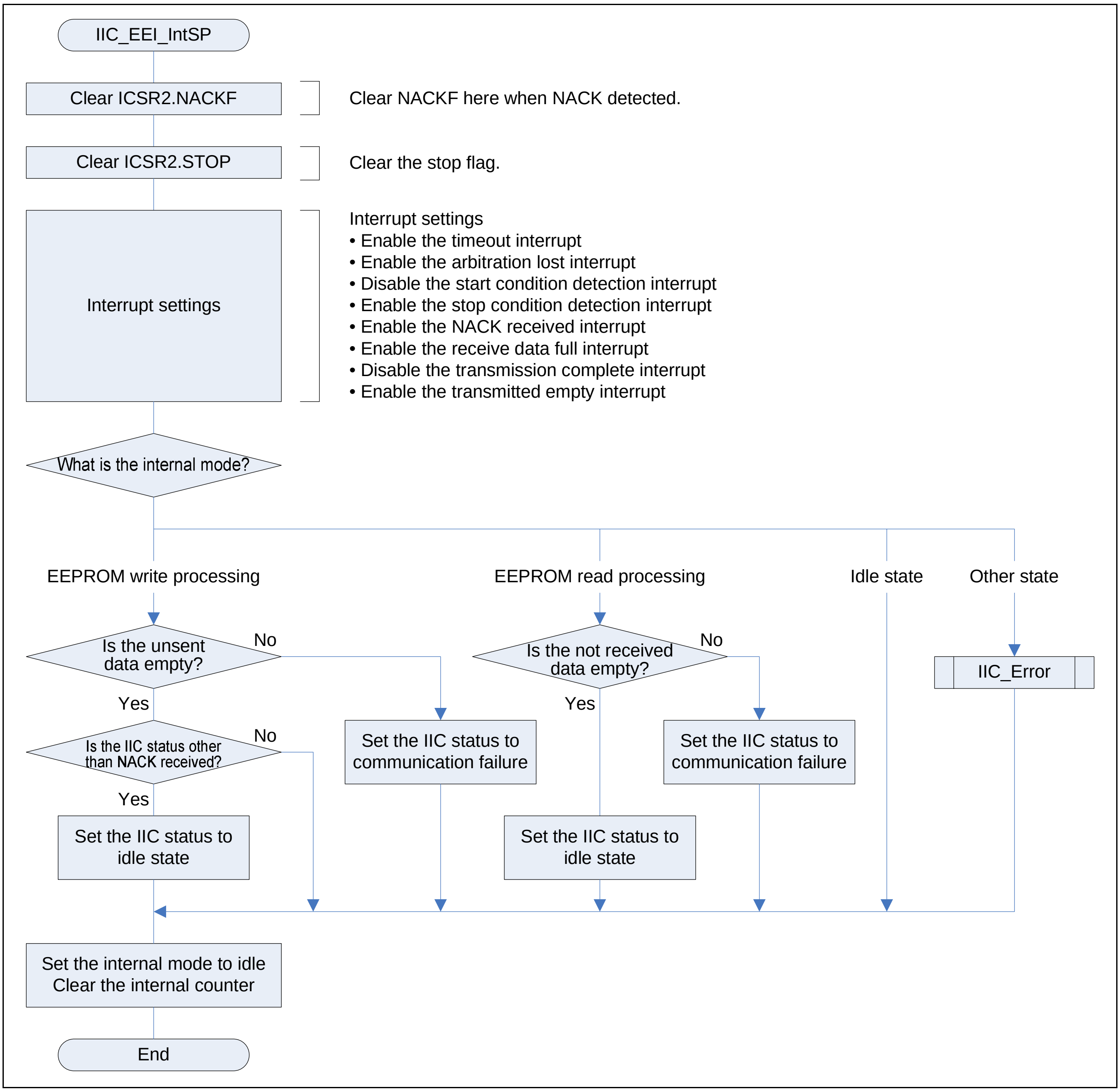

Figure 22 Stop Condition Detection Interrupt

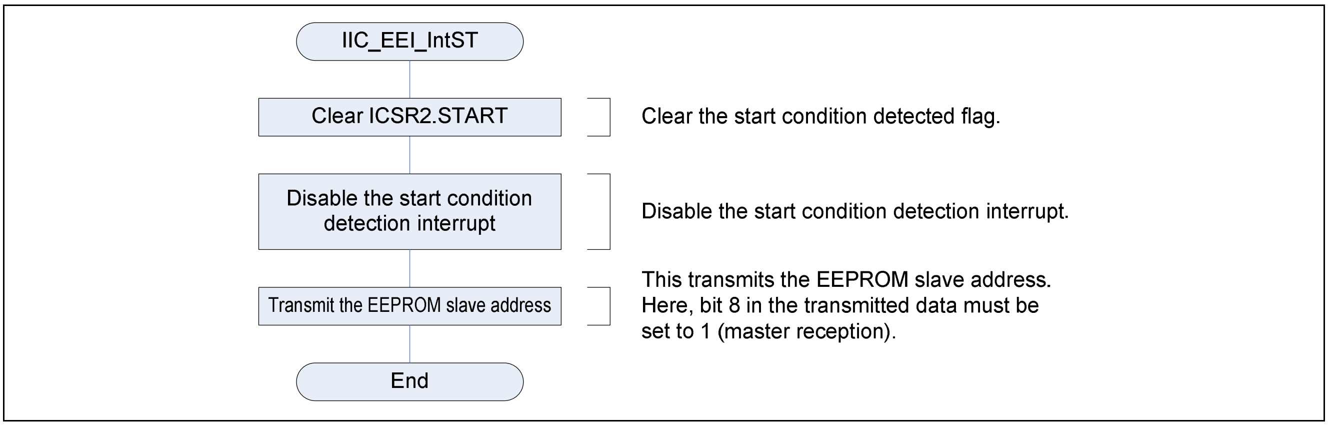

