RL78/G13

Serial Array Unit (SAU)
(EEPROM Control Using Simplified IIC)
[for CubeSuite+, IAR, and e2studio]

R01AN1351EJ0201
Rev. 2.01
2014.5.16

Introduction

This application note explains how to control EEPROM using the simplified IIC function of the serial array unit (SAU). The sample application covered in this application note carries out the read from and writes to EEPROM devices on an IIC bus using the simplified IIC function in an interrupt-driven mode.

Target Device

RL78/G13

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specifications

The sample application covered in this application note carries out the read from and writes to EEPROM devices on an IIC bus using the simplified IIC function of the serial array unit.

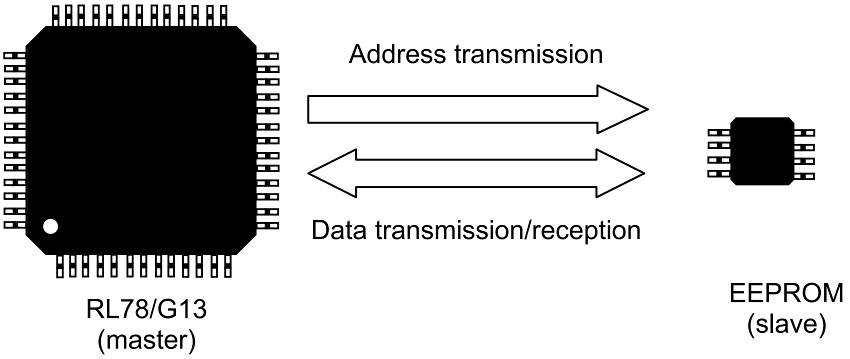
- An access to an EEPROM is accomplished by specifying necessary parameters in a structure and calling a function.
- Control of EEPROM devices is done in an interrupt-driven mode whenever possible with due consideration to the function as an API.
- One EEPROM type can be selected as the target EEPROM from 2 to 512 kbits of EEPROM. (The 16 kbits of R1EX24016A is selected as the default EEPROM. The main function performs test processing that assumes this EEPROM.)
- The application exercises the access control that supports the selected EEPROM.
- When a reset occurs while the application is reading EEPROM and processing is subsequently resumed when the EEPROM sets the SDA signal low, the application provides the support of releasing the bus that is held in an occupied state.
- The standard simplified IIC channel to be used is channel 00. Provisions, however, are provided that this can easily be altered.

Table 1.1 lists the peripheral function to be used and its use. Figure 1.1 presents an overview of IIC communication.

Figures 1.2 through 1.8 show timing charts for explaining the IIC communication.

Table 1.1 Peripheral Function to be Used and Its Use

Peripheral Function	Use
Serial array unit	Performs IIC master transmission and reception using the simplified IIC function (using the SCL00 and SDA00 pins).



Address transmission: Initial action taken by the master to identify the target of transfer (slave) in IIC communication. Address bits (7 bits) and a transfer direction bit (1 bit) is transmitted as 1 byte of data after a start condition is generated.

Data transmission/reception: Sending/receiving of data to/from the destination slave after address transmission. After completion of data transmission/reception, the master generates a stop condition and releases the bus.

Remarks: The simplified IIC function supports only single master communication. It accepts no wait requests from slaves.

Figure1.1 Outline of IIC Communication

(1) Master-to-slave communication 1 (start condition – address – data)

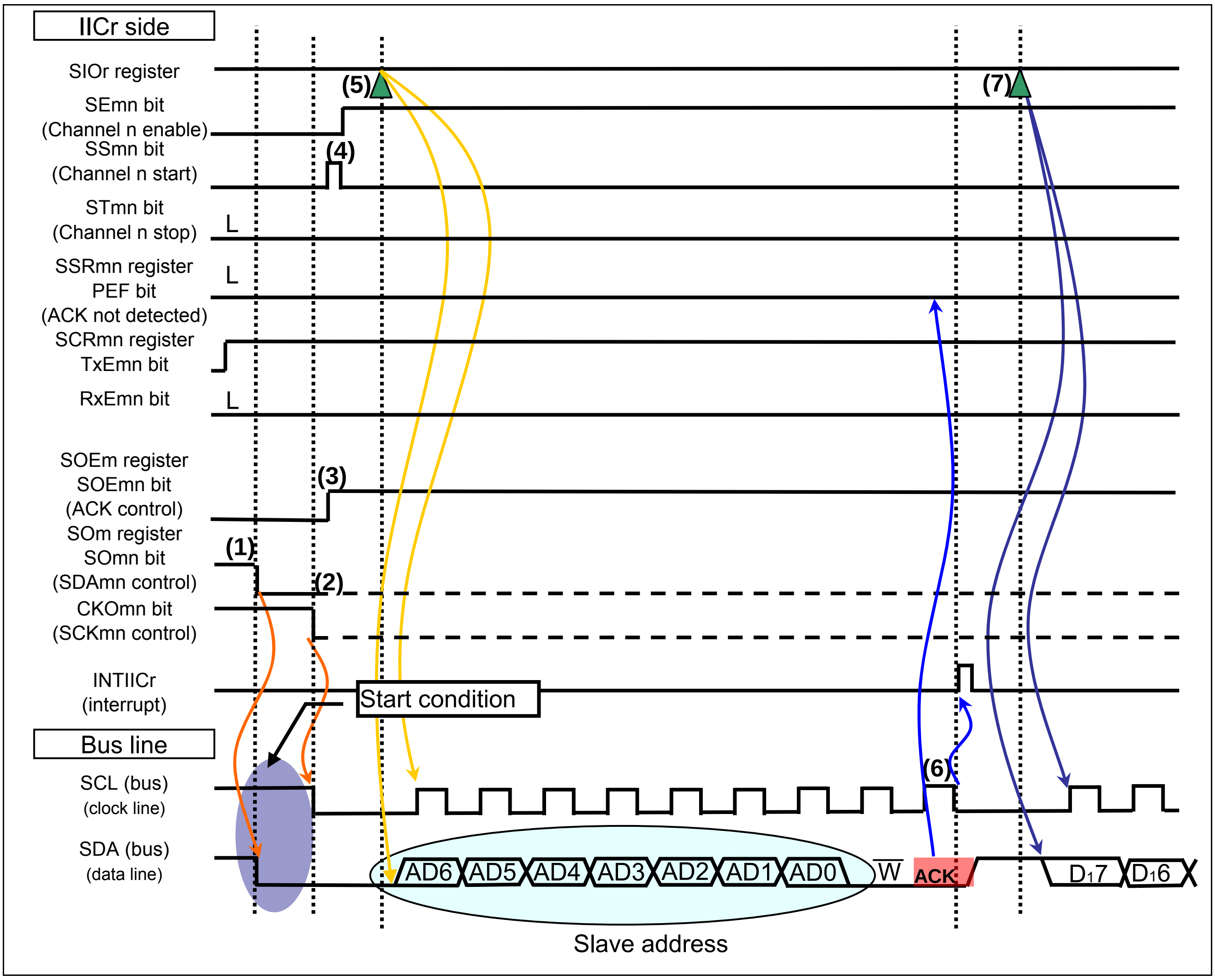


Figure 1.2 IIC Communication Timing Chart (Master-to-Slave Communication Example) (1/4)

(1) After IICmn initialization (here the SCRmn register is set up such that TxEmn is 1 and RxEmn is 0) is completed, set the SDA signal low by setting the SOmn bit of the SOm register to 0 to issue a start condition.

(2) After the lapse of the start condition hold time (4.0 µs in standard mode and 0.6 µs in fast mode), set the CKOmn bit of the SOm register to 0 to set the SCL signal low.

(3) For communication, set the SOEmn bit of the SOEm register to 1 to enable output.

(4) Set the SSmn bit of the SSm register to 1 to enable channel n.

(5) Write the address of the slave into the SIOr register, and communication will start.

(6) An INTIICr is generated when the transmission of the slave address is completed.

(7) Check the ACK response from the slave by testing the PEF bit of the SSRmn register and load the SIOr register with the transmit data if the PEF bit is found to be 0. Abort the transmission is if the PEF bit is set to 1.

(2) Master-to-slave communication 2 (address – data – data)

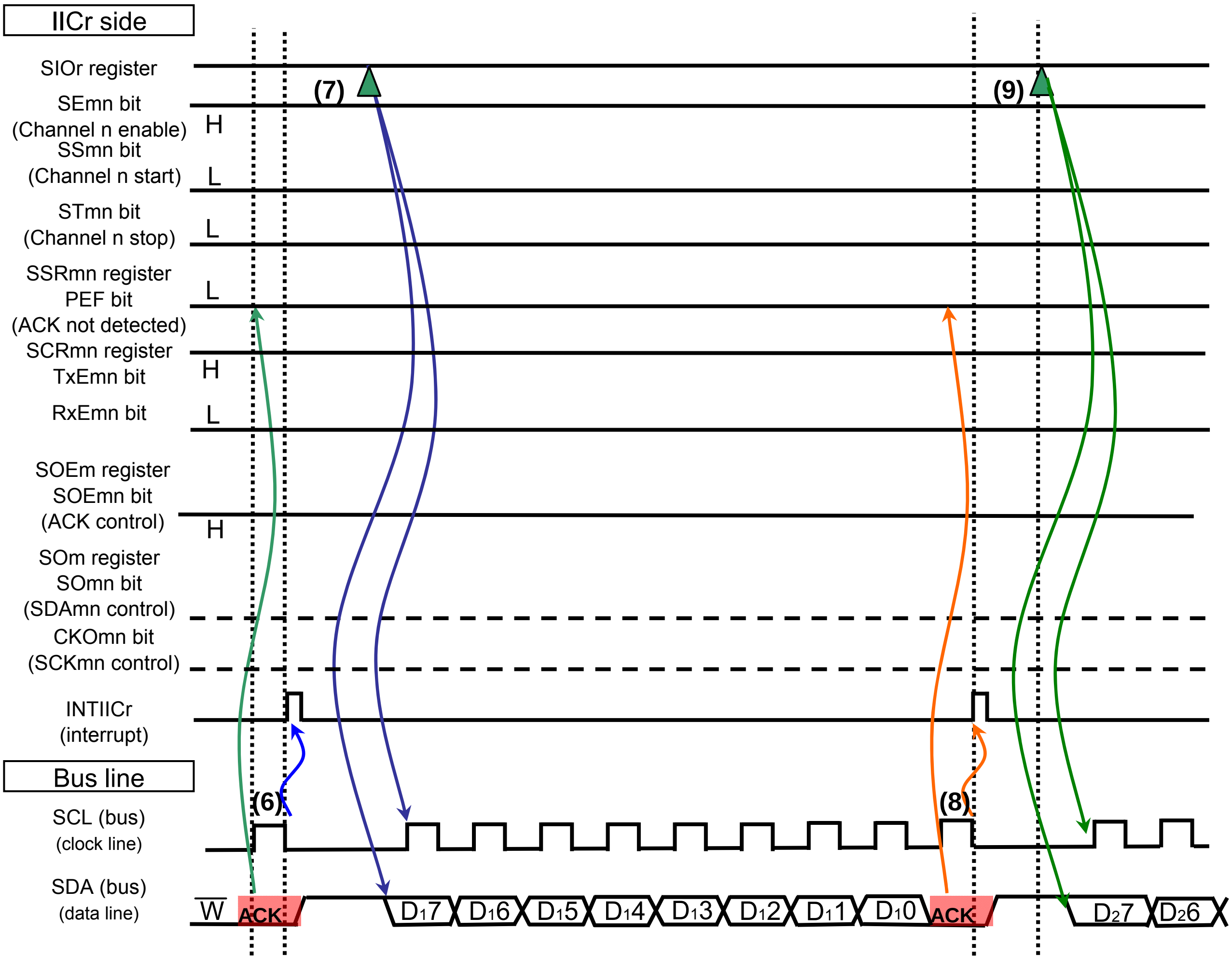


Figure 1.3 IIC Communication Timing Chart (Master-to-Slave Communication Example) (2/4)

(6) An INTIICr is generated when the transmission of the slave address is completed.

(7) Check the ACK response from the slave by testing the PEF bit of the SSRmn register and load the SIOr register with the transmit data if the PEF bit is found to be 0. Abort the transmission is if the PEF bit is set to 1.

(8) An INTIICr is generated when the transmission of the data is completed.

(9) Check the ACK response from the slave by testing the PEF bit of the SSRmn register and load the SIOr register with the transmit data if the PEF bit is found to be 0. Abort the transmission is if the PEF bit is set to 1.

(3) Master-to-slave communication 3 (data – data – stop condition)

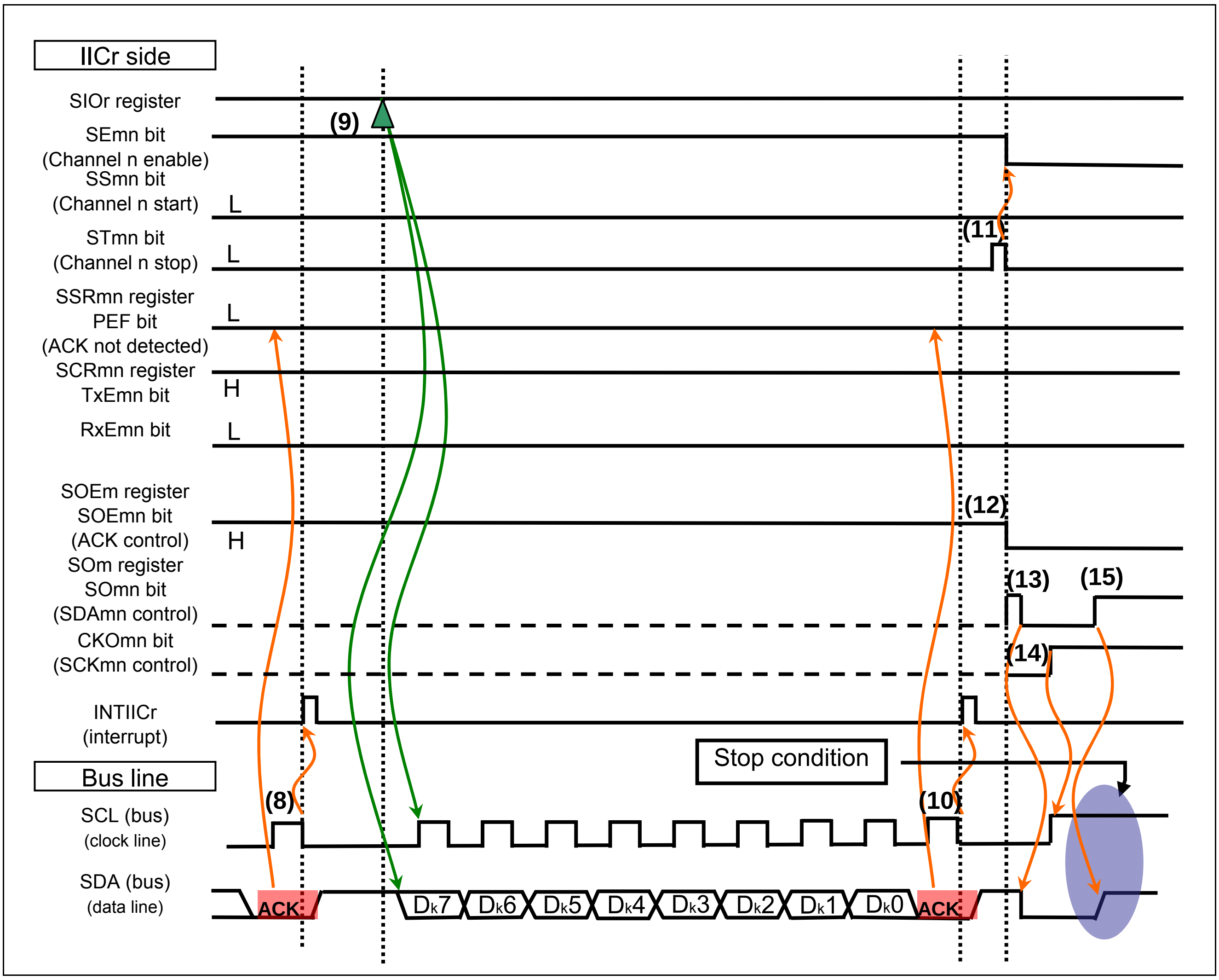


Figure 1.4 IIC Communication Timing Chart (Master-to-Slave Communication Example) (3/4)

(8) An INTIICr is generated when the transmission of the data is completed.

(9) Check the ACK response from the slave by testing the PEF bit of the SSRmn register and load the SIOr register with the transmit data if the PEF bit is found to be 0. Abort the transmission is if the PEF bit is set to 1.

(10) An INTIICr is generated when the transmission of the data is completed.

(11) Set the STmn bit of the STm register to 1 to disable channel n.

(12) To issue a stop condition, set the SOEmn bit of the SOEm register to 0 to disable output.

(13) To set SDA low in preparation for issuing a stop condition, set the SOmn bit of the SOm register to 0.

(14) To set SCL high in preparation for issuing a stop condition, set the CKOmn bit of the SOm register to 1.

(15) After the lapse of the stop condition setup time, set the SOmn bit of the SOm register to 1, and a stop condition will be issued.

(4) Master-to-slave communication 4 (data – restart condition – address)

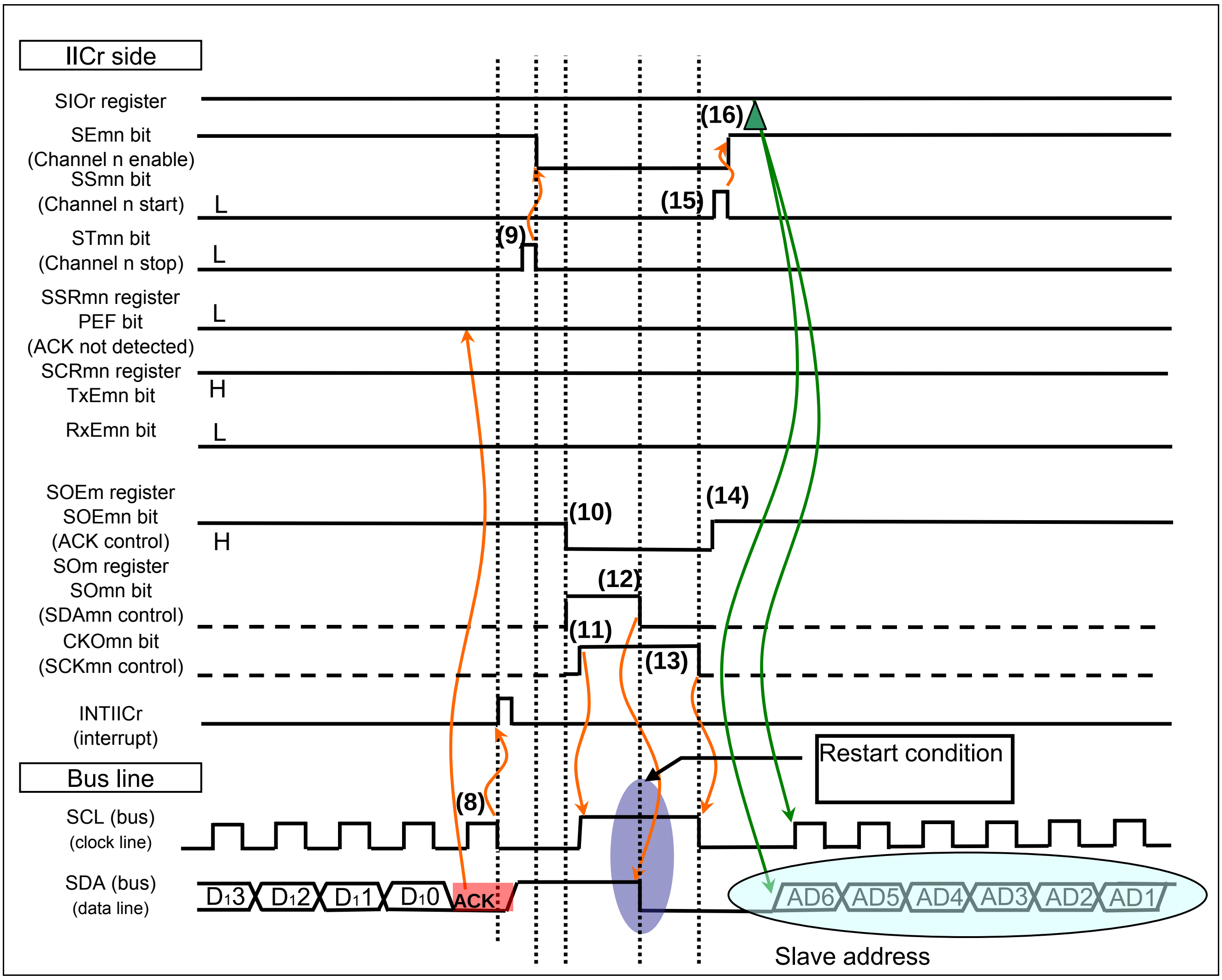


Figure 1.5 IIC Communication Timing Chart (Master-to-Slave Communication Example) (4/4)

(8) An INTIICr is generated when the transmission of the data is completed. Check the ACK response from the slave by testing the PEF bit of the SSRmn register.

(9) Set the STmn bit of the STm register to 1 to disable channel n.

(10) To issue a restart condition, set the SOEmn bit of the SOEm register to 0 to enable output.

(11) To set SCL high following SDA in preparation for issuing a restart condition, set the CKOmn bit of the SOm register to 1.

(12) Set the SOmn bit of the SOm register to 0, and a restart condition will be issued.

(13) After the lapse of the start condition hold time, set the CKOmn bit of the SOm register to 0 to set the SCL signal low.

(14) For communication, set the SOEmn bit of the SOEm register to 1 to enable output.

(15) Set the SSmn bit of the SSm register to 1 to enable channel n.

(16) Load the SIOr register with the address of the slave, and communication will start.

Remarks: This processing is used during EEPROM read processing, i.e., when specifying the cell address of the EEPROM and reading data from the specified address through master-to-slave communication.

(5) Slave-to-master communication 1 (start condition – address – data)

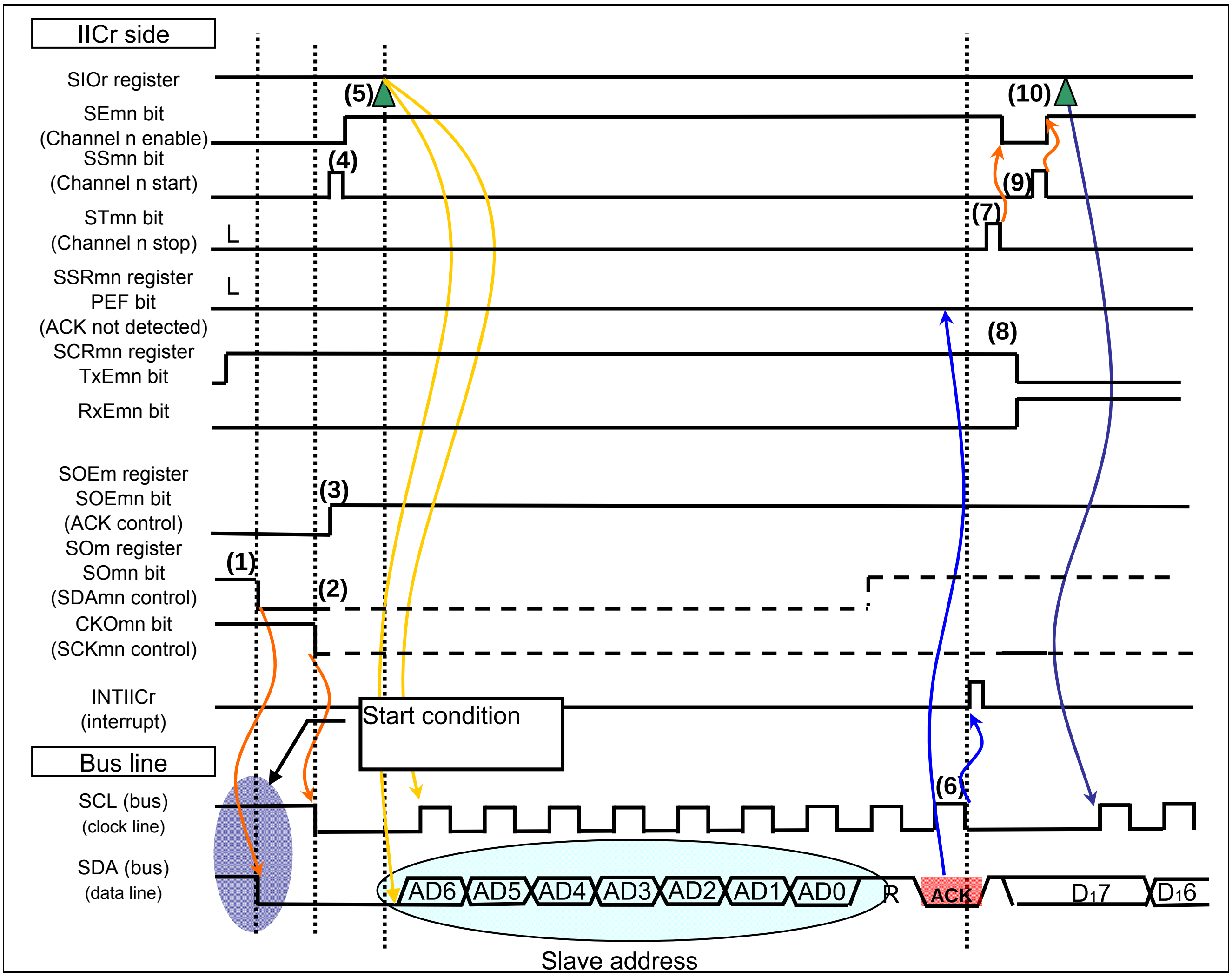


Figure 1.6 IIC Communication Timing Chart (Slave-to-Master Communication Example) (1/3)

(1) After IICmn initialization (here the SCRmn register is set up such that TxEmn is 1 and RxEmn is 0) is completed, set the SDA signal low by setting the SOmn bit of the SOm register to 0 to issue a start condition.

(2) After the lapse of the start condition hold time (4.0 μs in standard mode and 0.6 μs in fast mode), set the CKOmn bit of the SOm register to 0 to set the SCL signal low.

(3) For communication, set the SOEmn bit of the SOEm register to 1 to enable output.

(4) Set the SSmn bit of the SSm register to 1 to enable channel n.

(5) Write the address of the slave into the SIOr register, and communication will start.

(6) An INTIICr is generated when the transmission of the slave address is completed. Check the ACK response from the slave by testing the PEF bit of the SSRmn register.

(7) Disable the IICmn to switch the direction of communication.

(8) Set up the SCRmn register so that TxEmn is set to 0 and RxEmn to 1.

(9) Enable the IICmn.

(10) Write dummy data (0FFH) into the SIOr register to start receive processing.

(6) Slave-to-master communication 2 (address – data – data)

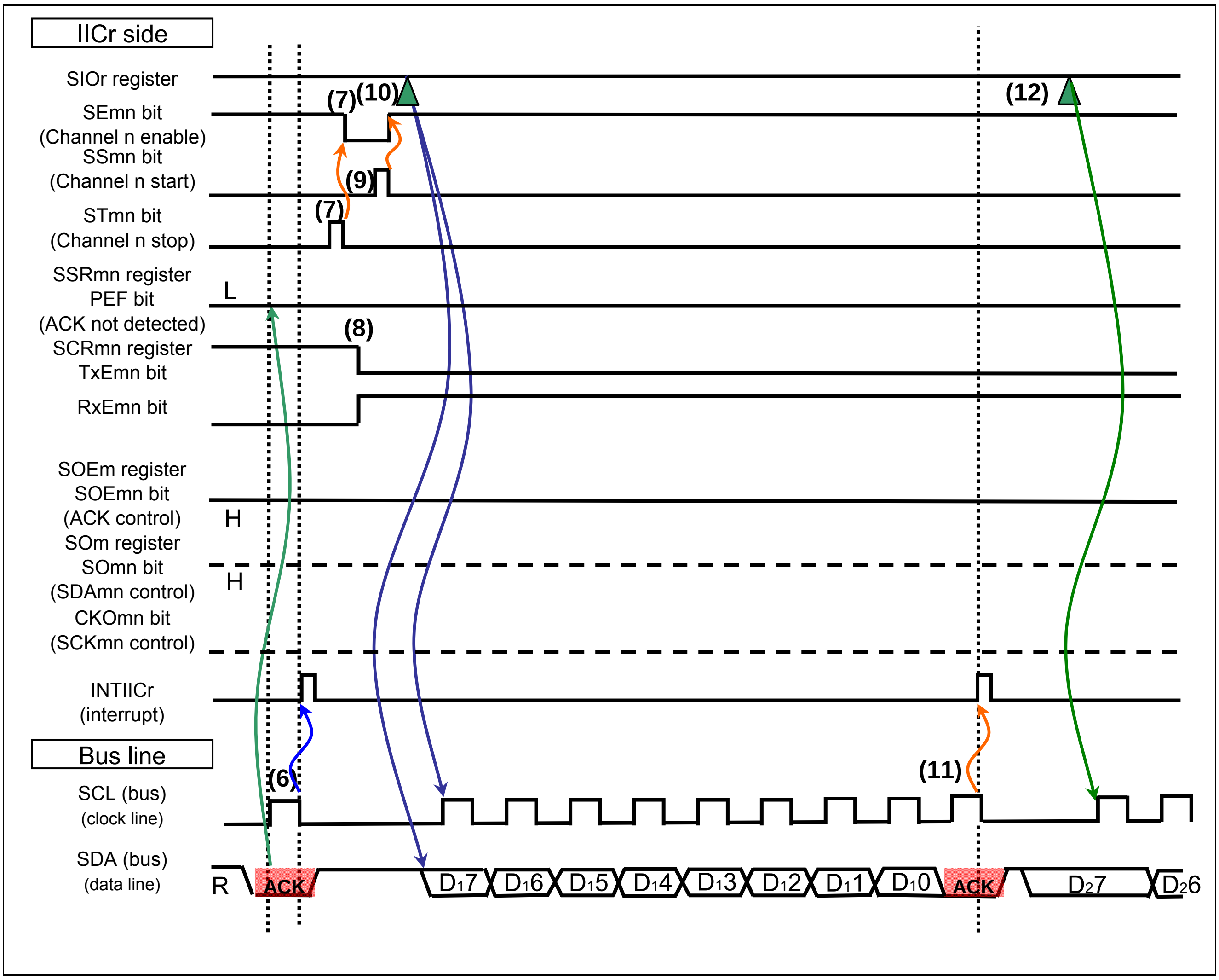


Figure 1.7 IIC Communication Timing Chart (Slave-to-Master Communication Example) (2/3)

(6) An INTIICr is generated when the transmission of the slave address is completed. Check the ACK response from the slave by testing the PEF bit of the SSRmn register.

(7) Disable the IICmn to switch the direction of communication.

(8) Set up the SCRmn register so that TxEmn is set to 0 and RxEmn to 1.

(9) Enable the IICmn.

(10) Write dummy data (0FFH) into the SIOr register to start receive processing.

(11) Since the SOEmn bit is set to 1, an ACK response is made on the 9th clock of SCL and an INTIICr is generated to signal the completion of receive processing.

(12) To receive the next data (not the last data), write dummy data (0FFH) into the SIOr register to start receive processing.

