RL78/G23

Serial Array Unit (SAU)
(EEPROM Control Using Simplified IIC)

R01AN7733EJ0100
Rev. 1.00
Apr. 11, 2025

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/G23

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.

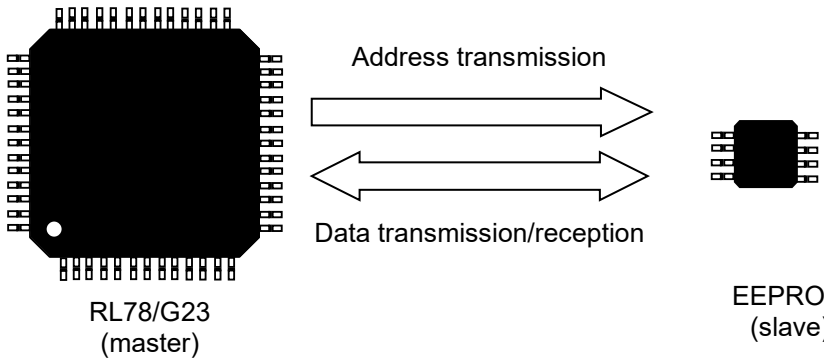
- An access to an EEPROM is accomplished by specifying necessary parameters in a structure and calling a function.
- Control of EEPROM devices is done in an interrupt-driven mode whenever possible with due consideration to the function as an API.
- One EEPROM type can be selected as the target EEPROM 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 03. 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 SCL11 and SDA11 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)

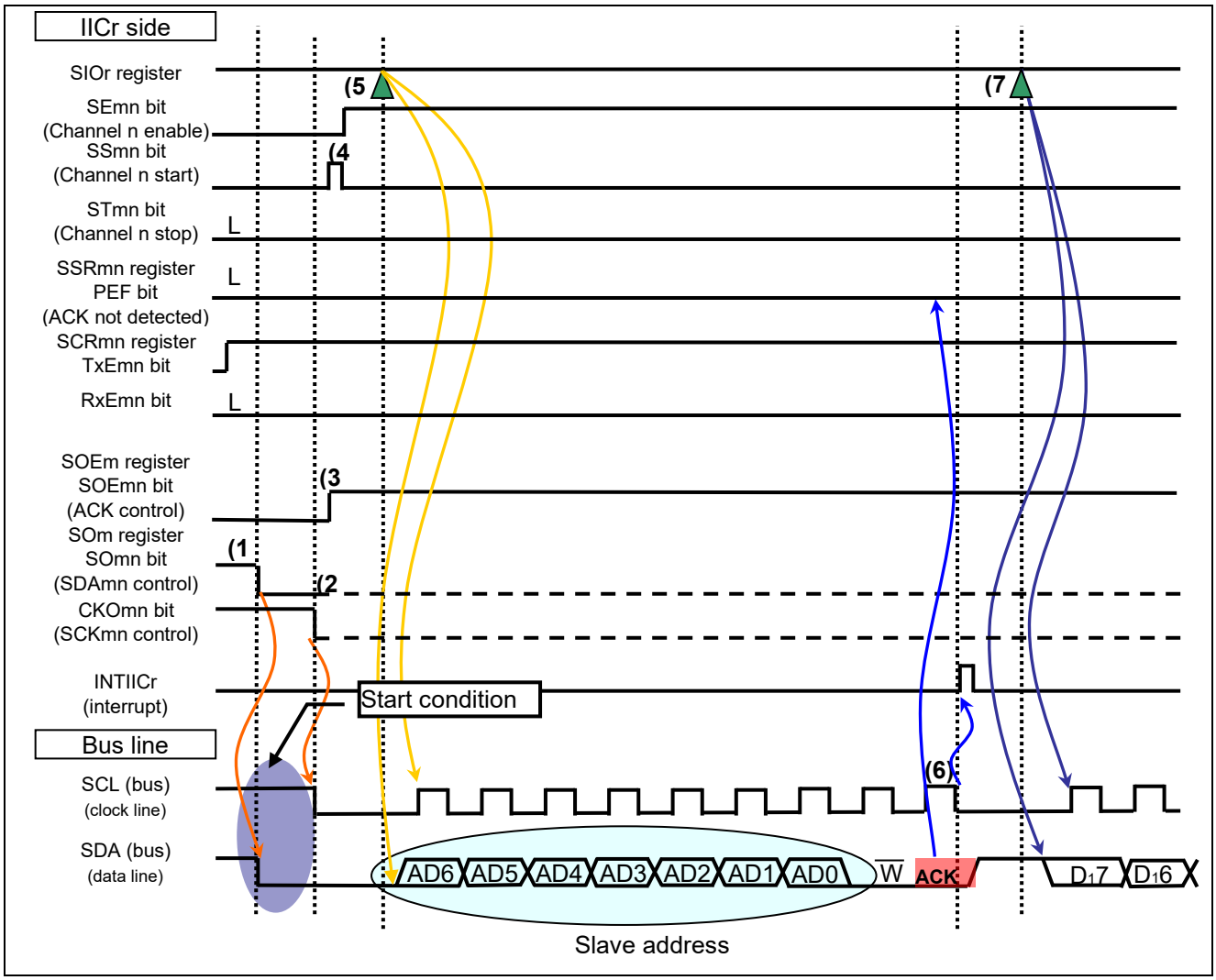


Figure 1.2 IIC Communication Timing Chart (Master-to-Slave Communication Example) (1/4)

(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)

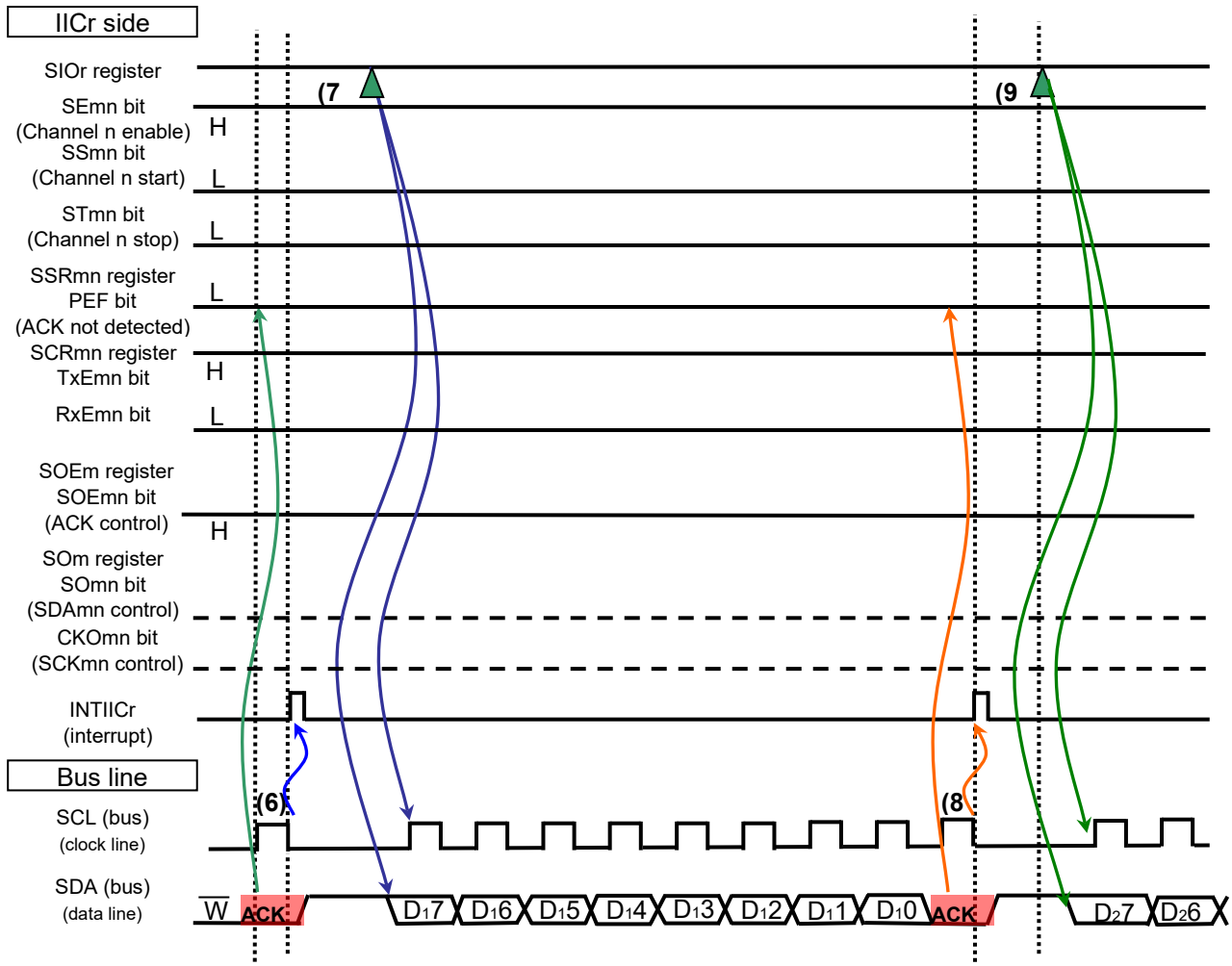


Figure 1.3 IIC Communication Timing Chart (Master-to-Slave Communication Example) (2/4)

(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)

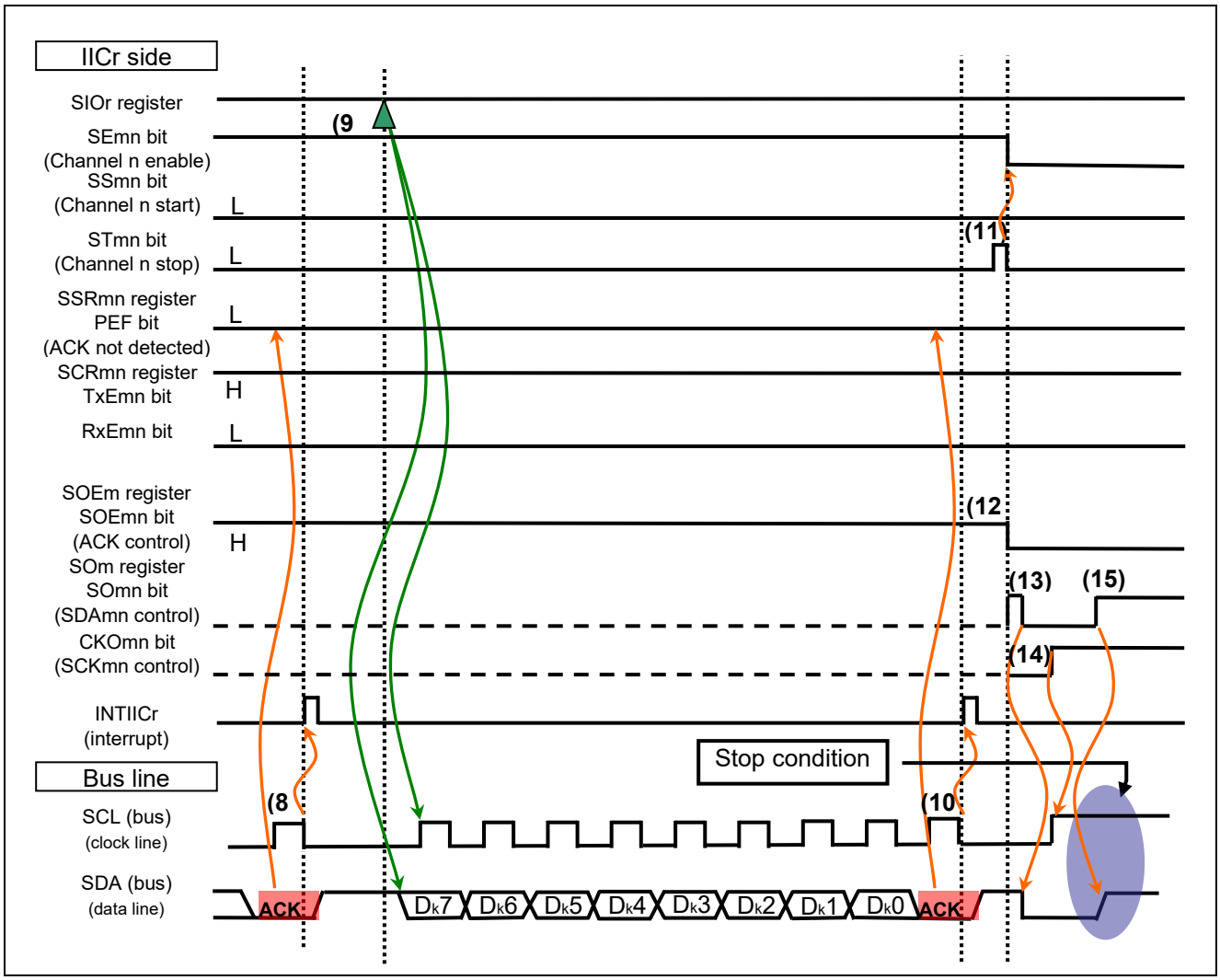


Figure 1.4 IIC Communication Timing Chart (Master-to-Slave Communication Example) (3/4)

(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)

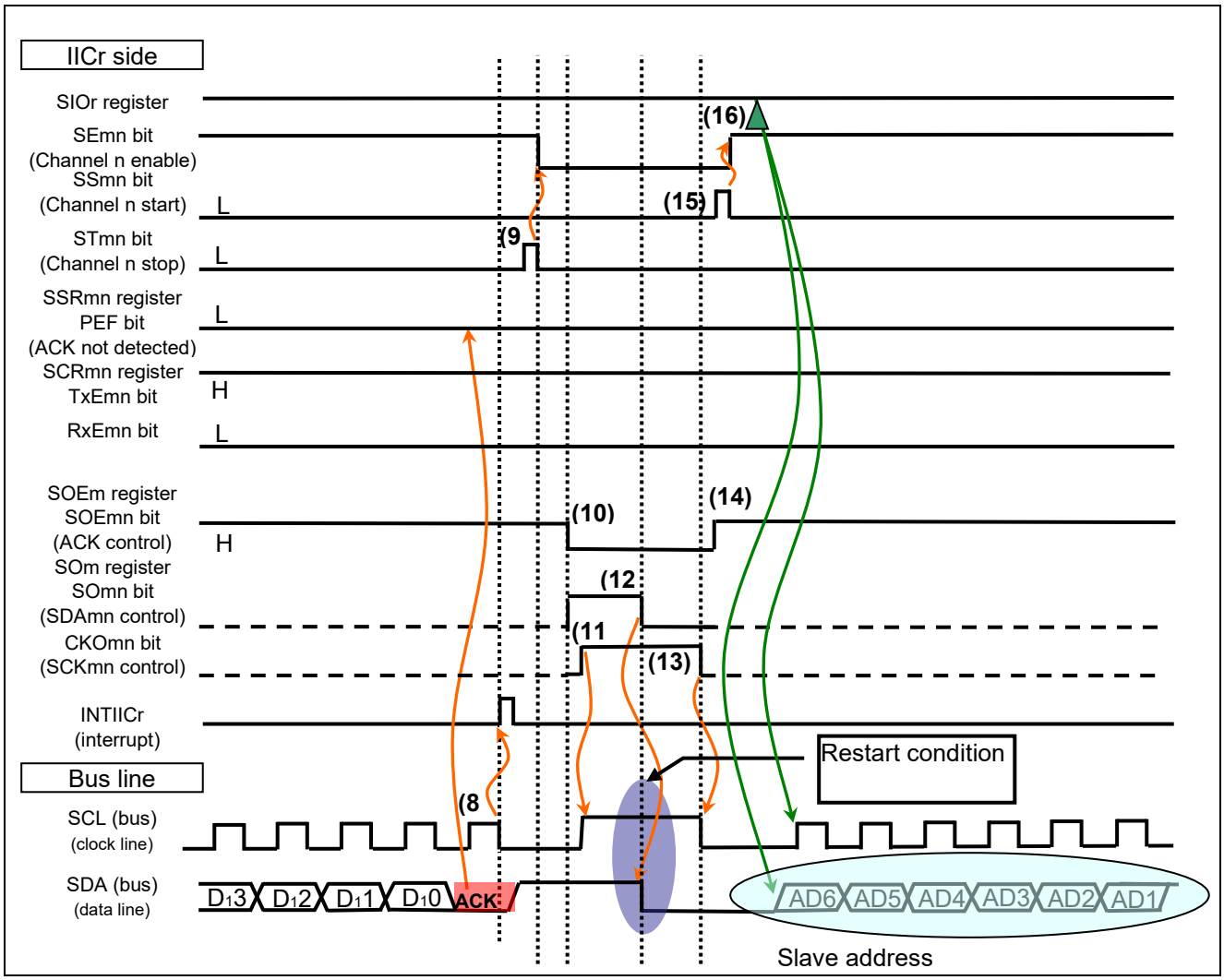


Figure 1.5 IIC Communication Timing Chart (Master-to-Slave Communication Example) (4/4)

(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)