Figure 23 Start Condition Detected Interrupt

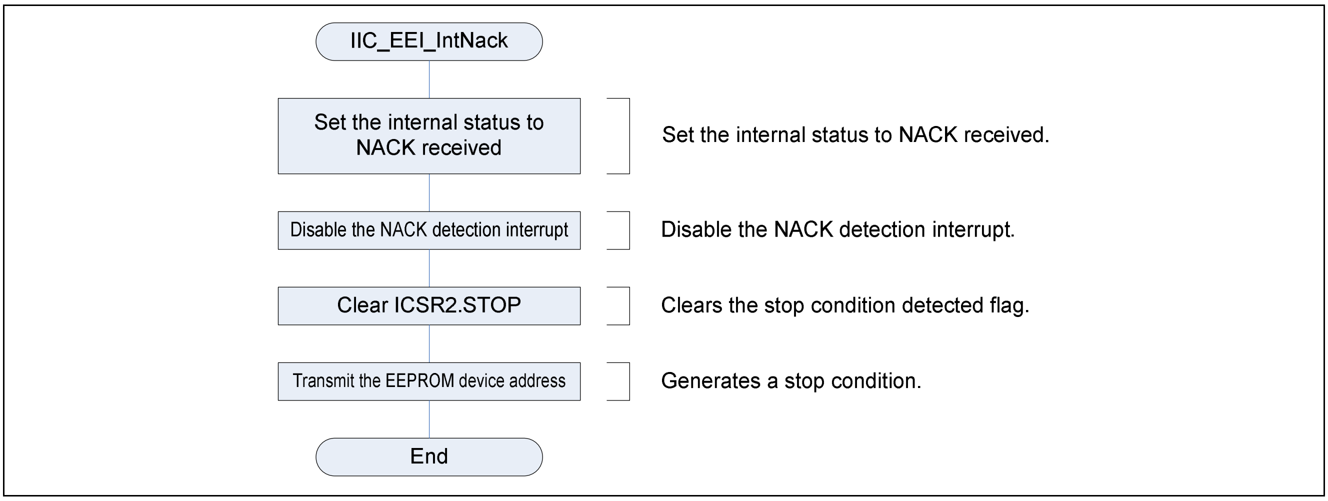

Figure 24 NACK Detection Interrupt

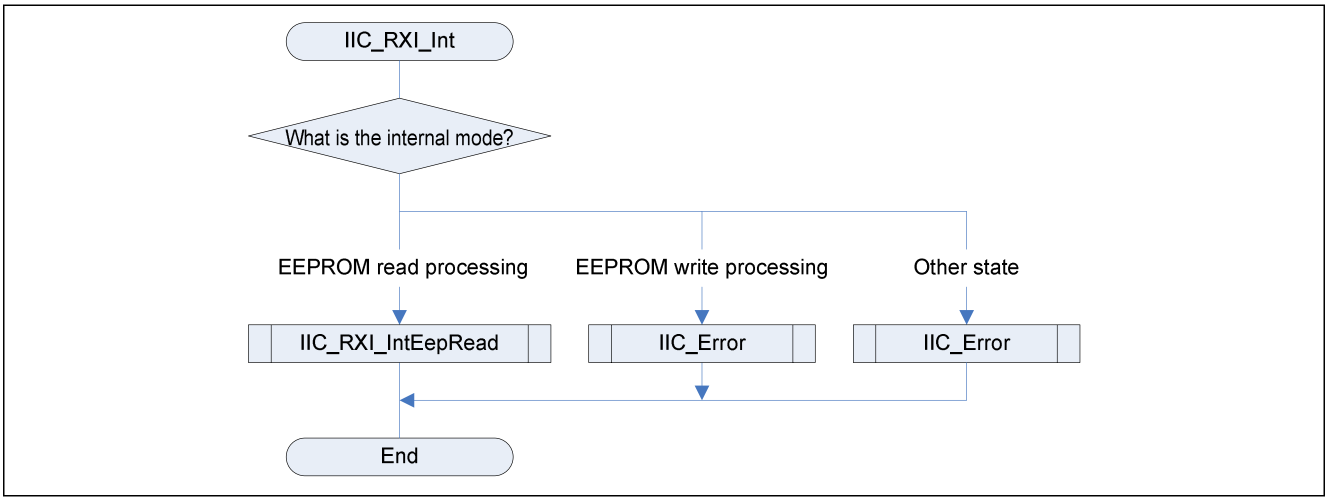