(7) Slave-to-master communication 3 (data – data – stop condition)

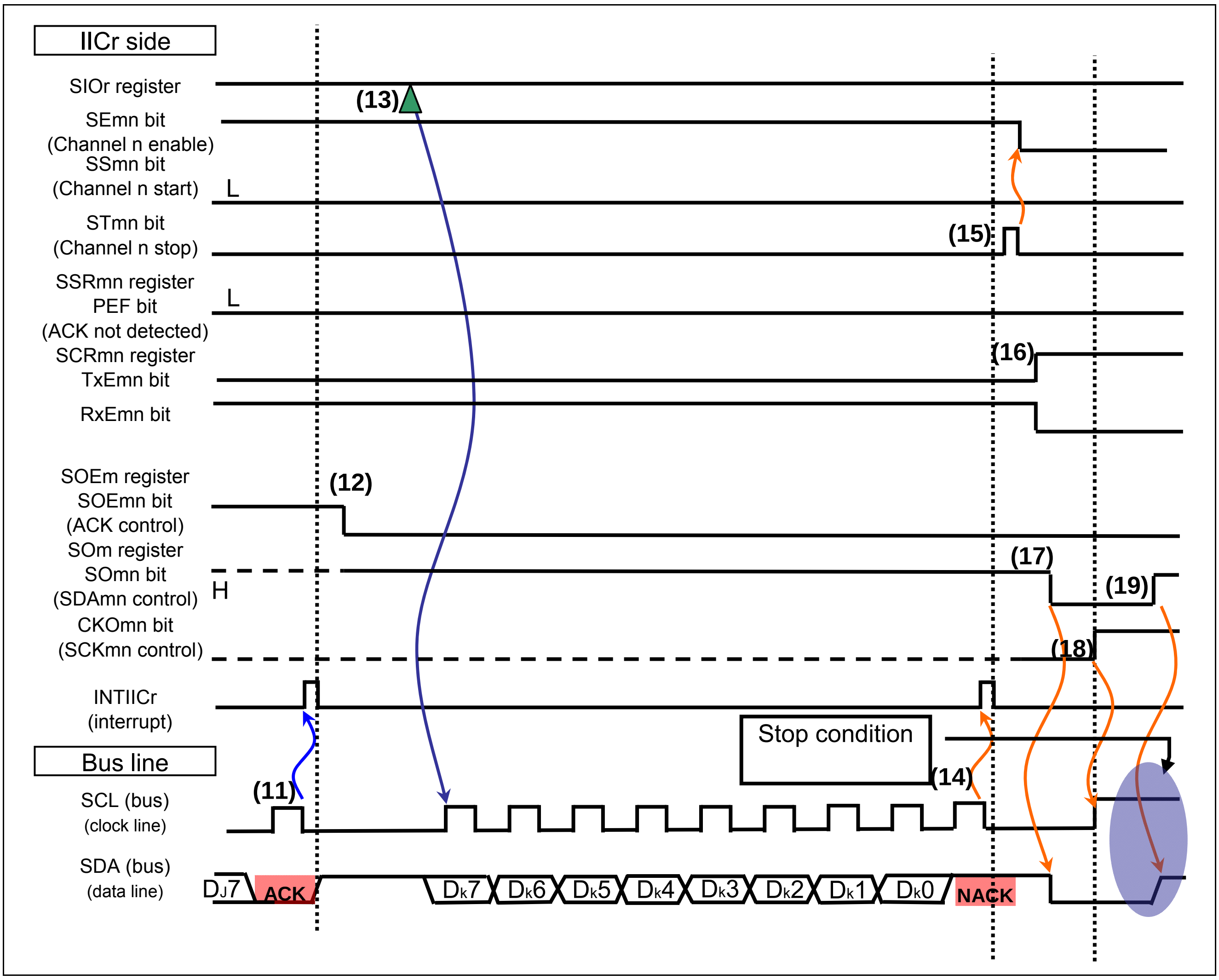


Figure 1.8 IIC Communication Timing Chart (Slave-to-Master Communication Example) (3/3)

(11) Since the SOEmn bit is set to 1, an ACK response is made on the 9th clock of SCL and an INTIICr is generated to signal the completion of receive processing.

(12) To respond with NACK for the last receive data, set the SOEmn bit of the SOEm register to 0 and disable serial output.

(13) Write dummy data (0FFH) into the SIOr register to start receive processing.

(14) Since the SOEmn bit is set to 0, a NACK response is made on the 9th clock of SCL and an INTIICr is generated to signal the completion of receive processing.

(15) Set the STmn bit of the STm register to 1 to disable channel n.

(16) For the next communication operation, set up the SCRmn register into its initial state.

(17) To set SDA low in preparation for issuing a stop condition, set the SOmn bit of the SOm register to 0.

(18) To set SCL high in preparation for issuing a stop condition, set the CKOmn bit of the SOm register to 1.

(19) After the lapse of the stop condition setup time, set the SOmn bit of the SOm register to 1, and a stop condition will be issued.

2. Operation Check Conditions

The sample code contained in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
Microcontroller used	RL78/G13 (R5F100LE)
Operating frequency	• High-speed on-chip oscillator (HOCO) clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	5.0 V (Operation is possible over a voltage range of 2.9 to 5.5 V.) LVD operation (V_{LVI}): Reset mode 2.81 V (2.76 to 2.87 V)
Integrated development environment (CubeSuite+)	CubeSuite + V1.01.01
C compiler (CubeSuite+)	CA78K0R V1.30
Integrated development environment (e2studio)	e2studio V2.0.1.3
C compiler (e2studio)	KPIT GNURL78-ELF Toolchain V13.02
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V1.30.2
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V1.30.2.50607
Board used	QB-R5F100LE-TB + EEPROM (R1EX24016, R1EX24032)

Note: This sample code is compatible with the RL78/G13 devices with 64 pins.

3. Related Application Note

The application notes that are related to this application note are listed below for reference.

- RL78/G13 Initialization (R01AN0451E) Application Note
- RL78/G13 Timer Array Unit (Interval Timer) (R01AN0456E) Application Note

4. Description of the Hardware

4.1 Hardware Configuration Example

Figure 4.1 shows an example of hardware configuration that is used for this application note.

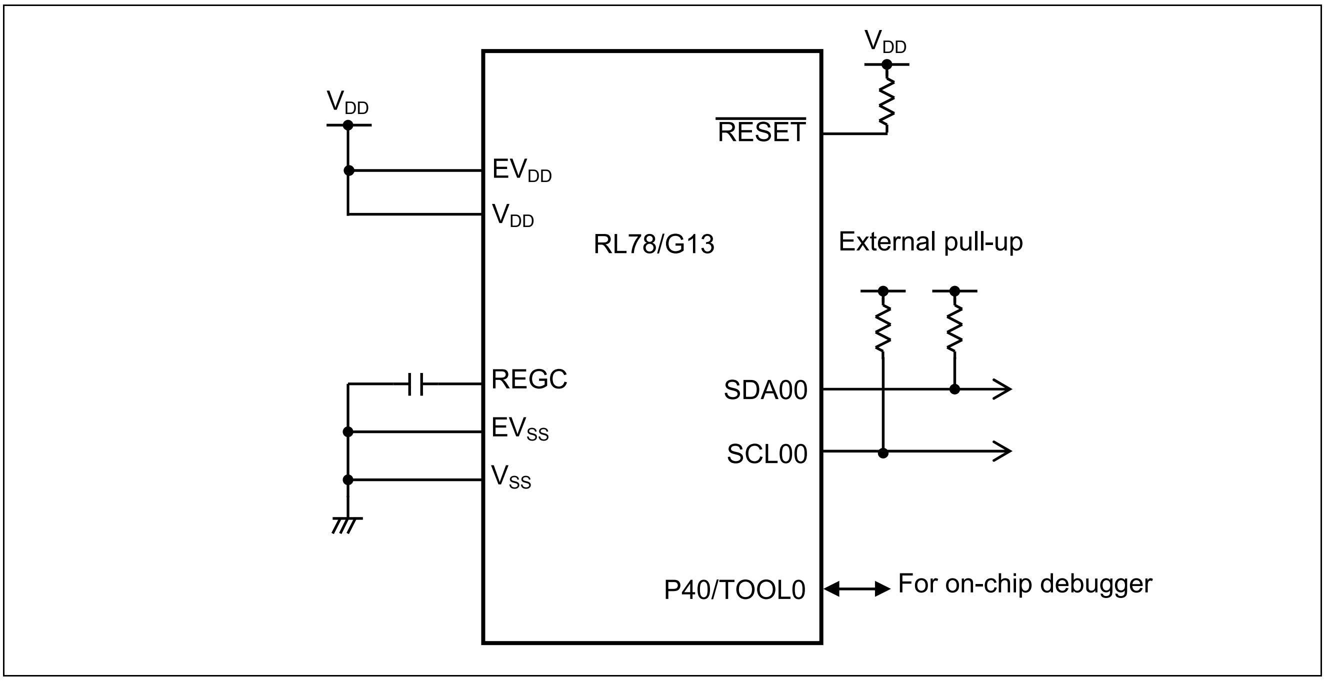


Figure 4.1 Hardware Configuration

Cautions: 1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).

2. Connect any pins whose name begins with EV_{SS} to V_{SS} and any pins whose name begins with EV_{DD} to V_{DD}, respectively.

3. V_{DD} must be held at not lower than the reset release voltage (V_{LVI}) that is specified as LVD.

4.2 List of Pins to be Used

Table 4.1 lists the pins to be used and their functions.

Table 4.2 Pins to be Used and their Functions

Pin Name	I/O	Description
P10/SCL00	Input/Output	IIC00 serial clock output pin
P11/SDA00	Input/Output	IIC00 serial data transmit/receive pin

5. Description of the Software

5.1 Operation Outline

The sample application covered in this application note controls operations (read and write) on EEPROM devices using the IIC master transmit/receive functions of the IIC00 serial interface. Control of EEPROM devices is done in an interrupt-driven mode whenever possible with due consideration to the function as an API.

(1) Initializes channel 0 of the serial array unit 0 through simplified IIC.
<Setting conditions>

- Sets the operating clock to CK00 (32 MHz).
- Sets the operating mode to "simplified IIC."
- Enables transfer end interrupts.
- Sets the phase of data and clock to type 1.
- Sets the data length to 8 bits, the stop bit length to 1 bit, no parity, and MSB first transfer.
- Sets the transfer clock to 381 kHz of fast mode.
- Sets SO00 and CKO00 to 1.
- Sets the P10/SCL00 pin for transfer clock output and the P11/SDA00 pin for data transmit/receive.

(2) Sets channel 2 of the timer array unit as a 100-µs interval timer for checking the completion of write processing.
(3) Copies the parameters for the EEPROM (16 kbits) to be used into the processing parameter structure.
(4) Issues a stop condition to set the bus free.
(5) Creates 256 bytes of write data (increment pattern).
(6) Sets up the access parameters (structure g_PARAI).
(7) Writes 256 bytes of data to the EEPROM starting at address 0x400.
(8) Reads out the written data with the leading and tailing padding bytes.
(9) Prepares 16 bytes of data of the same pattern (0xkk) (kk = 00, 11, 22, …, 77).
(10) Writes that data to the EEPROM starting at address 0xk00.
(11) Repeats steps (9) and (10) while changing the value of k from 0 to 7.
(12) Reads out the data that is written in steps (9) and (10) including 8 bytes each of the leading and trailing padding data, a total of 32 bytes.

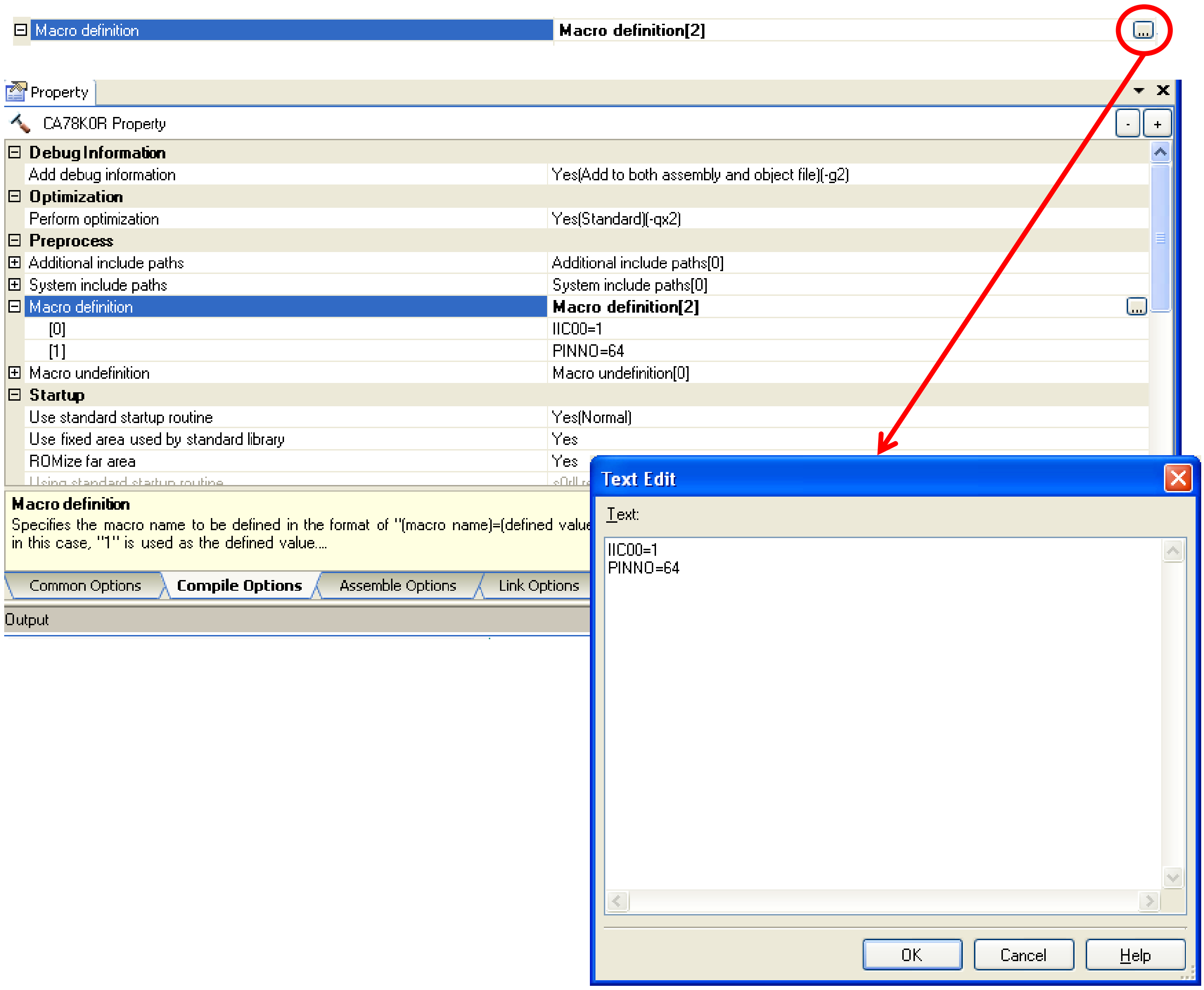
Note: This sample code is available to illustrate the example of controlling the EEPROM (R1EX24016 or R1EX24032) on an IIC bus using the simplified IIC00 function of the RL78/G13 series. When the channel or EEPROM to be used with this sample code is changed, conduct extensive evaluation of the modified code.

Remarks: In the project that is used, the simplified IIC channel and device's pin count are specified using the [Macro definition] function of the [Compile option]. The following illustration is for CubeSuite+ environment.

The Macro definition[0] selects IIC00 as the simplified IIC channel to be used. When using channel 20, for example, modify there to "IIC20 = 1."

The Macro definition[1] specifies the number of pins of the product to be used. It is used to check whether the channel specified in the Macro definition[0] is available for the target product.

To change these, click the button at the right-most end of [Macro definition] to open the text edit window, then edit the definitions in that window.



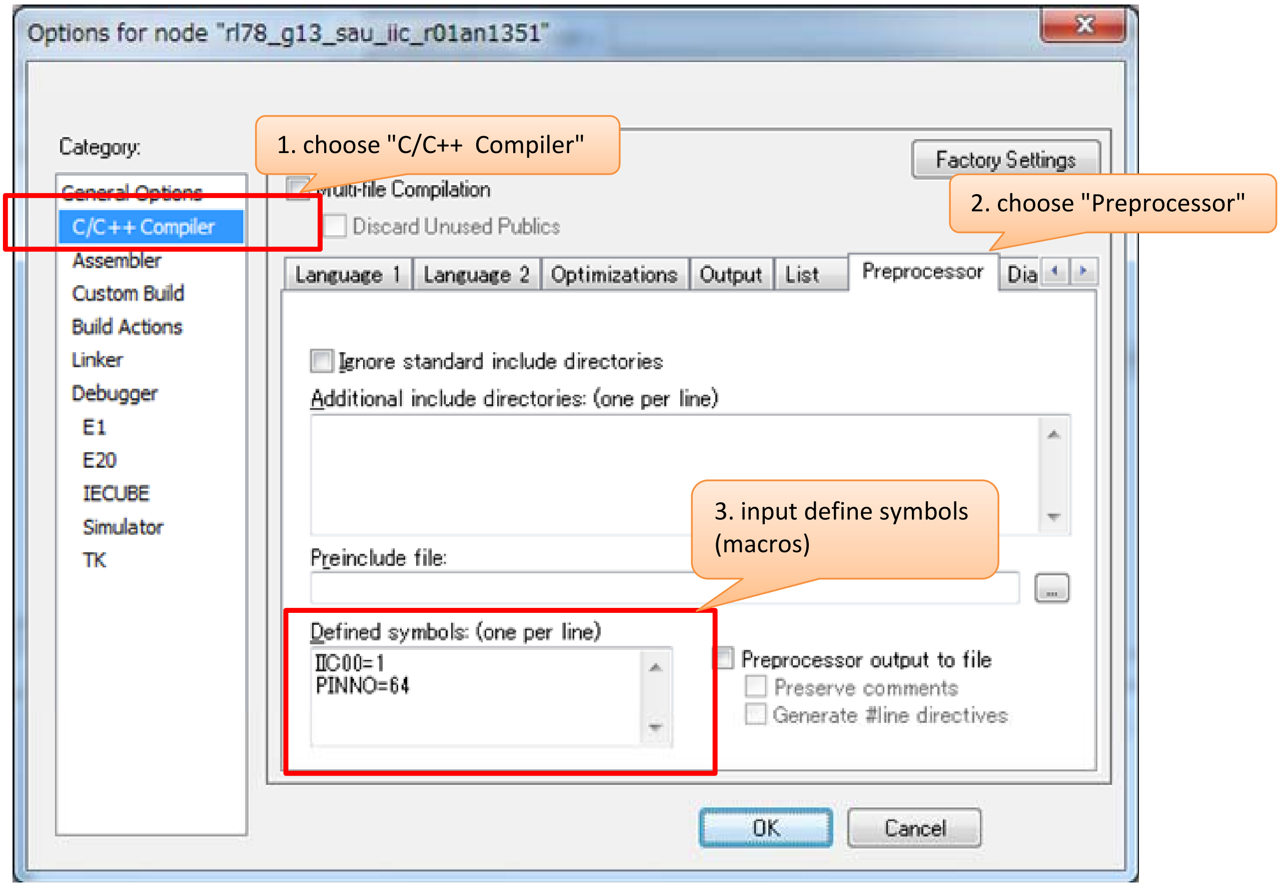
⊟ Macro definition	Macro definition[2]

⊟ **Debug Information**	
Add debug information	Yes(Add to both assembly and object file)(-g2)
⊟ **Optimization**	
Perform optimization	Yes(Standard)(-qx2)
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊟ Macro definition	**Macro definition[2]**
[0]	IIC00=1
[1]	PINNO=64
⊞ Macro undefinition	Macro undefinition[0]
⊟ **Startup**	
Use standard startup routine	Yes(Normal)
Use fixed area used by standard library	Yes
ROMize far area	Yes

Macro definition
Specifies the macro name to be defined in the format of "(macro name)=(defined value in this case, "1" is used as the defined value....

```
IIC00=1
PINNO=64
```

The IAR environment has a similar interface for this definition. Right click on the project name, and choose "Options". Choose the tab "C/C++ Compiler", and "Preprocessor" in the diarog shown below. In the last place, input your necessary define symbols in the field of "Defined symbols: (one per line)".



1. choose "C/C++ Compiler"

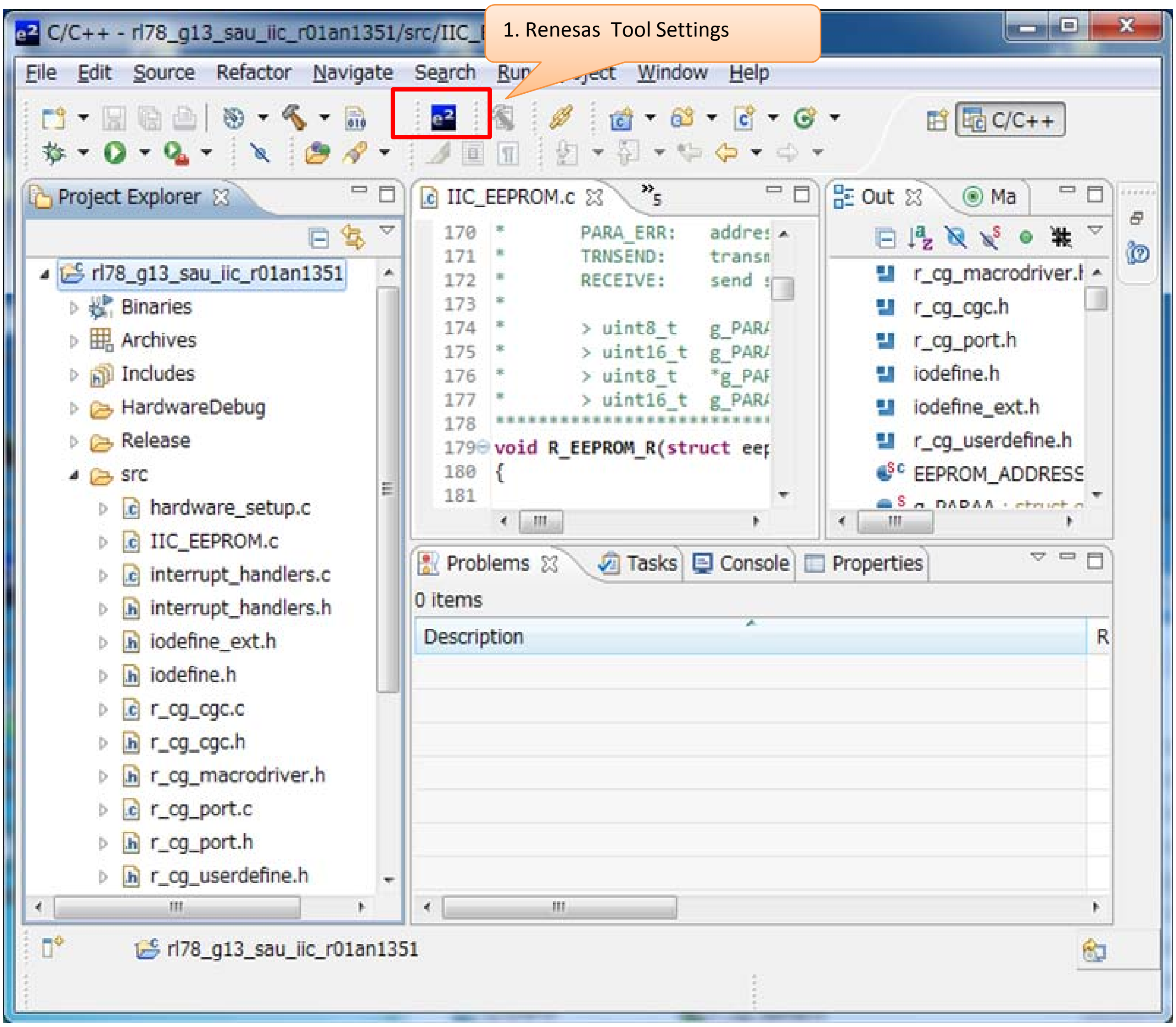
2. choose "Preprocessor"

3. input define symbols (macros)

Defined symbols: (one per line)

```
IIC00=1
PINNO=64
```

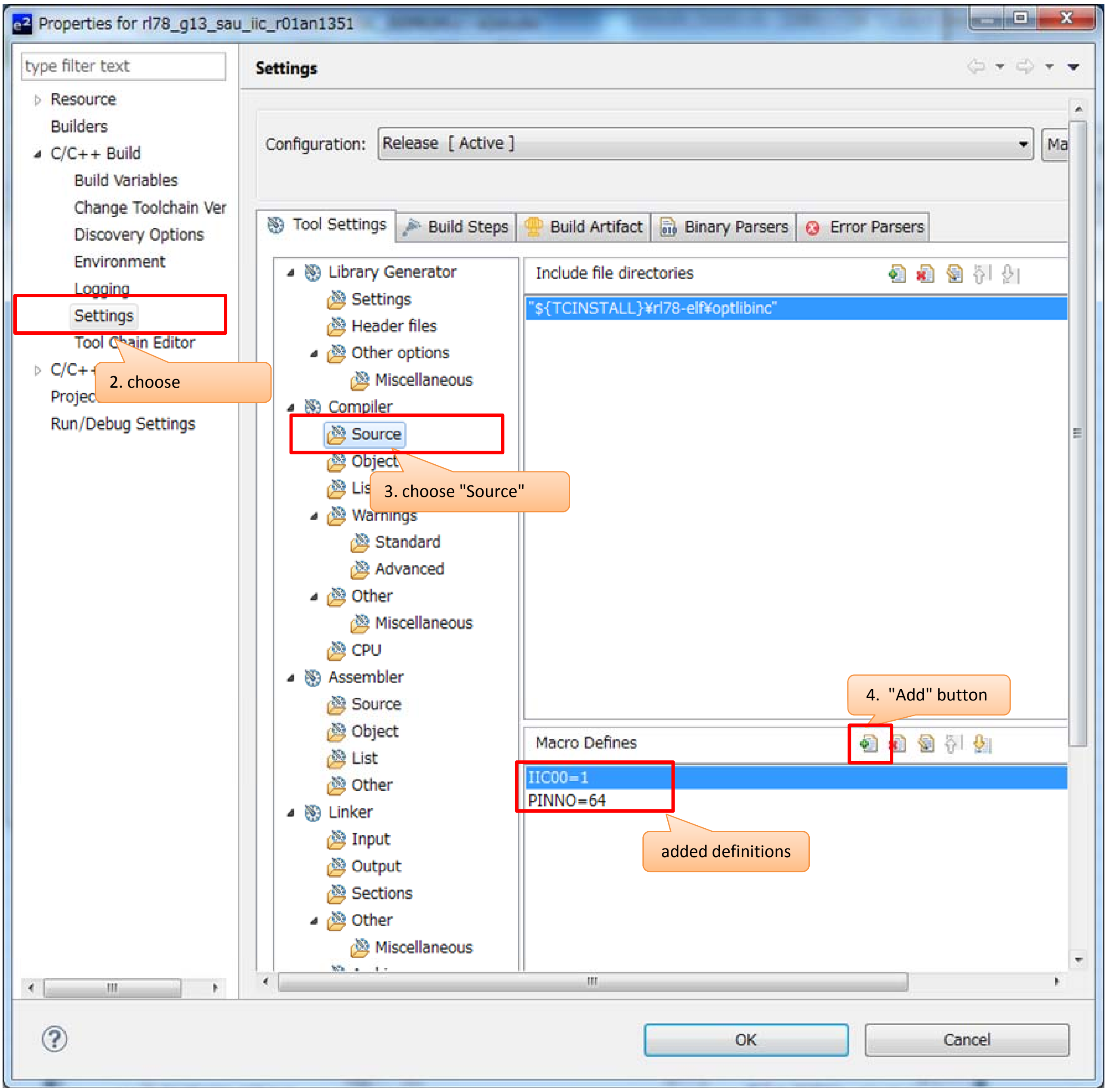
In the e2studio environment, please follow the following procedure. First, click "Rnesas Tool Settings".



1. Renesas Tool Settings

```
 *        PARA_ERR:    addres
 *        TRNSEND:     transm
 *        RECEIVE:     send s
 *
 *        > uint8_t    g_PARA
 *        > uint16_t   g_PARA
 *        > uint8_t    *g_PAR
 *        > uint16_t   g_PARA
 ***************************
void R_EEPROM_R(struct eep
{
```

Next, choose "Setting" in "C/C++ Build", "Tool Settings" tab, and "Source" in "Compiler". Then write in the field of "Macro Defines" the definition macros which you used in Macro definition in CubeSuite+ environment. In the last place, click "Add" button and you can add the setting to the system.



Configuration: Release [Active]

Include file directories: "${TCINSTALL}¥rl78-elf¥optlibinc"

Macro Defines:

IIC00=1
PINNO=64

5.2 List of Option Byte Settings

Table 5.1 summarizes the settings of the option bytes.

Table 5.1 Option Byte Settings

Address	Value	Description
000C0H/010C0H	01101110B	Disables the watchdog timer. (Stops counting after the release from the reset state.)
000C1H/010C1H	01111111B	LVD reset mode, 2.81 V (2.76 to 2.87 V)
000C2H/010C2H	11101000B	HS mode, HOCO: 32 MHz
000C3H/010C3H	10000100B	Enables the on-chip debugger.

5.3 List of Constants

Constants for the Sample Program (Table 5.2 and Table 5.3) list the constants that are used in this sample program.

Table5.4 shows a list of names of macros specifying the channel numbers to be used in the sample program.

Table 5.2 Constants for the Sample Program (1/2)

Constant	Setting	Description
PAGE_16	0x000F	Number of data bytes per page of 2 to 16 kbits of EEPROM
PAGE_32	0x001F	Number of data bytes per page of 32 or 64 kbits of EEPROM
PAGE_64	0x003F	Number of data bytes per page of 128 or 256 kbits EEPROM
PAGE_128	0x007F	Number of data bytes per page of 512 kbits of EEPROM
MEMORY_2K	0x0001	2 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_4K	0x0002	4 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_8K	0x0004	8 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_16K	0x0008	16 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_32K	0x0010	32 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_64K	0x0020	64 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_128K	0x0040	128 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_256K	0x0080	256 kbits of EEPROM capacity (unit of 256 bytes)
MEMORY_512K	0x0100	512 kbits of EEPROM capacity (unit of 256 bytes)
ADDR0BIT	0b00000000	Do not use the slave address for cell addresses.
ADDR1BIT	0b00000001	Specify A8 with bit 1 of the slave address.
ADDR2BIT	0b00000011	Specify A9 and 8 with bits 2 and 1 of the slave address.
ADDR3BIT	0b00000111	Specify A10 to A8 with bits 3 to 1 of the salve address.
I2C_OK	0x00	Normal termination
PARA_ERR	0x20	Parameter error
NO_ACK1	0x40	No ACK response to slave address
NO_ACK2	0x41	No ACK response to EEPROM address
NO_ACK3	0x42	No ACK response to transmit data
BUS_ERR	0x60	Bus is not free (SDA is not high).
SVAMSK	0b11111110	Mask data for bit 0 of the slave address
SCLLOWW	0x05	SCL low level time measurement data
SCLHIGHW	0x02	SCL high level time measurement data
RETRYCNT	0x09	Number of SCL dummy clock pulses

Table 5.3 Constants for the Sample Program (2/2)

Constant	Setting	Description
R1EX24002A	0x00	Constants for specifying the EEPROM to be used. Defined by the enumeration type constant eeprom_name. Used to reference the EEPROM parameters through the eeprom_info type structure EEPROM_ADDRESS.
R1EX24004A	0x01	
R1EX24008A	0x02	
R1EX24016A	0x03	
R1EX24032A	0x04	
R1EX24064A	0x05	
R1EX24128B	0x06	
R1EX24256B	0x07	
R1EX24512B	0x08	

Table 5.4 Macro Names Used in the Sample Program

Macro Name	Setting (Channel 0)	Description
SAUmEN	SAU0EN	Peripheral enable register
SPSm	SPS0	Serial clock selection register
SMRmn	SMR00	Serial mode register
SCRmn	SCR00	Serial communication register
SDRmn	SDR00	Serial data register
SIOr	SIO00	Serial data register (for data transmission/reception)
SSRmn	SSR00	Serial status register
SIRmn	SIR00	Serial flag clear trigger register
SSmL	SS0L	Serial channel start register
STmL	ST0L	Serial channel stop register
TRGONn	0b00000001	Trigger bit
SOEmL	SOE0L	Serial output enable register
SOEON	TRGONn	Serial output enable register setting bit
SOEOFF	(uint8_t)(~SOEON)	Bits for clearing the serial output enable register
SOm	SO0	Serial output register
SDAHIGH	TRGONn	Bit for setting SDA high
SDALOW	~SDAHIGH	Bit for setting SDA low
SCLHIGH	TRGONn × 0x100	Bit for setting SCL high
SCLLOW	~SCLHIGH	Bit for setting SCL low
IICIFr	IICIF00	Interrupt request flag register
IICMKr	IICMK00	Interrupt request mask register
PM_IICr	PM1	IICr port mode register (specifying SDA signal)
PM_SDAr	PM1.1	SDA port mode register bits
PM_SCLr	PM1.0	SCL port mode register bits
POM_IICr	POM1	Port output mode register
P_IICr	P1	IICr port
P_SDAr	P1.1	SDA port
P_SCLr	P1.0	SCL port
SDAINMODE	0b00000010	Bits for specifying port mode register input
SDAOUTMODE	0b11111100	Bits for port mode register output
SDASCLON	(uint8_t)(~SDAOUTMODE)	Port setting bits
IICPR0r	IICPR000	Interrupt priority setting registers
IICPR1r	IICPR100	

5.4 List of Variables

Table 5.5 lists the global variables that are used in this sample program.

g_eeprom_type and the subsequent entries contain global variables that are valid within the module.