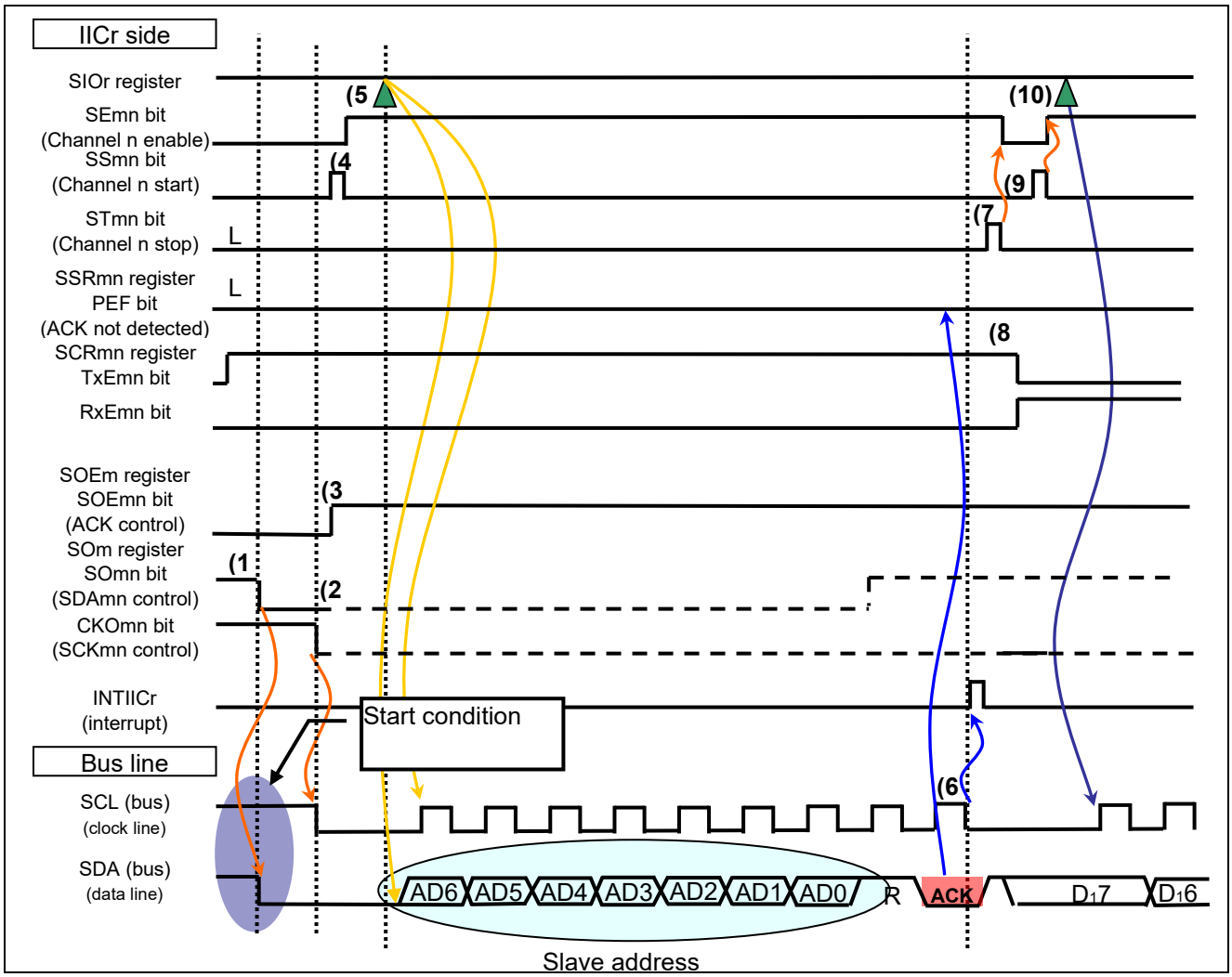


Figure 1.6 IIC Communication Timing Chart (Slave-to-Master Communication Example) (1/3)

(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)

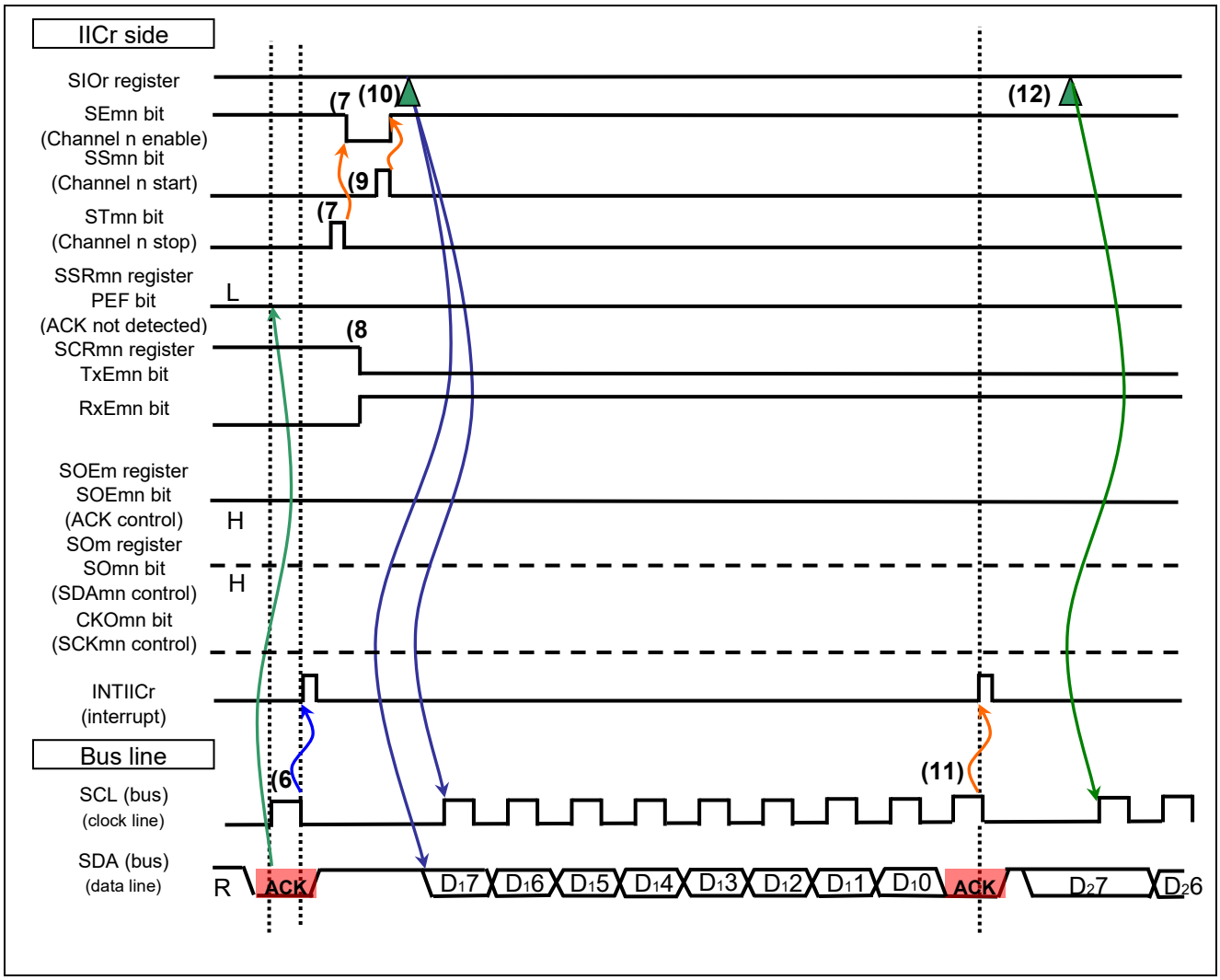


Figure 1.7 IIC Communication Timing Chart (Slave-to-Master Communication Example) (2/3)

(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)**

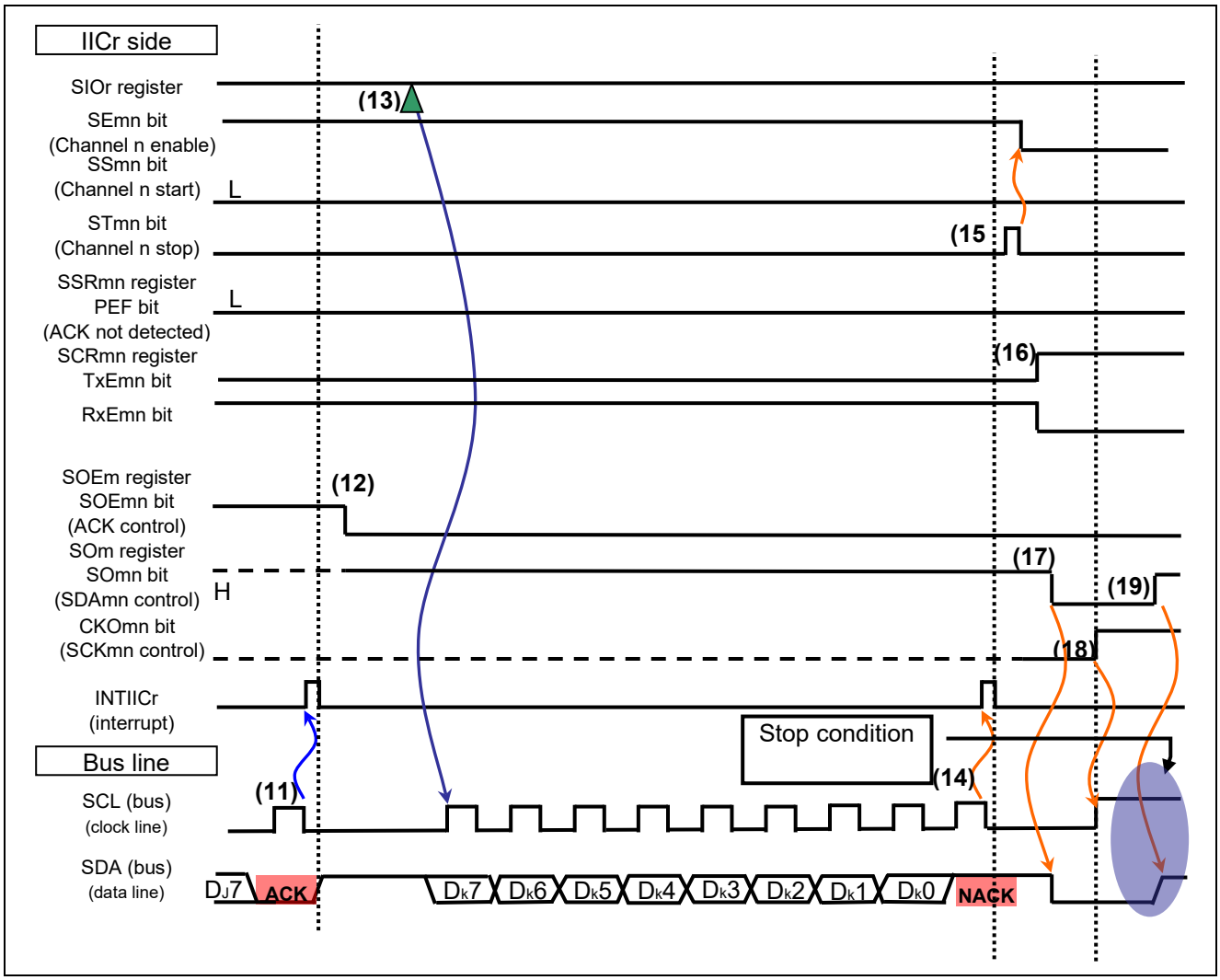


Figure 1.8 IIC Communication Timing Chart (Slave-to-Master Communication Example) (3/3)

(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/G23 (R7F100GLG)
Operating frequency	• High-speed on-chip oscillator (HOCO) clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	• 3.3V • LVD0 operations (VLVD0): Reset mode Rising edge TYP.1.90V Falling edge TYP.1.86V
Integrated development environment (CS+)	CS + V8.13.00 from Renesas Electronics Corpration
C compiler (CS+)	CC-RL V1.15.00 from Renesas Electronics Corporation
Integrated development environment (e^2 studio)	e^2 studio V2025-01 (25.1.0) from Renesas Electronics Corporation
C compiler (e^2 studio)	CC-RL V1.15.00 from Renesas Electronics Corporation
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V5.10.3 from IAR Systems
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V5.10.3.2716 from IAR Systems
Board used	RL78/G23 Fast Prototyping Board (RTK7RLG230CLG000BJ) + EEPROM (R1EX24016)

3. Related Application Note

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

- RL78/G23 I2C Supporting Multiple Slave Address (Master) (R01AN5825E) Application Note
- RL78/G23 I2C Supporting Multiple Slave Addresses (Slave) (R01AN5670E) 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.

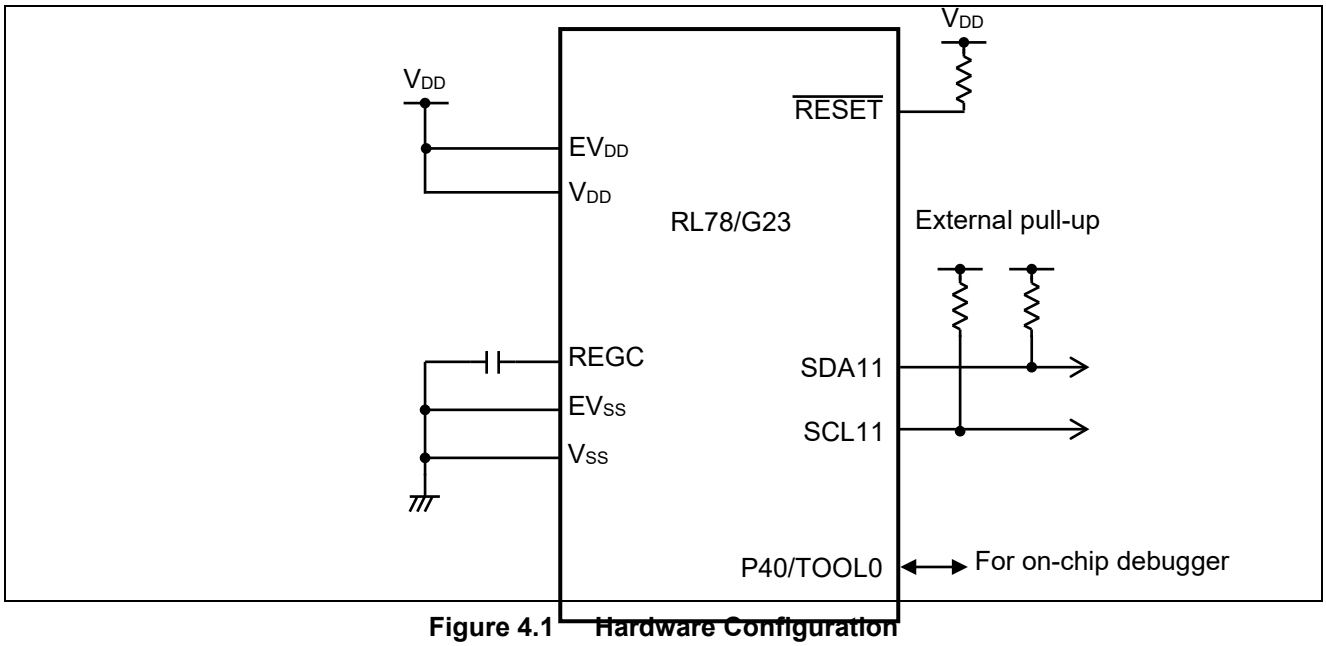


Figure 4.1 Hardware Configuration

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_{LVD0}) that is specified as LVD0.

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
P30/SCL11	Input/Output	IIC11 serial clock output pin
P50/SDA11	Input/Output	IIC11 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 SO03 and CKO03 to 1.
- Sets the P30/SCL11 pin for transfer clock output and the P50/SDA11 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.

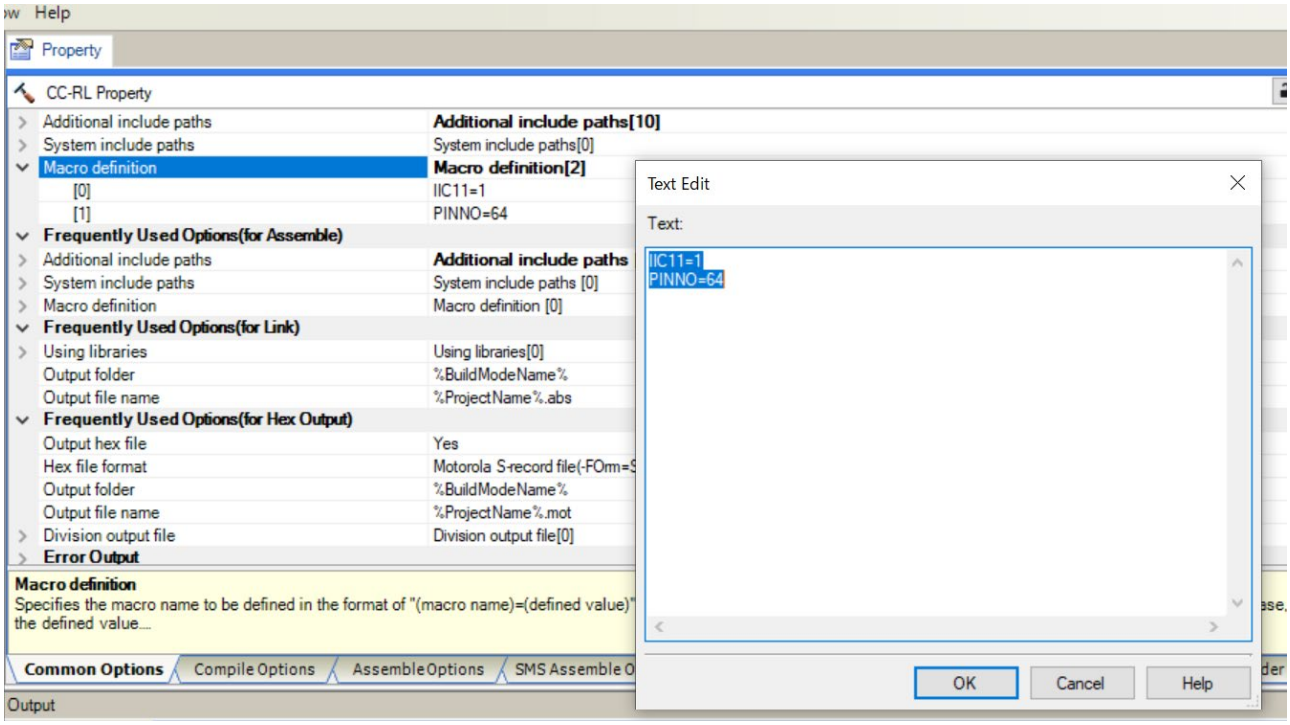
Note: This sample code is available to illustrate the example of controlling the EEPROM (R1EX24016) on an IIC bus using the simplified IIC11 function of the RL78/G23 series. When the channel or EEPROM to be used with this sample code is changed, conduct extensive evaluation of the modified code.

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 CS+ environment.

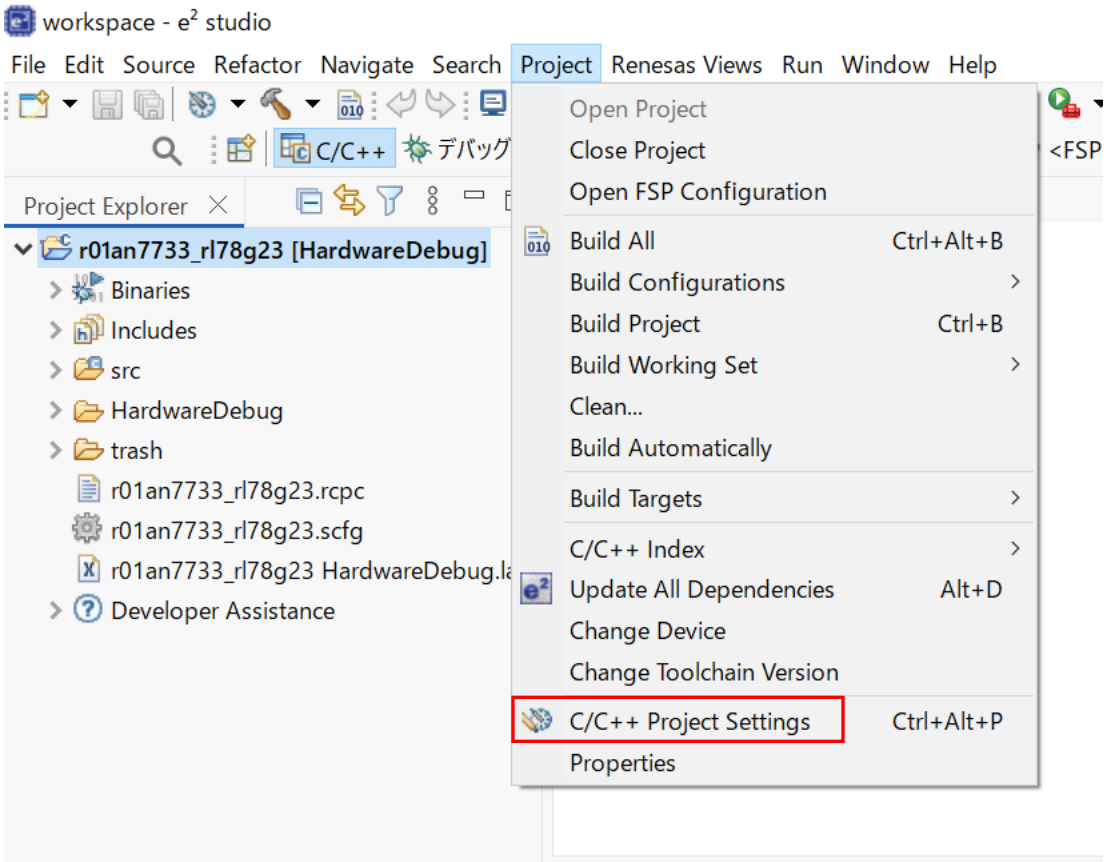
The Macro definition[0] selects IIC00 as the simplified IIC channel to be used. When using channel 20, for example, modify there to "IIC20 = 1."