Figure 25 Receive Data Full Interrupt

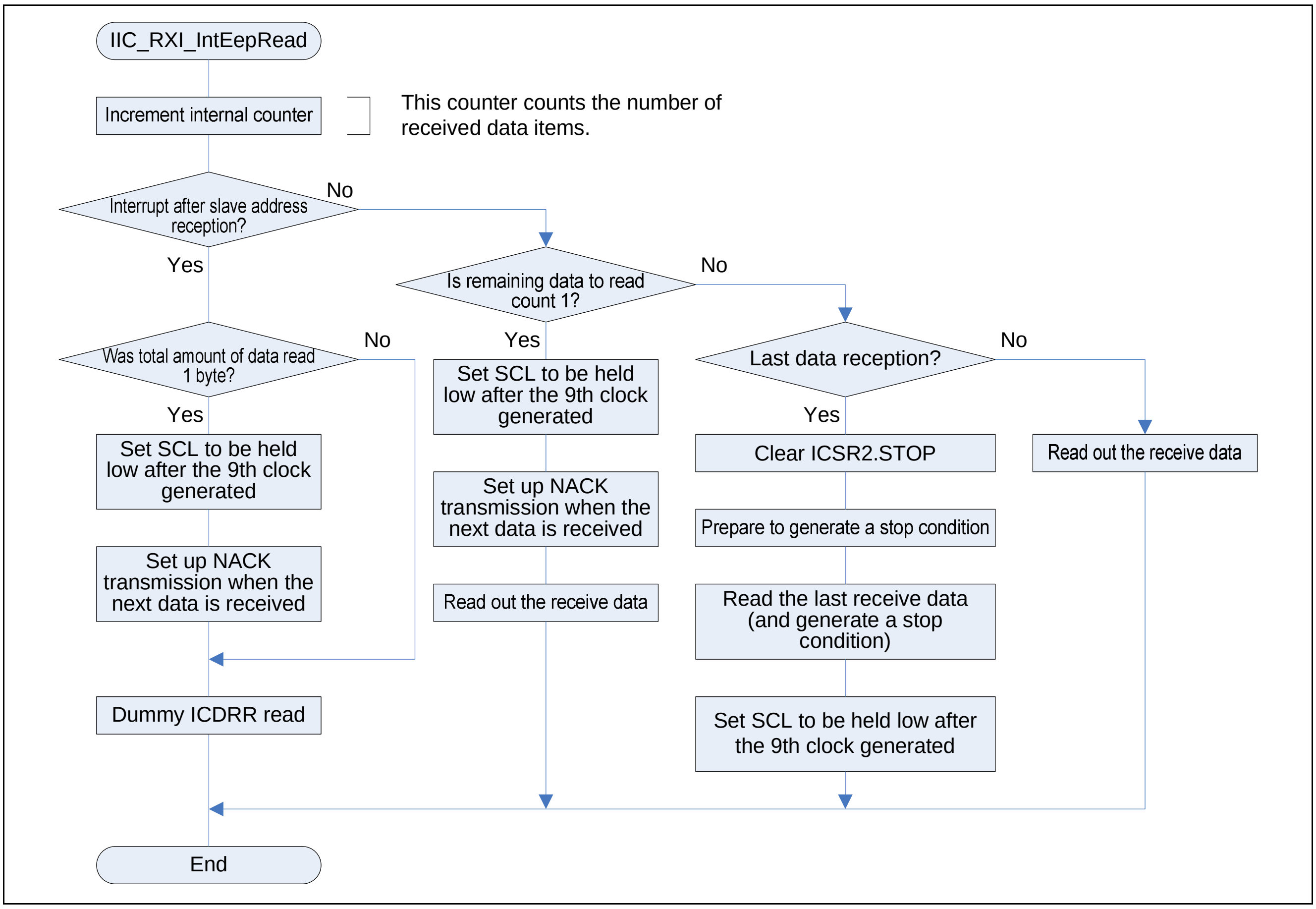

Figure 26 EEPROM Read Processing (Master Reception Section)