Table 5.5 Global Variables for the Sample Program

Type	Variable Name	Contents	Function Used

Structure eeprom_paraA16	g_PARAI	Parameters for specifying EEPROM access	main()
			check_EEPROM_Addr()

uint8_t	g_comstatus	Operation information/result flag	main()
			check_EEPROM_Addr()
			R_EEPROM_R()
			R_EEPROM_wait_read()
			R_IICr_Tx_addr1()
			R_IICr_Tx_addr2()
			R_IICr_Rx_RST()
			R_IICr_RxData_ST()
			R_IICr_RxData()
			R_IICr_Rx_LastData()
			R_EEPROM_W()
			R_EEPROM_wait_write()
			R_IICr_TxDataST()
			R_IICr_TxData()
			R_EEPROM_next_page()
			IINTIICr()

uint8_t array (256)	g_data_bufferW1	Write data buffer	main()
uint8_t array (256)	g_data_bufferW2	Write data buffer	main()
uint8_t array (512)	g_data_bufferR1	Read data buffer	main()

Type	Variable Name	Contents	Function Used
uint8_t	g_eeprom_type	No. of EEPROM to be used	R_device_select()

Structures eeprom_paraA16
uint8_t slaveaddr;
uint16_t eepromaddr;
uint8_t *bufferaddr;
uint16_t number;
	g_PARAA	Parameters for accessing EEPROM	R_EEPROM_R()
			R_IICr_Tx_addr1()
			R_IICr_Tx_addr2()
			R_IICr_Rx_RST()
			R_IICr_RxData_ST()
			R_IICr_RxData()
			R_IICr_Rx_LastData()
			R_EEPROM_W()
			R_IICr_TxDataST()
			R_IICr_TxData()
			R_EEPROM_Devide()
			SINTTM02()
			R_EEPROM_wait_write()

Structure eeprom_paraA16	g_PARAC	Copy of parameters for specifying access to EEPROM	check_EEPROM_Addr()
			R_EEPROM_R()
			R_EEPROM_W()
			R_IICr_TxData()
			get_slave_Addr()
			R_EEPROM_Devide()

Structure eeprom_info
uint16_t page_size;
uint16_t rom_size;
uint8_t addr_mask;
uint8_t mask2;
	EEPROM_Info	For holding parameters for the EEPROM to be used (for processing)	R_device_select()
			R_IICr_Tx_addr1()
			get_slave_Addr()
			R_EEPROM_Devide()

5.5 List of Functions

Table 5.6 summarizes the functions that are used in this sample program.

Table 5.6 Functions

Function Name	Outline
R_device_select	Specifies the EEPROM to be used.
R_EEPROM_R	Reads data from the EEPROM according to the accessing parameters in the structure that is pointed to by the argument pointer.
R_EEPROM_wait_read	Waits for the completion of the read from EEPROM.
R_EEPROM_W	Writes data into the EEPROM according to the accessing parameters in the structure that is pointed to by the argument pointer.
R_EEPROM_wait_write	Waits for the completion of the write to EEPROM.
check_EEPROM_Addr	Checks if the specified parameter exceeds the EEPROM capacity. Copies the accessing parameter if the parameter falls within the specified capacity.
get_slave_Addr	Includes the upper bits of the cell address into the slave address when the EEPROM capacity is 4 to 16 kbits.
R_EEPROM_Devide	Corrects the 1-write parameter so that the write data falls within 1 page in write mode.
R_IICr_Tx_addr1	Performs slave address transmission completion processing and sends the cell address of the EEPROM.
R_IICr_Tx_addr2	Sends the lower bit address when the cell address of the EEPROM is 2 bytes long.
R_IICr_Rx_RST	Restarts processing in receive mode to read data following the completion of the transmission of the EEPROM cell address.
R_IICr_RxData_ST	Starts the data receive processing for receiving data with TxE set to 0 and RxE set to 1. If the data to read is 1 byte long, start the receive processing after setting up a NACK response.
R_IICr_RxData	Stores the received data in the buffer and starts the next data reception processing.
R_IICr_Rx_LastData	Stores the last received data in the buffer, sets TxE to 1 and RxE to 0, and terminates processing after issuing a stop condition in preparation for the next communication.
R_IICr_TxDataST	Starts transmission of write data to the EEPROM.
R_IICr_TxData	After data transmission is completed and the transmission of all data to the current page is ended, issues a stop condition and specifies the write to the EEPROM. Terminates processing if processing of all data is completed. If there is remaining data, makes preparation for the next page write and starts the timer for awaiting the completion of the write processing.
R_EEPROM_next_page	Upon completion of the write to EEPROM, stops the timer and sends the cell address for the next EEPROM page.

Function Name	Outline
R_IICr_StartCond	Manipulates the SOm register, issues a start condition, and enable the IICr.
R_IICr_StopCond	Issues a stop condition and checks if the bus is freed. If the bus is not free, sends dummy clocks to SCL and reissues a stop condition.
R_IICr_send_Stop	Disables the IICr and manipulates the SOm register to issue a stop condition.
R_IICr_wait_bus	Sends 9 dummy clock pulses on the SCL signal line and checks if the SDA signal is becomes free (goes high).
R_IICr_SCL_pulse	Sends dummy clock pulses to the SCL signal.
R_IICr_SCL_high	Sets the SCL signal high and waits for the duration equal to the high level width.
R_IICr_SCL_low	Sets the SCL signal low and waits for the duration equal to the low level width.
R_IICr_NACK	Checks for an ACK/NACK response from the slave.
R_IICr_SCL_Time	Waits for the duration equal to the low period of the SCL signal.
R_IICr_SCL_highTime	Waits for the duration equal to the high period of the SCL signal.
IINTIICr	Checks the ACK response from the slave on an IICr transfer completion interrupt and allocates program control to the next processing.
IINTTM02	Sends the slave address that is used to confirm the completion of write processing during the TM02 interval interrupt processing.
R_IICr_Init	Initializes the IICr and TM02.

5.6 Function Specifications

This section describes the specifications for the functions that are used in this sample program

[Function Name] R_device_select

Synopsis	Specify EEPROM to be used.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	MD_STATUS R_device_select(enum eeprom_name);	
Explanation	This function copies the EEPROM parameter specified in the argument into the structure EEPROM_Info.	
Arguments	[EEPROM's name]	Name defined in the Enum type constant eeprom_name
Return value	When [I2C_OK]: Normal termination When [PARA_ERR]: Wrong name specified	
Remarks	None	

[Function Name] R_EEPROM_R

Synopsis	Process request to start read from EEPROM.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	void R_EEPROM_R(struct eeprom_paraA16 *PARA);	
Explanation	This function performs a read from EEPROM according to the parameters specified in the structure that is pointed to by the pointer in the argument.	
Arguments	*PARA	Pointer to the eeprom_paraA16 type structure
Return value	None	
Remarks	The processes of awaiting the completion of the read and getting the result must be done by R_EEPROM_wait_read.	

[Function Name] R_EEPROM_wait_read

Synopsis	Wait for completion of read from EEPROM.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	MD_STATUS R_EEPROM_wait_read (void);	
Explanation	This function waits for the completion of the read processing started by R_EEPROM_R.	
Arguments	None	
Return value	When [I2C_OK]	: Read terminates normally.
	When [PARA_ERR]	: Specified parameter out of valid EEPROM address range.
	When [NO_ACK1]	: No ACK response to slave address
	When [NO_ACK2]	: No ACK response to EEPROM address
Remarks		

[Function Name] R_EEPROM_W

Synopsis	Start write to EEPROM.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	void R_EEPROM_W(struct eeprom_paraA16 *PARA);	
Explanation	This function performs a write to EEPROM according to the parameters specified in the structure that is pointed to by the pointer in the argument.	
Arguments	*PARA	Pointer to the eeprom_paraA16 type structure
Return value	None	
Remarks	The processes of awaiting the completion of the write and getting the result must be done by R_EEPROM_wait_write.	

[Function Name] R_EEPROM_wait_write

Synopsis	Wait for completion of write to EEPROM.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	MD_STATUS R_EEPROM_wait_write(void);	
Explanation	This function waits for the completion of the write processing started by R_EEPROM_W.	
Arguments	None	-
Return value	When [I2C_OK]	: Writer terminates normally.
	When [PARA_ERR]	: Specified parameter out of valid EEPROM address range.
	When [NO_ACK1]	: No ACK response to slave address
	When [NO_ACK2]	: No ACK response to EEPROM address
	When [NO_ACK3]	: No ACK response to transmit data
Remarks	None	

[Function Name] check_EEPROM_Addr

Synopsis	Check access area in EEPROM.
Header	r_cg_macrodriver.h r_cg_userdefine.h
Declaration	static MD_STATUS check_EEPROM_Addr(struct eeprom_paraA16 *PARA);
Explanation	This function checks the paramers for accessing the EEPROM to determine whether the area to be read or written falls within the valid EEPROM address range.
Arguments	*PARA Pointer to the eeprom_paraA16 type structure
Return value	When [I2C_OK]: Accessing area falls within valid EEPROM address range. When [PARA_ERR]: Accessing area not falls within valid EEPROM address range.
Remarks	None

[Function Name] get_slave_Addr

Synopsis	Compute slave address of EEPROM
Header	r_cg_userdefine.h
Declaration	static void get_slave_Addr(void);
Explanation	This function qualifies the slave address of 4 to 16 kbits of EEPROM with the upper 1 to 3 bits of the cell address.
Arguments	None -
Return value	None
Remarks	The function qualifies the member slaveaddr with the value of the member eepromaddr of the structure g_PARAC.

[Function Name] R_EEPROM_Devide

Synopsis	Split into pages in EEPROM write mode.
Header	r_cg_userdefine.h
Declaration	static void R_EEPROM_Devide(void);
Explanation	This function splits the write data according to the page size of the EEPROM. It stores the parameters to be written during the current operation in the structure g_PARAA and the remaining data in the structure g_PARAC.
Arguments	None -
Return value	None
Remarks	The members eepromaddr and bufferaddr of the structure g_PARAC contains the current write parameters and the member number contains the data count for the next and subsequent write operations.

[Function Name] R_IICr_Tx_addr1

Synopsis	Transmit EEPROM address.
Header	r_cg_userdefine.h
Declaration	static void R_IICr_Tx_addr1(void);
Explanation	This function transmits the upper byte of the address of 32 kbits or more of EEPROM. For 16 kbits or less of EEPROM, the function transmits the 1 byte of cell address.
Arguments	None -
Return value	None
Remarks	Used by the INTIICmn processing (function name: IINTIICr).

[Function Name] R_IICr_Tx_addr2

Synopsis	Transmit lower address of EEPROM cell.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_Tx_addr2(void);	
Explanation	This function transmits the lower byte of the address of 32 kbits or more of EEPROM.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_Rx_RST

Synopsis	Restart processing in receive mode	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_Rx_RST(void);	
Explanation	Restarts processing in receive mode to read data following the completion of the transmission of the EEPROM cell address.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_RxData_ST

Synopsis	Start data read processing.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_RxData_ST(void);	
Explanation	This function switches the state of the IICr from transmit (TxE = 1, RxE = 0) to receive (TxE = 0, RxE = 1) upon completion of the transmission of the slave address in read mode and starts the receive processing (writing dummy data into SIOr). If the size of the data to be read is 1 byte long, the function sets up a NACK response before starting the receive processing.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_RxData

Synopsis	Data receive processing.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_RxData (void);	
Explanation	This function stores the received data in the buffer and starts the next data receive processing. If the data to be read out is 1 byte long, the function sets up a NACK response before starting the receive processing.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_Rx_LastData

Synopsis	Complete the reception of last data.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_Rx_LastData (void);	
Explanation	This function stores the received data in the buffer and stops the IICr. In preparation for the next communication, the function sets TxE to 1 and RxE to 0 and issues a stop condition before exiting.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_TxDataST

Synopsis	Start data transmission.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_ TxDataST (void);	
Explanation	This function starts the data transmit processing when the transmission of the EEPROM address is completed.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_TxData

Synopsis	Data transmission processing	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_ TxData (void);	
Explanation	This function transmits the next data after the transmission of 1 byte of data is completed. When the transmission of all data to be written into one page is completed, the function issues a stop condition and specifies the write into the EEPROM. The function terminates processing if the transmission of all data has been completed. If there is remaining data, the function makes preparation for writing into the next page, and invokes the timer for awaiting the completion of write processing.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_EEPROM_next_page

Synopsis	Specify address of next page following completion of wait for write.	
Header	r_cg_userdefine.h	
Declaration	static void R_EEPROM_next_page(void);	
Explanation	This function stops the waiting timer for writing upon completion of the write into one page and transmits the cell address of the next page in the EEPROM.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr). This function is invoked on an ACK response to the slave address transmission received during the write completion check.	

[Function Name] R_IICr_StartCond

Synopsis	Issue start condition.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_StartCond(void);	
Explanation	This function temporarily disables the IICr and issues a start condition to enable the IICr.	
Arguments	None	-
Return value	None	
Remarks	Used by the INTIICmn processing (function name: IINTIICr).	

[Function Name] R_IICr_StopCond

Synopsis	Perform bus freeing processing.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	MD_STATUS R_IICr_StopCond(void);	
Explanation	This function generates a stop condition for the IICr. If the bus cannot be freed, the function generates 9 dummy clocks on the SCL line and, when SDA is found high, reissues a stop condition.	
Arguments	None	-
Return value	When [I2C_OK]: Bus freeing complete When [BUS_ERR]: Could not free the bus.	
Remarks	None	

[Function Name] R_IICr_send_Stop

Synopsis	Issues stop condition.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_send_Stop (void);	
Explanation	This function disables the IICr and manipulates the SOm register to issue a stop condition.	
Arguments	None	-
Return value	None	
Remarks	This function is used to issue a stop condition internally.	

[Function Name] R_IICr_wait_bus

Synopsis	Check bus free state.	
Header	r_cg_macrodriver.h r_cg_userdefine.h	
Declaration	static MD_STATUS R_IICr_wait_bus(void);	
Explanation	This function places 9 dummy pulses on the SCL line with the IICr stopped and tests the SDA signal.	
Arguments	None	-
Return value	When [I2C_OK]: Successful in freeing the bus. When [BUS_ERR]: Could not free the bus (SDA signal did not go high.)	
Remarks		

[Function Name] R_IICr_SCL_pulse

Synopsis	Output dummy clocks to SCL signal	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_SCL_pulse(void);	
Explanation	This function manipulates the SOm register to output low pulses to SCL.	
Arguments	None	-
Return value	None	
Remarks	None	

[Function Name] R_IICr_SCL_high

Synopsis	Set SCL high.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_SCL_high(void);	
Explanation	This function manipulates the SOm register to set and hold the SCL signal high for the SCL high level period.	
Arguments	None	-
Return value	None	
Remarks	None	

[Function Name] R_IICr_SCL_low

Synopsis	Set SCL low.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_SCL_low(void);	
Explanation	This function manipulates the SOm register to set and hold the SCL signal low for the SCL low level period.	
Arguments	None	-
Return value	None	
Remarks	None	

[Function Name] R_IICr_NACK

Synopsis	Check ACK/NACK response from slave.	
Header	r_cg_userdefine.h	
Declaration	static MD_STATUS R_IICr_NACK(void);	
Explanation	This function returns the value of bit 1 (PEF bit = NACK) of the SSRMn register.	
Arguments	None	-
Return value	When [0x00]: ACK response present When [0x02]: No ACK response	
Remarks	This status flag should only be checked and not cleared.	

[Function Name] R_IICr_SCL_Time

Synopsis	Wait for SCL signal low period.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_SCL_Time(void);	
Explanation	This function waits for the SCL signal low period.	
Arguments	None	-
Return value	None	
Remarks		

[Function Name] R_IICr_SCL_highTime

Synopsis	Wait for SCL signal high period.	
Header	r_cg_userdefine.h	
Declaration	static void R_IICr_SCL_highTime(void);	
Explanation	This function waits for the SCL signal high period.	
Arguments	None	-
Return value	None	
Remarks		

[Function Name] IINTIICr

Synopsis	Process IICr transfer completion interrupt.	
Header	r_cg_userdefine.h	
Declaration	__interrupt void IINTIICr(void);	
Explanation	This function is invoked on an INTIICr and calls the necessary processing according to the state of communication. When processing other than EEPROM write completion wait is in progress and a NACK response from the slave is detected, the function sets an error flag in g_comstatus in according to the condition of processing.	
Arguments	None	-
Return value	None	
Remarks		

[Function Name] IINTTM02

Synopsis	Process 100-µs interval timer completion interrupt.	
Header	r_cg_userdefine.h	
Declaration	__interrupt void IINTTM02 (void);	
Explanation	This function sends a start condition and a slave address to check for the completion of write processing.	
Arguments	None	-
Return value	None	
Remarks	The evaluation of the result is done by IINTIICr.	

[Function Name] R_IICr_Init

Synopsis	Perform IICr initialization.
Header	r_cg_userdefine.h
Declaration	void R_IICr_Init (void);
Explanation	This function makes settings according to the IIC channel to be used. After completing the setup of the IICr, the function configures timer 02 as an interval timer for checking for the completion of write processing. After completing all settings, the function sets the initial value of the EEPROM to be used to 16 kbits.
Arguments	None -
Return value	None
Remarks	

5.7 Flowcharts

Figure 5.1 shows the overall flow of the sample program described in this application note.

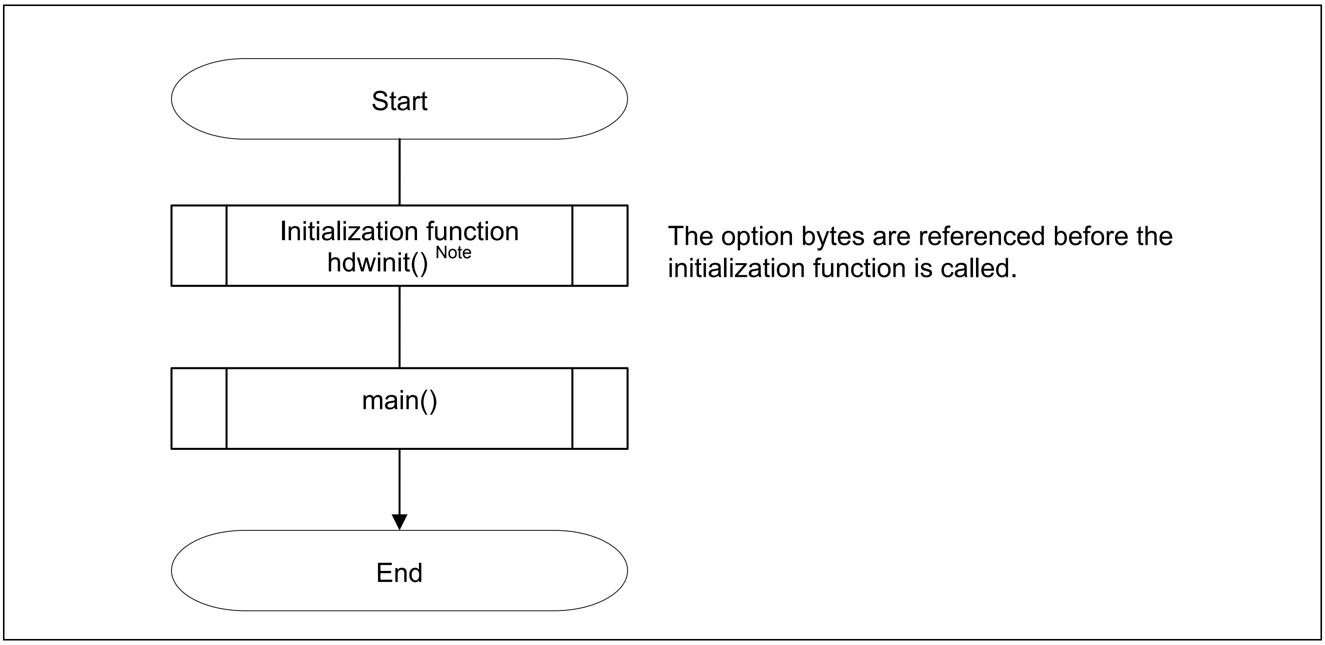

Start
Initialization function
hdwinit() Note
The option bytes are referenced before the initialization function is called.
main()
End

Figure 5.1 Overall Flow

5.7.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function

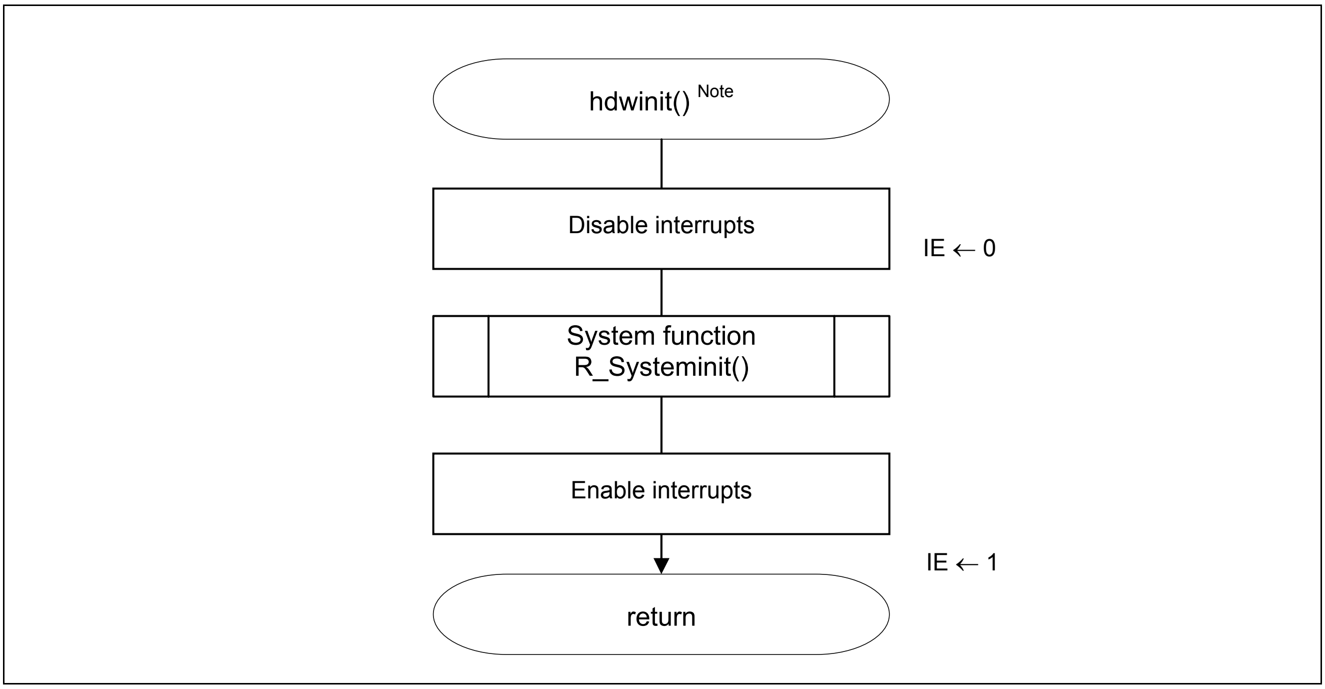


Figure 5.2 Initialization Function

Note: This function is used in the Renesas CubeSuite+ sample code. The function __low_level_init initializes the system in the IAR Workbench IDE-based sample code insted.

5.7.2 System Function

Figure 5.3 shows the flowchart for the system function.

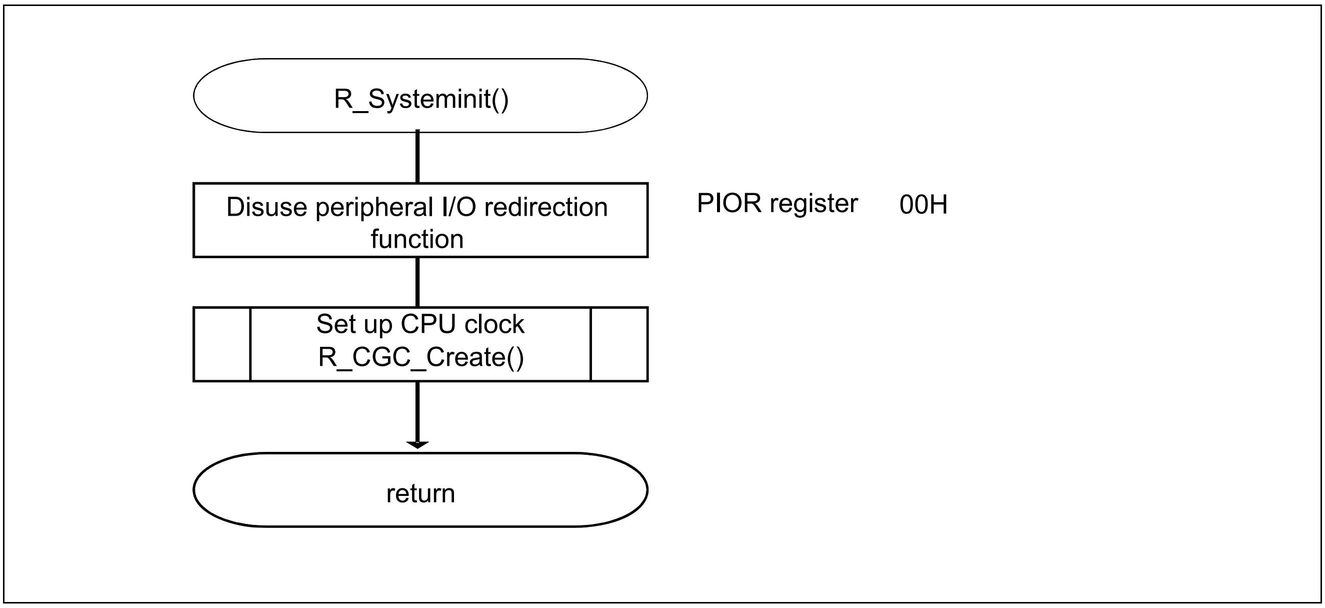


Figure 5.3 System Function

Note: Since the pins to be used differ depending on the IIC channel to be used, no settings are made for unused ports.

5.7.3 CPU Clock Setup

Figure 5.4 shows the flowchart for CPU clock setup.

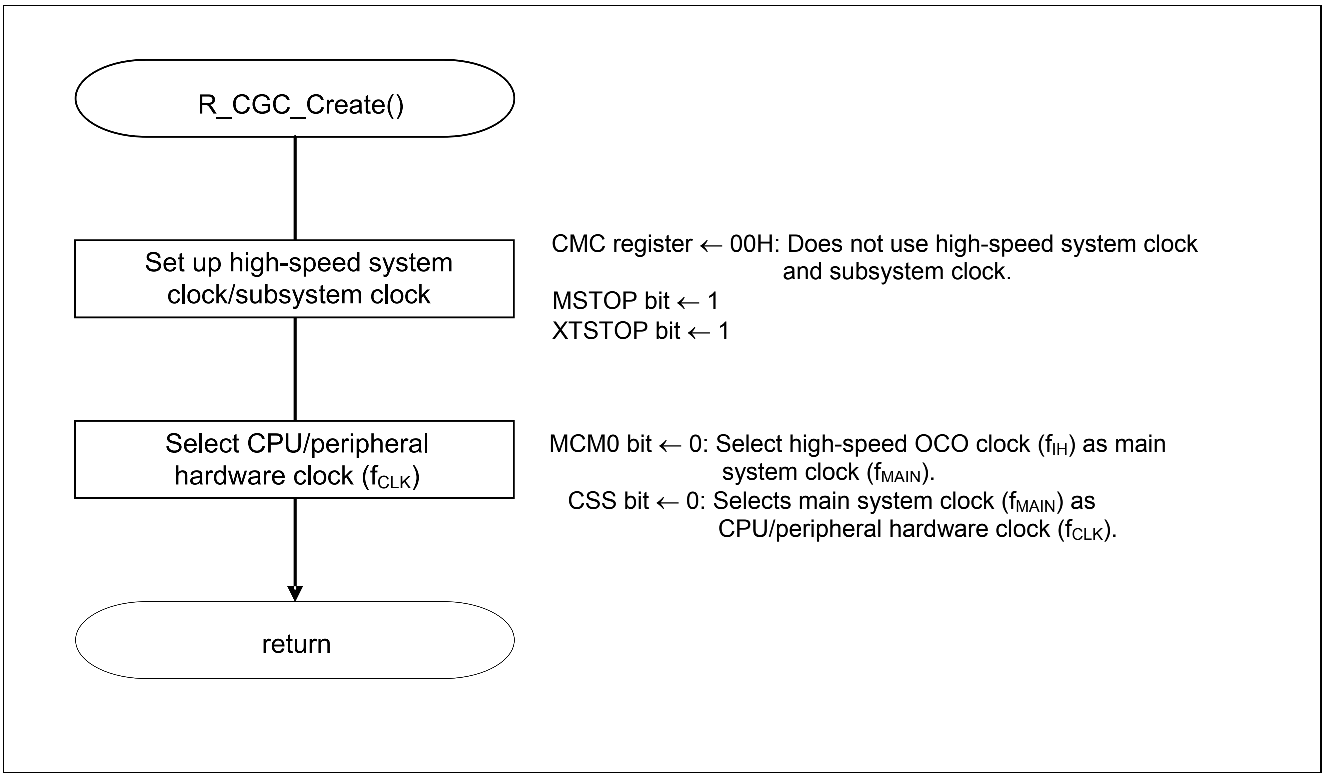

R_CGC_Create()
Set up high-speed system clock/subsystem clock
CMC register ← 00H: Does not use high-speed system clock and subsystem clock.
MSTOP bit ← 1
XTSTOP bit ← 1
Select CPU/peripheral hardware clock (f_{CLK})
MCM0 bit ← 0: Select high-speed OCO clock (f_{IH}) as main system clock (f_{MAIN}).
CSS bit ← 0: Selects main system clock (f_{MAIN}) as CPU/peripheral hardware clock (f_{CLK}).
return

Figure 5.4 CPU Clock Setup

5.7.4 Serial Array Unit Setup

Figures 5.5 and 5.6 show the flowcharts for setting up the serial interface.