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.

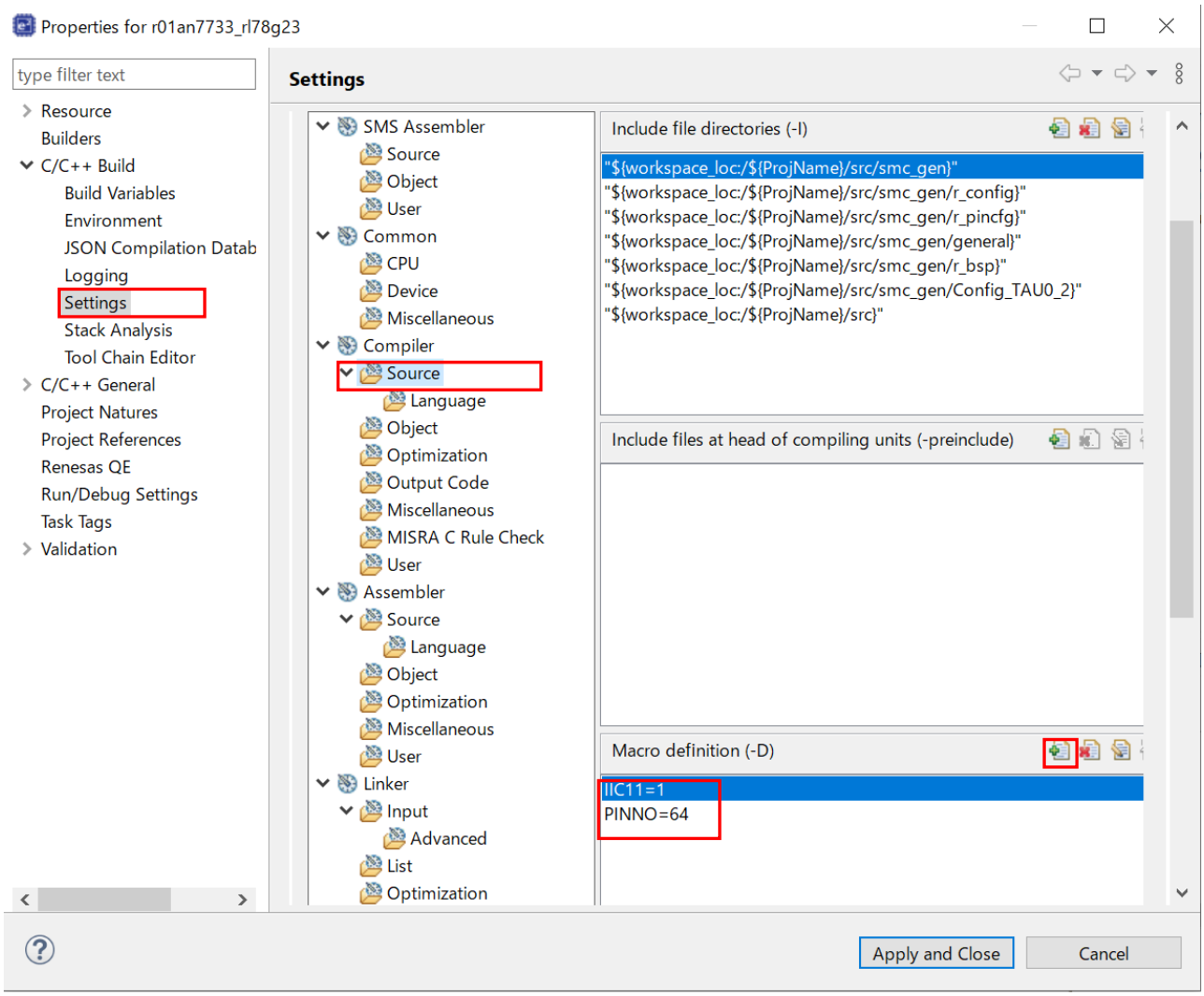
To change these, click the button at the right-most end of [Macro definition] to open the text edit window, then edit the definitions in that window.



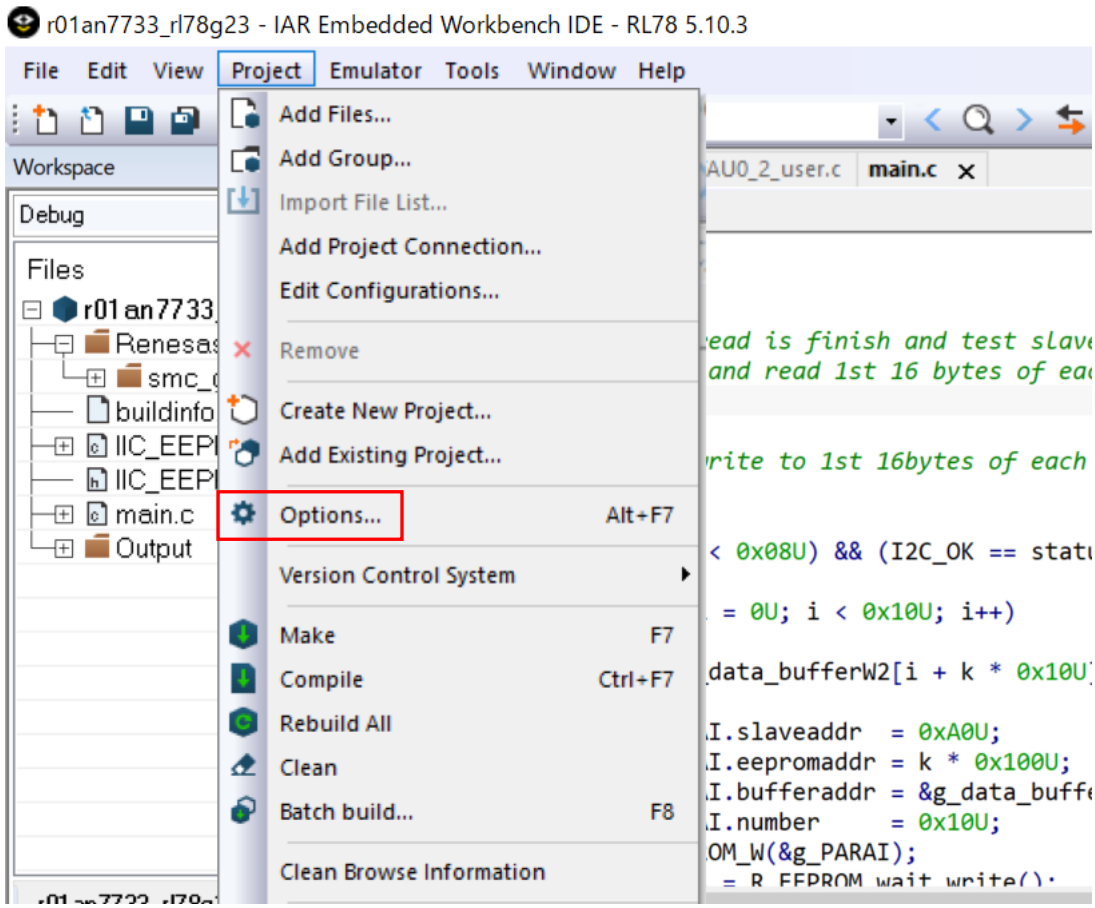
In the e² studio environment, please follow the following procedure. First, click "C/CC++ Project Settings".



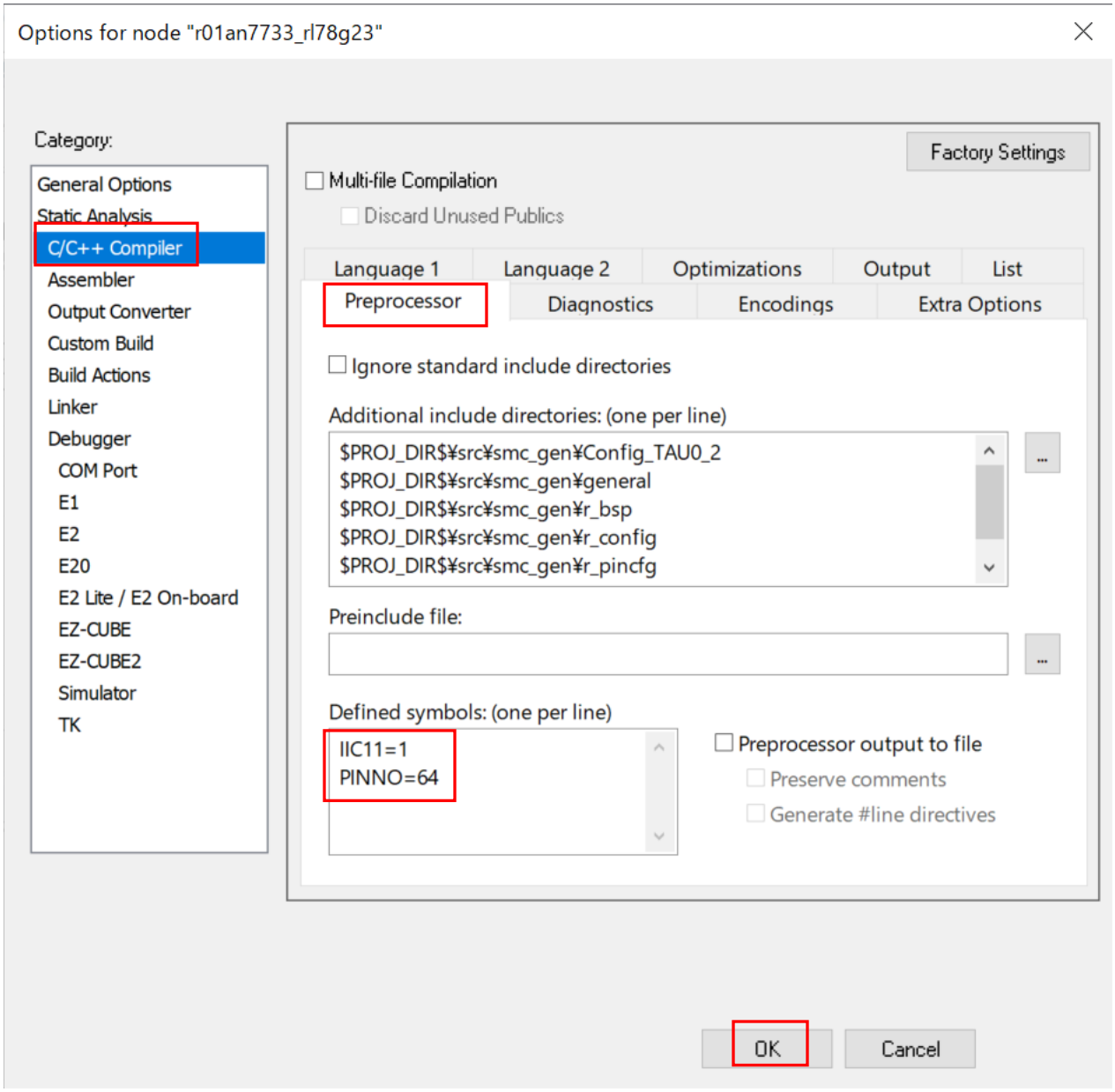
Next, choose "Setting" in "C/C++ Build", "Tool Settings" tab, and "Source" in "Compiler". Then write in the field of "Macro Defines" the definition macros which you used in Macro definition in CS+ environment. In the last place, click "Add" button and you can add the setting to the system.



In the IAR environment, please follow the following procedure. First, click "Options".



Next, select “Preprocessor” under “C/C++ Compiler”. Then, in the “Symbol Definition” field, set the definition set by the definition macro function of CS+. Finally, click the “OK” button to add the definition.



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/040C0H	01101110B	Disables the watchdog timer. (Stops counting after the release from the reset state.)
000C1H/040C1H	11111110B	LVD0 operating mode: reset mode Detection voltage: Rising edge 1.90V Falling edge 1.86V
000C2H/040C2H	11101000B	HS mode, HOCO: 32 MHz
000C3H/040C3H	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 3)	Description
SAUmEN	SAU0EN	Peripheral enable register
SPSm	SPS0	Serial clock selection register
SMRmn	SMR03	Serial mode register
SCRmn	SCR03	Serial communication register
SDRmn	SDR03	Serial data register
SIOr	SIO03	Serial data register (for data transmission/reception)
SSRmn	SSR03	Serial status register
SIRmn	SIR03	Serial flag clear trigger register
SSmL	SS0L	Serial channel start register
STmL	ST0L	Serial channel stop register
TRGONn	0b00001000	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	IICIF11	Interrupt request flag register
IICMKr	IICMK11	Interrupt request mask register
PM_IICr	PM5	IICr port mode register (specifying SDA signal)
PM_SDAr	PM5_bit.no0	SDA port mode register bits
PM_SCLr	PM3_bit.no0	SCL port mode register bits
POM_IICr	POM5	Port output mode register
P_IICr	P3	IICr port
P_SDAr	P5_bit.no0	SDA port
P_SCLr	P3_bit.no0	SCL port
SDAINMODE	0b00000001	Bits for specifying port mode register input
SDAOUTMODE	0b11111100	Bits for port mode register output
SDASCLON	(uint8_t)(~SDAOUTMODE)	Port setting bits
IICPR0r	IICPR011	Interrupt priority setting registers
IICPR1r	IICPR111	

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.
INTTM02	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.

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(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(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(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(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	static void __near 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] INTTM02

Synopsis	Process 100-μs interval timer completion interrupt.	
Header	r_cg_userdefine.h	
Declaration	static void _near INTTM02 (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. The function sets the initial value of the EEPROM to be used to 16 kbits.
Arguments	None -
Return value	None
Remarks	

5.7 Flowcharts

5.7.1 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.

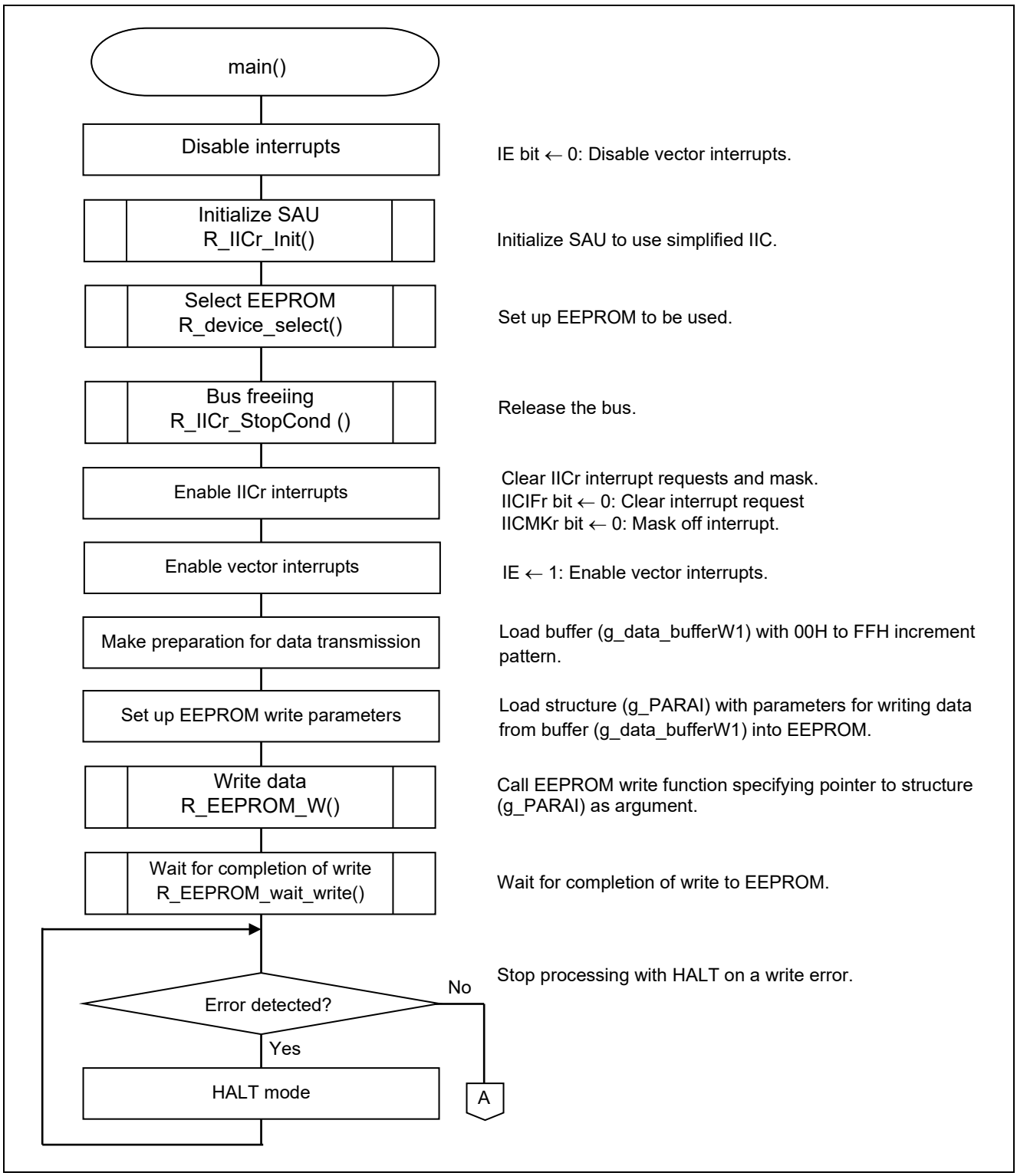
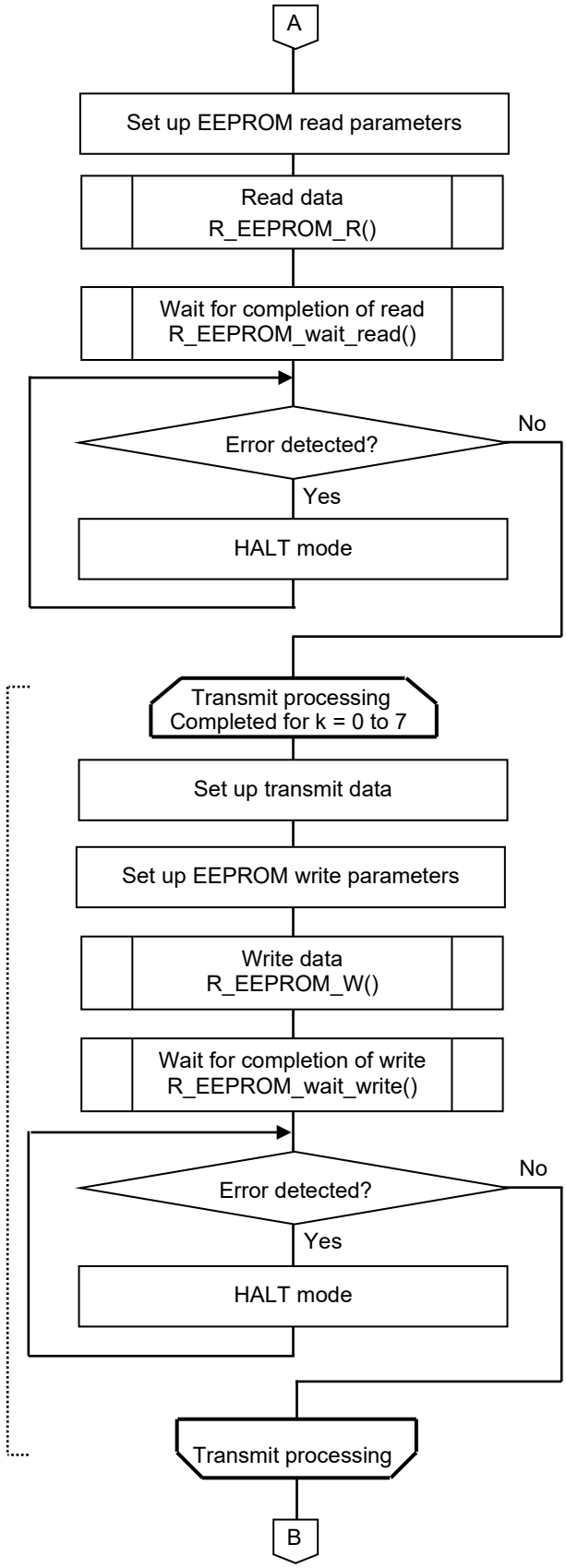