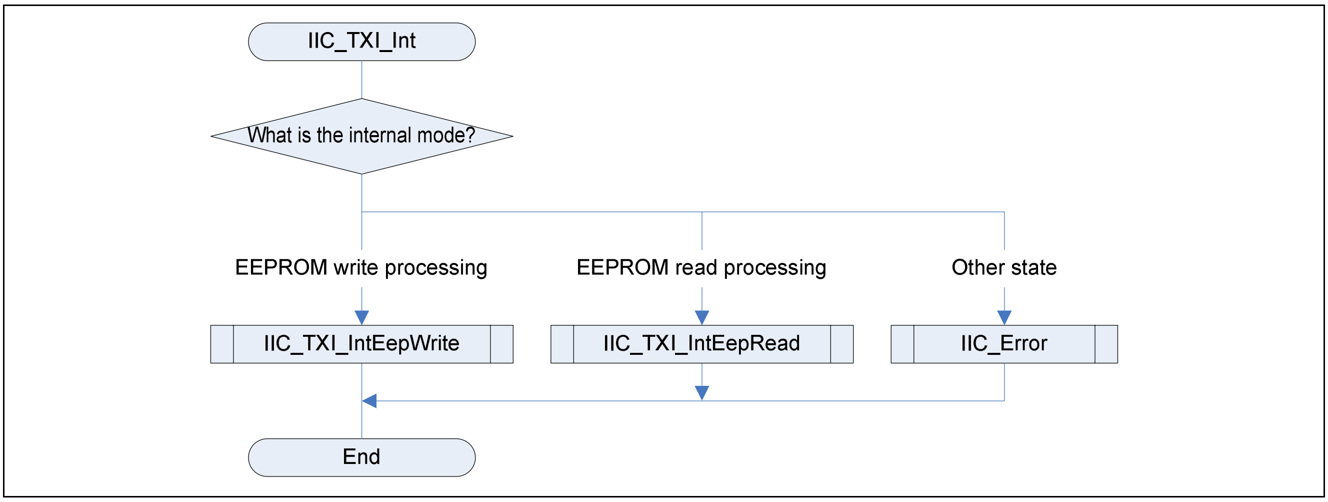

Figure 27 Transmit Data Empty Interrupt

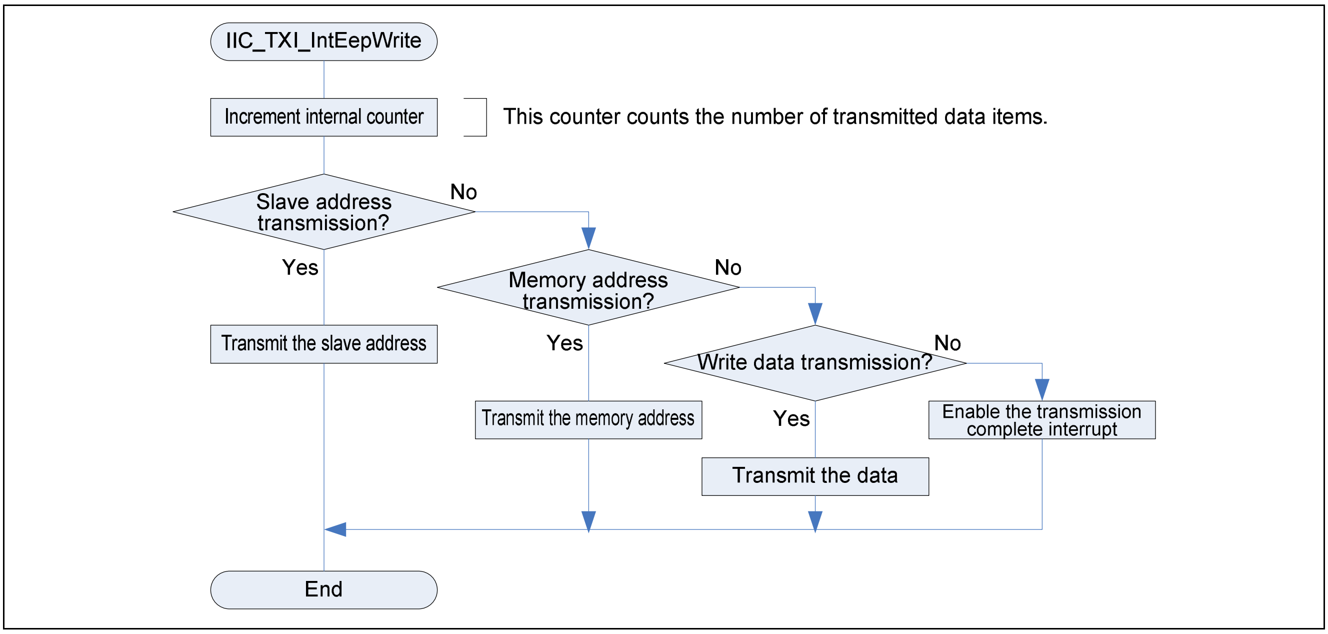