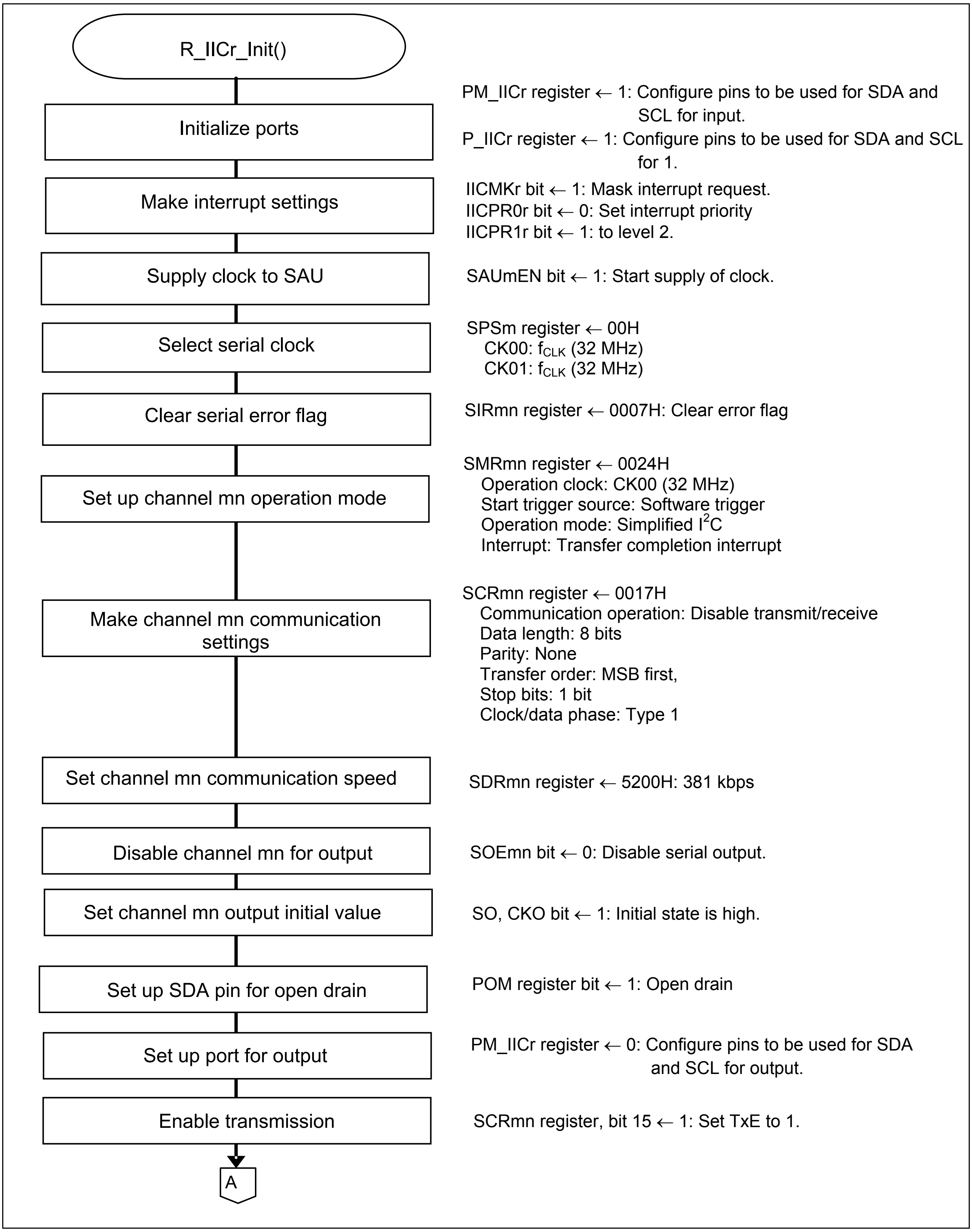

R_IICr_Init()
Initialize ports
PM_IICr register ← 1: Configure pins to be used for SDA and SCL for input.
P_IICr register ← 1: Configure pins to be used for SDA and SCL for 1.
Make interrupt settings
IICMKr bit ← 1: Mask interrupt request.
IICPR0r bit ← 0: Set interrupt priority
IICPR1r bit ← 1: to level 2.
Supply clock to SAU
SAUmEN bit ← 1: Start supply of clock.
Select serial clock
SPSm register ← 00H
CK00: f_{CLK} (32 MHz)
CK01: f_{CLK} (32 MHz)
Clear serial error flag
SIRmn register ← 0007H: Clear error flag
Set up channel mn operation mode
SMRmn register ← 0024H
Operation clock: CK00 (32 MHz)
Start trigger source: Software trigger
Operation mode: Simplified I^2C
Interrupt: Transfer completion interrupt
Make channel mn communication settings
SCRmn register ← 0017H
Communication operation: Disable transmit/receive
Data length: 8 bits
Parity: None
Transfer order: MSB first,
Stop bits: 1 bit
Clock/data phase: Type 1
Set channel mn communication speed
SDRmn register ← 5200H: 381 kbps
Disable channel mn for output
SOEmn bit ← 0: Disable serial output.
Set channel mn output initial value
SO, CKO bit ← 1: Initial state is high.
Set up SDA pin for open drain
POM register bit ← 1: Open drain
Set up port for output
PM_IICr register ← 0: Configure pins to be used for SDA and SCL for output.
Enable transmission
SCRmn register, bit 15 ← 1: Set TxE to 1.
A

Figure 5.5 SAU Setup (1/2)

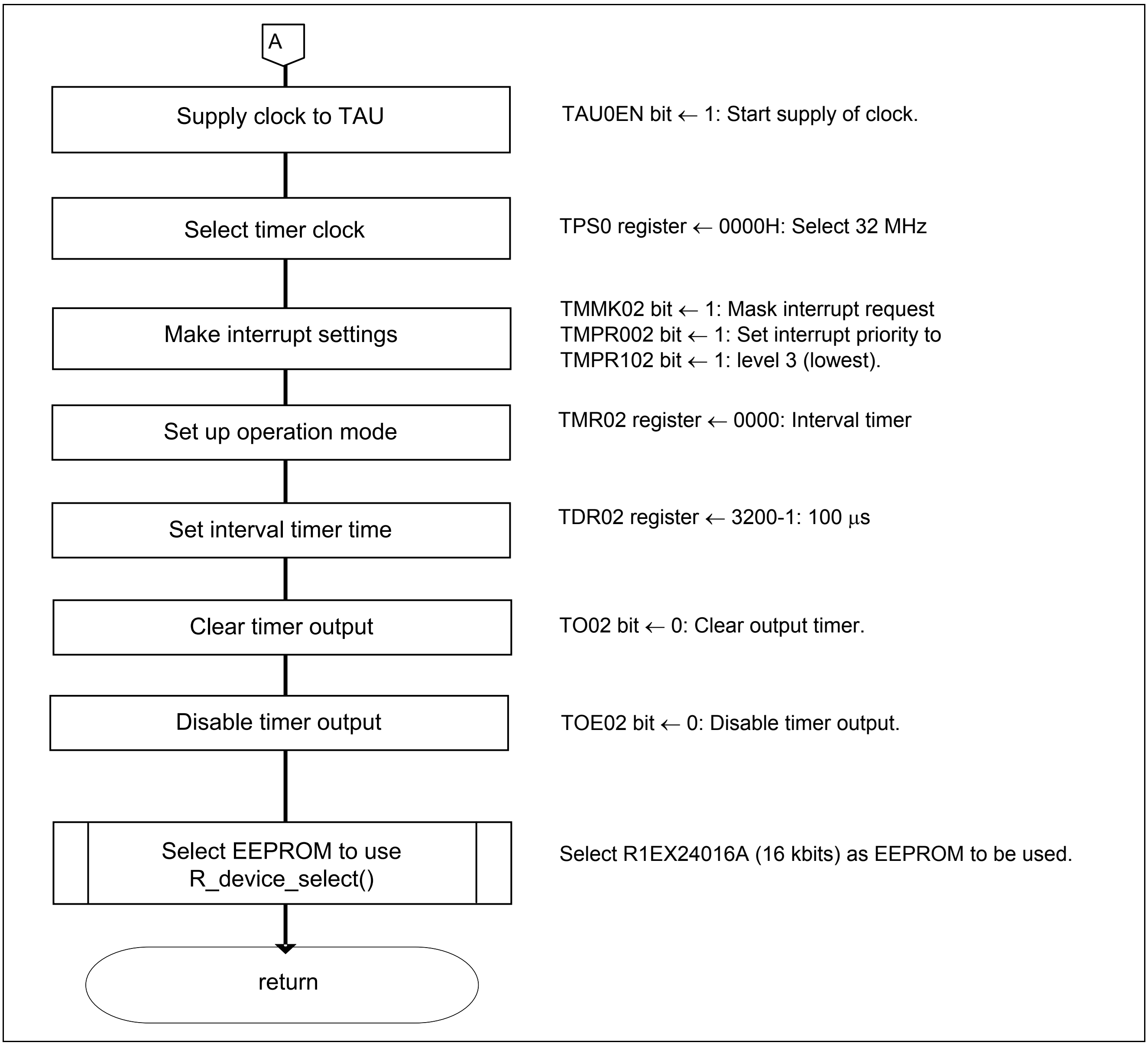

A
Supply clock to TAU
TAU0EN bit ← 1: Start supply of clock.
Select timer clock
TPS0 register ← 0000H: Select 32 MHz
Make interrupt settings
TMMK02 bit ← 1: Mask interrupt request
TMPR002 bit ← 1: Set interrupt priority to
TMPR102 bit ← 1: level 3 (lowest).
Set up operation mode
TMR02 register ← 0000: Interval timer
Set interval timer time
TDR02 register ← 3200-1: 100 µs
Clear timer output
TO02 bit ← 0: Clear output timer.
Disable timer output
TOE02 bit ← 0: Disable timer output.
Select EEPROM to use
R_device_select()
Select R1EX24016A (16 kbits) as EEPROM to be used.
return

Figure 5.6 SAU Setup (2/2)

Starting the supply of clock to the serial array unit SAUm

- Peripheral enable register 0 (PER0)
  Manipulate SAUmEN to start the supply of clock to the SAUm.

Symbol: PER0

7	6	5	4	3	2	1	0
RTCEN	0	ADCEN	IICA0EN	SAU1EN Note	SAU0EN	0	TAU0EN
x	0	x	x	**1** Note	**1**	0	x

Bits 3 Note and 2

SAUmEN	Control of serial array unit m input clock supply
0	Stops input clock supply.
1	**Enables input clock supply.**

Note: This is not provided in the 20, 24 and25-pin products.

Selecting the clock to the serial array unit SAUm

- Serial clock selection register m (SPSm)
  Operation clock: CK00 = 32 MHz, CK01 = 32 MHz

Symbol: SPSm

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	PRS 013	PRS 012	PRS 011	PRS 010	PRS 003	PRS 002	PRS 001	PRS 000
0	0	0	0	0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits 3 to 0

PRSm 03	PRSm 02	PRSm 01	PRSm 00	Selection of operation clock (CK00)					
					f_{CLK} = 2 MHz	f_{CLK} = 5 MHz	f_{CLK} = 10 MHz	f_{CLK} = 20 MHz	**f_{CLK} = 32 MHz**
0	**0**	**0**	**0**	**f_{CLK}**	2 MHz	5 MHz	10 MHz	20 MHz	**32 MHz**
0	0	0	1	$f_{CLK}/2$	1 MHz	2.5 MHz	5 MHz	10 MHz	16 MHz
0	0	1	0	$f_{CLK}/2^{2}$	500 kHz	1.25 MHz	2.5 MHz	5 MHz	8 MHz
0	0	1	1	$f_{CLK}/2^{3}$	250 kHz	625 kHz	1.25 MHz	2.5 MHz	4 MHz
0	1	0	0	$f_{CLK}/2^{4}$	125 kHz	313 kHz	625 kHz	1.25 MHz	2 MHz
0	1	0	1	$f_{CLK}/2^{5}$	62.5 kHz	156 kHz	313 kHz	625 kHz	1 MHz
0	1	1	0	$f_{CLK}/2^{6}$	31.3 kHz	78.1 kHz	156 kHz	313 kHz	500 kHz
0	1	1	1	$f_{CLK}/2^{7}$	15.6 kHz	39.1 kHz	78.1 kHz	156 kHz	250 kHz
1	0	0	0	$f_{CLK}/2^{8}$	7.81 kHz	19.5 kHz	39.1 kHz	78.1 kHz	125 kHz
1	0	0	1	$f_{CLK}/2^{9}$	3.91 kHz	9.77 kHz	19.5 kHz	39.1 kHz	62.5 kHz
1	0	1	0	$f_{CLK}/2^{10}$	1.95 kHz	4.88 kHz	9.77 kHz	19.5 kHz	31.3 kHz
1	0	1	1	$f_{CLK}/2^{11}$	977 Hz	2.44 kHz	4.88 kHz	9.77 kHz	15.6 kHz
1	1	0	0	$f_{CLK}/2^{12}$	488 Hz	1.22 kHz	2.44 kHz	4.88 kHz	7.81 kHz
1	1	0	1	$f_{CLK}/2^{13}$	244 Hz	610 Hz	1.22 kHz	2.44 kHz	3.91 kHz
1	1	1	0	$f_{CLK}/2^{14}$	122 Hz	305 Hz	610 Hz	1.22 kHz	1.95 kHz
1	1	1	1	$f_{CLK}/2^{15}$	61 Hz	153 Hz	305 Hz	610 Hz	977 Hz

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Clearing the error flags

- Serial flag clear trigger register mn (SIRmn)
  Clear channel mn error flags.

Symbol: SIRmn

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	FECT 01	PECT 01	OVCT 01
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**

Bit 1

PECTmn	Clear trigger of parity error flag of channel mn (no ACK response)
0	Not cleared.
1	**Clears the PEFmn bit of the SSRmn register to 0.**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting up channel mn operation mode

- Serial mode register mn (SMRmn)
  Select an operation clock: CK00
  Set the start trigger: Soft trigger only
  Set the operation mode: Simplified I^{2}C
  Select an interrupt factor: Transfer end interrupt

Symbol: SMRmn

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKS mn	CCS mn	0	0	0	0	0	STS mn Note	0	SIS mn0 Note	1	0	0	MD mn2	MD mn1	MD mn0
0	**0**	0	0	0	0	0	**0**	0	**0**	1	0	0	**1**	**0**	**0**

Bit 15

CKSmn	Selection of operation clock (f_{MCK}) of channel mn
0	**Operation clock CK00 set by the SPS0 register**
1	Operation clock CK01 set by the SPS0 register

Bit 14

CCSmn	Selection of transfer clock (f_{TCLK}) of channel mn
0	**Divided operation clock f_{MCK} specified by the CKSmn bit**
1	Clock input f_{SCK} from the SCK00 pin (slave transfer in CSI mode)

Bit 8

STSmn Note	Selection of start trigger source
0	**Only software trigger is valid (selected for CSI, UART transmission, and simplified I^{2}C).**
1	Valid edge of the RXD0 pin (selected for UART reception)

Bits 2 and 1

MDmn2	MDmn1	Setting of operation mode of channel mn
0	0	CSI mode
0	1	UART mode
1	**0**	**Simplified I^{2}C mode**
1	1	Setting prohibited

Bit 0

MDmn0	Selection of interrupt source of channel mn
0	**Transfer end interrupt**
1	Buffer empty interrupt (Occurs when data is transferred from the SDR00 register to the shift register.)

Note: Odd channels only

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting up the communication format

- Serial communication operation setting register mn (SCRmn)
  Operation mode: Transmission only
  Parity bit setting: No parity
  Data transfer order: MSB first
  Stop bit length: 1 bit
  Data length: 8 bits

Symbol: SCRmn

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXE mn	RXE mn	DAP mn	CKP mn	0	EOC mn	PTC mn1	PTC mn0	DIR mn	0	SLC mn1	SLC mn0	0	1	DLS mn1	DLS mn0
1	**0**	**0**	**0**	0	**0**	**0**	**0**	**0**	0	**0**	**1**	0	1	**1**	**1**

Bits 15 and 14

TXEmn	RXEmn	Setting of operation mode of channel mn
0	0	Disable communication.
0	1	Reception only
1	**0**	**Transmission only**
1	1	Transmission/reception

Bits 13 and 12

DAPmn	CKPmn	Selection of data and clock phase in CSI mode
0	**0**	**Type 1**
0	1	Type 2
1	0	Type 3
1	1	Type 4

Be sure to set DAP01 and CKP01 to 0 in the UART mode and simplified I^{2}C mode.

Bit 10

EOCmn	Selection of masking of error interrupt signal (INTSREx (x = 0 to 3))
0	**Masks error interrupt INTSREx (INTSRx is not masked).**
1	Enables generation of error interrupt INTSREx (INTSRx is masked if an error occurs).

Set EOC01 to 0 during UART reception.

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Bits 9 and 8

PTCmn1	PTCmn0	Setting of parity bit in UART mode	
		Transmission	Reception
0	**0**	**Does not output the parity bit.**	**Receives without parity**
0	1	Outputs 0 parity.	No parity judgment
1	0	Outputs even parity.	Judges as even parity.
1	1	Outputs odd parity.	Judges as odd parity.

Bit 7

DIRmn	Selection of data transfer sequence in CSI and UART modes
0	**Inputs/outputs data with MSB first.**
1	Inputs/outputs data with LSB first.

Bits 5 and 4

SLCmn1	SLCmn0	Setting of stop bit in UART mode
0	0	No stop bit
0	**1**	**Stop bit length = 1 bit**
1	0	Stop bit length = 2 bits (mn = 00, 02, 10, 12 only)
1	1	Setting prohibited

Set 1 bit (SLCmn1, SLCmn0 = 0, 1) during UART reception and in the simplified I^{2}C mode.

Bits 1 and 0

DLSmn1	DLSmn0	Setting of data length in CSI and UART modes
0	1	9-bit data length (stored in bits 0 to 8 of the SDRmn register) (can be set in UART mode only)
1	0	7-bit data length (stored in bits 0 to 6 of the SDRmn register)
1	**1**	**8-bit data length (stored in bits 0 to 7 of the SDRmn register)**
Other than above		Setting prohibited

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Selecting an operation clock (f_{MCK}) frequency division

- Serial data register mn (SDRmn)
  Transfer clock: f_{MCK}/84 (381 kbps)

Symbol: SDRmn

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Division ratio of operation clock (f_{MCK})							Transmit/receive buffer								
0	**1**	**0**	**1**	**0**	**0**	**1**	x	x	x	x	x	x	x	x	x

Bits 15 to 9

b15	b14	b13	b12	b11	b10	b9	Transfer clock setting by dividing the operating clock (f_{MCK})
0	0	0	0	0	0	0	f_{MCK}/2
0	0	0	0	0	0	1	f_{MCK}/4
0	0	0	0	0	1	0	f_{MCK}/6
0	0	0	0	0	1	1	f_{MCK}/8
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
0	**1**	**0**	**1**	**0**	**0**	**1**	**f_{MCK}/84**
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
1	1	1	1	1	1	0	f_{MCK}/254
1	1	1	1	1	1	1	f_{MCK}/256

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Enabling/disabling target channel for serial data output

- Serial output enable register m (SOEm)
  Stop output

Symbol: SOEm

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	SOE m3	SOE m2	SOE m1	SOE m0
0	0	0	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**	**0**

SOEmn	Serial output enable/stop of channel mn
0	**Stops output by serial communication operation.**
1	Enables output by serial communication operation.

Setting SDA and SCL initial output level

- Serial output register m (SOm)
  Output level: 1

Symbol: SOm

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Serial clock Output								Serial data output							
0	0	0	0	CKO m3	CKO m2	CKO m1	CKO m0	0	0	0	0	SO m3	SO m2	SO m1	SO m0
0	0	0	0	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	**1**	**1**

Bits 11 to 8

CKOmn	Serial clock output of channel mn
0	Serial clock output value is “0”.
1	**Serial clock output value is “0”.**

Bits 3 to 0

SOmn	Serial data output of channel mn
0	Serial data output value is “0”.
1	**Serial data output value is “0”.**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting up the IICr pins

- Port register (Px)
- Port mode register (PMx)
- Port output mode register (POMx)
  Set up the SDA pin for open drain output.
  The pins that are to be used as SCL and SDA in output mode are listed below.

Products with 20 pins

Simplified I²C channel	IIC00	IIC01	IIC10	IIC11	IIC20	IIC21
SCL pin	P10	—	—	P30	—	—
SDA pin	P11	—	—	P17	—	—

Products with 24 to 64 pins

Simplified I²C channel	IIC00	IIC01 Note 3	IIC10 Note 4	IIC11	IIC20 Note 1	IIC21 Note 2
SCL pin	P10	P75	P04	P30	P15	P70
SDA pin	P11	P74	P03	P50	P14	P71

Notes 1. Products with 30 or more pins
2. Products with 36 or more pins
3. Products with 48 or more pins
4. Products with 64 pins only

Products with 80 or more pins

Simplified I²C channel	IIC00	IIC01	IIC10	IIC11	IIC20	IIC21	IIC30	IIC31
SCL pin	P10	P43	P04	P30	P15	P70	P142	P54
SDA pin	P11	P44	P03	P50	P14	P71	P143	P53

Symbol: Pxn

Pxn	Output data control
0	Output 0
1	**Output 1**

Symbol: PMxn

PMxn	Pxn pin I/O mode selection
0	**Output mode (output buffer on)**
1	Input mode (output buffer off)

Symbol: POMxn

POMxn	Pxn pin output mode selection
0	CMOS output mode
1	**N-ch open drain output mode**

Note: Only the pins that can be used as SDA pins are set up.

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting up timer 02 (interval timer)

- See the following application note for details on the timer settings:
  RL78/G13 Timer Array Unit (Interval Timer) (R01AN0456E) Application Note

5.7.5 Main Function

Figures 5.7 through 5.9 show the flowchart for the main function.

The main function performs read/write tests on 16 kbits of EEPROM to illustrate the use of API.

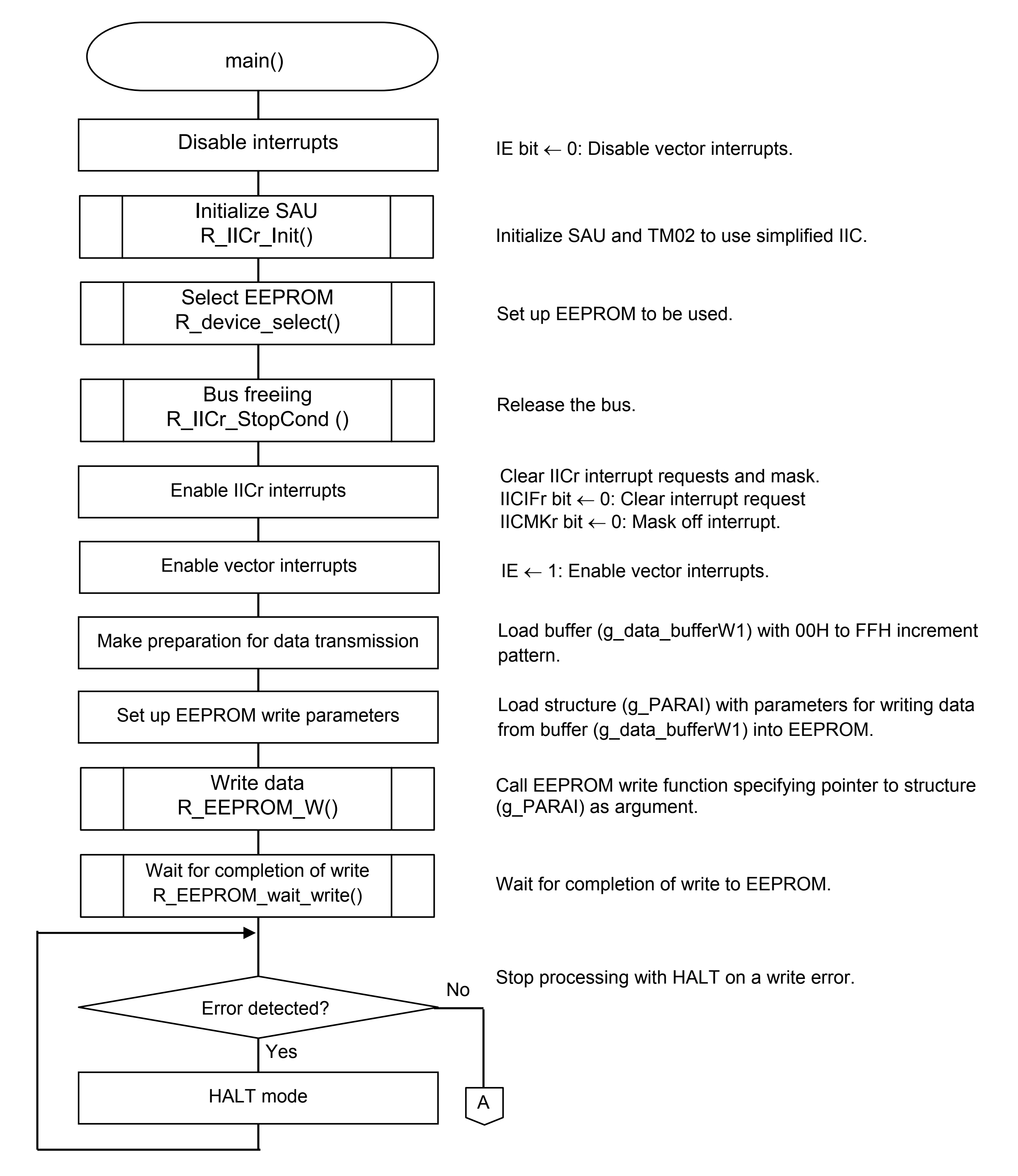
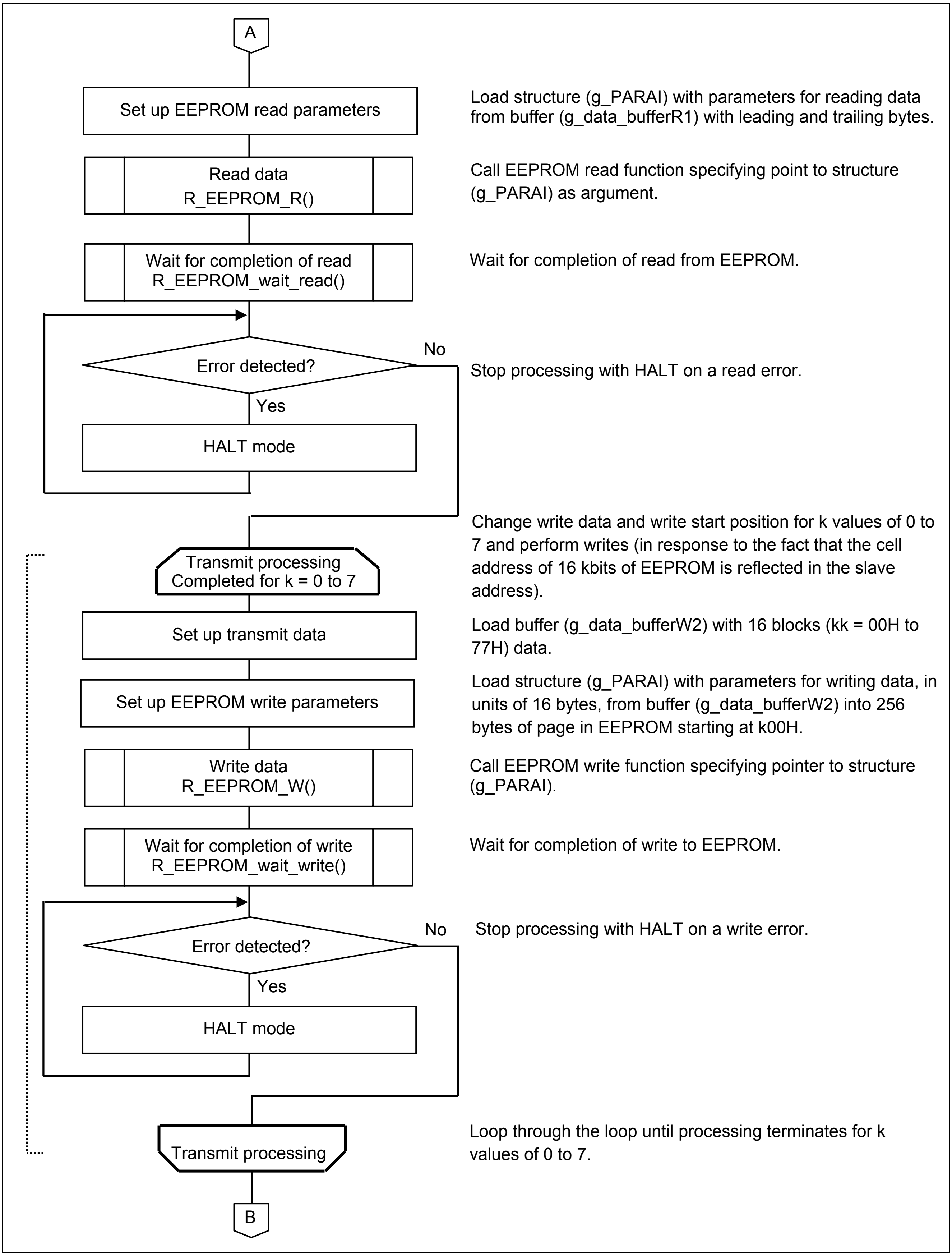

main()
Disable interrupts
IE bit ← 0: Disable vector interrupts.
Initialize SAU
R_IICr_Init()
Initialize SAU and TM02 to use simplified IIC.
Select EEPROM
R_device_select()
Set up EEPROM to be used.
Bus freeiing
R_IICr_StopCond ()
Release the bus.
Enable IICr interrupts
Clear IICr interrupt requests and mask.
IICIFr bit ← 0: Clear interrupt request
IICMKr bit ← 0: Mask off interrupt.
Enable vector interrupts
IE ← 1: Enable vector interrupts.
Make preparation for data transmission
Load buffer (g_data_bufferW1) with 00H to FFH increment pattern.
Set up EEPROM write parameters
Load structure (g_PARAI) with parameters for writing data from buffer (g_data_bufferW1) into EEPROM.
Write data
R_EEPROM_W()
Call EEPROM write function specifying pointer to structure (g_PARAI) as argument.
Wait for completion of write
R_EEPROM_wait_write()
Wait for completion of write to EEPROM.
Stop processing with HALT on a write error.
Error detected?
No
Yes
HALT mode
A

Figure 5.7 Main Function (1/3)



Load structure (g_PARAI) with parameters for reading data from buffer (g_data_bufferR1) with leading and trailing bytes.

Call EEPROM read function specifying point to structure (g_PARAI) as argument.

Wait for completion of read from EEPROM.

Stop processing with HALT on a read error.

Change write data and write start position for k values of 0 to 7 and perform writes (in response to the fact that the cell address of 16 kbits of EEPROM is reflected in the slave address).

Load buffer (g_data_bufferW2) with 16 blocks (kk = 00H to 77H) data.

Load structure (g_PARAI) with parameters for writing data, in units of 16 bytes, from buffer (g_data_bufferW2) into 256 bytes of page in EEPROM starting at k00H.

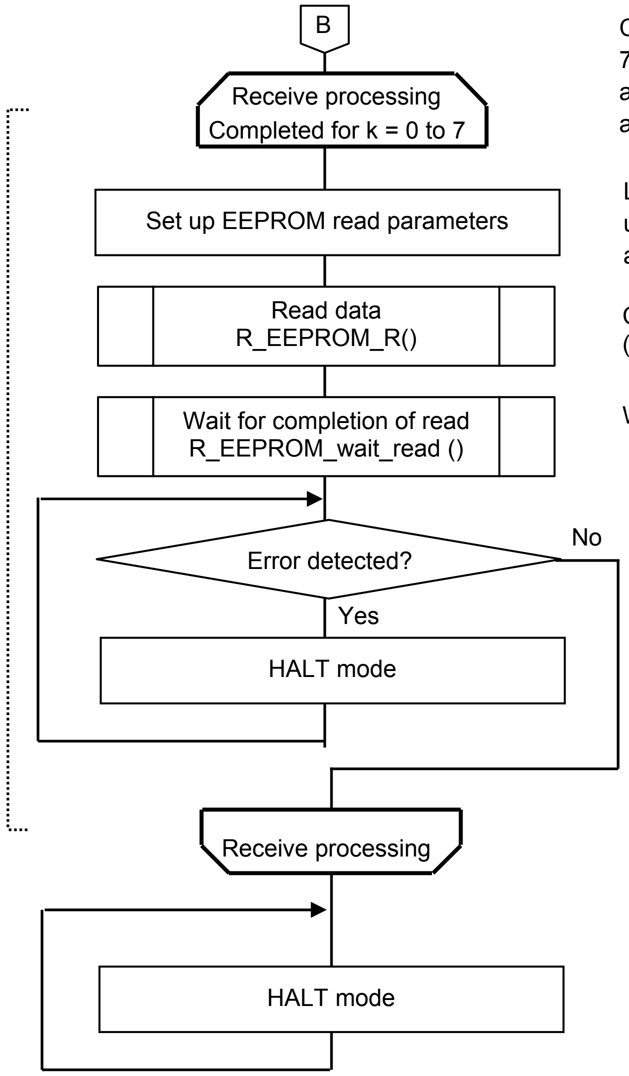
Call EEPROM write function specifying pointer to structure (g_PARAI).

Wait for completion of write to EEPROM.

Stop processing with HALT on a write error.

Loop through the loop until processing terminates for k values of 0 to 7.

Figure 5.8 Main Function (2/3)



Change read data and read start position for k values of 0 to 7 and perform reads (in response to the fact that the cell address of 16 kbits of EEPROM is reflected in the slave address).

Load structure (g_PARAI) with parameters for reading data, in units of 32 bytes, from 256 bytes of page in EEPROM starting at address (k00H)-8 (Note) into buffer (g_data_bufferR1).

Call EEPROM read function specifying pointer to structure (g_PARAI).

Wait for completion of read from EEPROM.

Stop processing with HALT on a read error.

Stop with HALT when processing is completed.

Note: The first subtraction (when K = 0) is not performed because subtracting 8 from 0 for the first read would cause an address out of range of the EEPROM.