Figure 5.7 Main Function (1/3)



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.

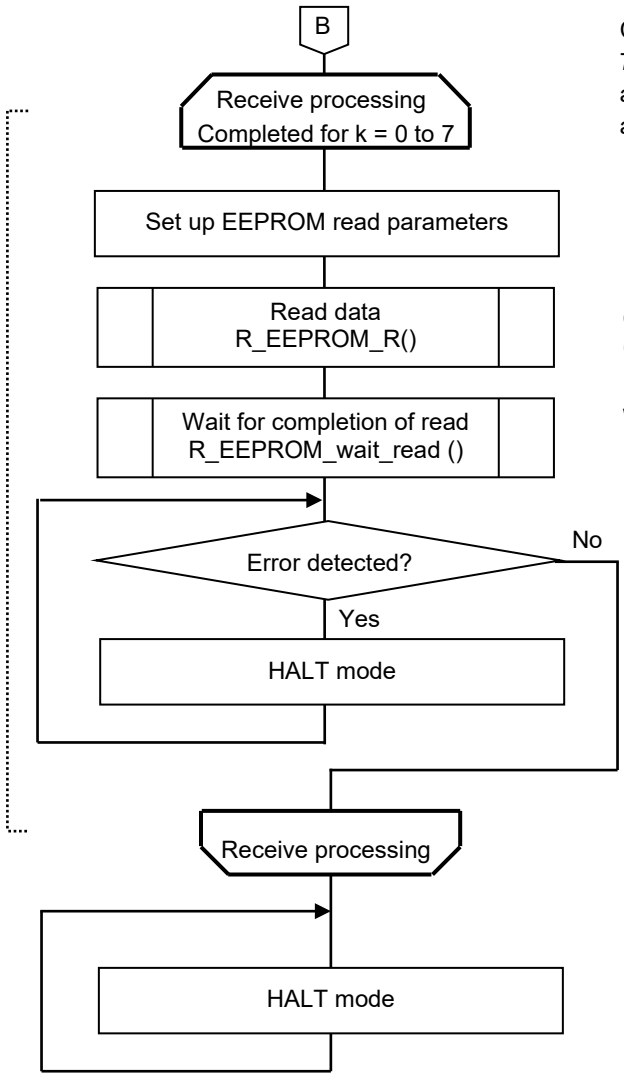
Call EEPROM write function specifying pointer to structure (g_PARAI).

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)

5.7.2 EEPROM Selection

Figure 5.10 shows the flowchart for selecting the EEPROM.

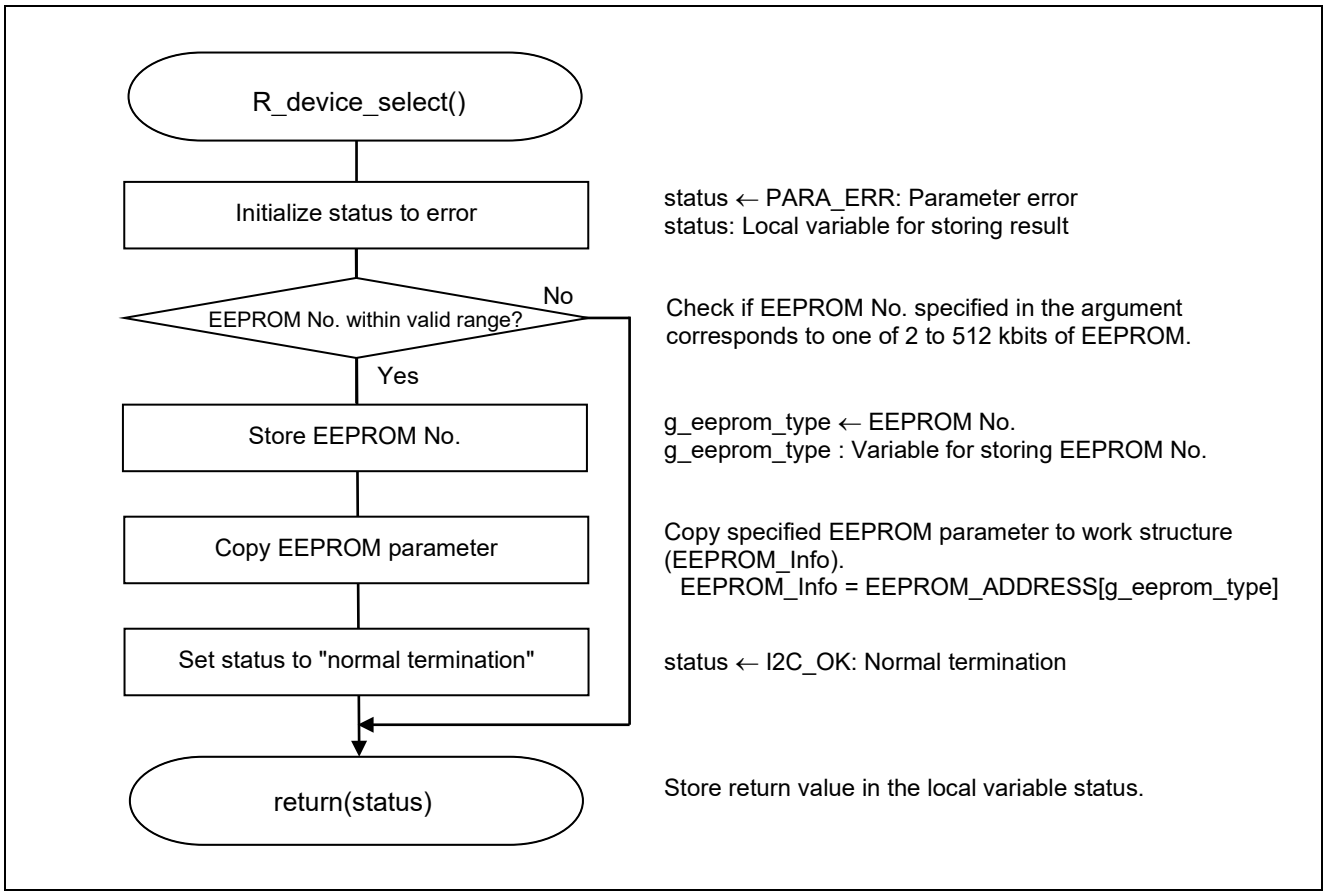


Figure5.10 EEPROM Selection

5.7.3 Bus freeing Processing

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

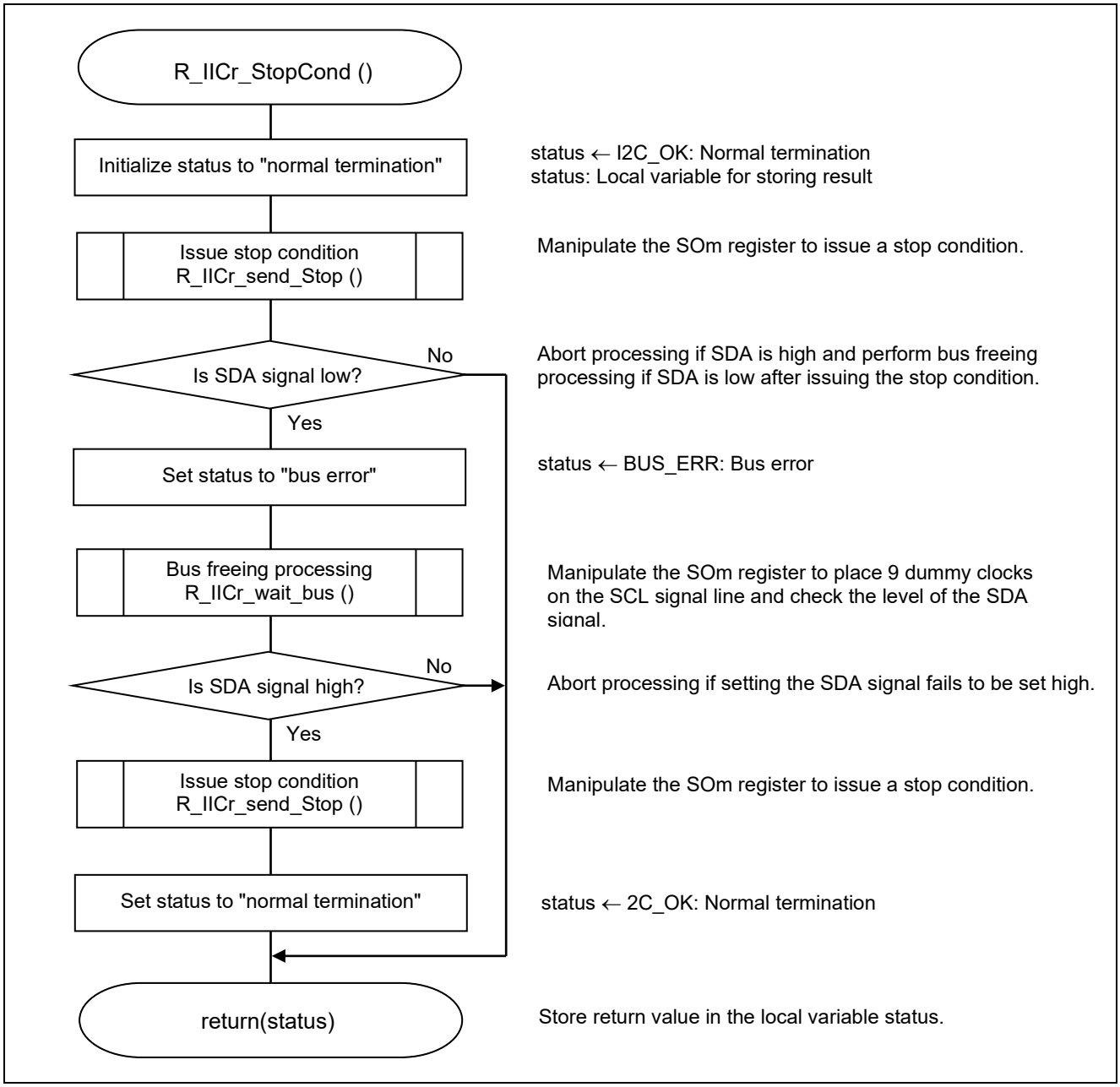


Figure 5.11 Bus freeing Processing

5.7.4 Stop Condition Generation Function

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

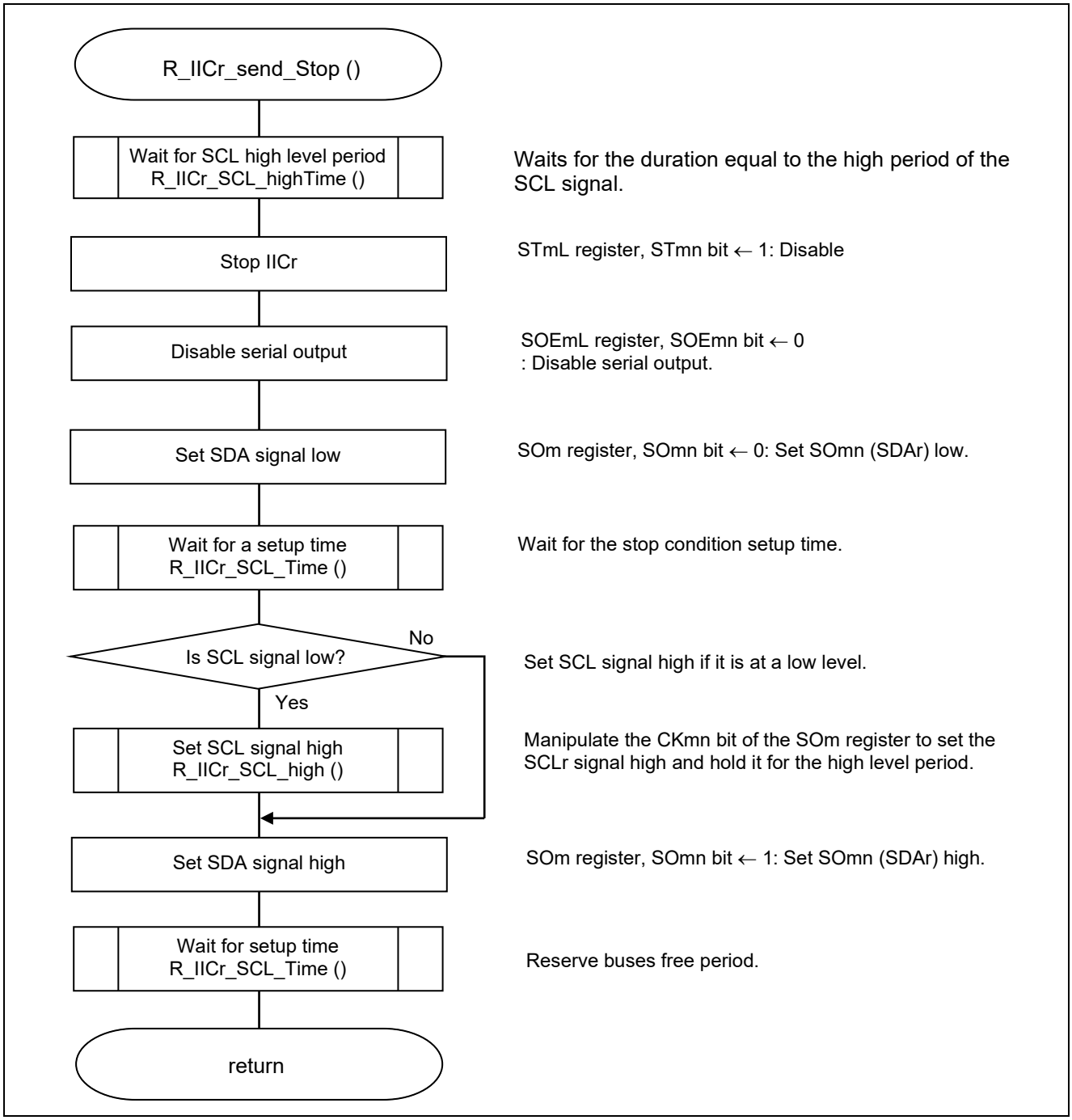


Figure 5.12 Stop Condition Generation Function

5.7.5 Bus Freeing Function

Figure 5.13 shows the flowchart for the bus freeing function.