Figure 28 EEPROM Write Processing

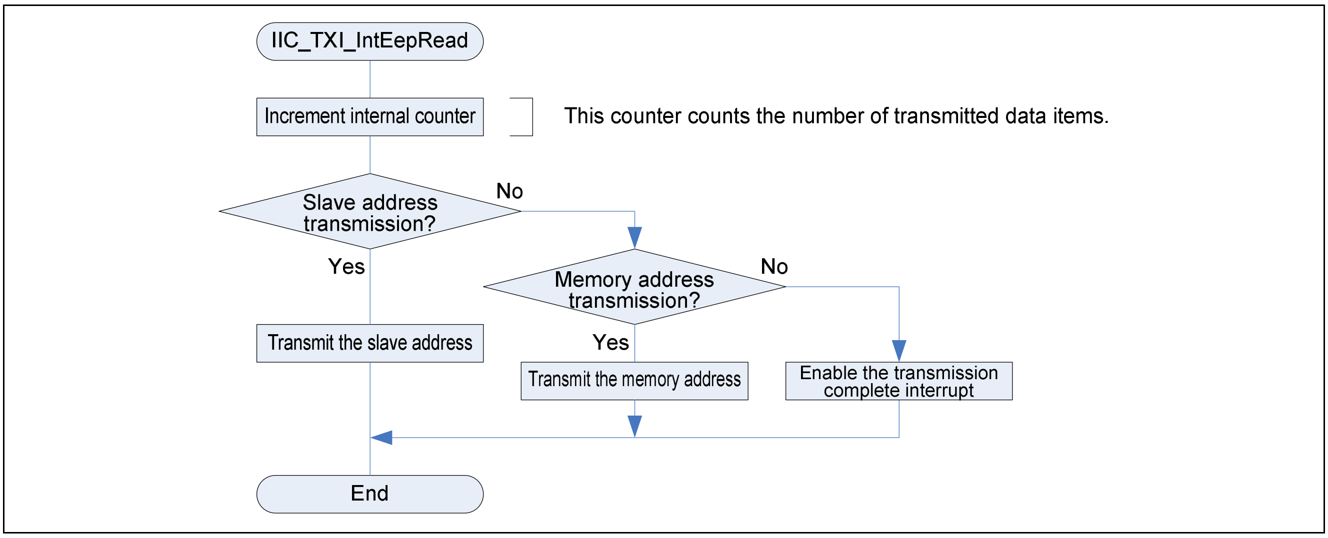

Figure 29 EEPROM Read Processing (Master Transmission Section)