Figure 5.9 Main Function (3/3)

5.7.6 EEPROM Selection

Figure 5.10 shows the flowchart for selecting the EEPROM.

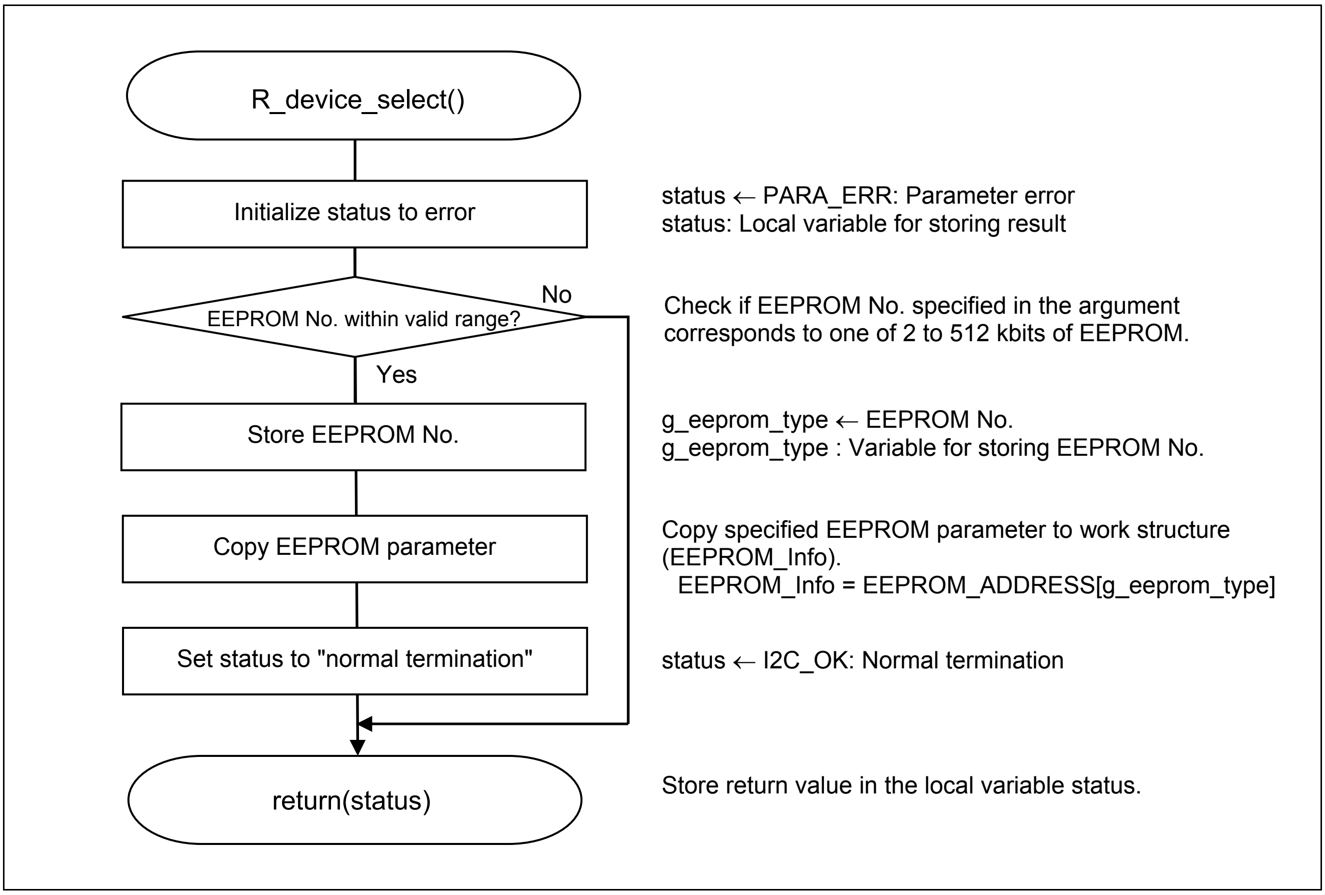

R_device_select()
Initialize status to error
status ← PARA_ERR: Parameter error
status: Local variable for storing result
EEPROM No. within valid range?
No
Yes
Check if EEPROM No. specified in the argument corresponds to one of 2 to 512 kbits of EEPROM.
Store EEPROM No.
g_eeprom_type ← EEPROM No.
g_eeprom_type : Variable for storing EEPROM No.
Copy EEPROM parameter
Copy specified EEPROM parameter to work structure (EEPROM_Info).
EEPROM_Info = EEPROM_ADDRESS[g_eeprom_type]
Set status to "normal termination"
status ← I2C_OK: Normal termination
return(status)
Store return value in the local variable status.

Figure5.10 EEPROM Selection

5.7.7 Bus freeing Processing

Figure 5.11 shows the flowchart for the function to free the bus.

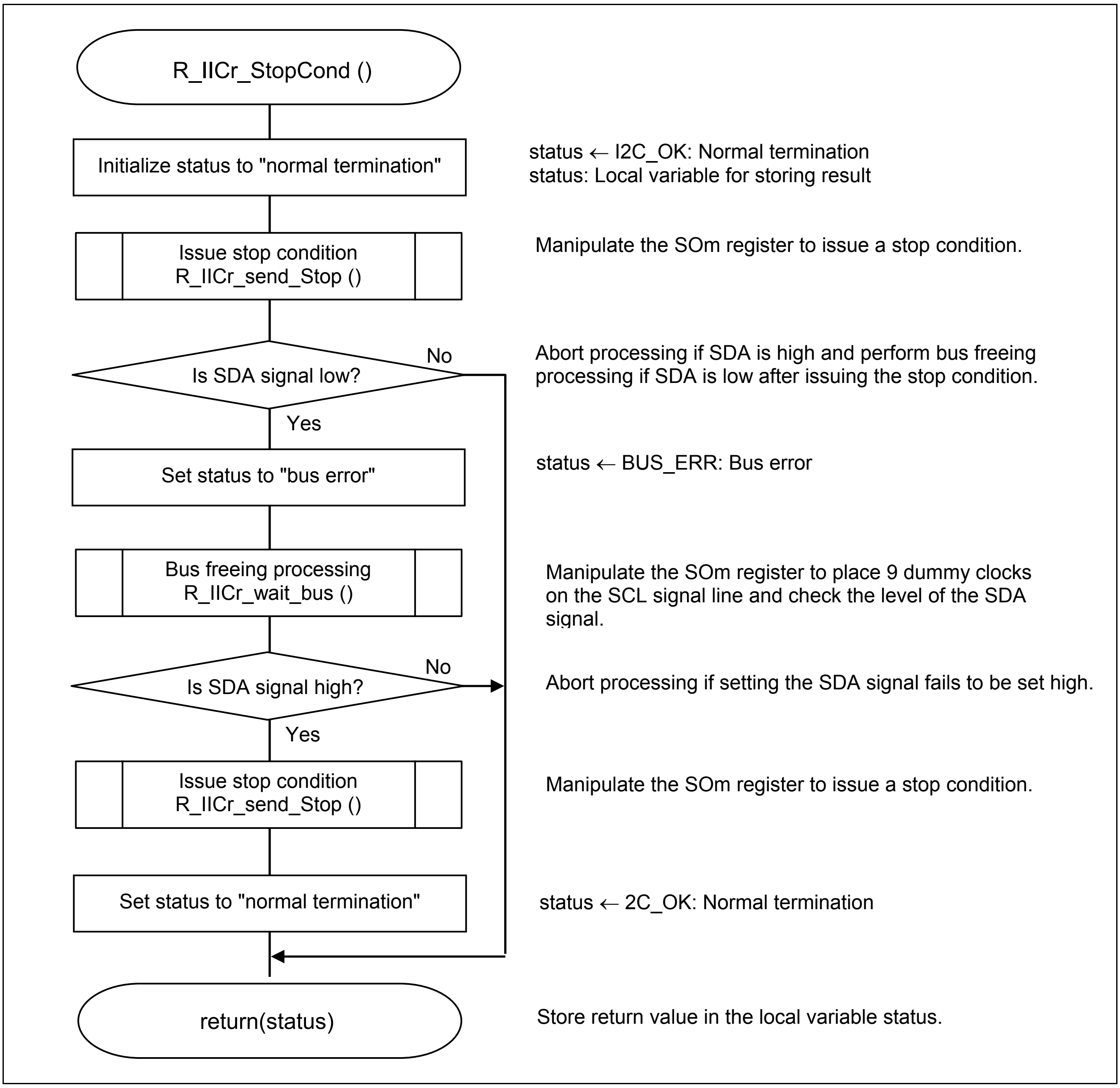

R_IICr_StopCond ()
Initialize status to "normal termination"
status ← I2C_OK: Normal termination
status: Local variable for storing result
Issue stop condition
R_IICr_send_Stop ()
Manipulate the SOm register to issue a stop condition.
Is SDA signal low?
No
Yes
Abort processing if SDA is high and perform bus freeing processing if SDA is low after issuing the stop condition.
Set status to "bus error"
status ← BUS_ERR: Bus error
Bus freeing processing
R_IICr_wait_bus ()
Manipulate the SOm register to place 9 dummy clocks on the SCL signal line and check the level of the SDA signal.
Is SDA signal high?
No
Yes
Abort processing if setting the SDA signal fails to be set high.
Issue stop condition
R_IICr_send_Stop ()
Manipulate the SOm register to issue a stop condition.
Set status to "normal termination"
status ← 2C_OK: Normal termination
return(status)
Store return value in the local variable status.

Figure 5.11 Bus freeing Processing

5.7.8 Stop Condition Generation Function

Figure 5.12 shows the flowchart for the function to generate a stop condition.

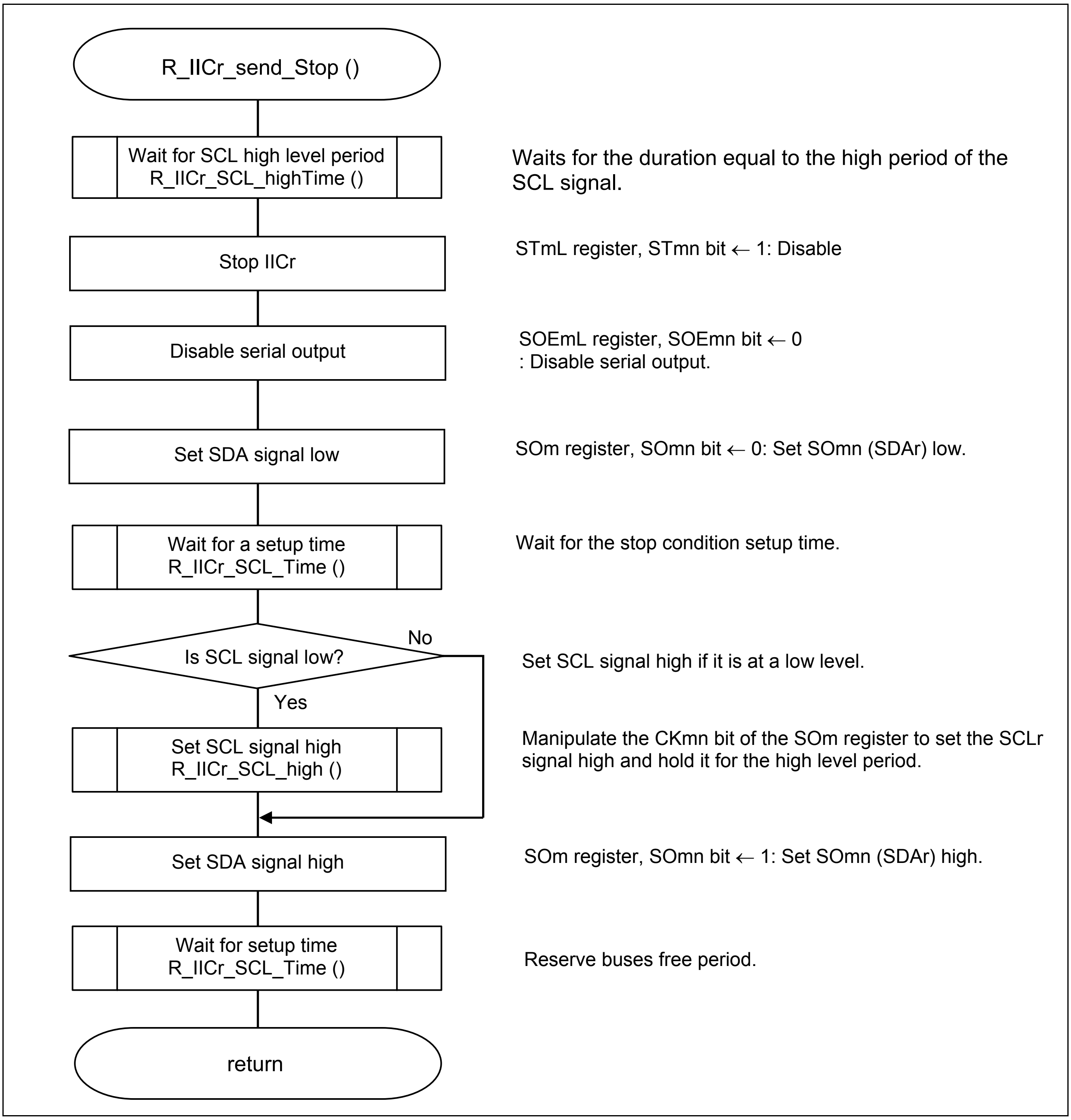

R_IICr_send_Stop ()
Wait for SCL high level period
R_IICr_SCL_highTime ()
Waits for the duration equal to the high period of the SCL signal.
Stop IICr
STmL register, STmn bit ← 1: Disable
Disable serial output
SOEmL register, SOEmn bit ← 0
: Disable serial output.
Set SDA signal low
SOm register, SOmn bit ← 0: Set SOmn (SDAr) low.
Wait for a setup time
R_IICr_SCL_Time ()
Wait for the stop condition setup time.
Is SCL signal low?
No
Yes
Set SCL signal high if it is at a low level.
Set SCL signal high
R_IICr_SCL_high ()
Manipulate the CKmn bit of the SOm register to set the SCLr signal high and hold it for the high level period.
Set SDA signal high
SOm register, SOmn bit ← 1: Set SOmn (SDAr) high.
Wait for setup time
R_IICr_SCL_Time ()
Reserve buses free period.
return

Figure 5.12 Stop Condition Generation Function

Stopping serial channel

- Serial channel stop register m (STm/STmL)
  Stops a given serial channel.

Symbol: STmL

7	6	5	4	3	2	1	0
0	0	0	0	STm3	STm2	STm1	STm0
0	0	0	0	**0/1**	**0/1**	**0/1**	**0/1**

Bits 3 to 0

STmn	Operation stop trigger of channel mn
0	**No trigger operation**
1	**Clears the SEmn bit to 0 and stops the communication operation.**

Disabling serial output

- Serial output enable register m (SOEm/SOEmL)
  Disables the serial channel for output.

Symbol: SOEmL

7	6	5	4	3	2	1	0
0	0	0	0	SOEm3	SOEm2	SOEm1	SOEm0
0	0	0	0	**0/1**	**0/1**	**0/1**	**0/1**

Bits 3 to 0

SOEmn	Serial output enable/stop of channel mn
0	**Stops output by serial communication operation.**
1	**Enables output by serial communication operation.**

Serial output manipulation

- Serial output register m (SOm)
  Manipulates serial output s (SCL and SDA).

Symbol: SOm

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CKOm3	CKOm2	CKOm1	CKOm0	0	0	0	0	SOm3	SOm2	SOm1	SOm0
0	0	0	0	**0/1**	**0/1**	**0/1**	**0/1**	0	0	0	0	**0/1**	**0/1**	**0/1**	**0/1**

Bits 11 to 8

CKOmn	Serial clock output of channel mn
0	**Serial clock output value is “0”.**
1	**Serial clock output value is “1”.**

Bits 3 to 0

SOmn	Serial data output of channel mn
0	**Serial data output value is “0”.**
1	**Serial data output value is “1”.**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

5.7.9 Bus Freeing Function

Figure 5.13 shows the flowchart for the bus freeing function.

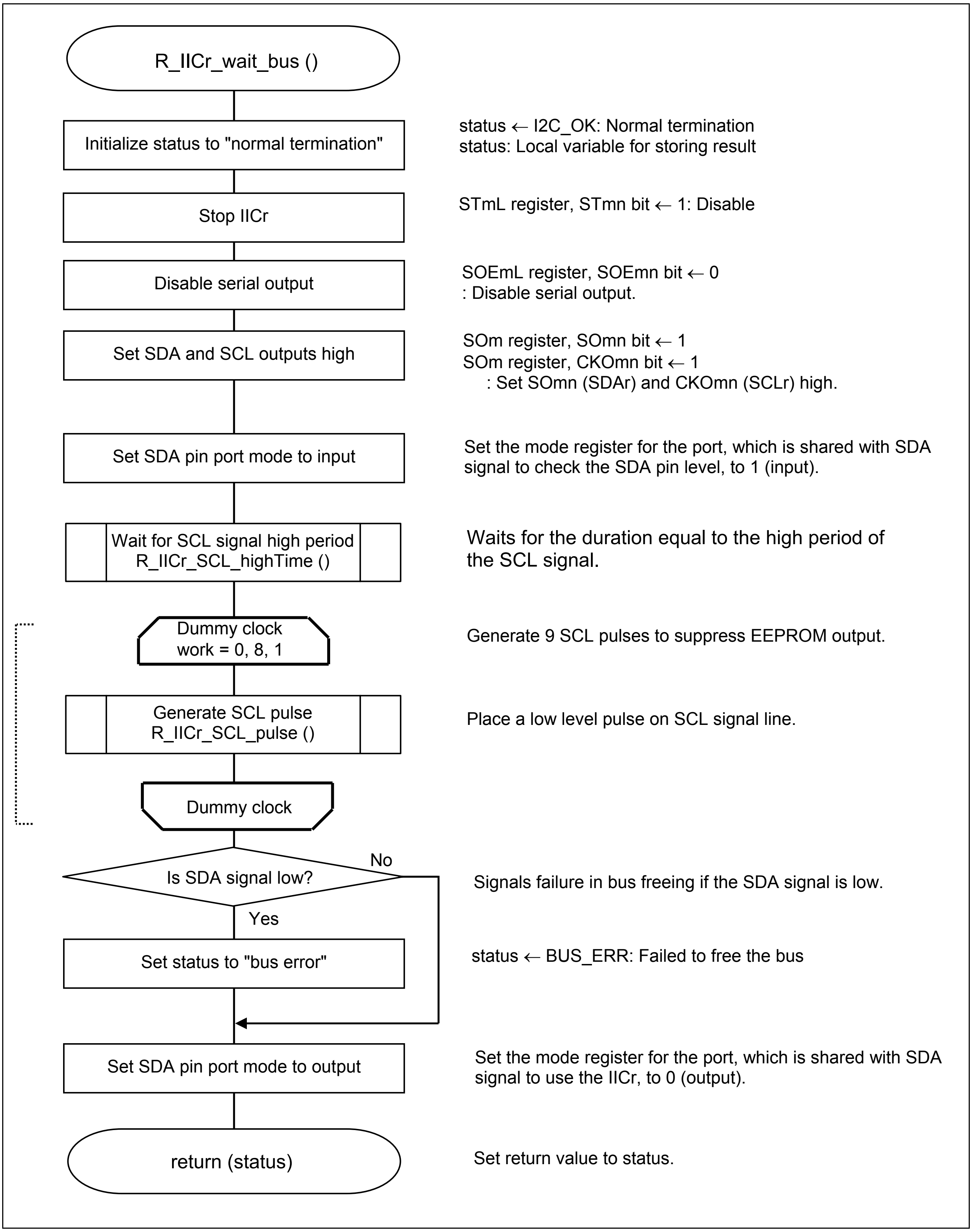
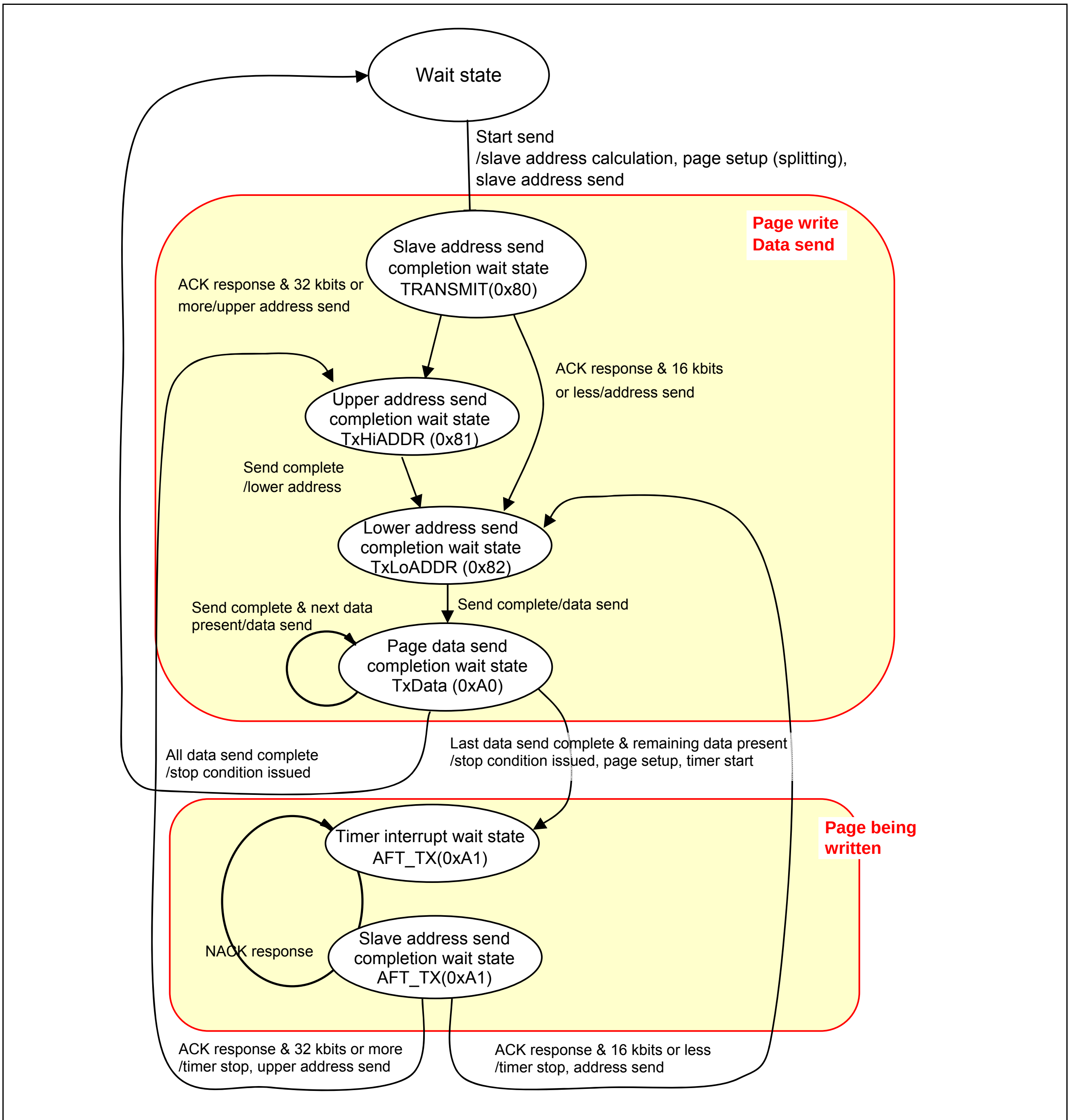

R_IICr_wait_bus ()
Initialize status to "normal termination"
status ← I2C_OK: Normal termination
status: Local variable for storing result
Stop IICr
STmL register, STmn bit ← 1: Disable
Disable serial output
SOEmL register, SOEmn bit ← 0
: Disable serial output.
Set SDA and SCL outputs high
SOm register, SOmn bit ← 1
SOm register, CKOmn bit ← 1
: Set SOmn (SDAr) and CKOmn (SCLr) high.
Set SDA pin port mode to input
Set the mode register for the port, which is shared with SDA signal to check the SDA pin level, to 1 (input).
Wait for SCL signal high period
R_IICr_SCL_highTime ()
Waits for the duration equal to the high period of the SCL signal.
Dummy clock
work = 0, 8, 1
Generate 9 SCL pulses to suppress EEPROM output.
Generate SCL pulse
R_IICr_SCL_pulse ()
Place a low level pulse on SCL signal line.
Dummy clock
Is SDA signal low?
No
Yes
Signals failure in bus freeing if the SDA signal is low.
Set status to "bus error"
status ← BUS_ERR: Failed to free the bus
Set SDA pin port mode to output
Set the mode register for the port, which is shared with SDA signal to use the IICr, to 0 (output).
return (status)
Set return value to status.

Figure 5.13 Bus Freeing Function

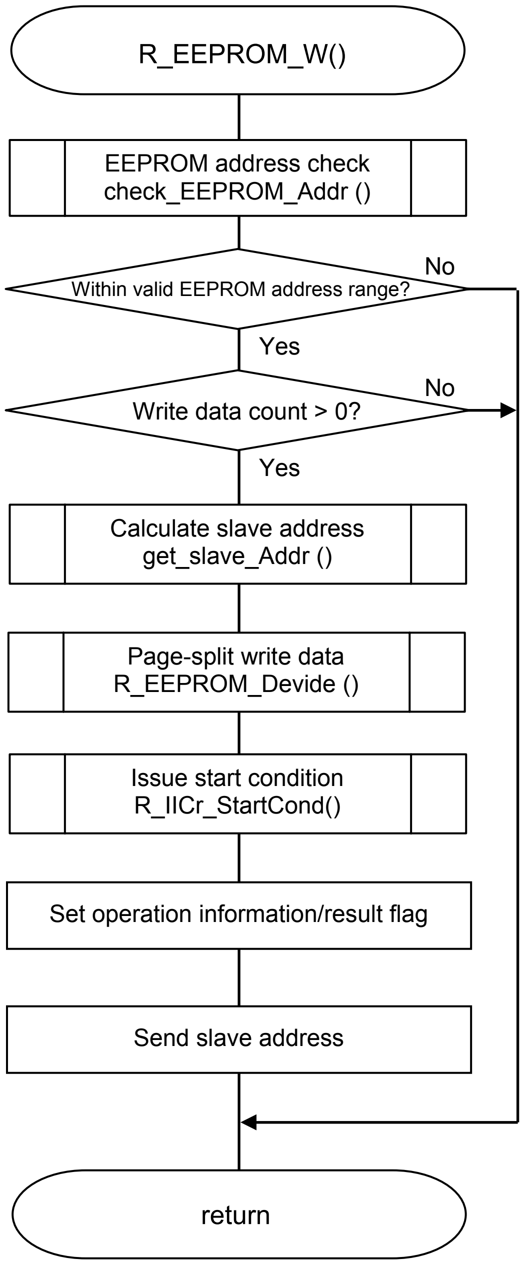
5.7.10 EEPROM Write Processing

Figure 5.14 shows the state transition diagram of EEPROM write processing and figure 5.15 shows its flow chart.



Note: Error processing is omitted. The 3rd line in a state contains the value of operation information (/result flag).

Figure 5.14 EEPROM Write Processing State Transition Diagram



Check if the area to be accessed falls within valid EEPROM address range and place the result in operation information/result flag (g_comstatus).

Abort processing if the area to be accessed falls does not fit within valid EEPROM address range or the write data count is 0.

Reflect the cell address of the EEPROM to be accessed in the slave address.
(for 4 to 16 kbits of EEPROM)

Set up the number of data bytes according to the page size of the EEPROM.

Issue a start condition to start the use of the IIC bus.

Change the value of the operation information/result flag (g_comstatus) to "slave address transmission" (TRANSMIT).

Transmit the slave address following a start condition.

Figure 5.15 EEPROM Write Processing

5.7.11 EEPROM Address Check Processing

Figure 5.16 shows the flowchart for the EEPROM address check processing.

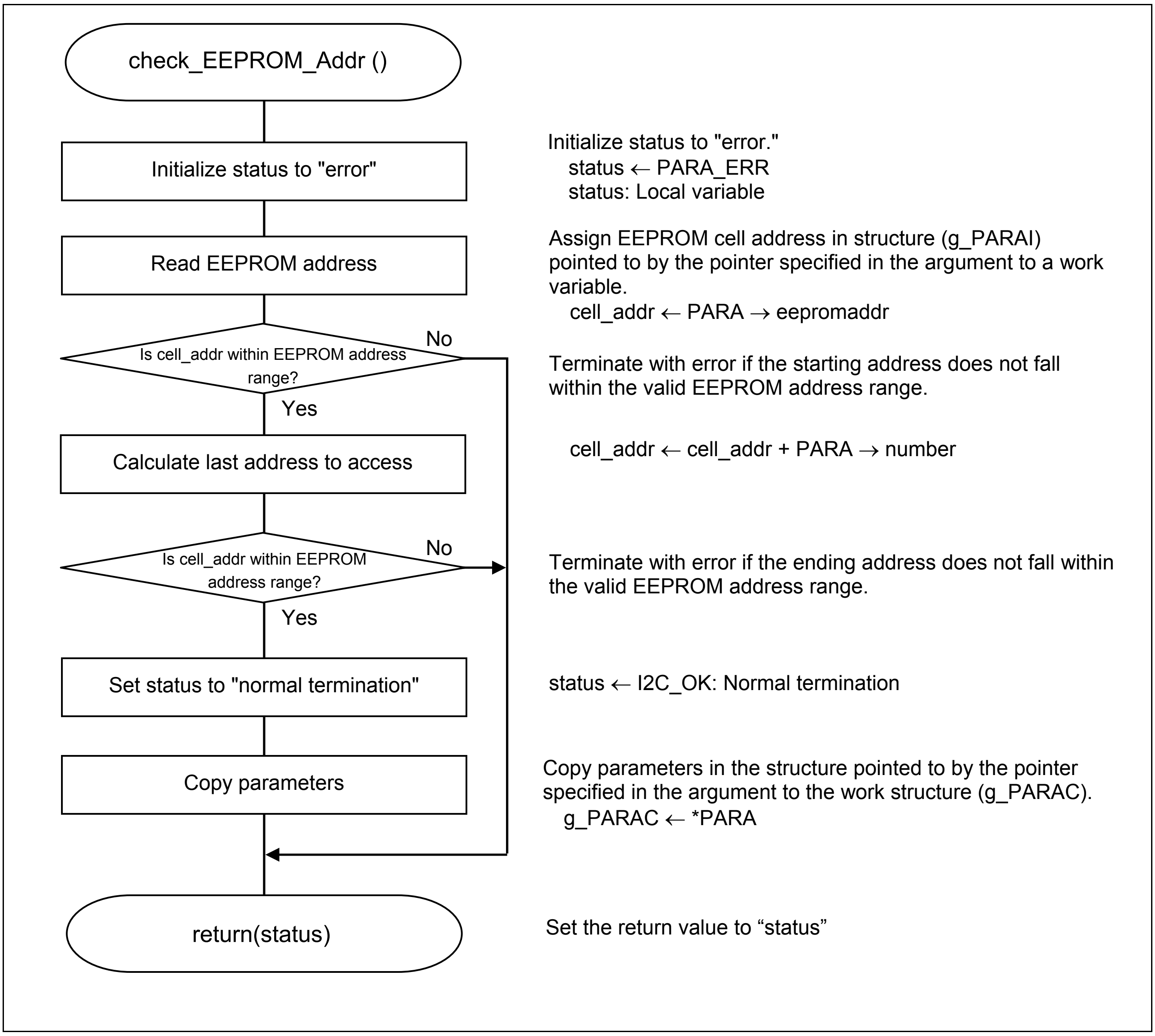
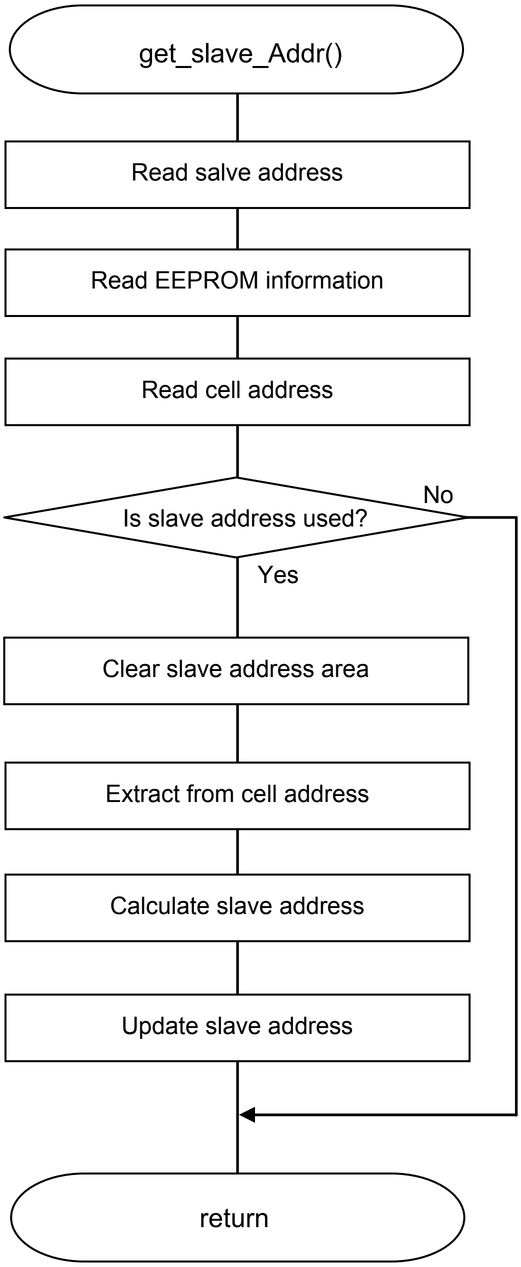

check_EEPROM_Addr ()
Initialize status to "error"
Read EEPROM address
Is cell_addr within EEPROM address range?
No
Yes
Calculate last address to access
Is cell_addr within EEPROM address range?
No
Yes
Set status to "normal termination"
Copy parameters
return(status)
Initialize status to "error."
status ← PARA_ERR
status: Local variable
Assign EEPROM cell address in structure (g_PARAI) pointed to by the pointer specified in the argument to a work variable.
cell_addr ← PARA → eepromaddr
Terminate with error if the starting address does not fall within the valid EEPROM address range.
cell_addr ← cell_addr + PARA → number
Terminate with error if the ending address does not fall within the valid EEPROM address range.
status ← I2C_OK: Normal termination
Copy parameters in the structure pointed to by the pointer specified in the argument to the work structure (g_PARAC).
g_PARAC ← *PARA
Set the return value to “status”

Figure 5.16 EEPROM Address Check Processing

5.7.12 Slave Address Calculation

Figure 5.17 shows the flowchart for the slave address calculation processing.