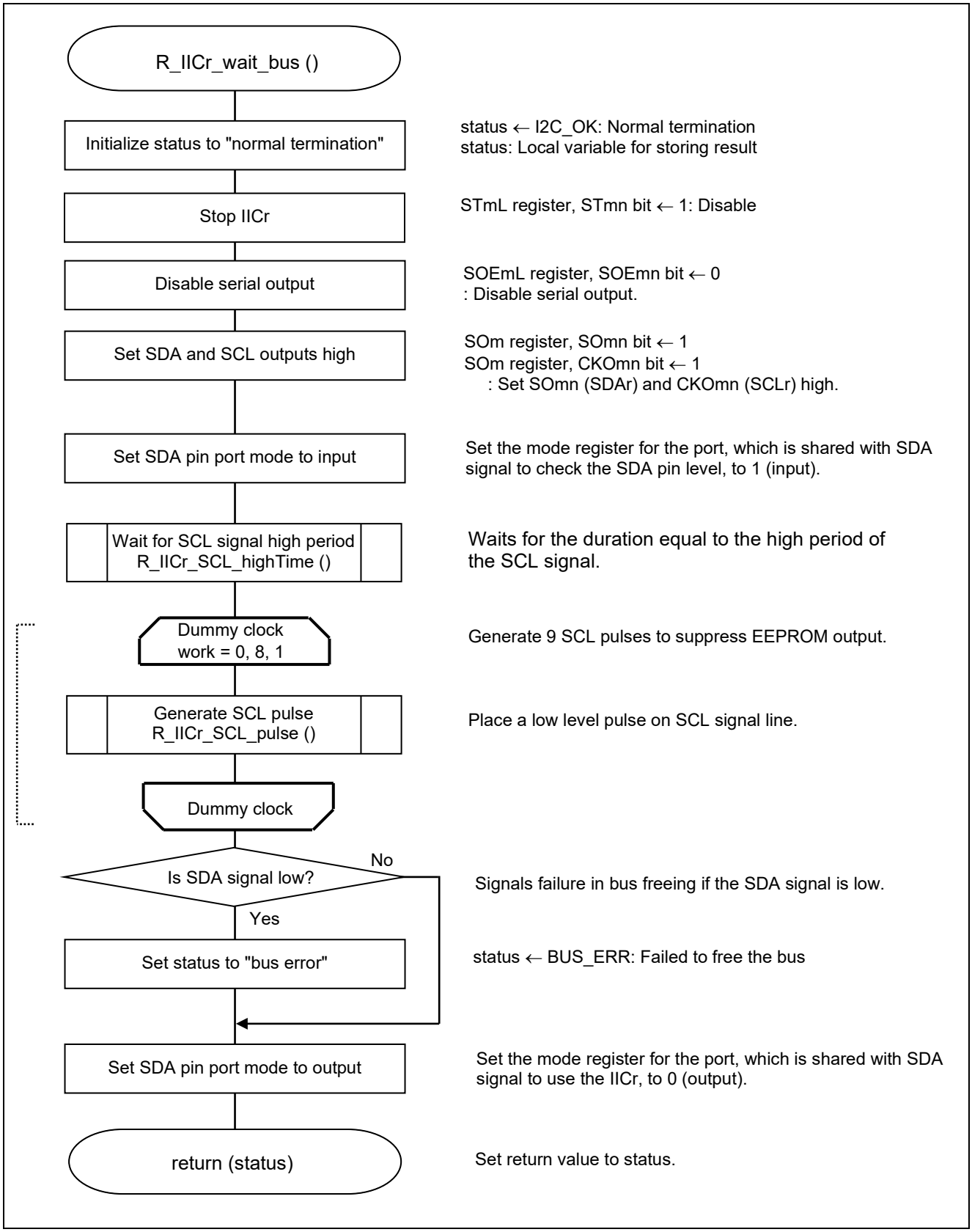
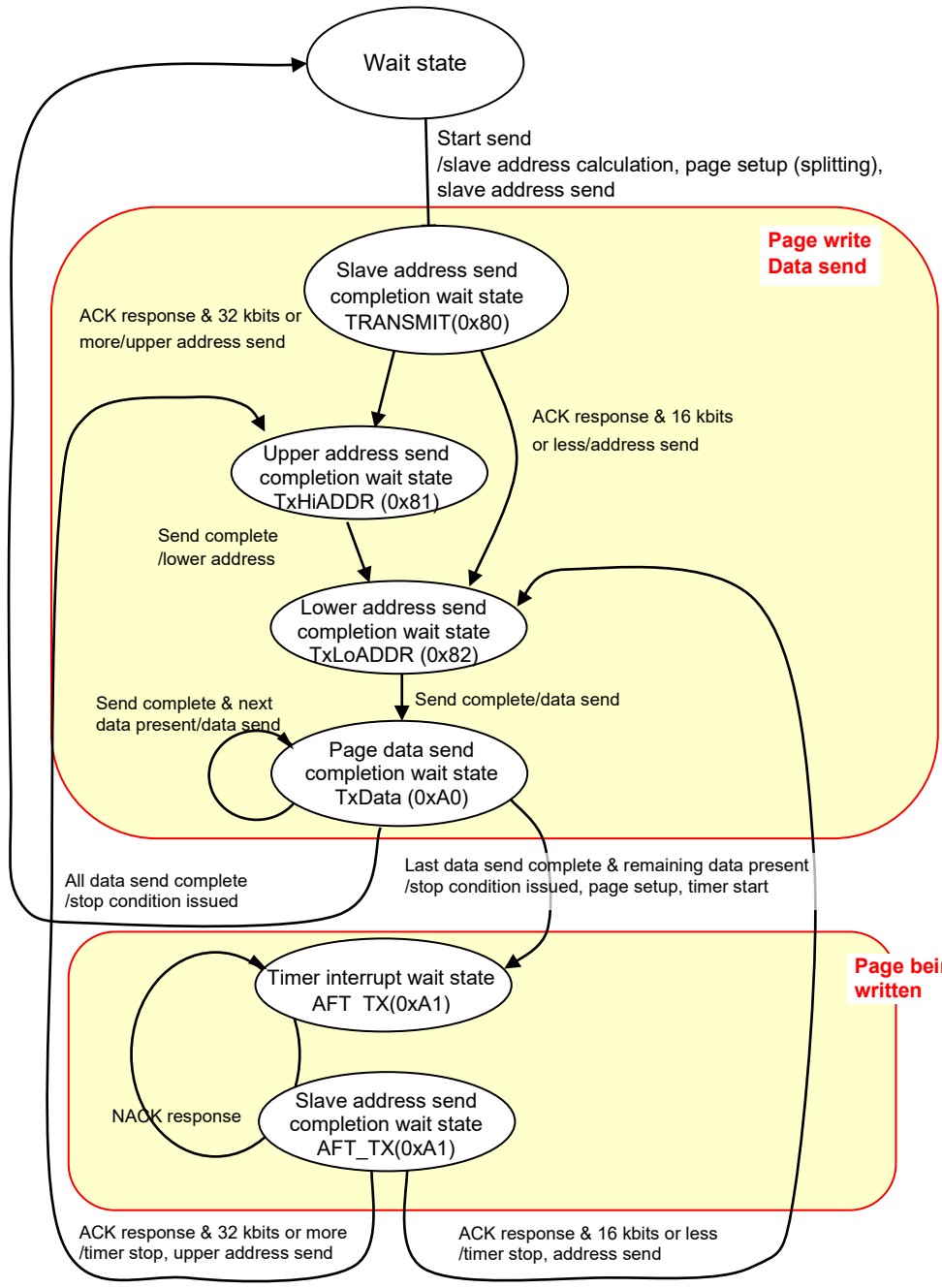


Figure 5.13 Bus Freeing Function

5.7.6 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

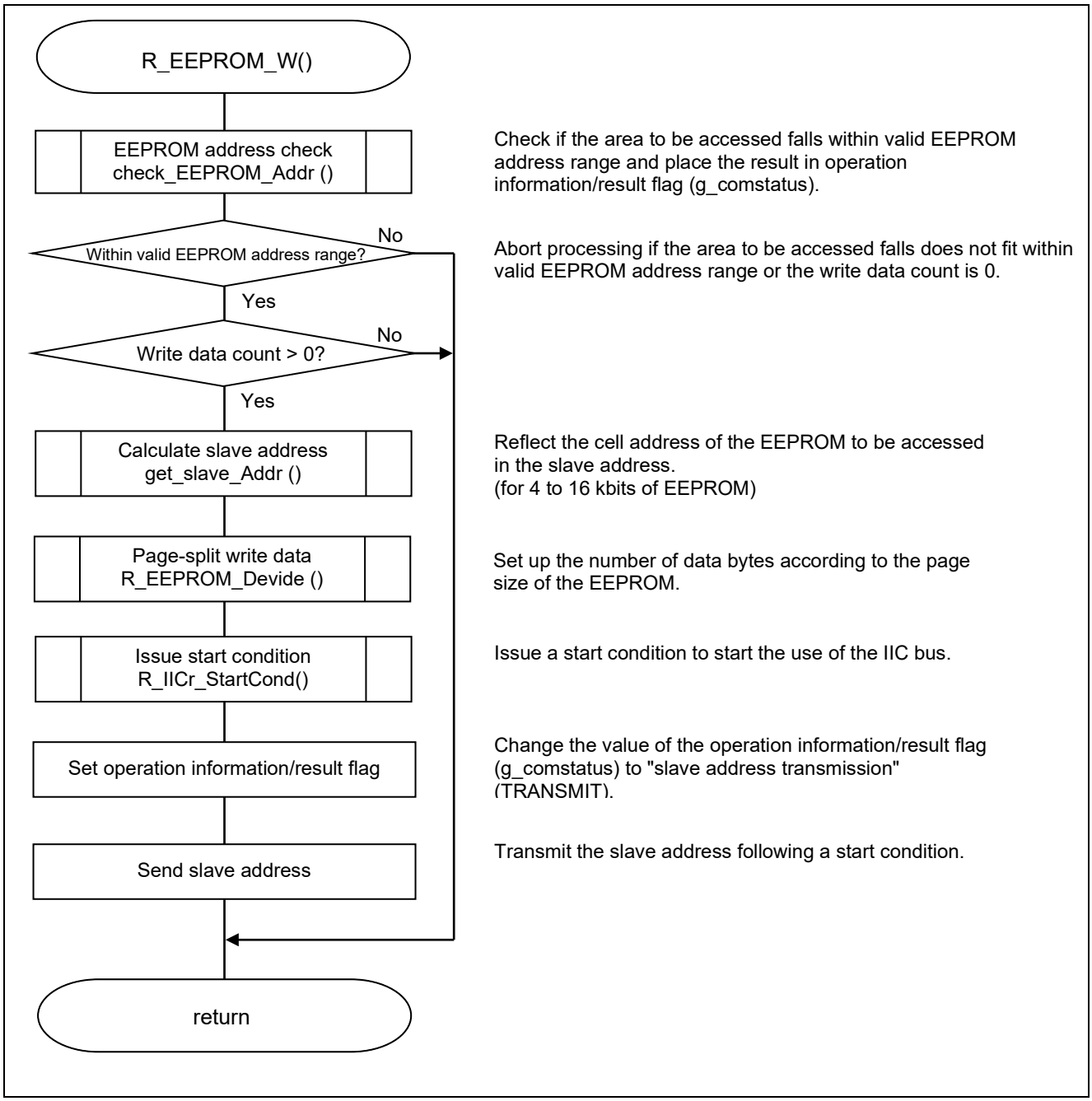


Figure 5.15 EEPROM Write Processing

5.7.7 EEPROM Address Check Processing

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

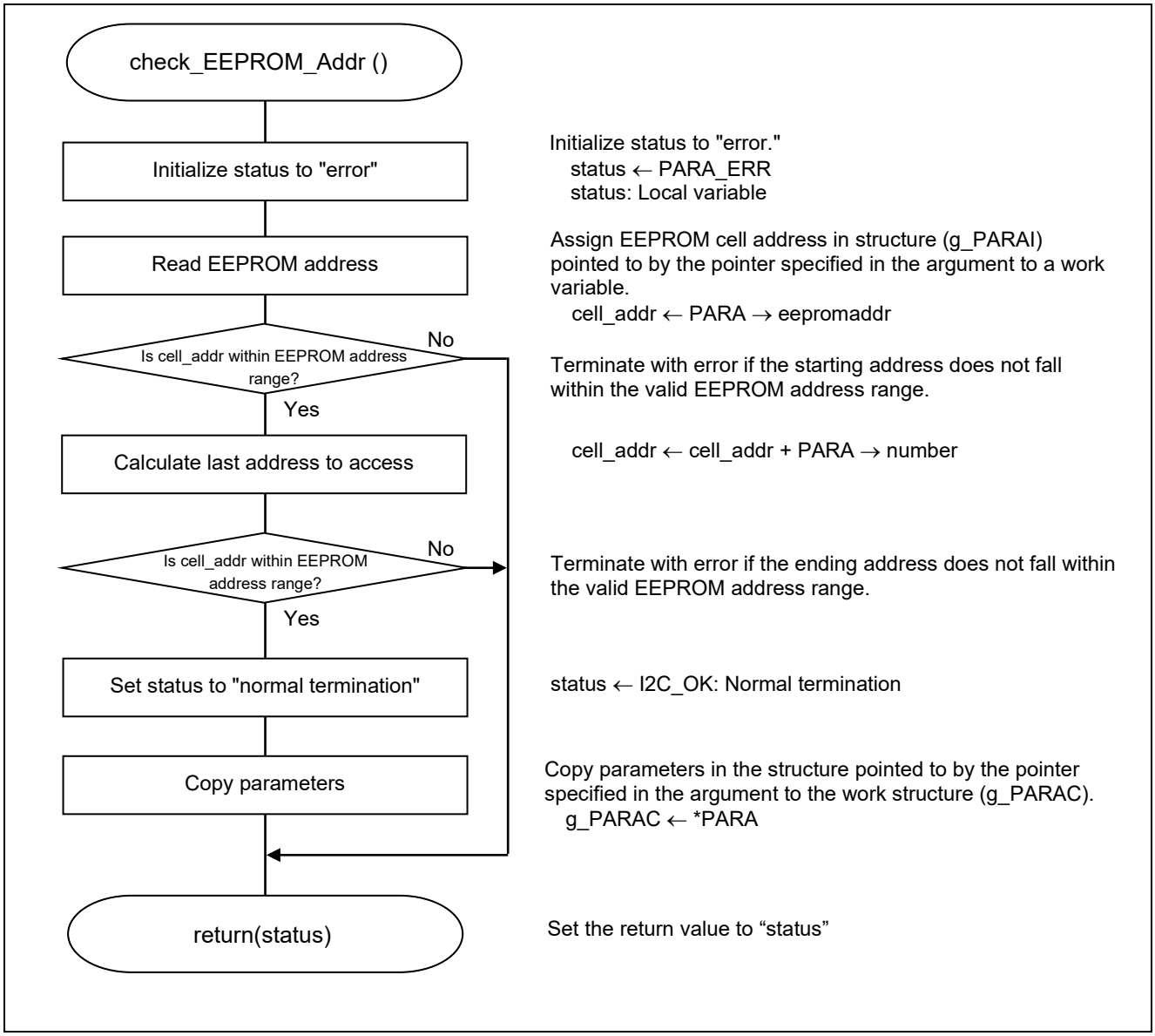
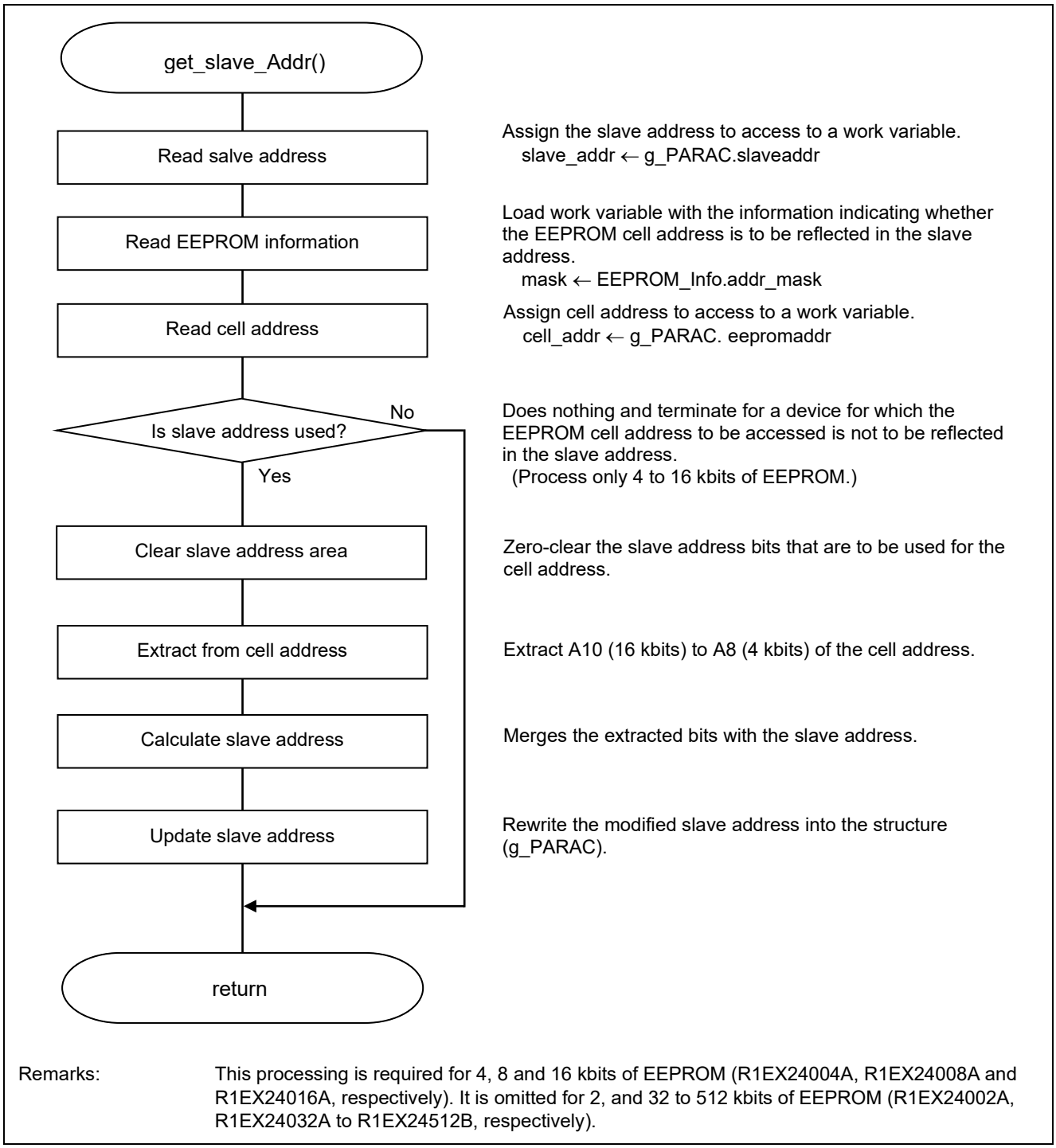