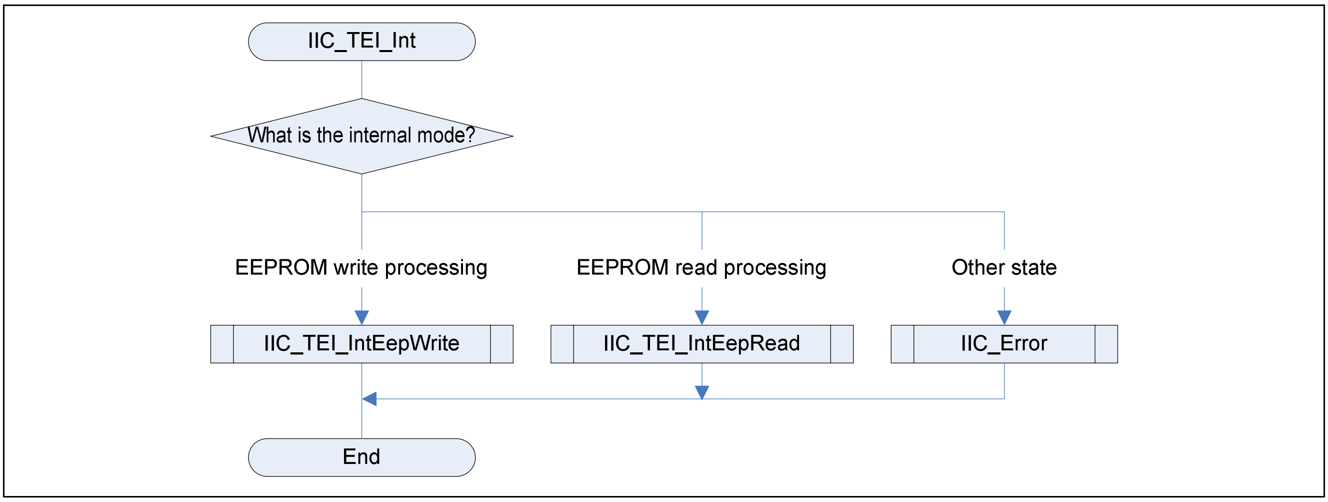

Figure 30 Transmission Complete Interrupt

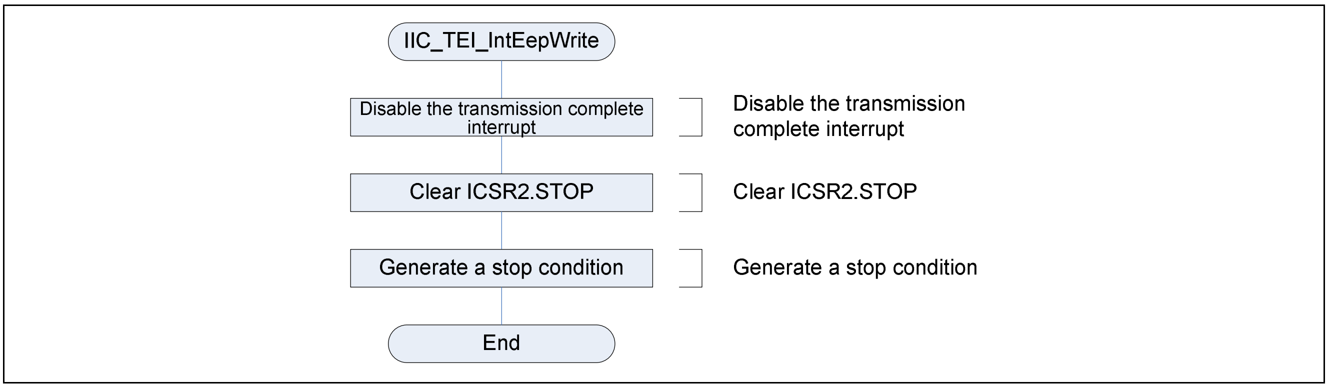

Figure 31 Transmission Complete Processing after EEPROM Write Processing

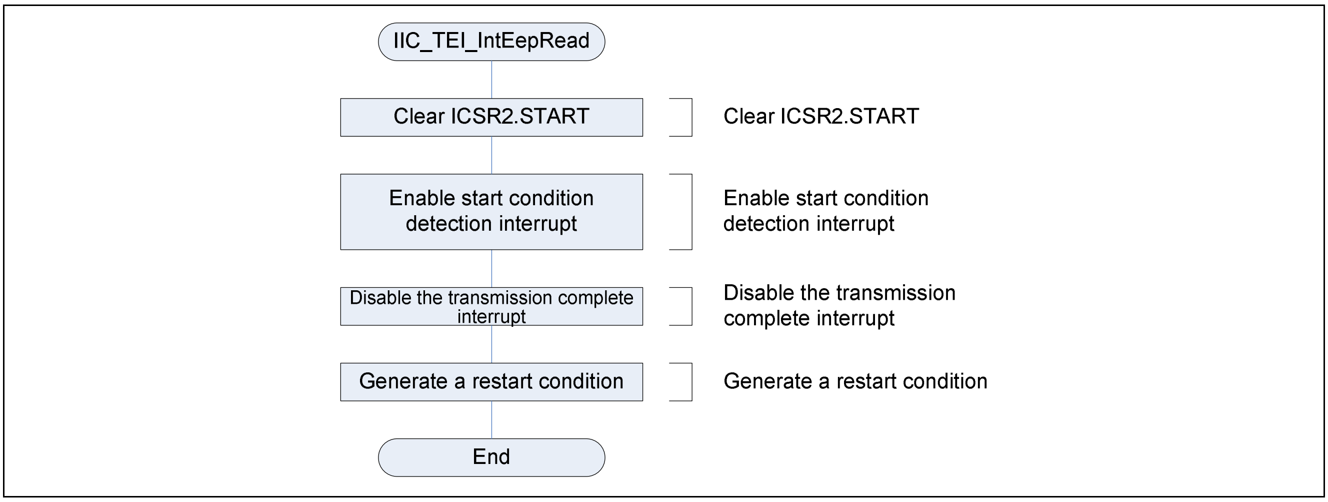

Figure 32 Transmission Complete Processing after EEPROM Read Processing

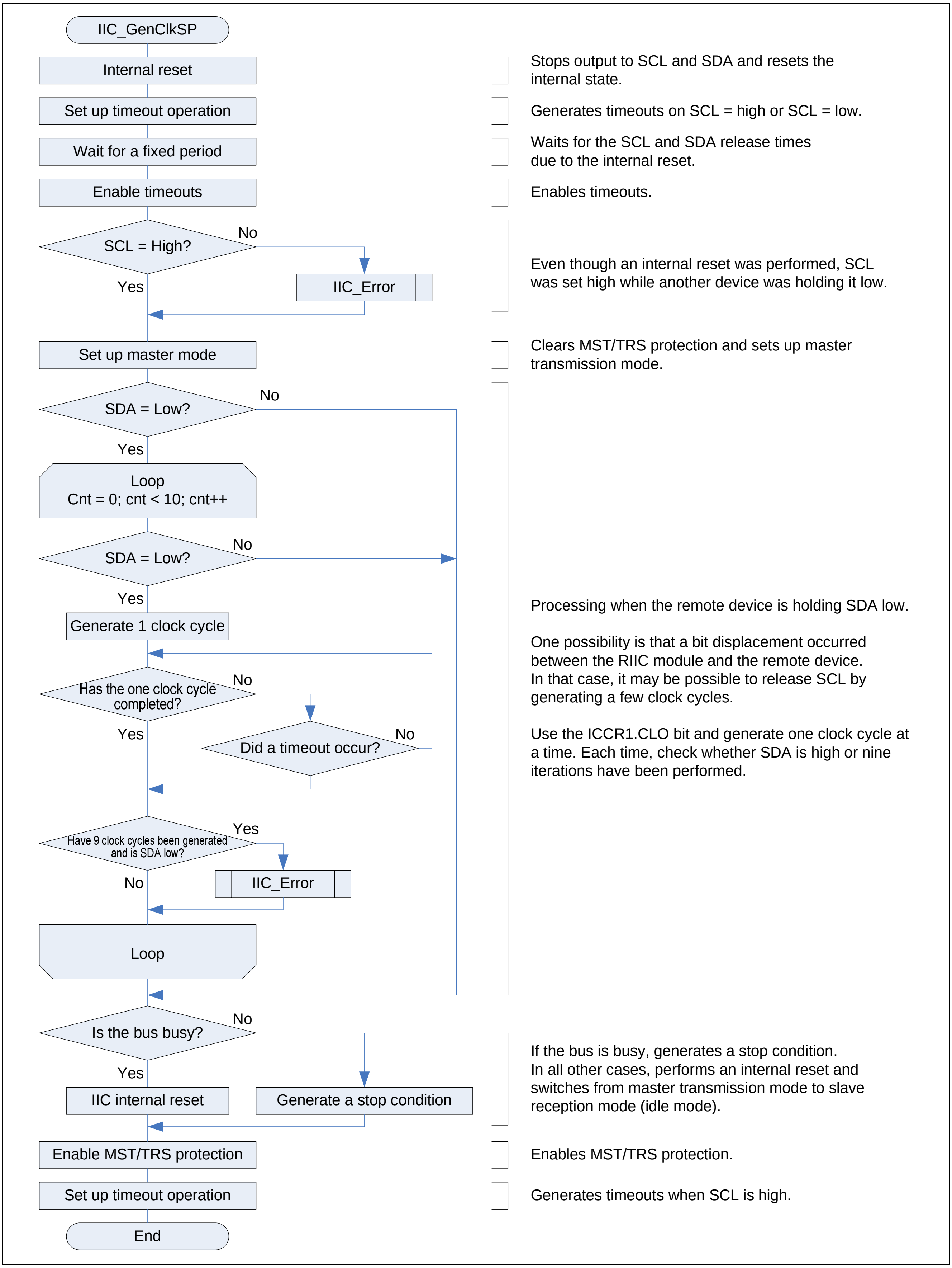

Figure 33 Stop Condition Generation Processing when an Abnormal State Occurs

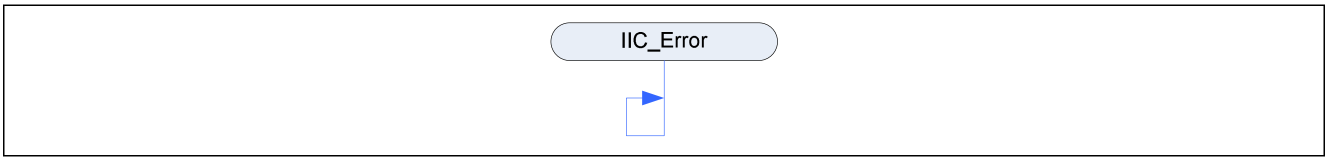

Figure 34 Error Handling

5. Reference Documents

- Hardware Manual
  RX610 Group User's Manual: Hardware
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Software Manual
  RX Family User's Manual: Software
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Development Environment Manual
  RX Family C/C++ Compiler Package User's Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Technical Updates
  (The latest information can be downloaded from the Renesas Electronics Web site.)

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.18.11	—	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 part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0