Assign the slave address to access to a work variable.
slave_addr ← g_PARAC.slaveaddr

Load work variable with the information indicating whether the EEPROM cell address is to be reflected in the slave address.
mask ← EEPROM_Info.addr_mask

Assign cell address to access to a work variable.
cell_addr ← g_PARAC. eepromaddr

Does nothing and terminate for a device for which the EEPROM cell address to be accessed is not to be reflected in the slave address.
(Process only 4 to 16 kbits of EEPROM.)

Zero-clear the slave address bits that are to be used for the cell address.

Extract A10 (16 kbits) to A8 (4 kbits) of the cell address.

Merges the extracted bits with the slave address.

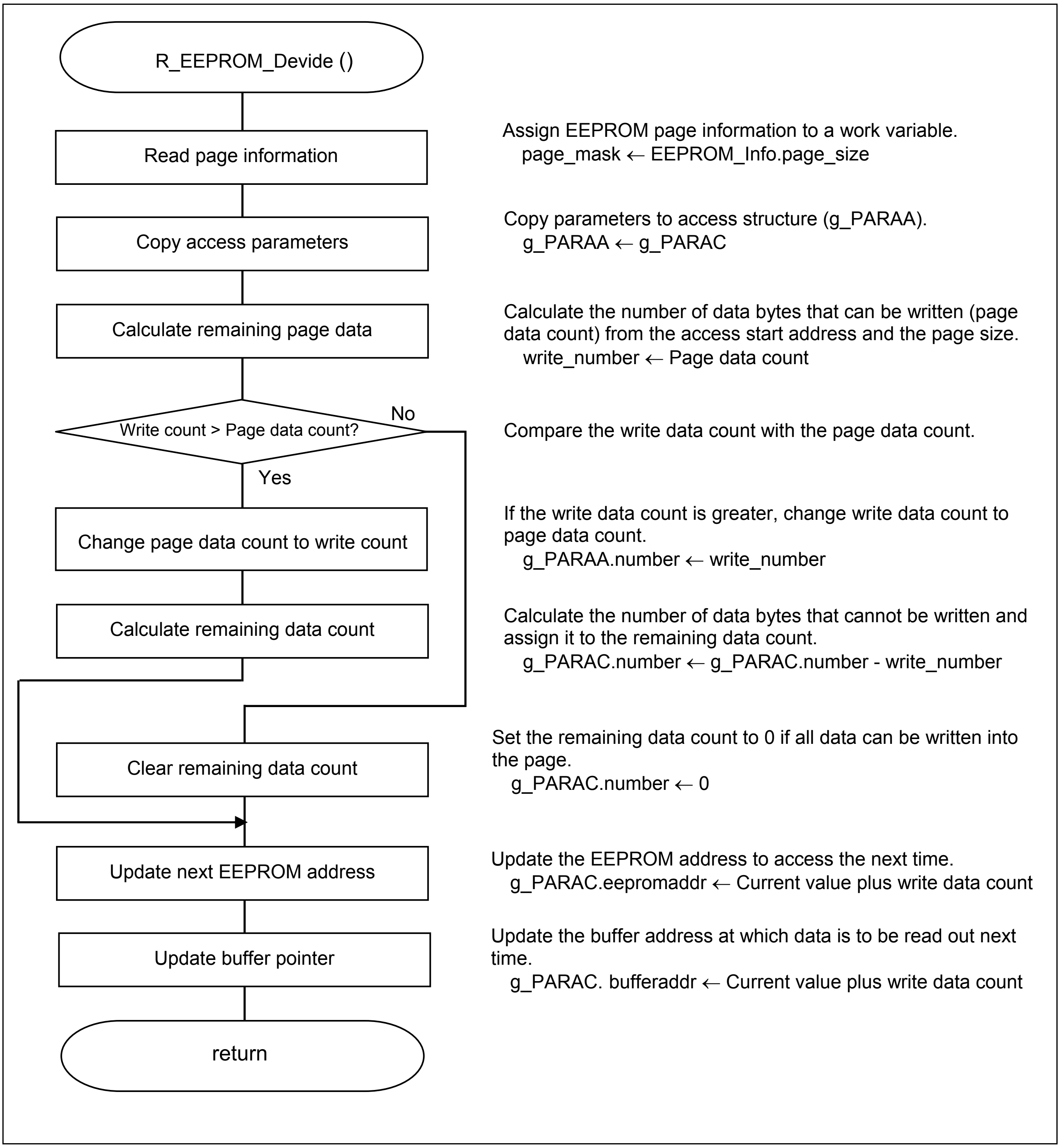
Rewrite the modified slave address into the structure (g_PARAC).

Remarks: This processing is required for 4, 8 and 16 kbits of EEPROM (R1EX24004A, R1EX24008A and R1EX24016A, respectively). It is omitted for 2, and 32 to 512 kbits of EEPROM (R1EX24002A, R1EX24032A to R1EX24512B, respectively).

Figure 5.17 Slave Address Calculation Processing

5.7.13 Write Data Page Split Processing

Figure 5.18 shows the flowchart for the write data page split processing.


R_EEPROM_Devide ()
Read page information
Copy access parameters
Calculate remaining page data
Write count > Page data count?
No
Yes
Change page data count to write count
Calculate remaining data count
Clear remaining data count
Update next EEPROM address
Update buffer pointer
return

Assign EEPROM page information to a work variable.
page_mask ← EEPROM_Info.page_size

Copy parameters to access structure (g_PARAA).
g_PARAA ← g_PARAC

Calculate the number of data bytes that can be written (page data count) from the access start address and the page size.
write_number ← Page data count

Compare the write data count with the page data count.

If the write data count is greater, change write data count to page data count.
g_PARAA.number ← write_number

Calculate the number of data bytes that cannot be written and assign it to the remaining data count.
g_PARAC.number ← g_PARAC.number - write_number

Set the remaining data count to 0 if all data can be written into the page.
g_PARAC.number ← 0

Update the EEPROM address to access the next time.
g_PARAC.eepromaddr ← Current value plus write data count

Update the buffer address at which data is to be read out next time.
g_PARAC. bufferaddr ← Current value plus write data count

Figure 5.18 Write Data Page Split Processing

5.7.14 Start Condition Issuing Processing

Figure 5.19 shows the flowchart for the function to issue a start condition.

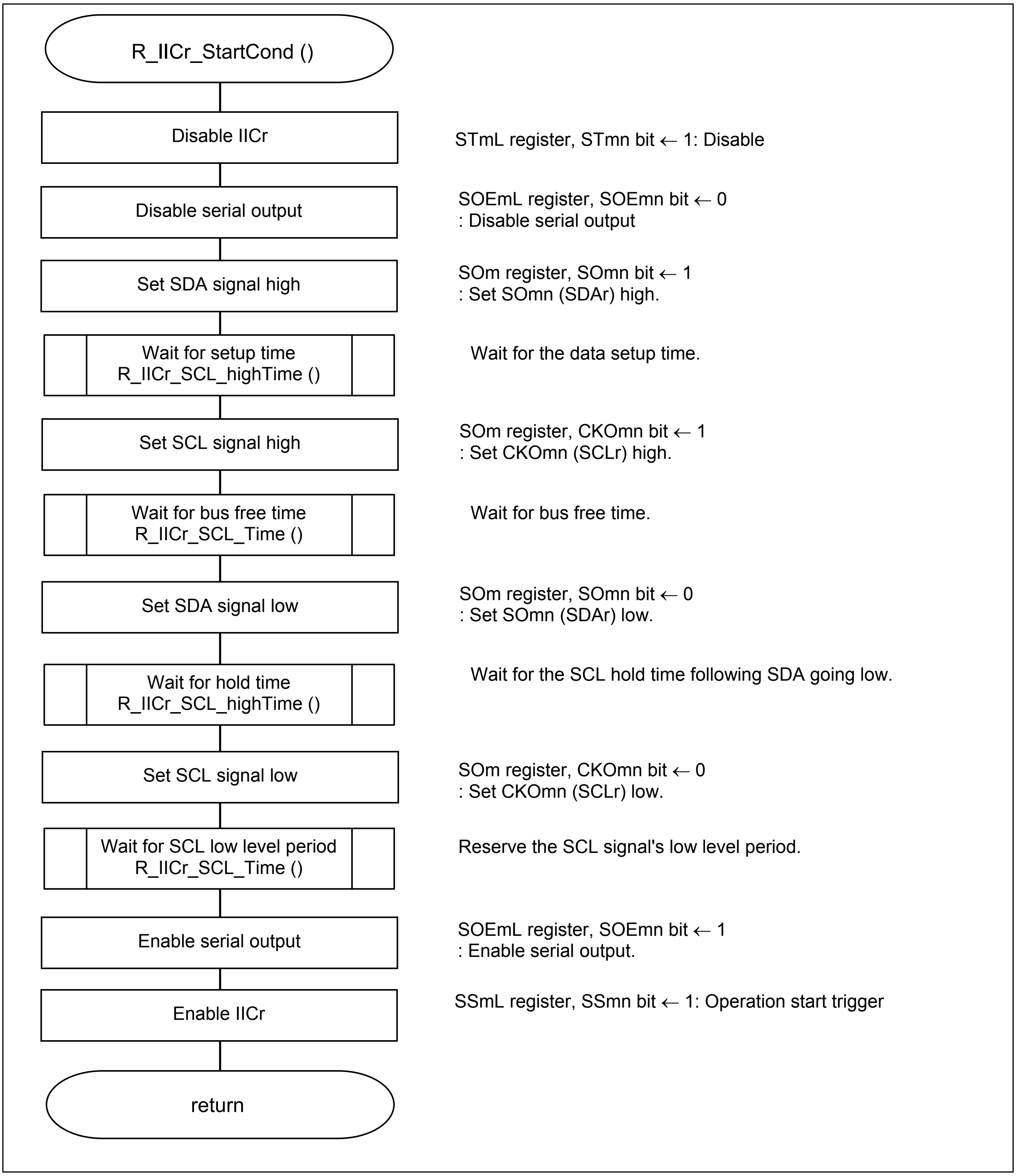


Figure 5.19 Start Condition Issuing Processing

Starting serial channel

- Serial channel start register m (SSm/SSmL)
  Starts a given serial channel.

Symbol: SSmL

7	6	5	4	3	2	1	0
0	0	0	0	SSm3	SSm2	SSm1	SSm0
0	0	0	0	**0/1**	**0/1**	**0/1**	**0/1**

Bits 3 to 0

SSmn	Operation start trigger of channel mn
0	**No trigger operation**
1	**Sets the SEmn bit to 1 and enters the communication wait status.**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

5.7.15 EEPROM Write Completion Wait Processing

Figure 5.20 shows the flowchart for the EEPROM write completion wait processing.

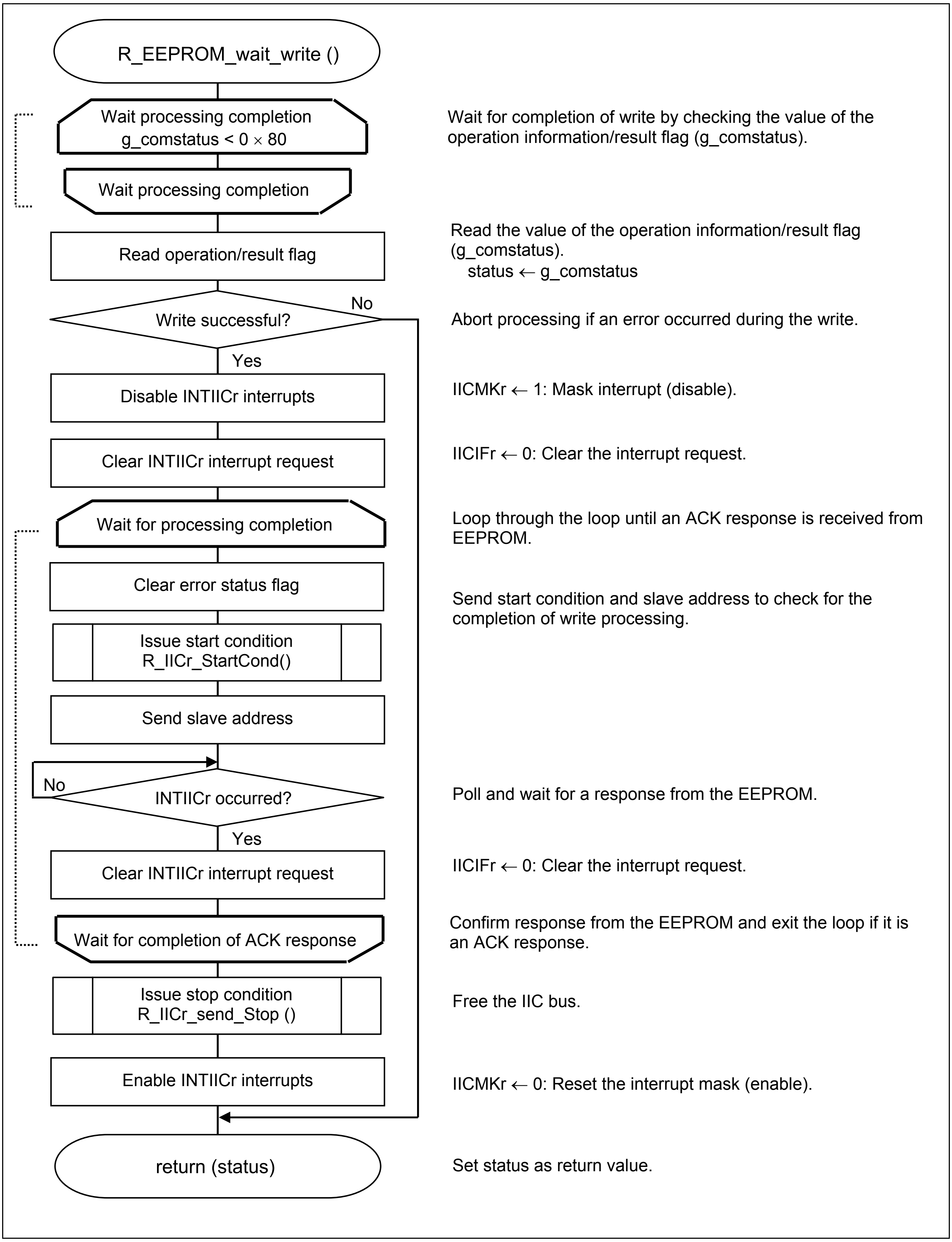
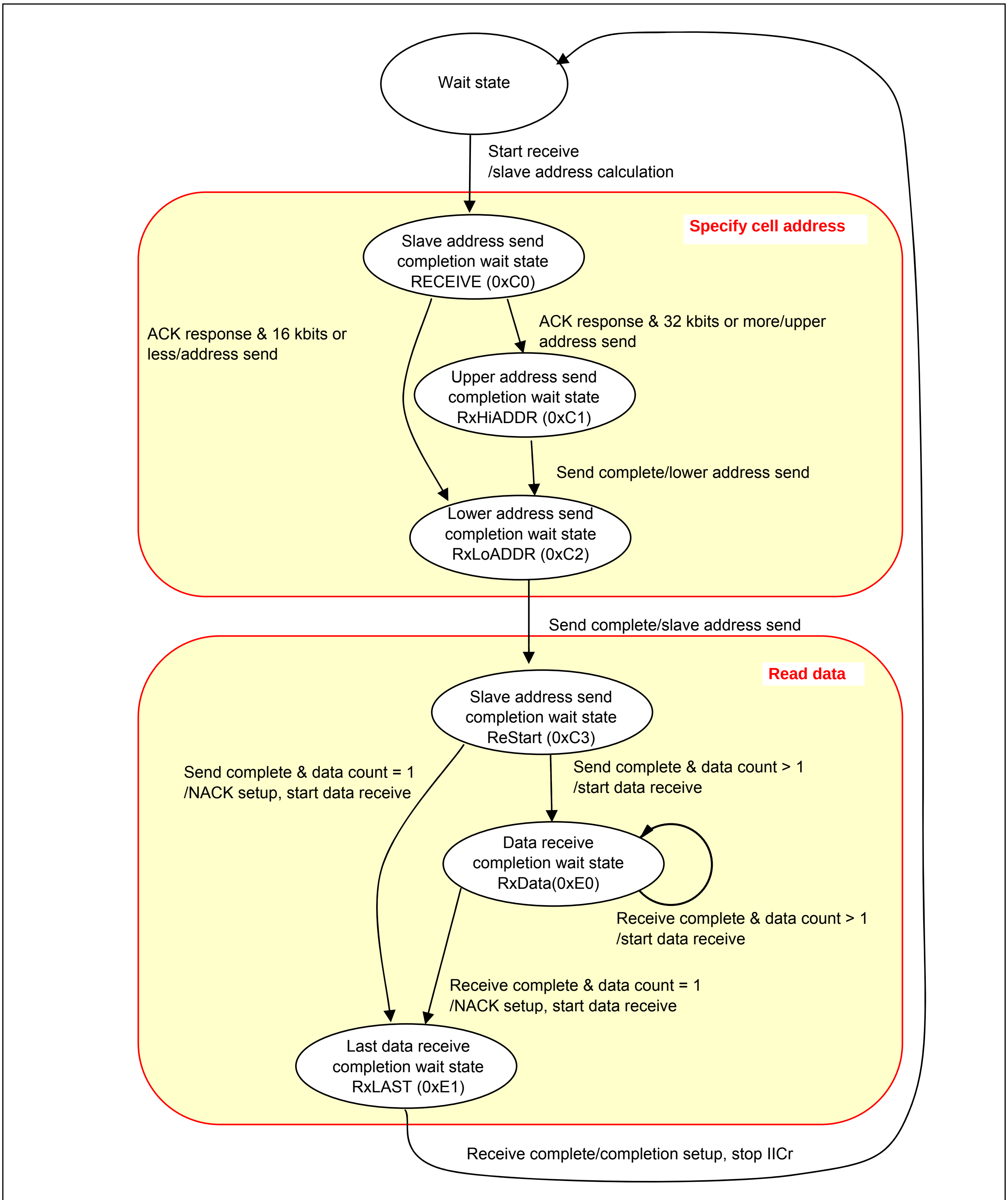

R_EEPROM_wait_write ()
Wait processing completion
g_comstatus < 0 × 80
Wait for completion of write by checking the value of the operation information/result flag (g_comstatus).
Wait processing completion
Read operation/result flag
Read the value of the operation information/result flag (g_comstatus).
status ← g_comstatus
Write successful?
No
Abort processing if an error occurred during the write.
Yes
Disable INTIICr interrupts
IICMKr ← 1: Mask interrupt (disable).
Clear INTIICr interrupt request
IICIFr ← 0: Clear the interrupt request.
Wait for processing completion
Loop through the loop until an ACK response is received from EEPROM.
Clear error status flag
Send start condition and slave address to check for the completion of write processing.
Issue start condition
R_IICr_StartCond()
Send slave address
No
INTIICr occurred?
Poll and wait for a response from the EEPROM.
Yes
Clear INTIICr interrupt request
IICIFr ← 0: Clear the interrupt request.
Wait for completion of ACK response
Confirm response from the EEPROM and exit the loop if it is an ACK response.
Issue stop condition
R_IICr_send_Stop ()
Free the IIC bus.
Enable INTIICr interrupts
IICMKr ← 0: Reset the interrupt mask (enable).
return (status)
Set status as return value.

Figure 5.20 EEPROM Write Completion Wait Processing

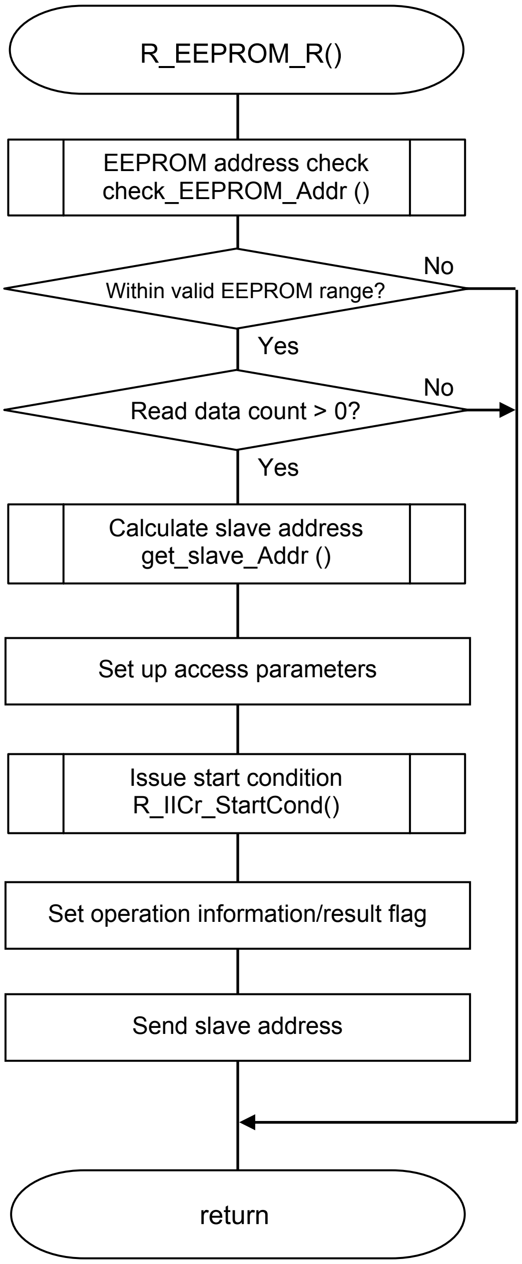
5.7.16 EEPROM Read Processing

Figure 5.21 shows the state transition diagram of EEPROM read processing and figure 5.22 shows its flow chart.



Note: Error processing is omitted. The 3rd line in a state contains the value of operation information (/result flag).

Figure 5.21 EEPROM Read Processing State Transition Diagram



Check if the area to be accessed falls within valid EEPROM address range and place the result in operation information/result flag (g_comstatus).

Abort processing if the area to be accessed falls does not fit within valid EEPROM address range or the read data count is 0.

Reflect the cell address of the EEPROM to be accessed in the slave address.
(for 4 to 16 kbits of EEPROM)

Copy EEPROM control parameters from the structure g_PARAC to the access structure g_PARAA.

Issue a start condition to start the use of the IIC bus.

Change the value of the operation information/result flag (g_comstatus) to "slave address receive" (RECEIVE).

Transmit the slave address following a start condition.

Figure 5.22 EEPROM Read Processing

5.7.17 EEPROM Read Completion Wait Processing

Figure 5.23 shows the flowchart for the EEPROM read completion wait processing.

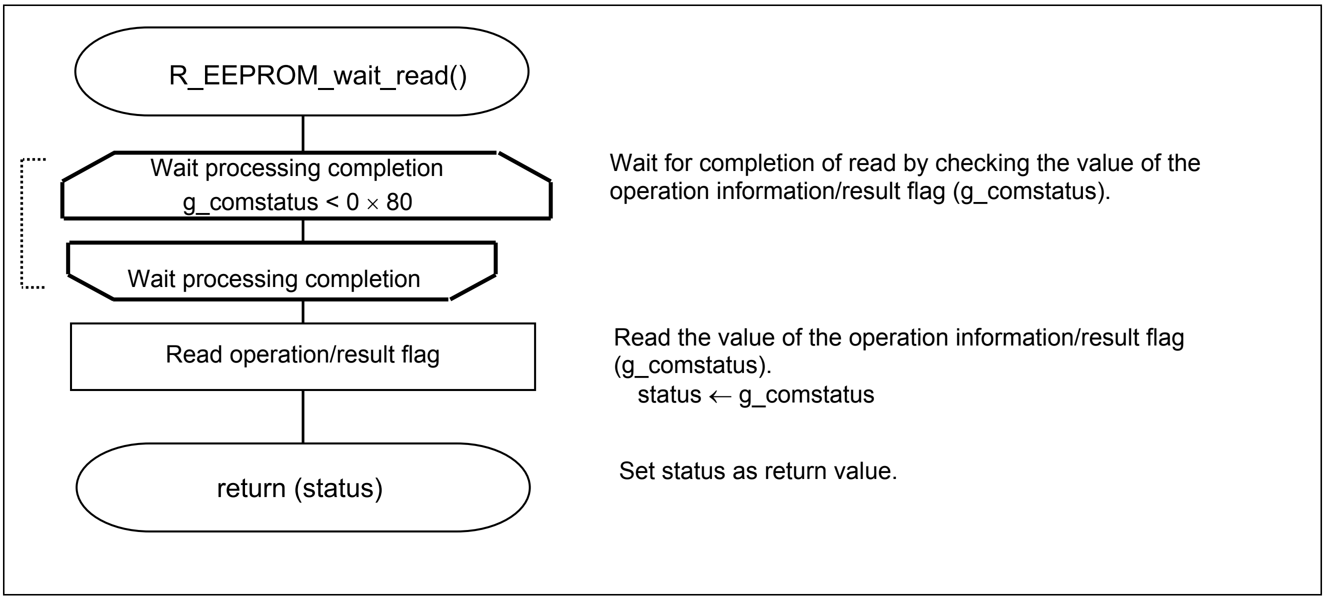

R_EEPROM_wait_read()
Wait processing completion
g_comstatus < 0 × 80
Wait processing completion
Read operation/result flag
return (status)
Wait for completion of read by checking the value of the operation information/result flag (g_comstatus).
Read the value of the operation information/result flag (g_comstatus).
status ← g_comstatus
Set status as return value.

Figure 5.23 EEPROM Read Completion Wait Processing

5.7.18 Slave Address Transmission Completion Processing

Figure 5.24 shows the flowchart for the slave address transmission completion processing.

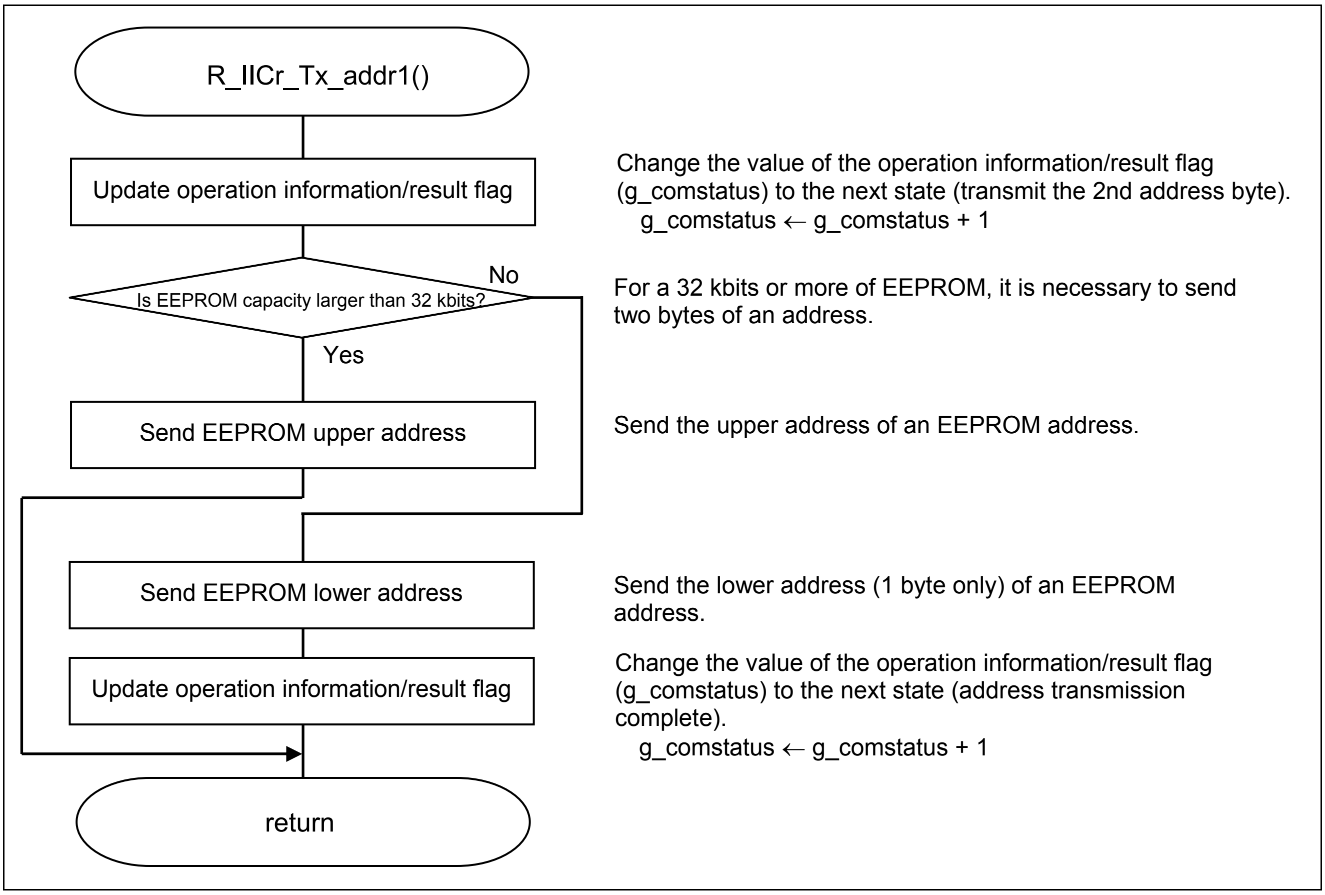

R_IICr_Tx_addr1()
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (transmit the 2nd address byte).
g_comstatus ← g_comstatus + 1
Is EEPROM capacity larger than 32 kbits?
No
Yes
For a 32 kbits or more of EEPROM, it is necessary to send two bytes of an address.
Send EEPROM upper address
Send the upper address of an EEPROM address.
Send EEPROM lower address
Send the lower address (1 byte only) of an EEPROM address.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (address transmission complete).
g_comstatus ← g_comstatus + 1
return

Figure 5.24 Slave Address Transmission Completion Processing

5.7.19 Upper Address Transmission Completion Processing

Figure 5.25 shows the flowchart for the upper address transmission completion processing.