Figure 5.16 EEPROM Address Check Processing

5.7.8 Slave Address Calculation

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



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

5.7.9 Write Data Page Split Processing

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

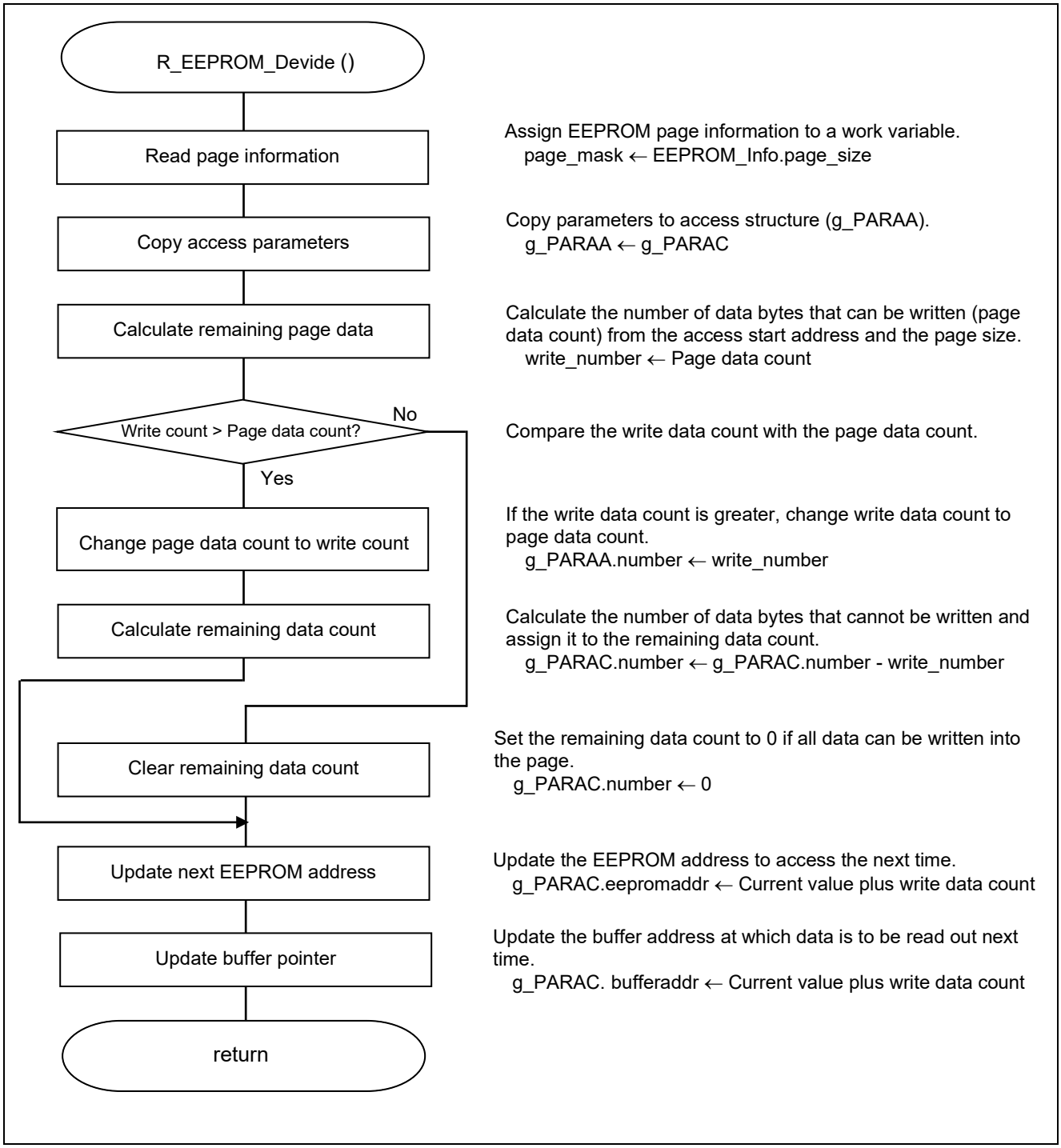


Figure 5.18 Write Data Page Split Processing

5.7.10 Start Condition Issuing Processing

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

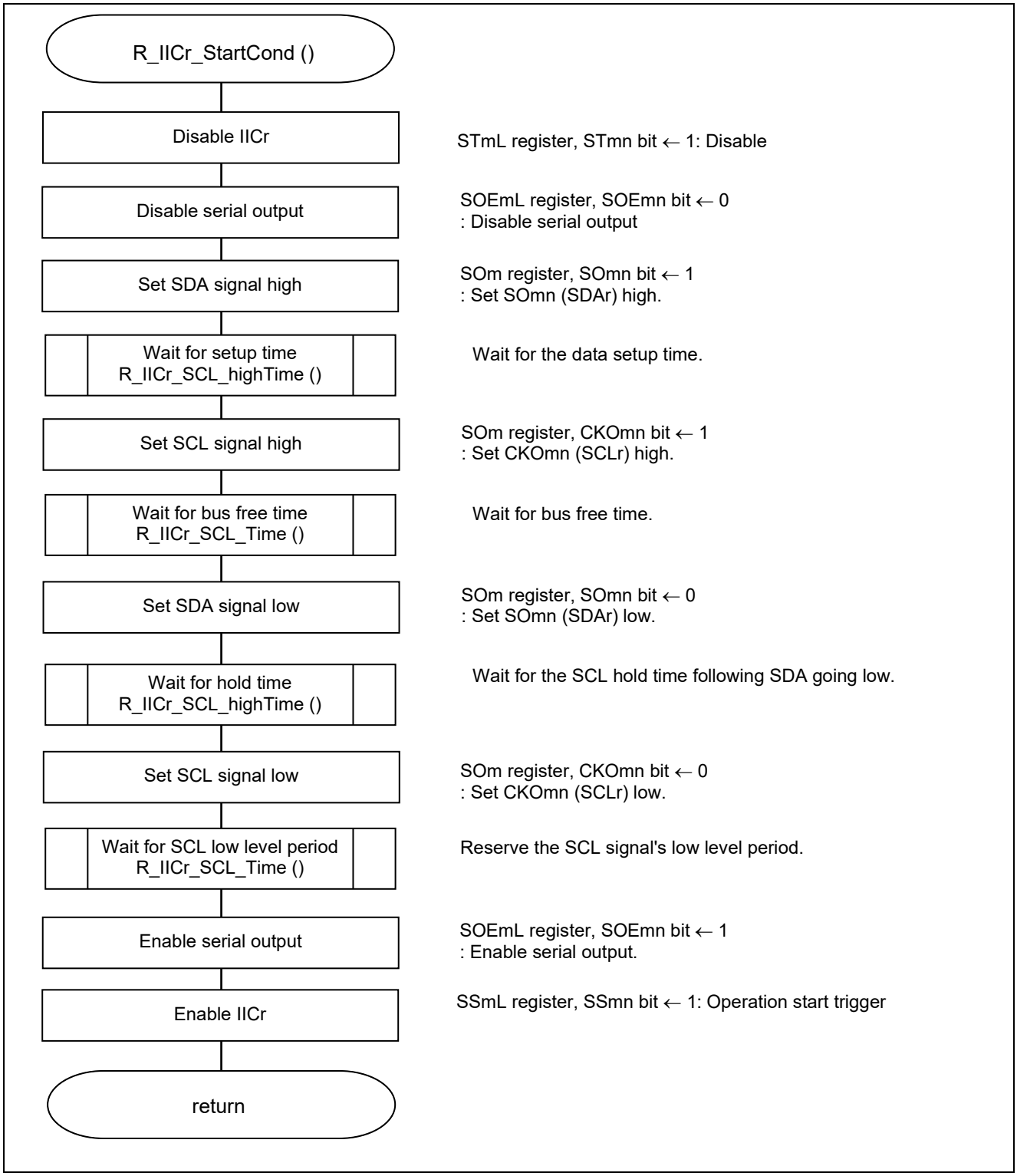


Figure 5.19 Start Condition Issuing Processing

5.7.11 EEPROM Write Completion Wait Processing

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

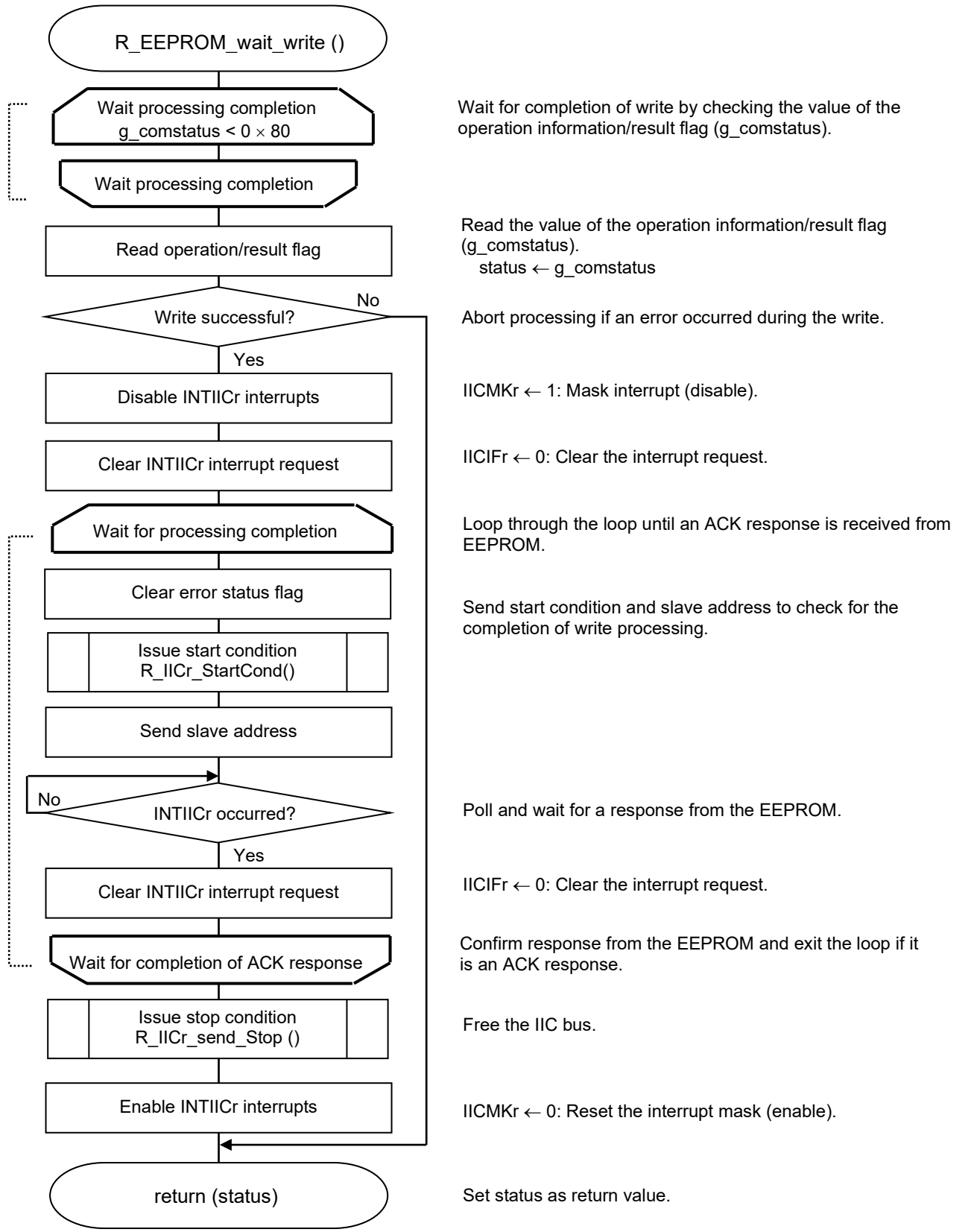
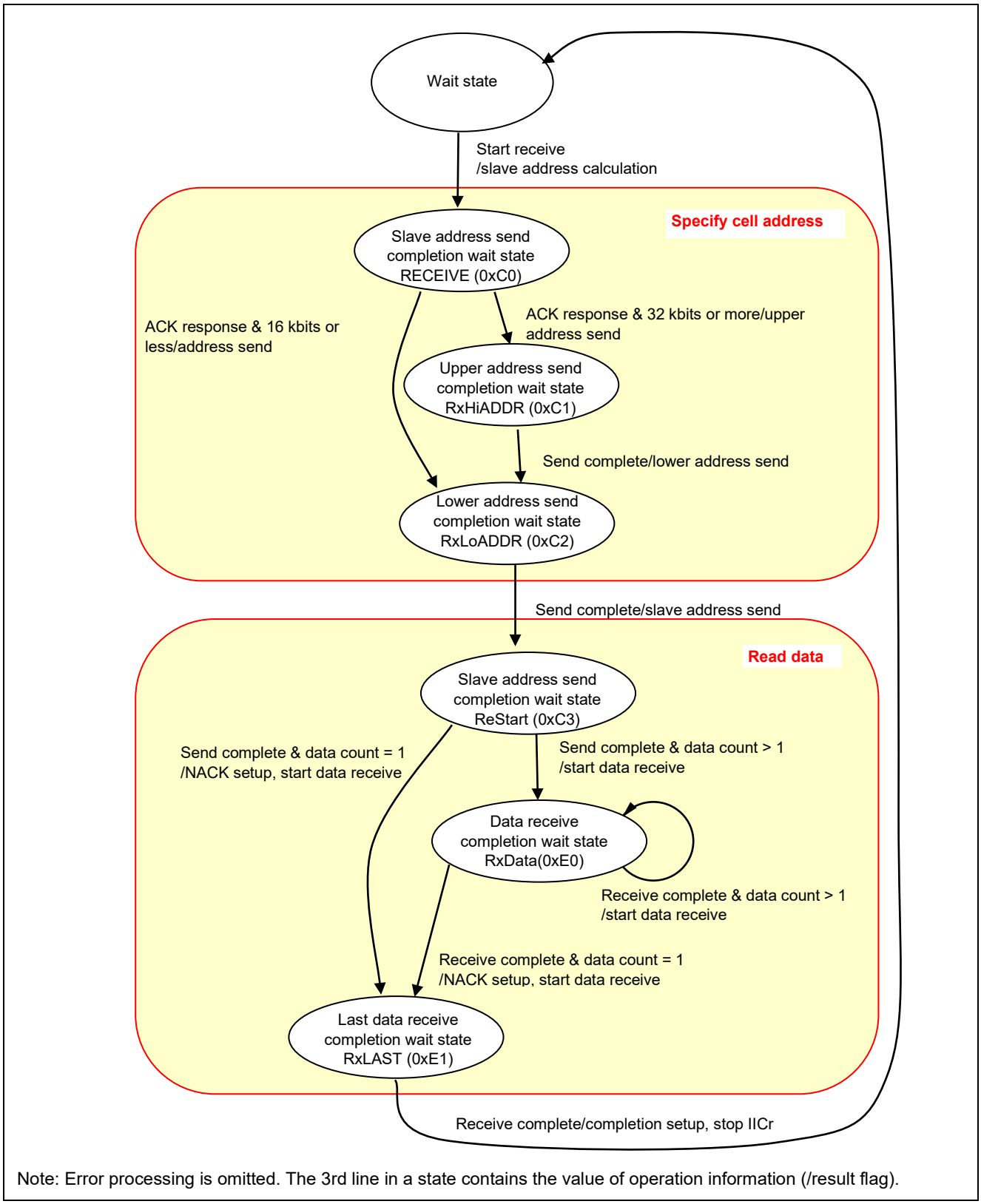


Figure 5.20 EEPROM Write Completion Wait Processing

5.7.12 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

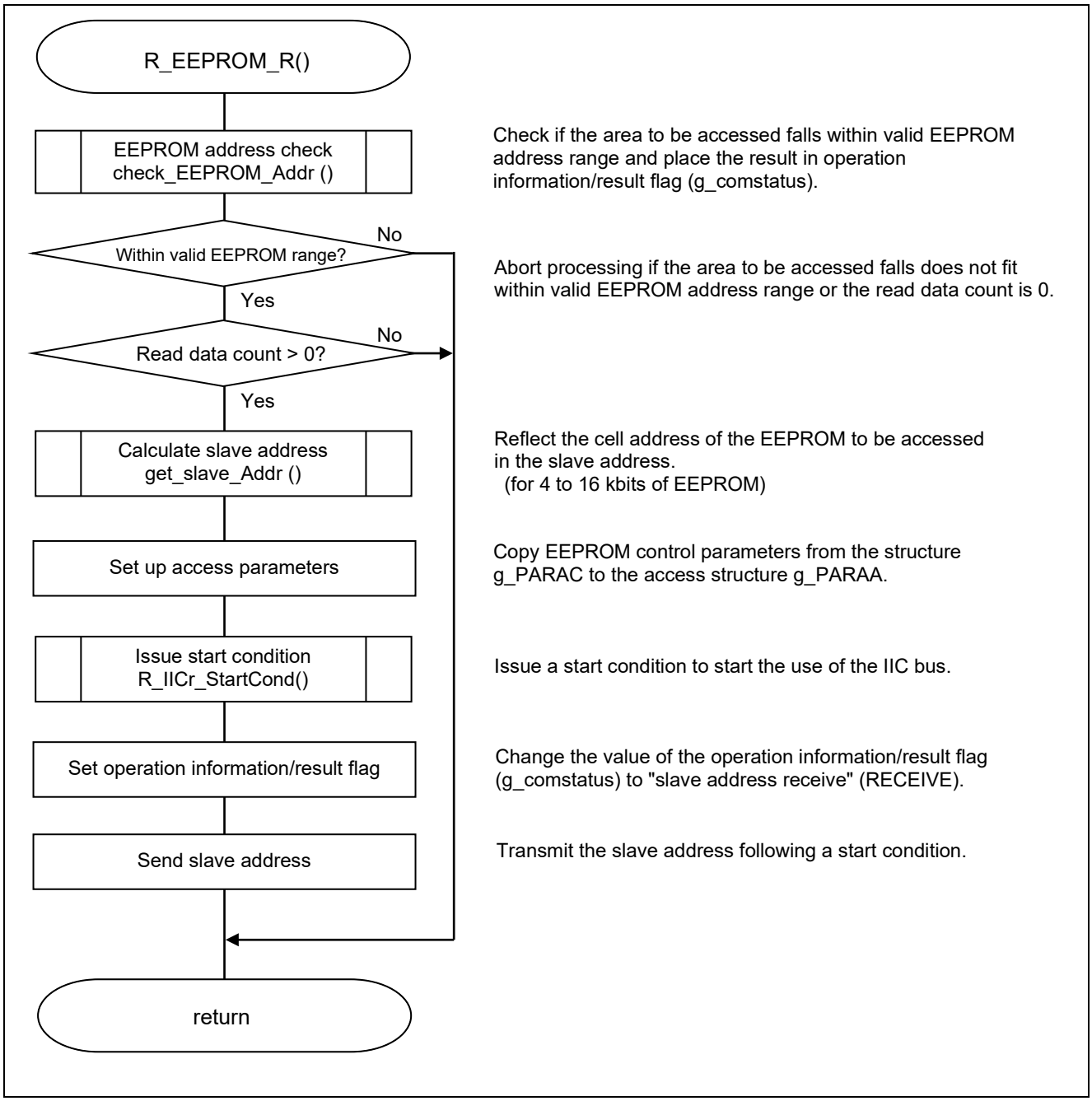


Figure 5.22 EEPROM Read Processing

5.7.13 EEPROM Read Completion Wait Processing

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

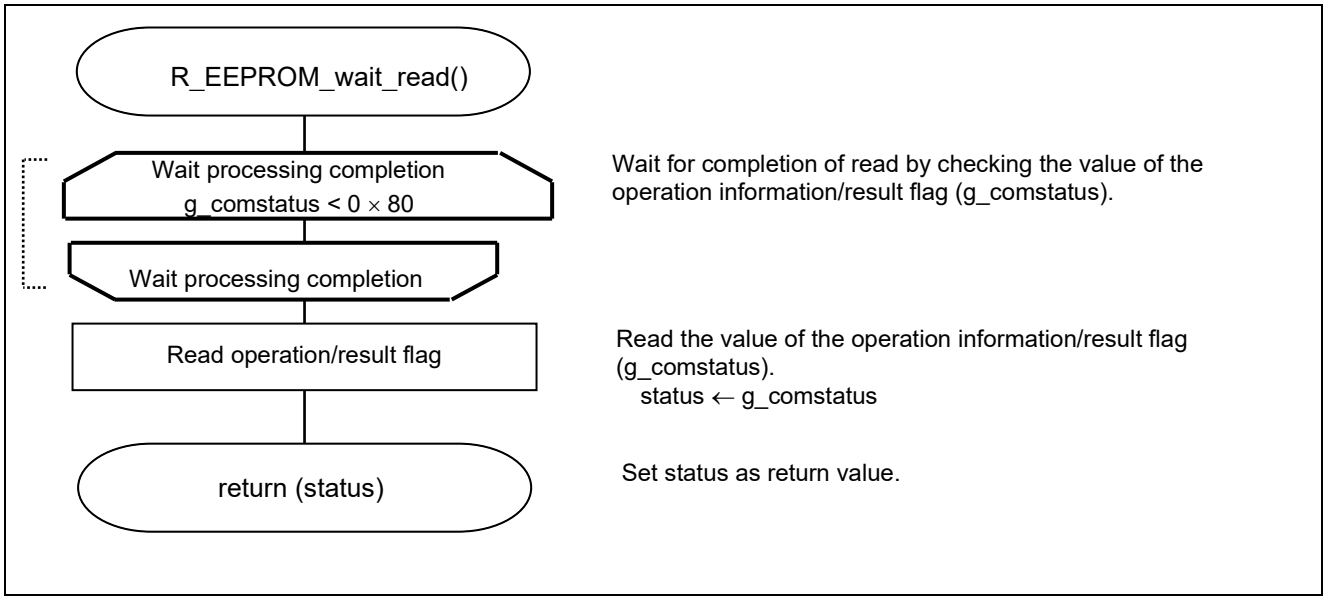


Figure 5.23 EEPROM Read Completion Wait Processing

5.7.14 Slave Address Transmission Completion Processing

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

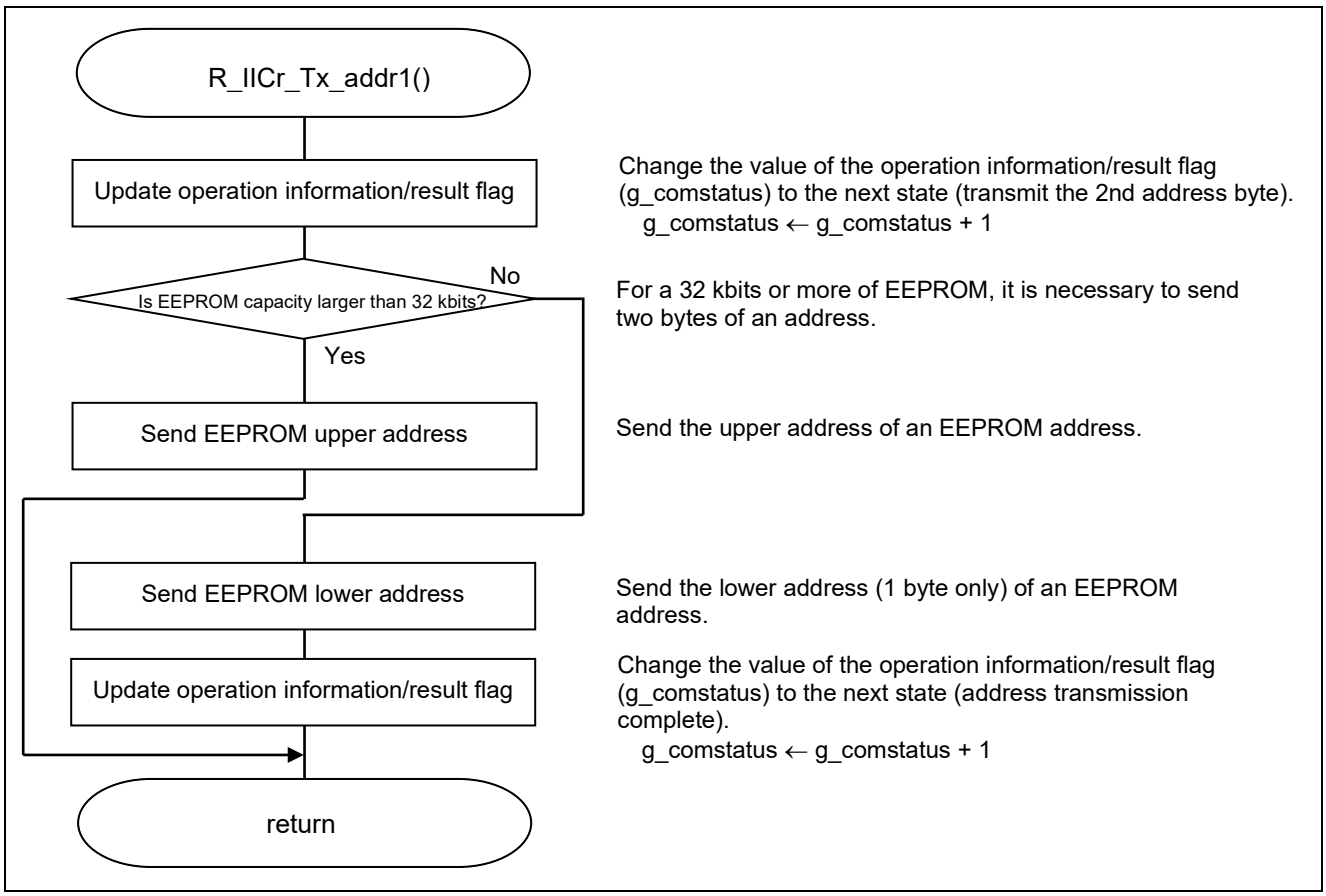


Figure 5.24 Slave Address Transmission Completion Processing

5.7.15 Upper Address Transmission Completion Processing

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

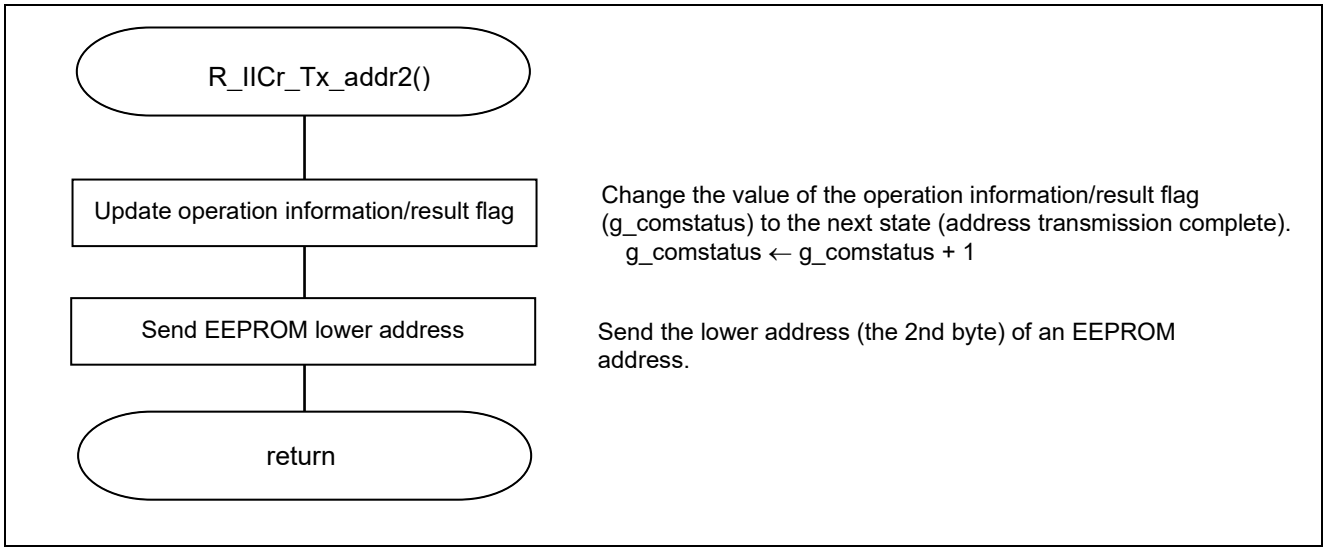


Figure 5.25 Upper Address Transmission Completion Processing

5.7.16 Restart Processing

Figure 5.26 shows the flowchart for the restart processing.

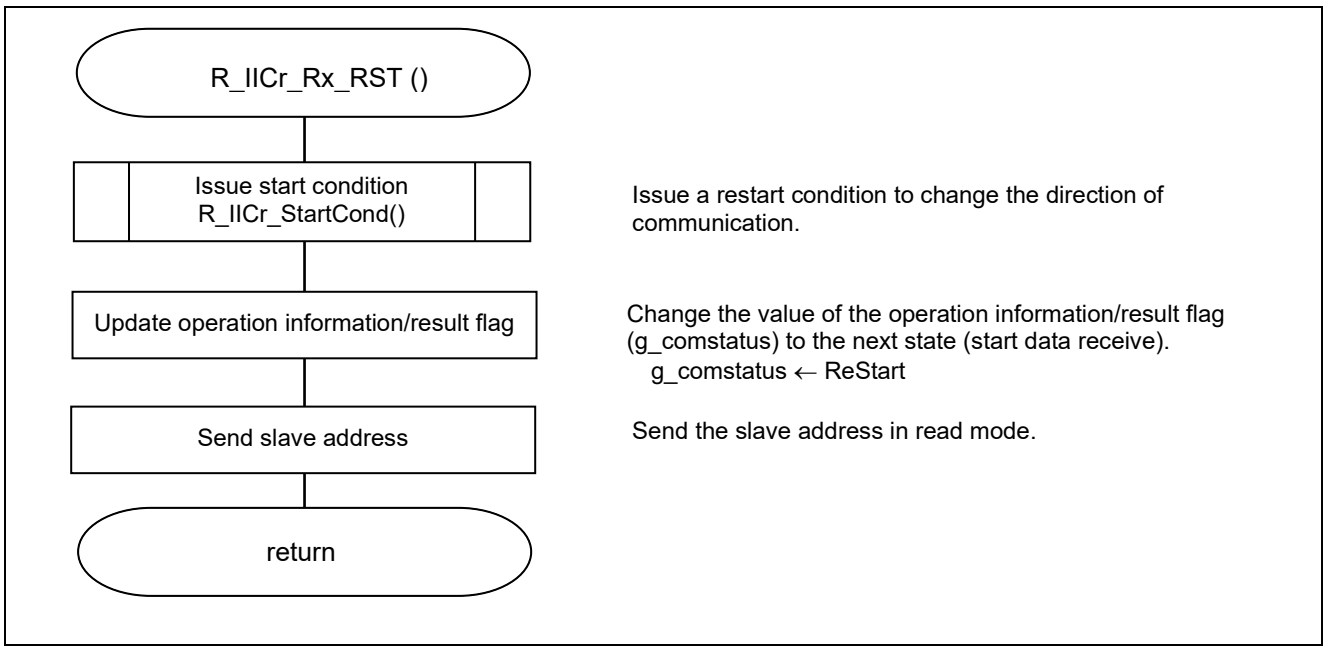


Figure 5.26 Restart Processing

5.7.17 Data Reception Start Processing

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

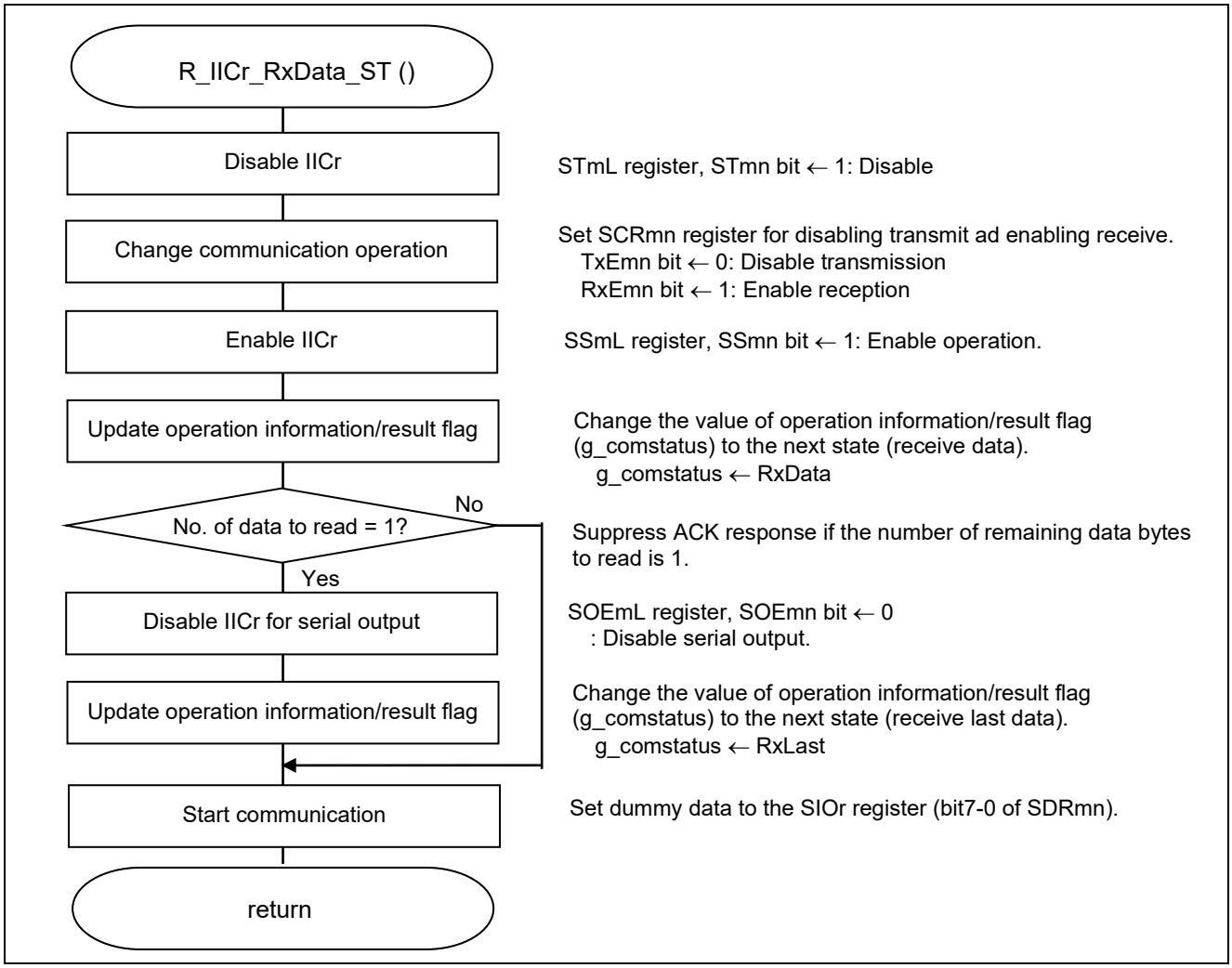


Figure 5.27 Data Reception Start Processing

5.7.18 Data Receive Processing

Figure 5.28 shows the flowchart for the data receive processing.