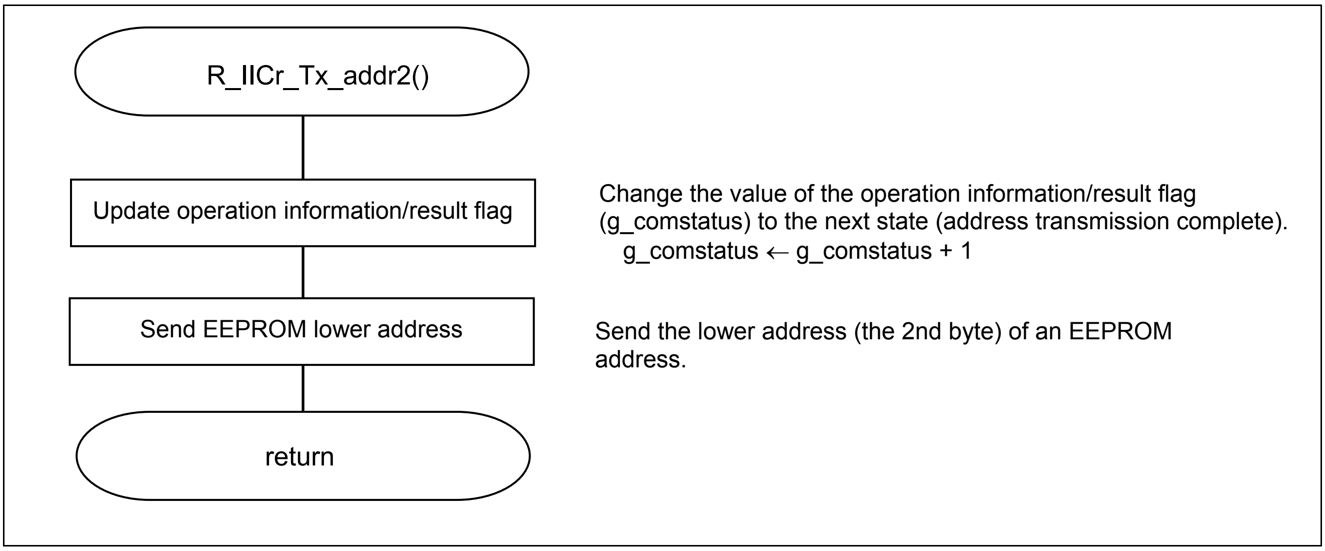

R_IICr_Tx_addr2()
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (address transmission complete).
g_comstatus ← g_comstatus + 1
Send EEPROM lower address
Send the lower address (the 2nd byte) of an EEPROM address.
return

Figure 5.25 Upper Address Transmission Completion Processing

5.7.20 Restart Processing

Figure 5.26 shows the flowchart for the restart processing.

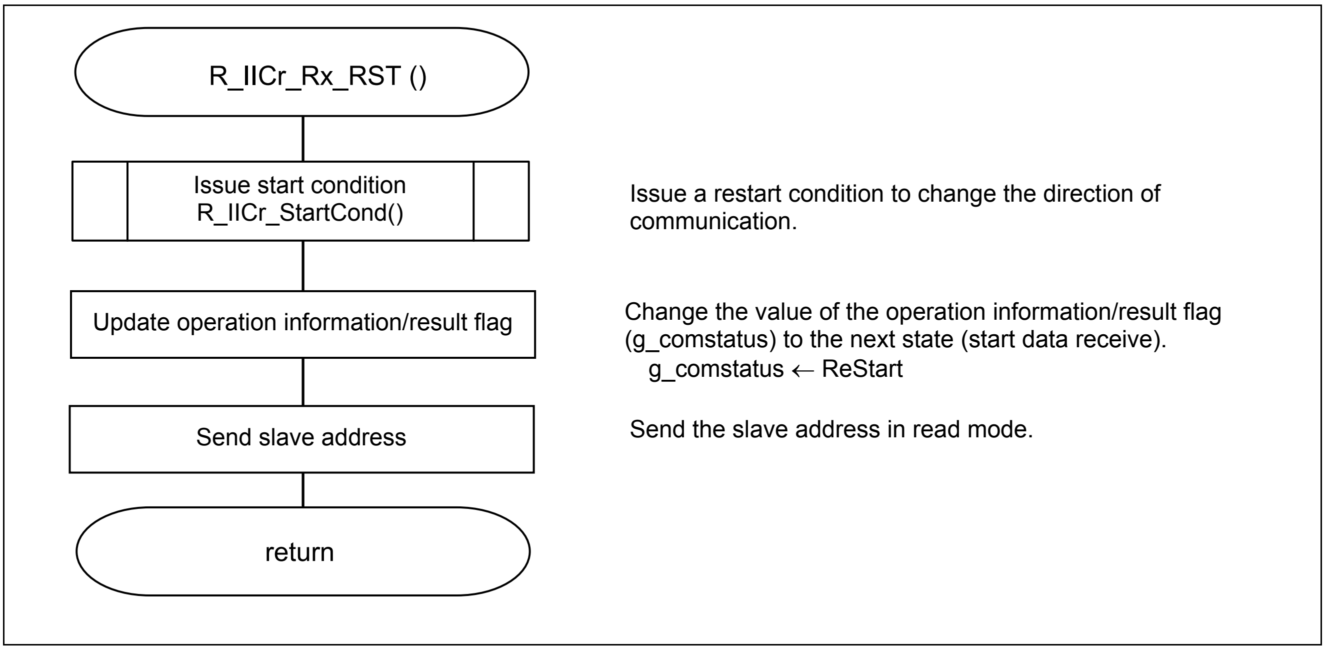

R_IICr_Rx_RST ()
Issue start condition
R_IICr_StartCond()
Issue a restart condition to change the direction of communication.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (start data receive).
g_comstatus ← ReStart
Send slave address
Send the slave address in read mode.
return

Figure 5.26 Restart Processing

5.7.21 Data Reception Start Processing

Figure 5.27 shows the flowchart for the data reception start processing.

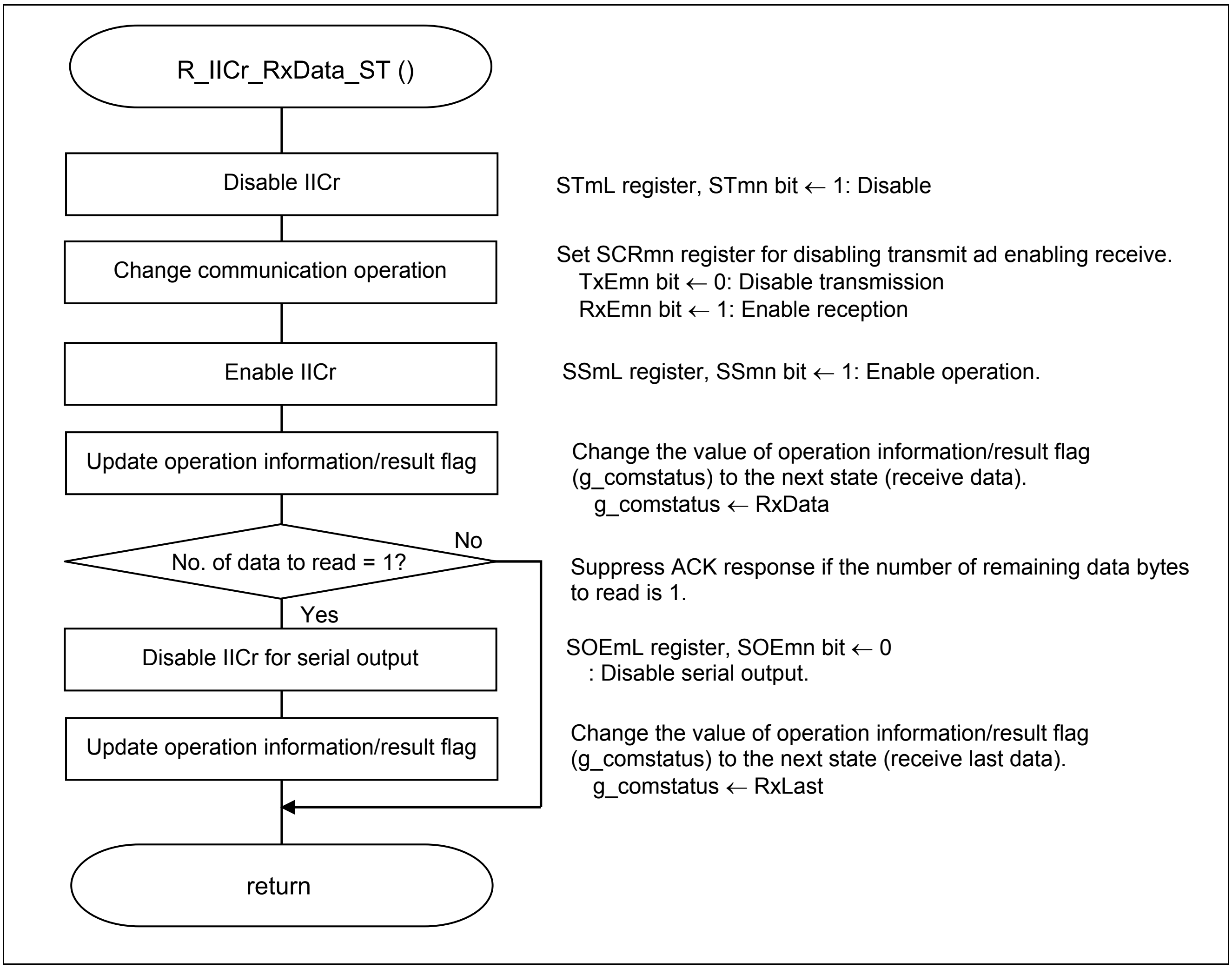

R_IICr_RxData_ST ()
Disable IICr
STmL register, STmn bit ← 1: Disable
Change communication operation
Set SCRmn register for disabling transmit ad enabling receive.
TxEmn bit ← 0: Disable transmission
RxEmn bit ← 1: Enable reception
Enable IICr
SSmL register, SSmn bit ← 1: Enable operation.
Update operation information/result flag
Change the value of operation information/result flag (g_comstatus) to the next state (receive data).
g_comstatus ← RxData
No. of data to read = 1?
No
Yes
Suppress ACK response if the number of remaining data bytes to read is 1.
Disable IICr for serial output
SOEmL register, SOEmn bit ← 0
: Disable serial output.
Update operation information/result flag
Change the value of operation information/result flag (g_comstatus) to the next state (receive last data).
g_comstatus ← RxLast
return

Figure 5.27 Data Reception Start Processing

5.7.22 Data Receive Processing

Figure 5.28 shows the flowchart for the data receive processing.

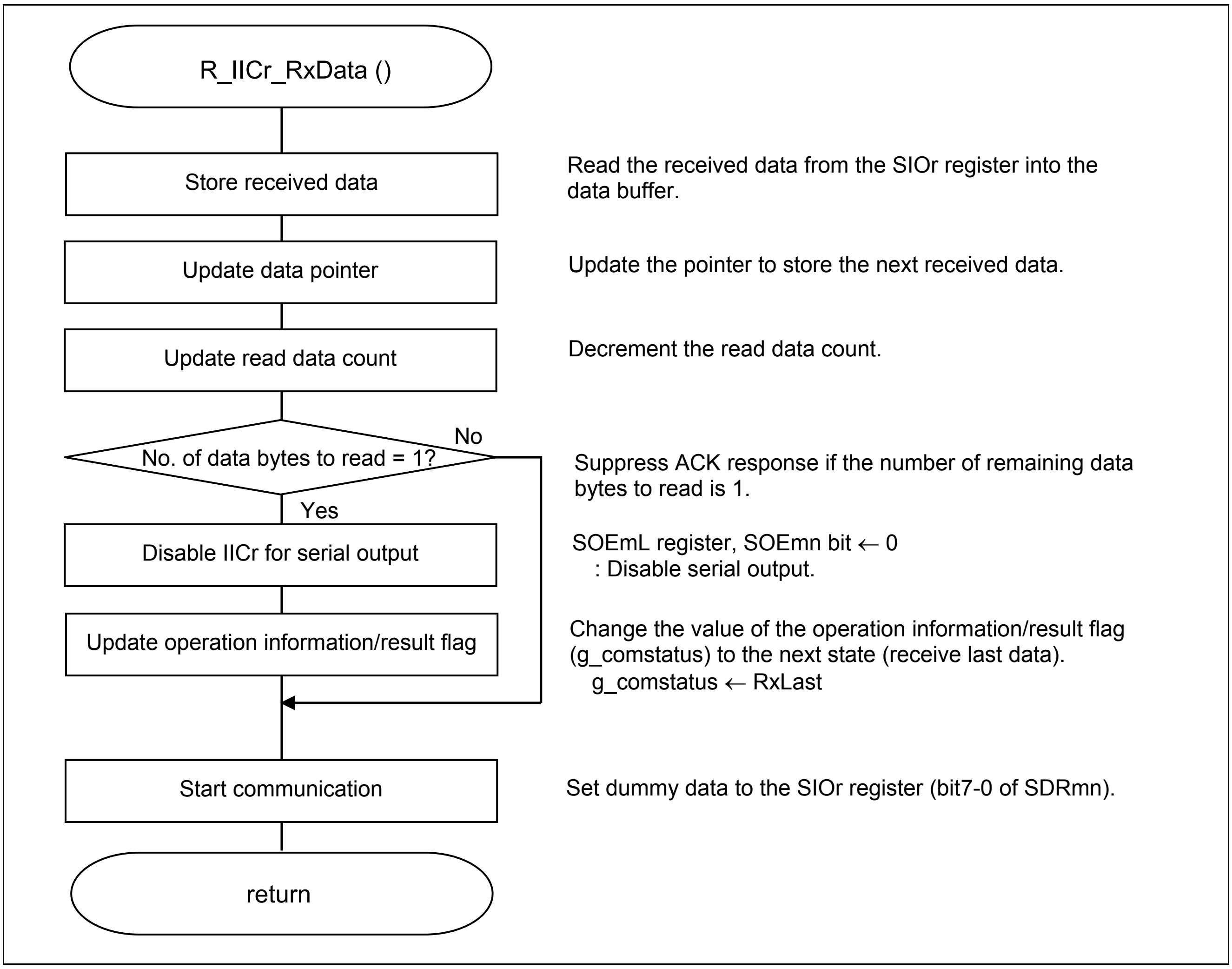

R_IICr_RxData ()
Store received data
Read the received data from the SIOr register into the data buffer.
Update data pointer
Update the pointer to store the next received data.
Update read data count
Decrement the read data count.
No. of data bytes to read = 1?
No
Yes
Suppress ACK response if the number of remaining data bytes to read is 1.
Disable IICr for serial output
SOEmL register, SOEmn bit ← 0
: Disable serial output.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (receive last data).
g_comstatus ← RxLast
Start communication
Set dummy data to the SIOr register (bit7-0 of SDRmn).
return

Figure 5.28 Data Receive Processing

5.7.23 Last Data Receive Processing

Figure 5.29 shows the flowchart for the last data receive processing.

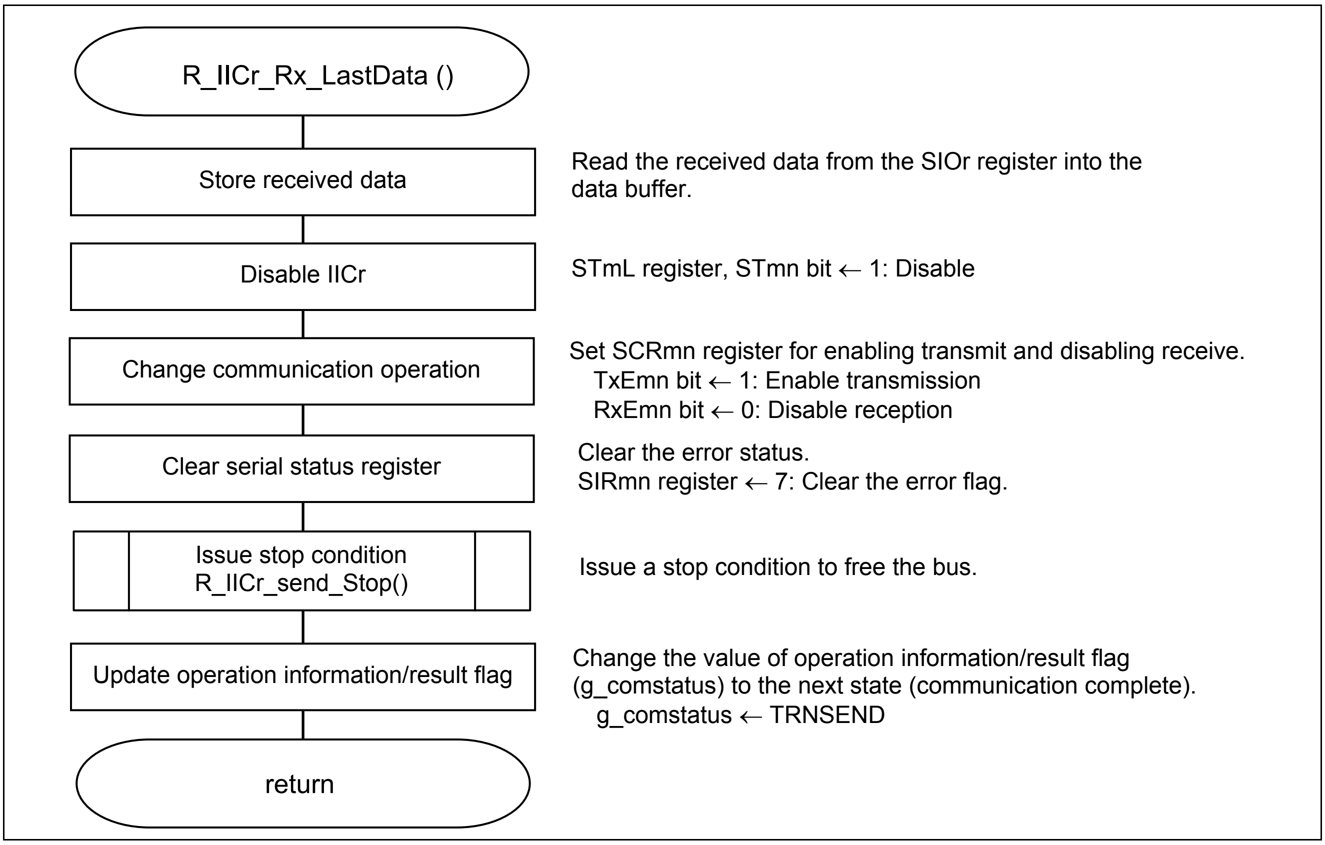

R_IICr_Rx_LastData ()
Store received data
Read the received data from the SIOr register into the data buffer.
Disable IICr
STmL register, STmn bit ← 1: Disable
Change communication operation
Set SCRmn register for enabling transmit and disabling receive.
TxEmn bit ← 1: Enable transmission
RxEmn bit ← 0: Disable reception
Clear serial status register
Clear the error status.
SIRmn register ← 7: Clear the error flag.
Issue stop condition
R_IICr_send_Stop()
Issue a stop condition to free the bus.
Update operation information/result flag
Change the value of operation information/result flag (g_comstatus) to the next state (communication complete).
g_comstatus ← TRNSEND
return

Figure 5.29 Last Data Receive Processing

5.7.24 Data Transmission Start Processing

Figure 5.30 shows the flowchart for the data transmission start processing.

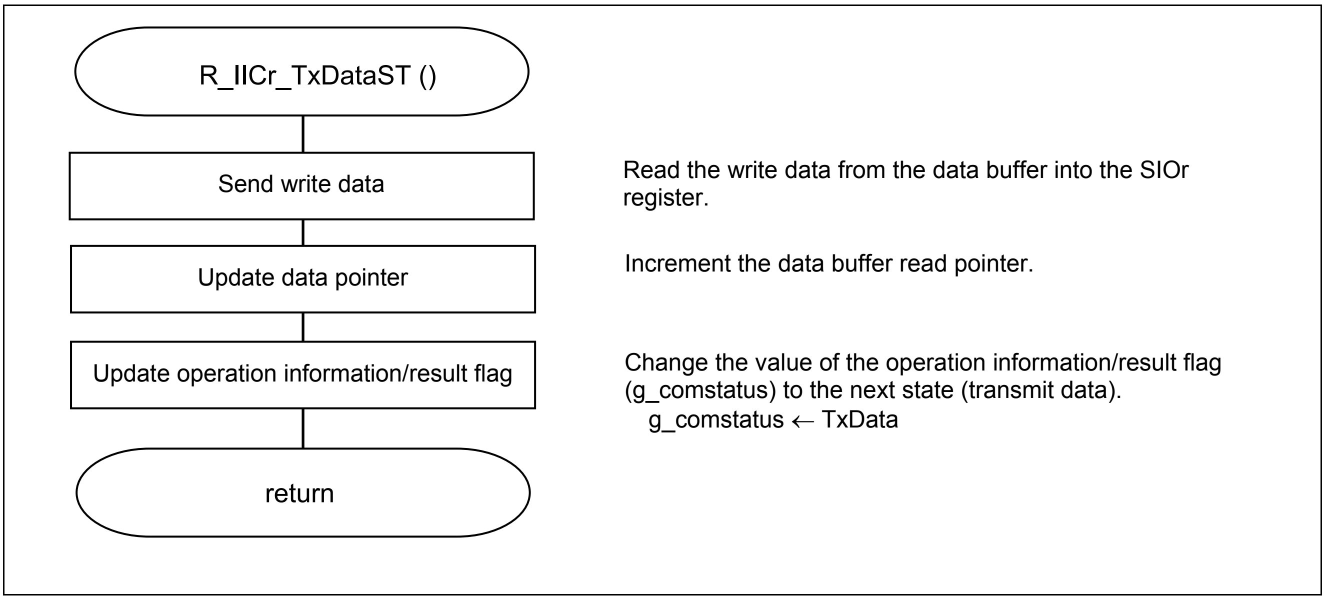

R_IICr_TxDataST ()
Send write data
Read the write data from the data buffer into the SIOr register.
Update data pointer
Increment the data buffer read pointer.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (transmit data).
g_comstatus ← TxData
return

Figure 5.30 Data Transmission Start Processing

5.7.25 Data Transmit Processing

Figure 5.31 shows the flowchart for the data transmit processing.

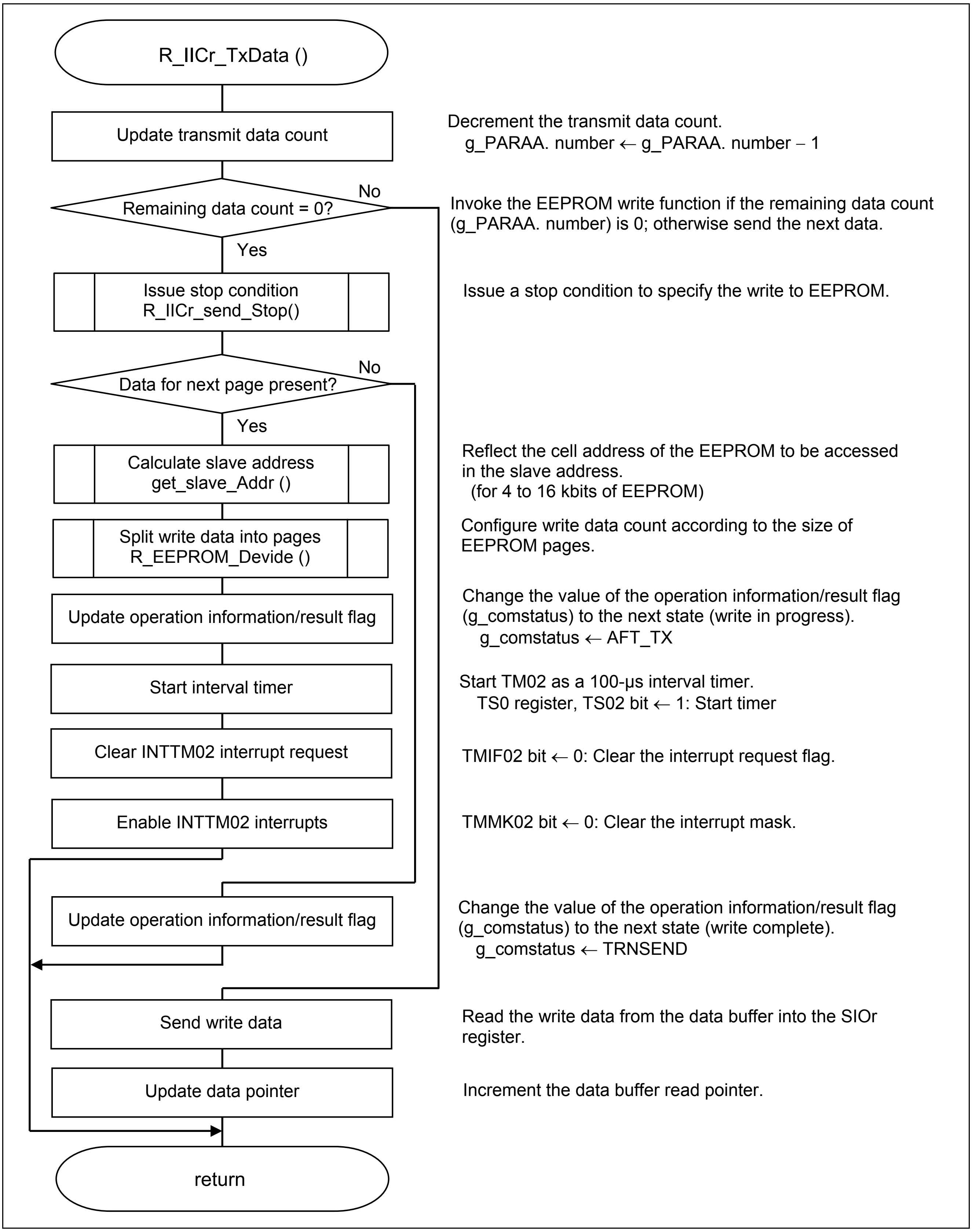

R_IICr_TxData ()
Update transmit data count
Decrement the transmit data count.
g_PARAA. number ← g_PARAA. number – 1
Remaining data count = 0?
No
Yes
Invoke the EEPROM write function if the remaining data count (g_PARAA. number) is 0; otherwise send the next data.
Issue stop condition
R_IICr_send_Stop()
Issue a stop condition to specify the write to EEPROM.
Data for next page present?
No
Yes
Calculate slave address
get_slave_Addr ()
Reflect the cell address of the EEPROM to be accessed in the slave address.
(for 4 to 16 kbits of EEPROM)
Split write data into pages
R_EEPROM_Devide ()
Configure write data count according to the size of EEPROM pages.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (write in progress).
g_comstatus ← AFT_TX
Start interval timer
Start TM02 as a 100-μs interval timer.
TS0 register, TS02 bit ← 1: Start timer
Clear INTTM02 interrupt request
TMIF02 bit ← 0: Clear the interrupt request flag.
Enable INTTM02 interrupts
TMMK02 bit ← 0: Clear the interrupt mask.
Update operation information/result flag
Change the value of the operation information/result flag (g_comstatus) to the next state (write complete).
g_comstatus ← TRNSEND
Send write data
Read the write data from the data buffer into the SIOr register.
Update data pointer
Increment the data buffer read pointer.
return

Figure 5.31 Data Transmit Processing

5.7.26 Next Page Write Start Processing

Figure 5.32 shows the flowchart for the next page write start processing.

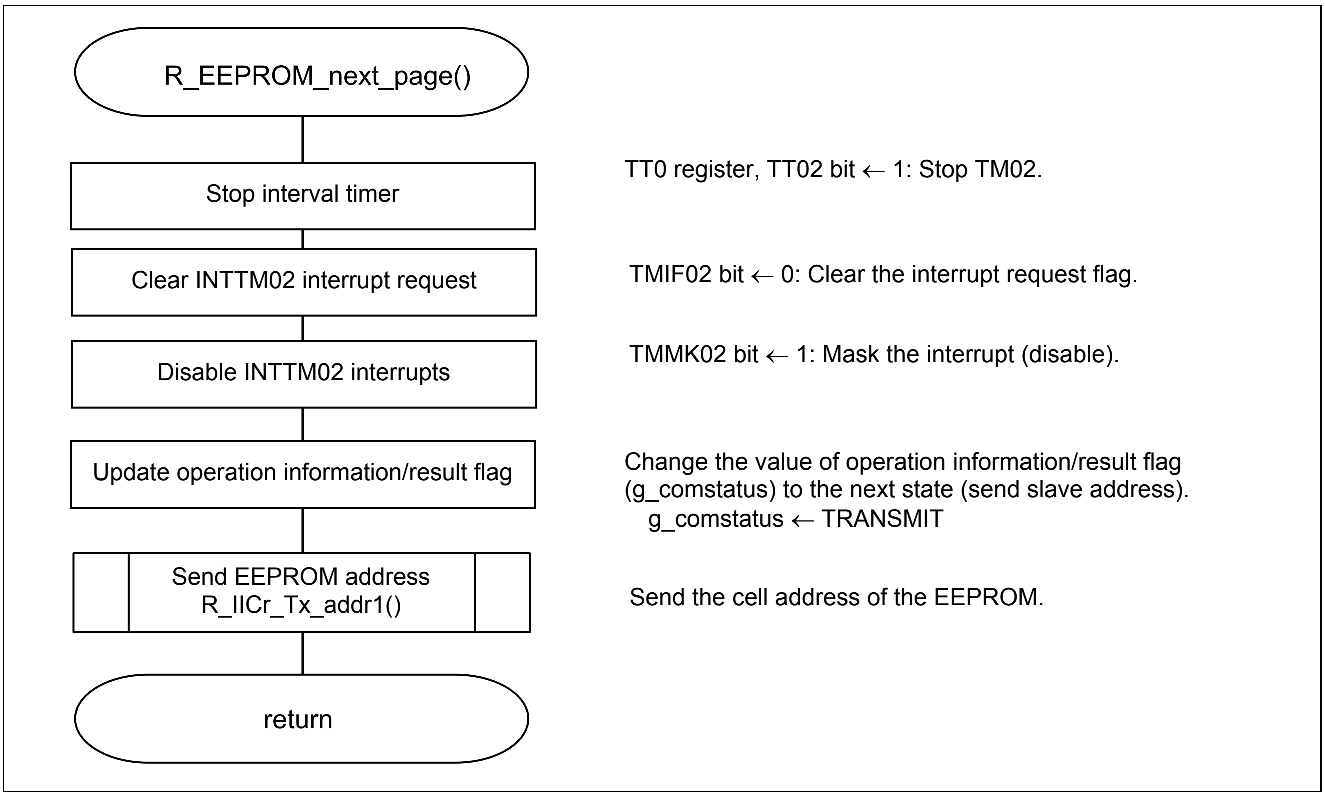

R_EEPROM_next_page()
Stop interval timer
TT0 register, TT02 bit ← 1: Stop TM02.
Clear INTTM02 interrupt request
TMIF02 bit ← 0: Clear the interrupt request flag.
Disable INTTM02 interrupts
TMMK02 bit ← 1: Mask the interrupt (disable).
Update operation information/result flag
Change the value of operation information/result flag (g_comstatus) to the next state (send slave address).
g_comstatus ← TRANSMIT
Send EEPROM address
R_IICr_Tx_addr1()
Send the cell address of the EEPROM.
return

Figure 5.32 Next Page Write Start Processing

5.7.27 SCL Dummy Clock Output Processing

Figure 5.33 shows the flowchart for the SCL dummy clock output processing.

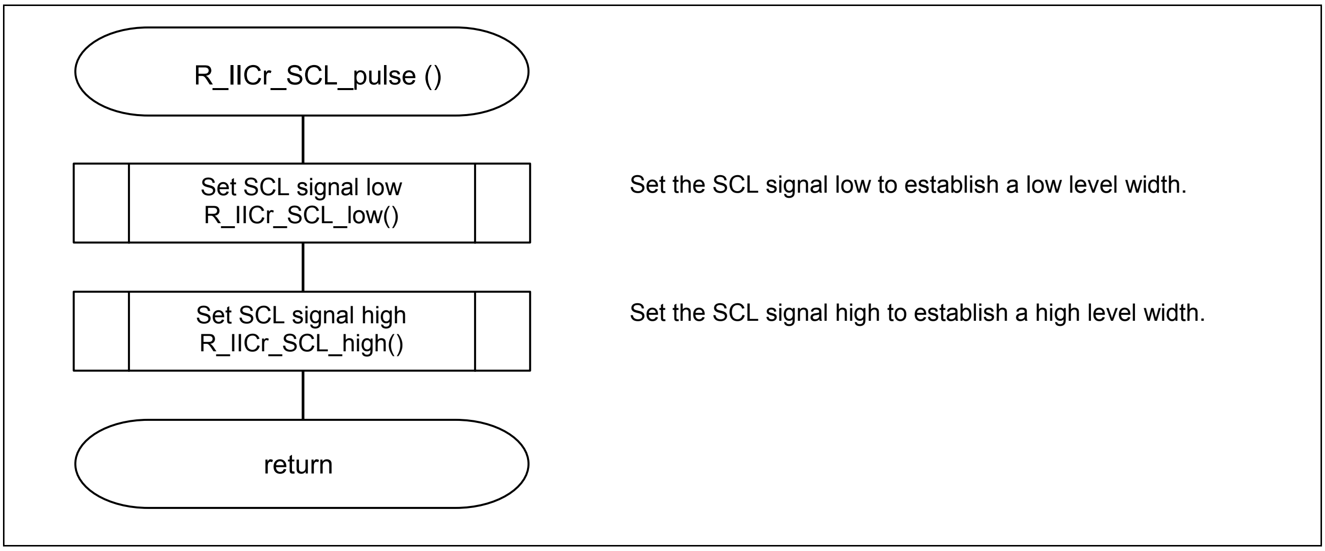

R_IICr_SCL_pulse ()
Set SCL signal low
R_IICr_SCL_low()
Set the SCL signal low to establish a low level width.
Set SCL signal high
R_IICr_SCL_high()
Set the SCL signal high to establish a high level width.
return

Figure 5.33 SCL Dummy Clock Output Processing

5.7.28 SCL Set High Processing

Figure 5.34 shows the flowchart for setting SCL high.

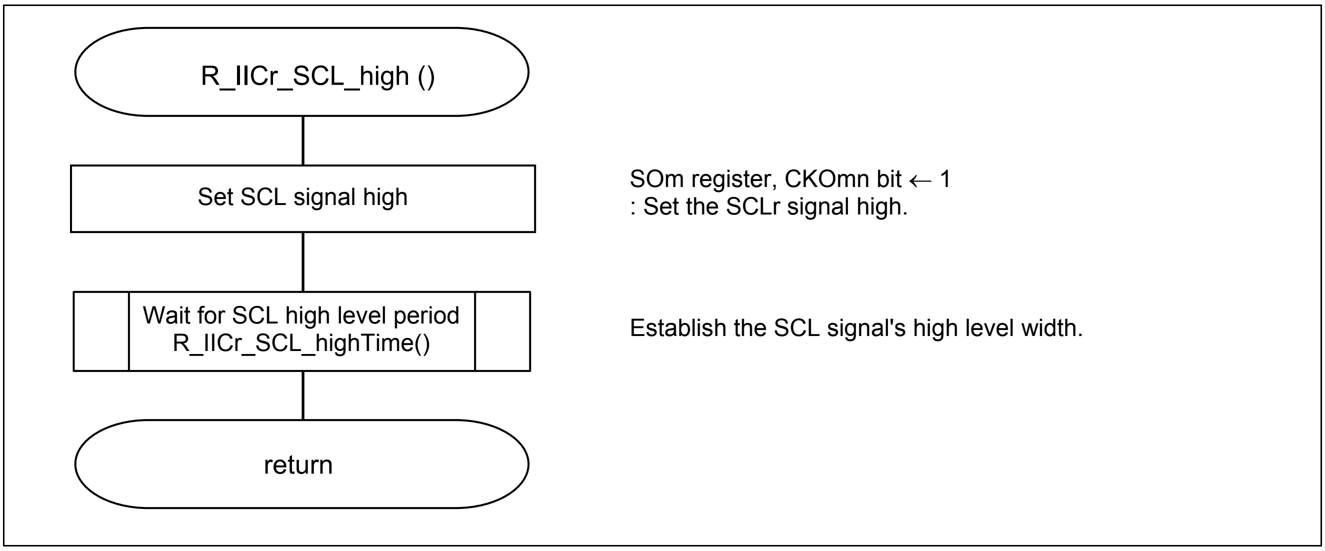

R_IICr_SCL_high ()
Set SCL signal high
SOm register, CKOmn bit ← 1
: Set the SCLr signal high.
Wait for SCL high level period
R_IICr_SCL_highTime()
Establish the SCL signal's high level width.
return

Figure 5.34 SCL Set High Processing

5.7.29 SCL Set Low Processing

Figure 5.35 shows the flowchart for setting SCL low.

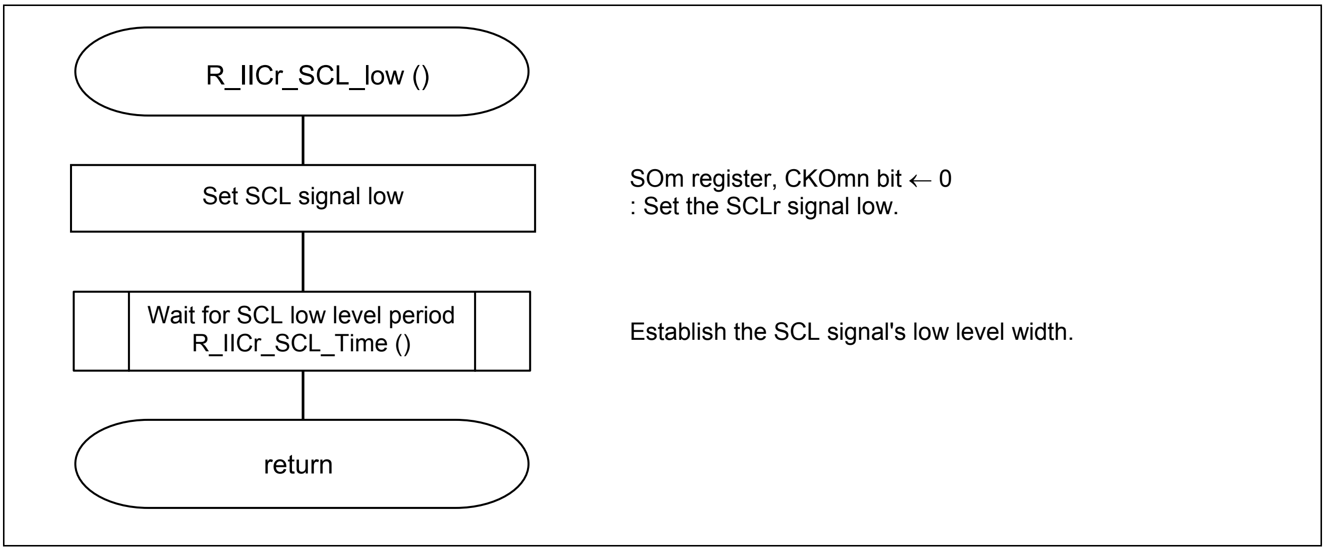

R_IICr_SCL_low ()
Set SCL signal low
SOm register, CKOmn bit ← 0
: Set the SCLr signal low.
Wait for SCL low level period
R_IICr_SCL_Time ()
Establish the SCL signal's low level width.
return

Figure 5.35 SCL Set Low Processing

5.7.30 ACK Confirmation Processing

Figure 5.36 shows the flowchart for checking the ACK response.

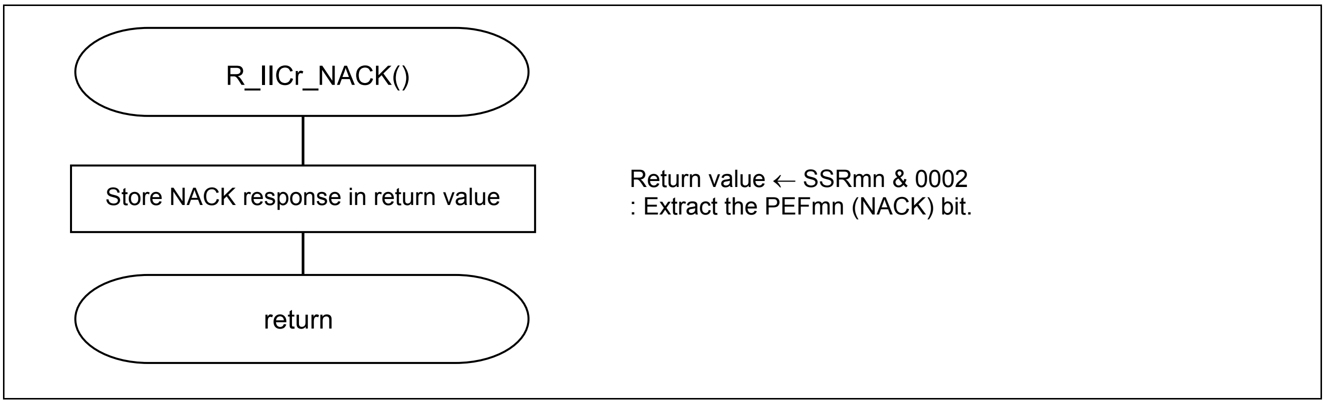

R_IICr_NACK()
Store NACK response in return value
Return value ← SSRmn & 0002
: Extract the PEFmn (NACK) bit.
return

Figure 5.36 ACK Response Check Processing

5.7.31 SCL Low Level Period Wait Processing

Figure 5.37 shows the flowchart for the SCL low level period wait processing.

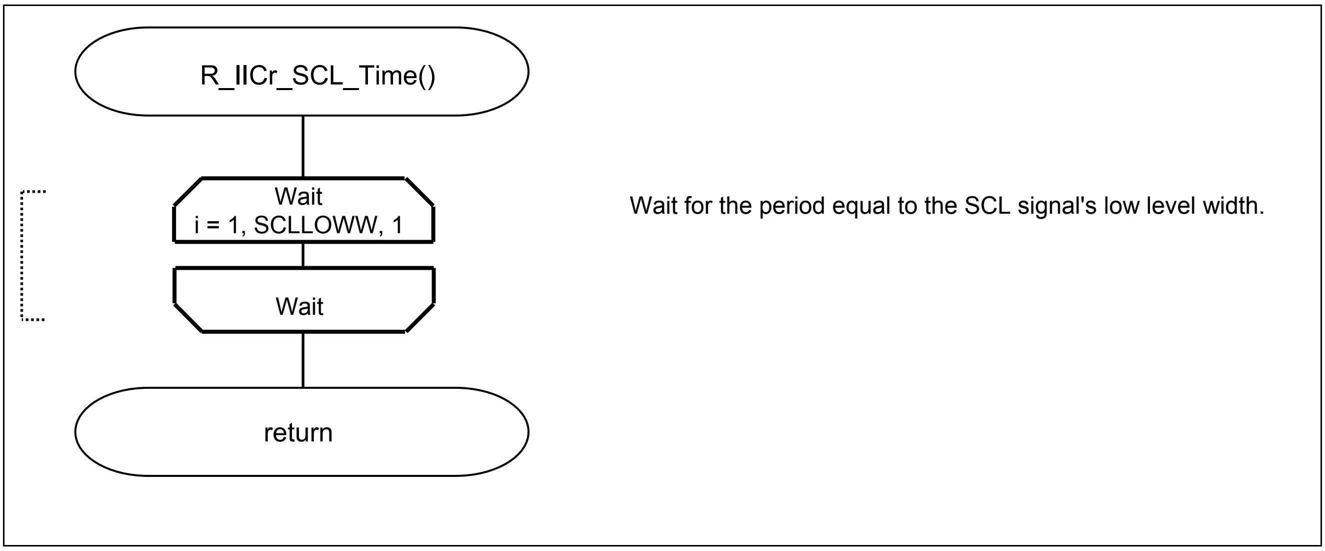

R_IICr_SCL_Time()
Wait
i = 1, SCLLOWW, 1
Wait
return
Wait for the period equal to the SCL signal's low level width.

Figure 5.37 SCL Low Level Period Wait Processing

5.7.32 SCL High Level Period Wait Processing

Figure 5.38 shows the flowchart for the SCL high level period wait processing.

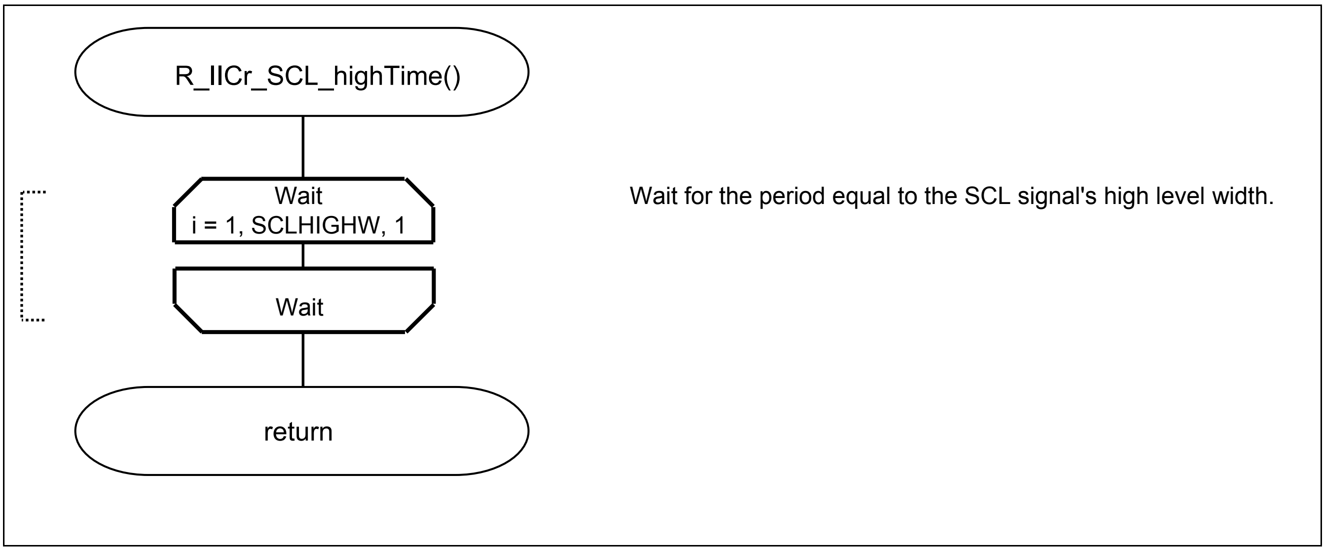
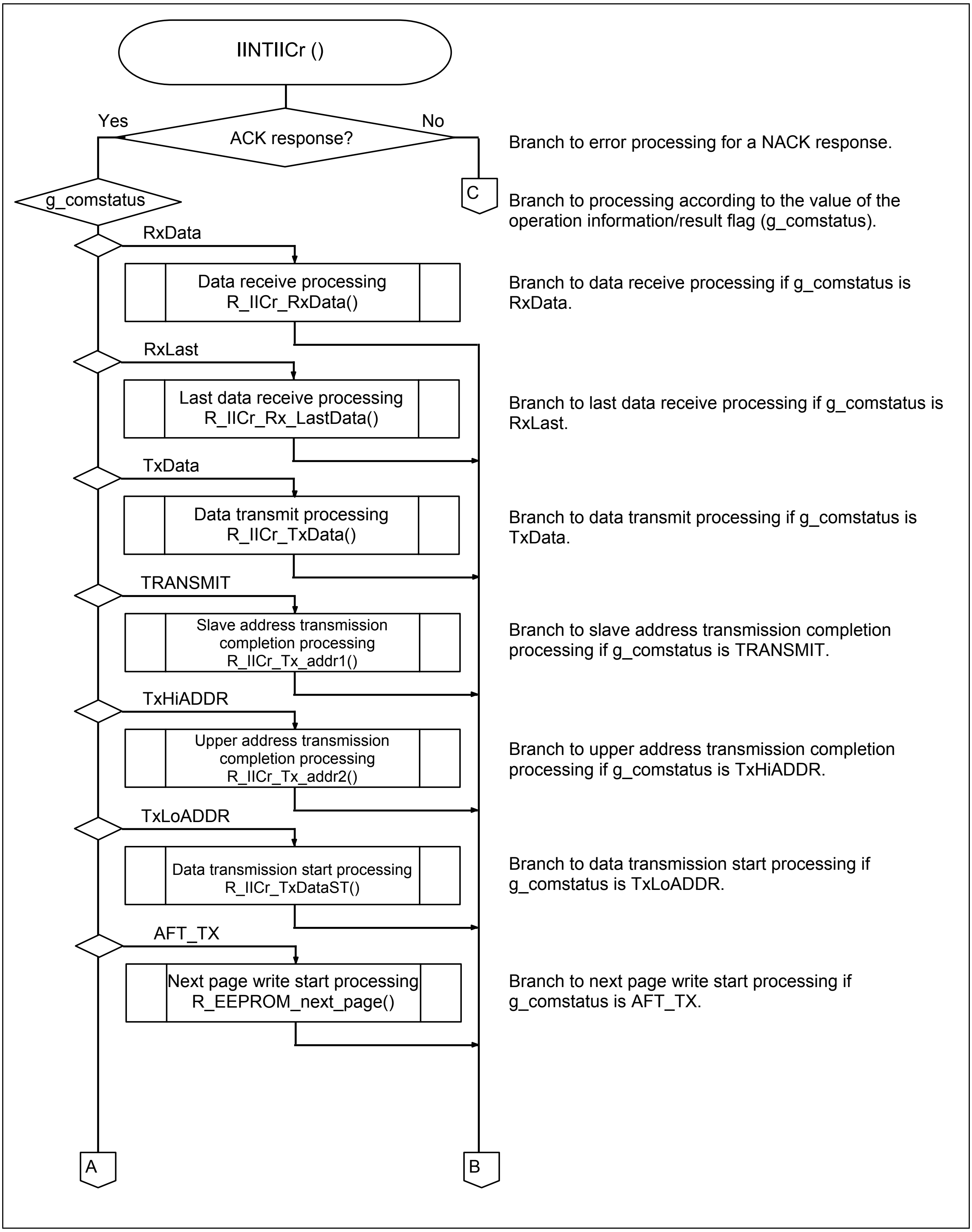

R_IICr_SCL_highTime()
Wait
i = 1, SCLHIGHW, 1
Wait
return
Wait for the period equal to the SCL signal's high level width.

Figure 5.38 SCL High Level Period Wait Processing

5.7.33 INTIICr Interrupt Processing

Figures 5.39 to 5.41 show the flowcharts for the INTIICr interrupt processing.



Branch to error processing for a NACK response.

Branch to processing according to the value of the operation information/result flag (g_comstatus).

Branch to data receive processing if g_comstatus is RxData.

Branch to last data receive processing if g_comstatus is RxLast.

Branch to data transmit processing if g_comstatus is TxData.

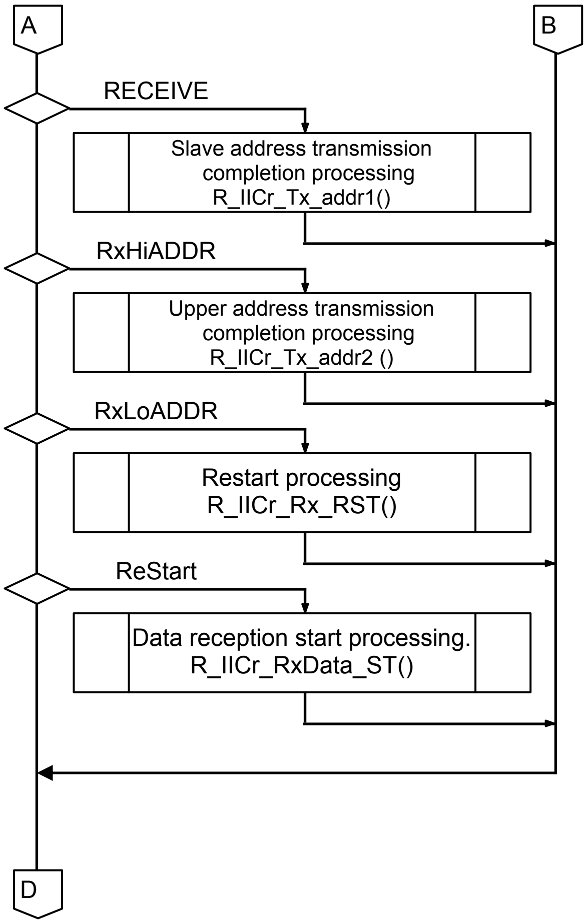
Branch to slave address transmission completion processing if g_comstatus is TRANSMIT.

Branch to upper address transmission completion processing if g_comstatus is TxHiADDR.

Branch to data transmission start processing if g_comstatus is TxLoADDR.

Branch to next page write start processing if g_comstatus is AFT_TX.

Figure 5.39 INTIICr Interrupt Processing (1/3)


A
B
RECEIVE
Slave address transmission completion processing R_IICr_Tx_addr1()
RxHiADDR
Upper address transmission completion processing R_IICr_Tx_addr2 ()
RxLoADDR
Restart processing R_IICr_Rx_RST()
ReStart
Data reception start processing. R_IICr_RxData_ST()
D

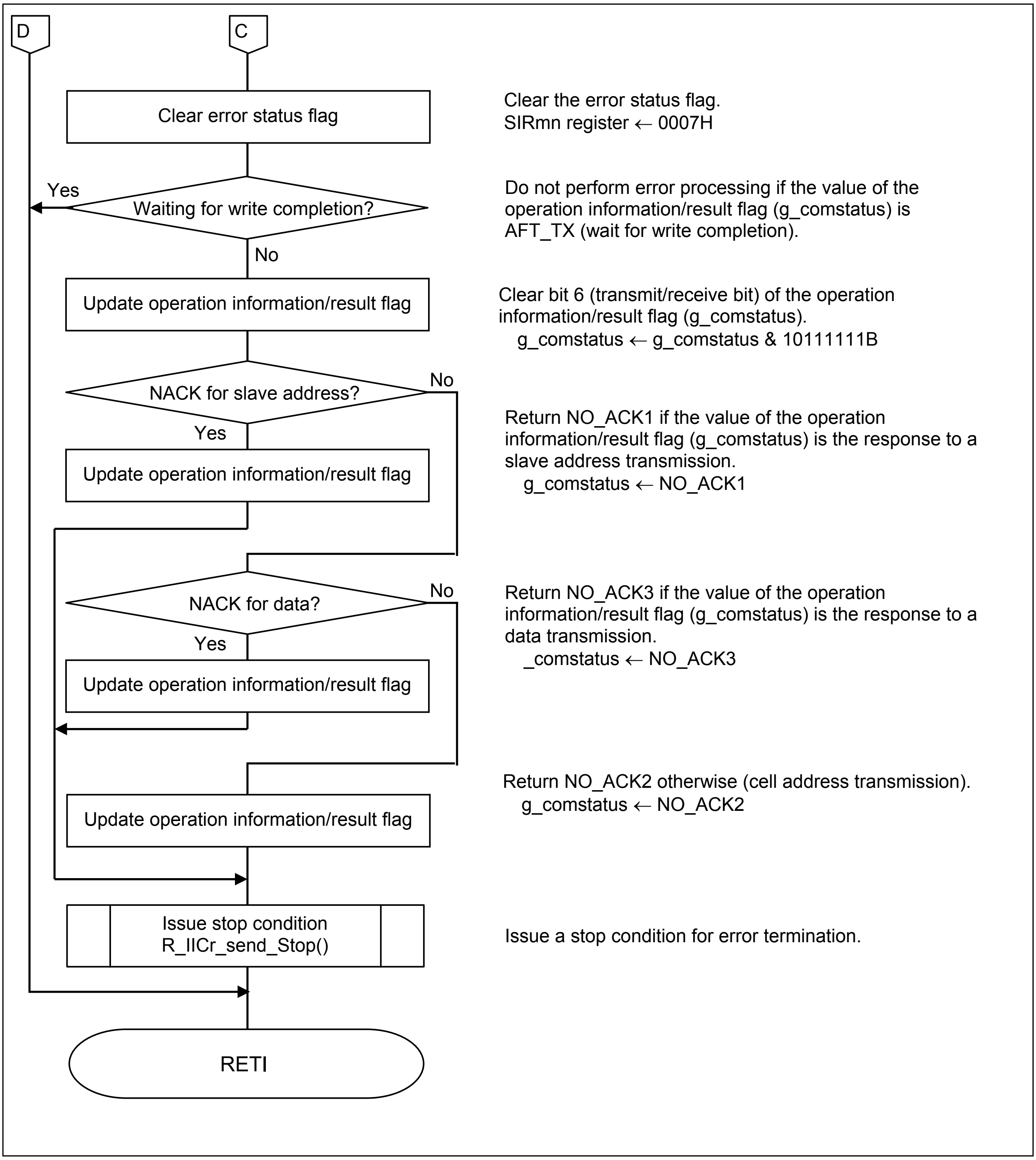
Branch to data receive mode slave address transmission completion processing if g_comstatus is RECEIVE.

Branch to data receive mode upper address transmission completion processing if g_comstatus is RxHiADDR.

Branch to data receive mode restart processing if g_comstatus is RxLoADDR.

Branch to data receptions start processing if g_comstatus is ReStart.

Figure 5.40 INTIICr Interrupt Processing (2/3)


D
C
Clear error status flag
Yes
Waiting for write completion?
No
Update operation information/result flag
NACK for slave address?
No
Yes
Update operation information/result flag
NACK for data?
No
Yes
Update operation information/result flag
Update operation information/result flag
Issue stop condition
R_IICr_send_Stop()
RETI

Clear the error status flag.
SIRmn register ← 0007H

Do not perform error processing if the value of the operation information/result flag (g_comstatus) is AFT_TX (wait for write completion).

Clear bit 6 (transmit/receive bit) of the operation information/result flag (g_comstatus).
g_comstatus ← g_comstatus & 10111111B

Return NO_ACK1 if the value of the operation information/result flag (g_comstatus) is the response to a slave address transmission.
g_comstatus ← NO_ACK1

Return NO_ACK3 if the value of the operation information/result flag (g_comstatus) is the response to a data transmission.
_comstatus ← NO_ACK3

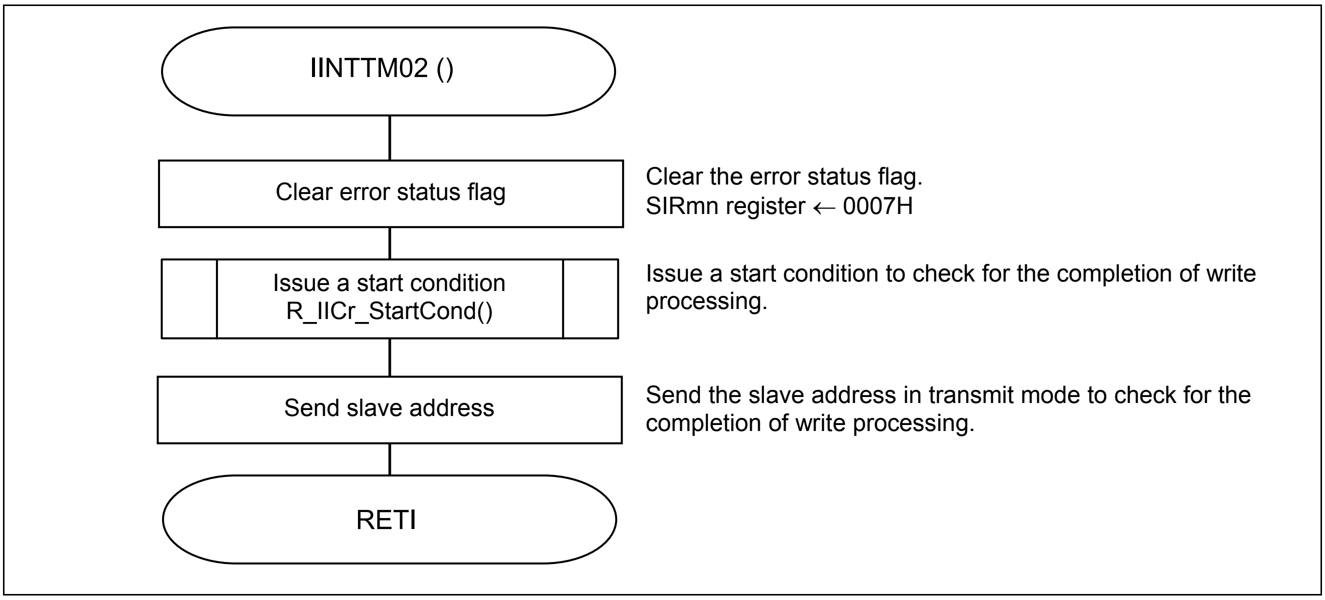
Return NO_ACK2 otherwise (cell address transmission).
g_comstatus ← NO_ACK2

Issue a stop condition for error termination.

Figure 5.41 INTIICr Interrupt Processing (3/3)

5.7.34 INTTM02 Interrupt Processing

Figure 5.42 shows the flowchart for the INTTM02 interrupt processing.



Clear the error status flag.
SIRmn register ← 0007H

Issue a start condition to check for the completion of write processing.

Send the slave address in transmit mode to check for the completion of write processing.

Figure 5.42 INTTM02 Interrupt Processing

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G13 User's Manual: Hardware (R01UH0146E)

RL78 Family User's Manual: Software (R01US0015E)

(The latest versions of the documents are available on the Renesas Electronics Website.)

Technical Updates/Technical Brochures

(The latest versions of the documents are available on the Renesas Electronics Website.)

Website and Support

Renesas Electronics Website

- http://www.renesas.com/index.jsp

Inquiries

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

Revision Record	RL78/G13 Serial Array Unit (SAU) (EEPROM Control Using Simplified IIC)

Rev.	Date	Description	
		Page	Summary
1.00	2012. 9.30	—	First edition issued
2.00	2014.3.24	12	Table 2.1: Added e2studio and IAR version information
		34	Fibure 5.1, Figure 5.2: Added note
		37	Figure 5.6: Updated
2.01	2014.5.16 -1	19	Table 5.2: Corrected
		38	Notes for Bit3-2: Added
		43	The setteing of $f_{MCK:}$ Modified
	2014.5.16 -2	5-11	Correction (SCKmn bit -> CKOmn bit)
		7,8	Correction (P.7 (12) and P.8 (10))
		24	Correction (Outline of R_IICr_send_Stop function)
		25	Correction (Declaration of R_device_select function)
		27-30	“Function Name: IINTIICr”: Added
		28	Synopsis of R_IICr_Rx_RST function: Modified
		28	Synopsis of R_IICr_RxData function: Added
		28	Explanation of R_IICr_Rx_RST function: Added
		29	Name of function “R_IICr_TxDataST”: Corrected
		29	Synopsis of R_IICr_TxData function: Modified
		29	Name of R_IICr_TxData function: Corrected
		30	Name of R_IICr_wait_bus function: Corrected
		30	Fonts Modified: Explanation in R_IICr_StartCond function, and R_IICr_wait_bus function
		32	Name of IINTIICr function: Modifed (mn -> r)
		32	Declaration of R_IICr_SCL_Time function and R_IICr_SCL_highTime function: Corrected
		33	Declaration of R_IICr_Init function: Corrected
		37	Name of R_device_select function: Corrected
		42	Explanation of bits 5 and 4: Modified
		42	“SDR01” was changed into “SDRmn”
		44	Serial output enable register m (SOEm): Register name was Modified
		46	Explanation of R_IICr_StopCond function function: Modifeid
		46	Name of Fig. 5.7: Modified
		47	Buffer name was changed to g_data_bufferW2
		49	Variable name: Corrected to g_eeprom_type
		52,53	Bits 11 to 8: Modified
		53	Fig. 5.13: Correction (Flow chart part)
		53	Fig. 5.13: Correction (Explanation part)
		56	Explanation of return: Modified
		63	Fig 5.22: Corrected
		69	Function name was modified to R_IICr_TxData
	2014.5.16 -3	22	R_EEPROM_wait_write function: Added to variable g_PARAA
		24	Outlline of IINTTM02 function: Modified
		25,30	Function Specifications of R_IICr_StopCond function were redundant, so needless one was deleted.
		67	Modification of Fig. 5.28 (Addition of a process)
		69	Modification of Fig. 5.31 (Sequence of processes)

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 manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

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 unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

- The state of the product is undefined at the moment when power is supplied.
  - — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
    In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
    In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

- Access to reserved addresses is prohibited.
  - — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

- After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
  - — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

- Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.
  - — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

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

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

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

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

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

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

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 3.0