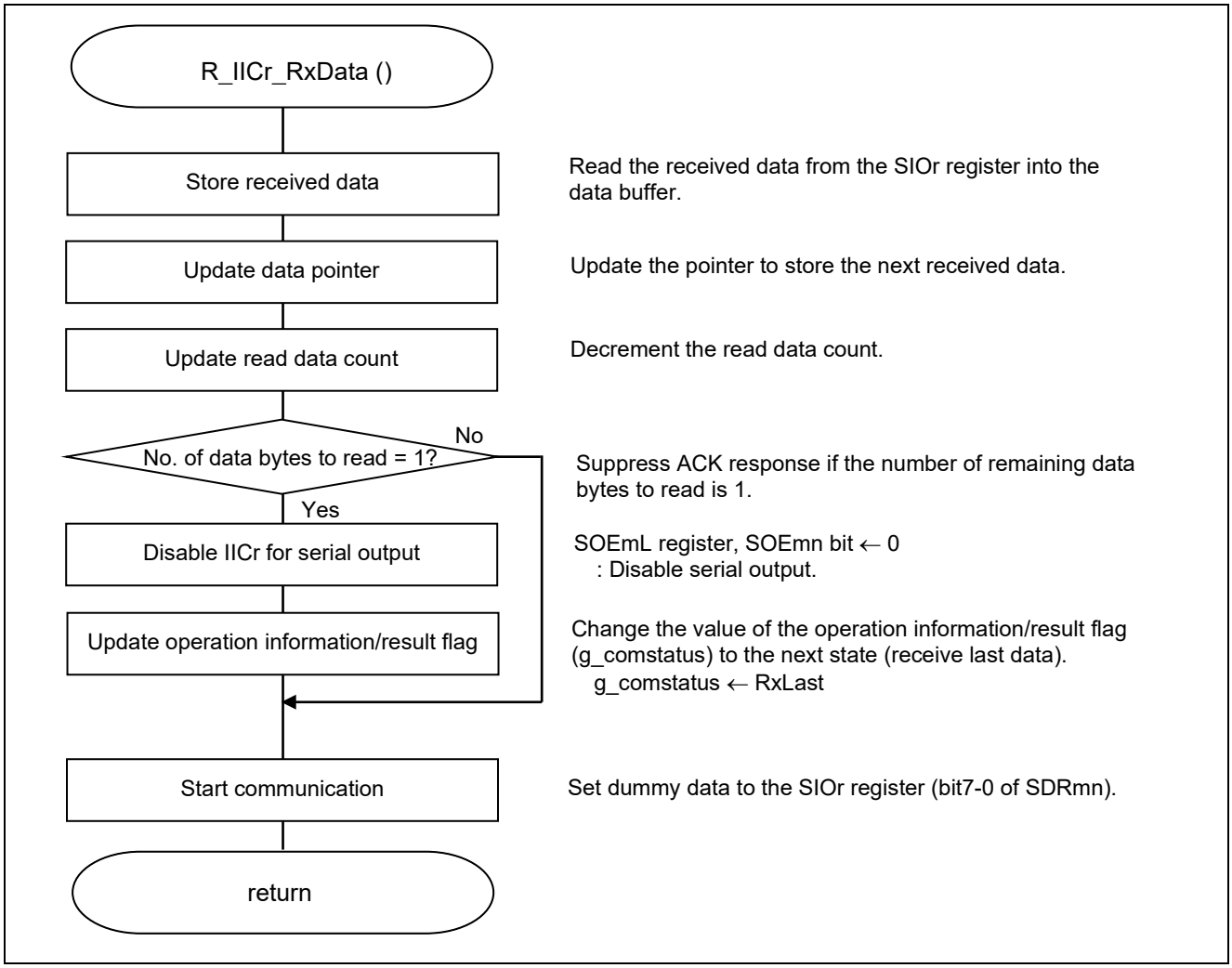


Figure 5.28 Data Receive Processing

5.7.19 Last Data Receive Processing

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

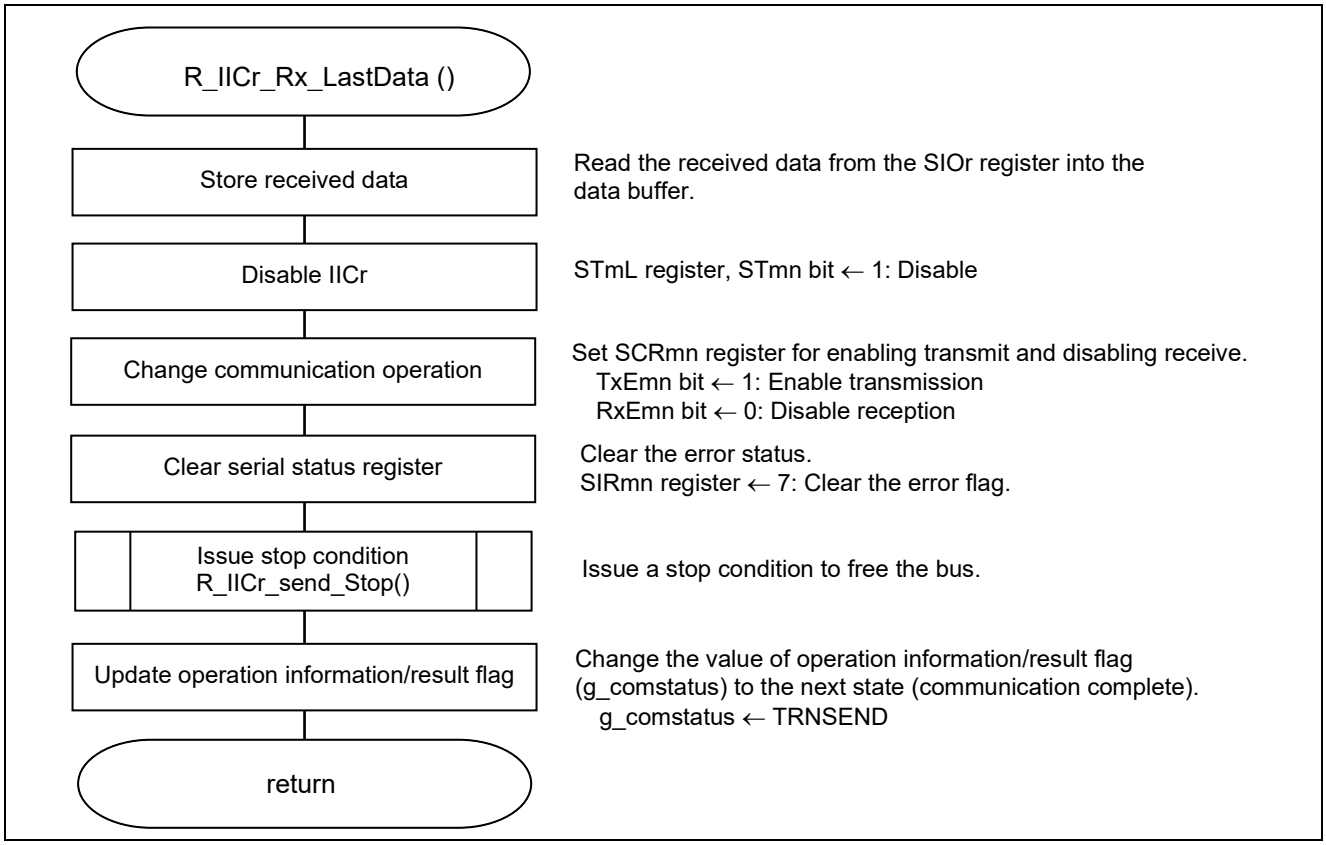


Figure 5.29 Last Data Receive Processing

5.7.20 Data Transmission Start Processing

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

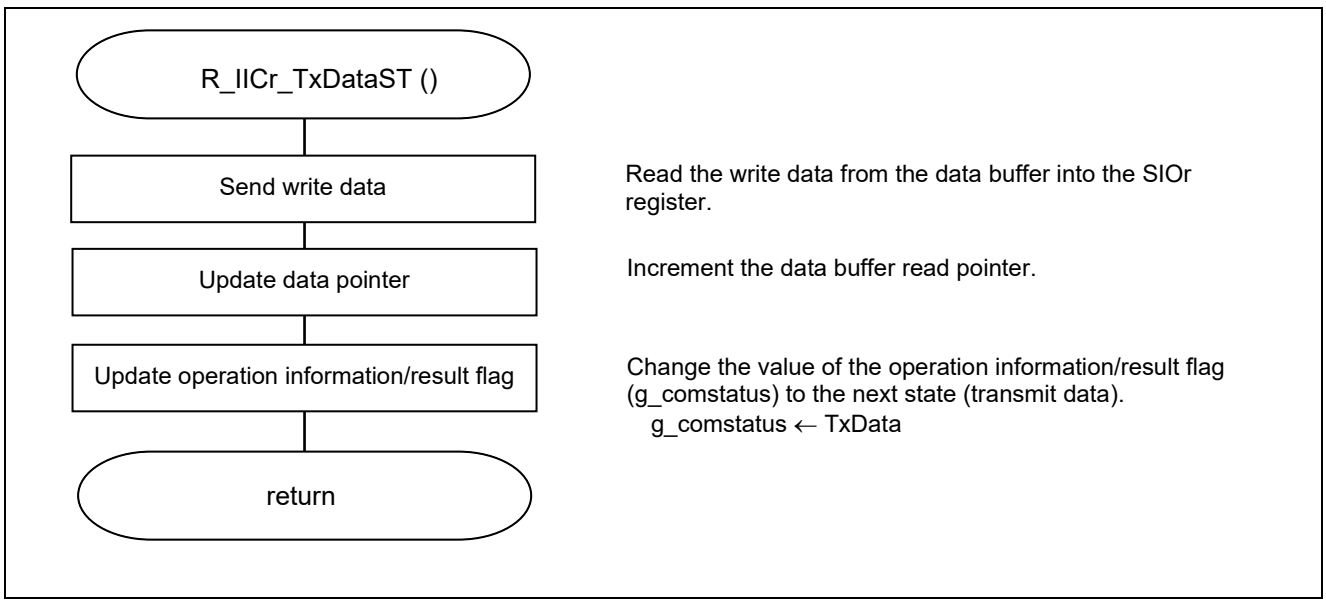


Figure 5.30 Data Transmission Start Processing

5.7.21 Data Transmit Processing

Figure 5.31 shows the flowchart for the data transmit processing.

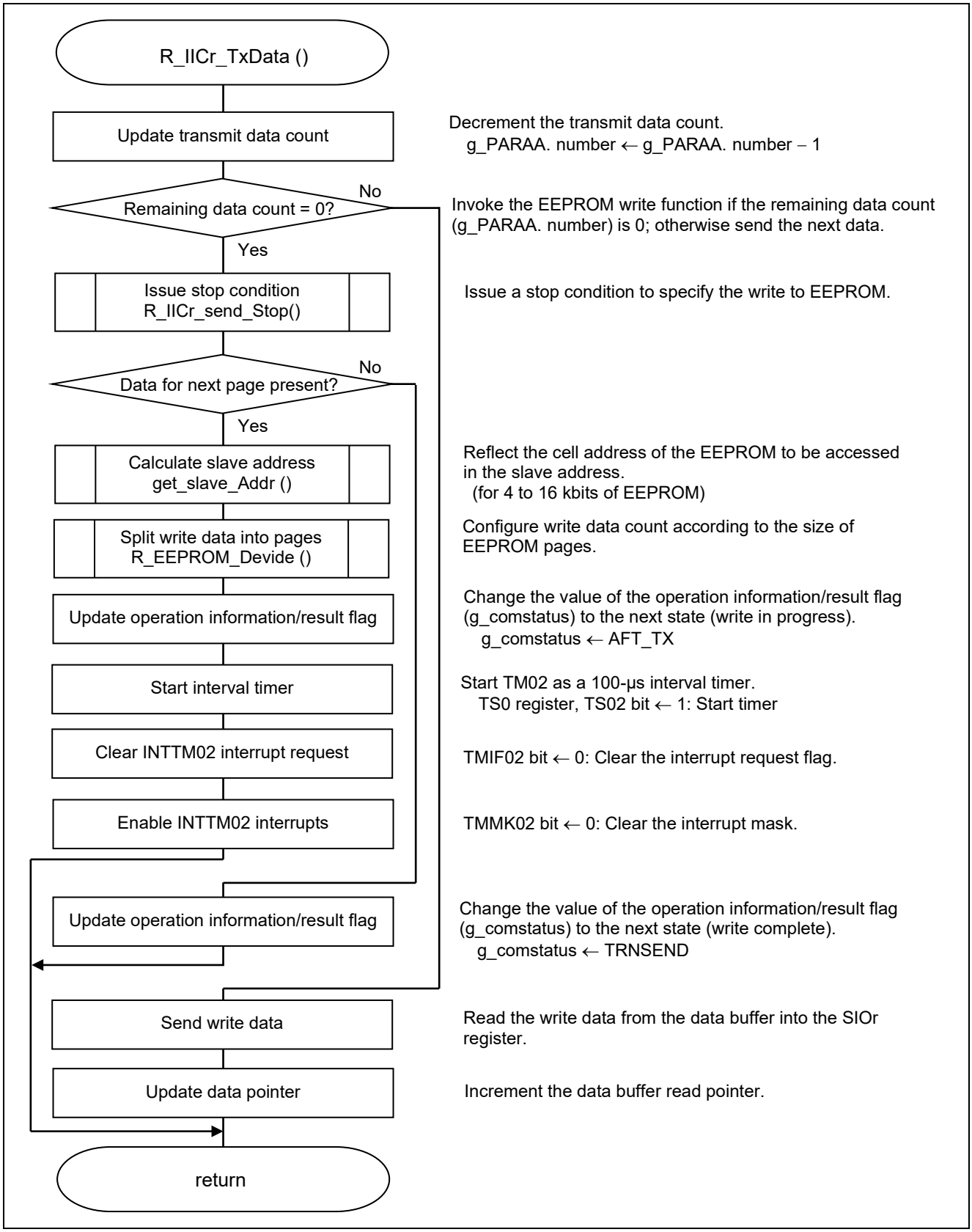


Figure 5.31 Data Transmit Processing

5.7.22 Next Page Write Start Processing

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

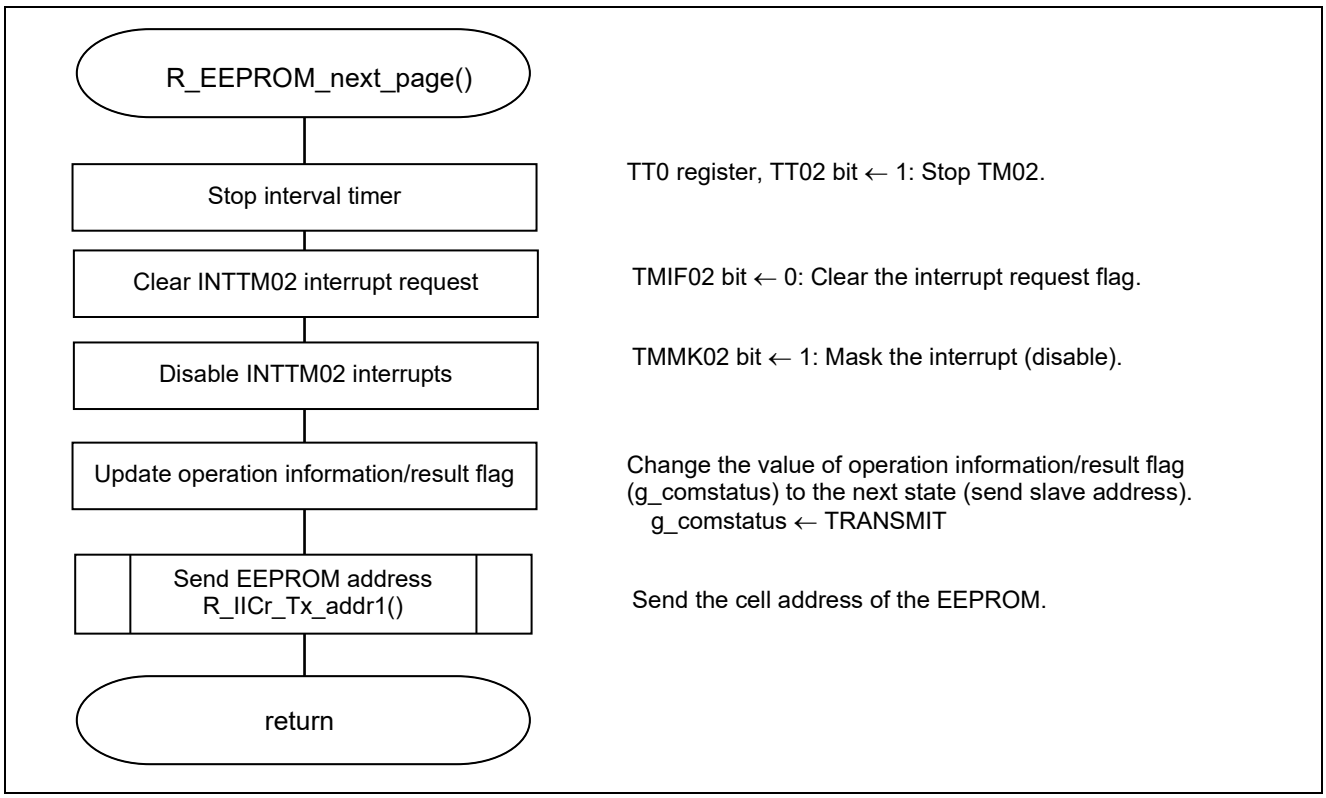


Figure 5.32 Next Page Write Start Processing

5.7.23 SCL Dummy Clock Output Processing

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

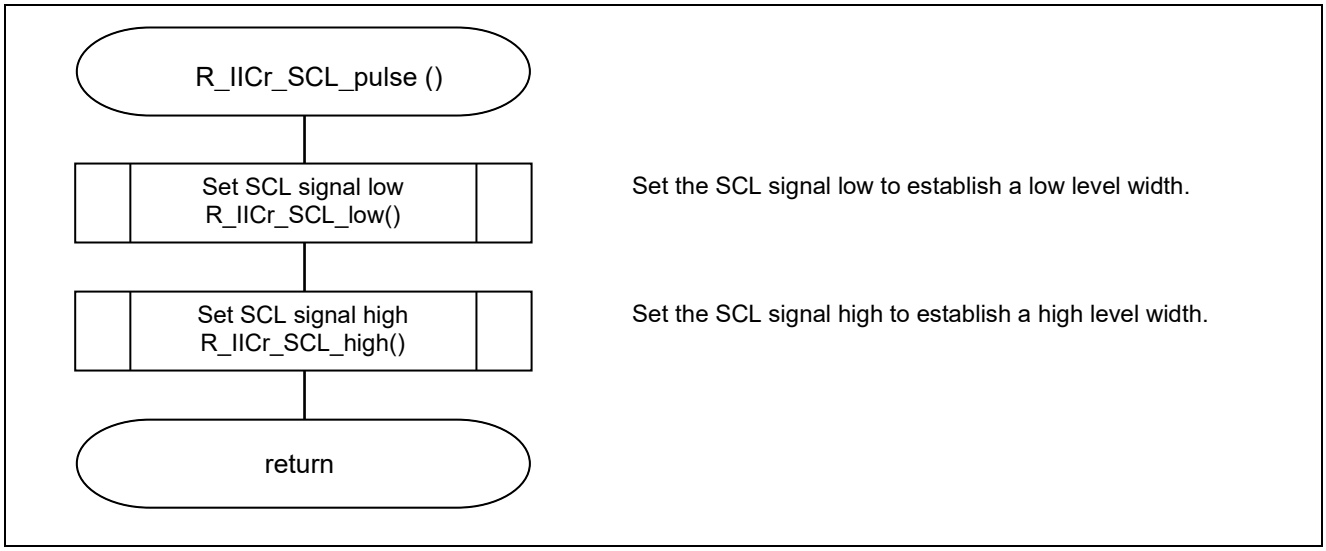


Figure 5.33 SCL Dummy Clock Output Processing

5.7.24 SCL Set High Processing

Figure 5.34 shows the flowchart for setting SCL high.

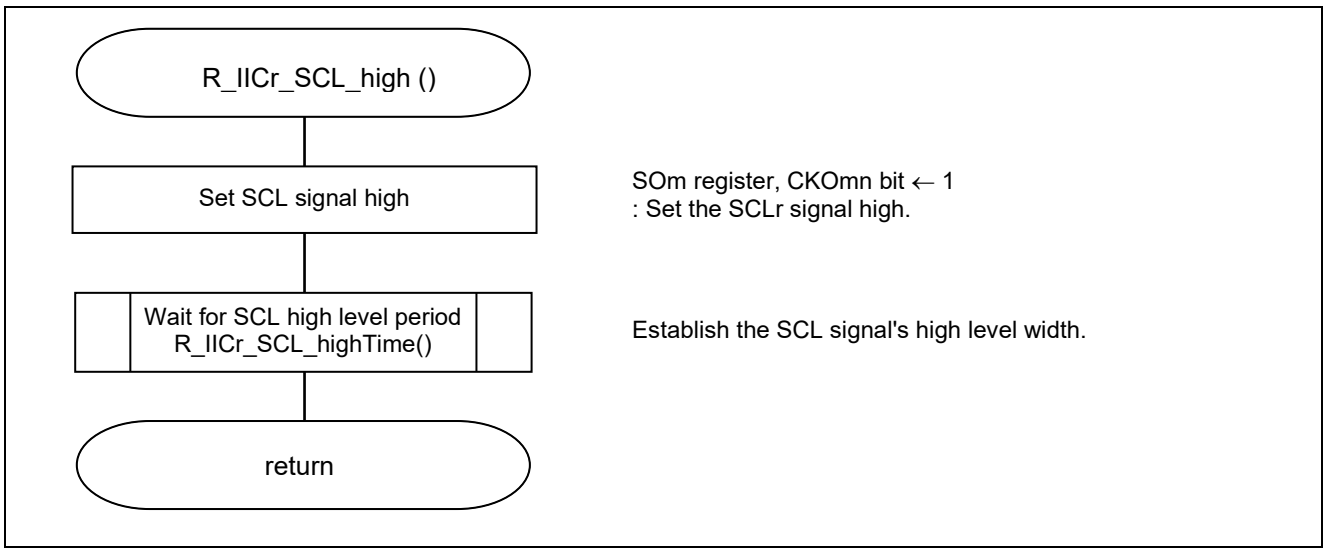


Figure 5.34 SCL Set High Processing

5.7.25 SCL Set Low Processing

Figure 5.35 shows the flowchart for setting SCL low.

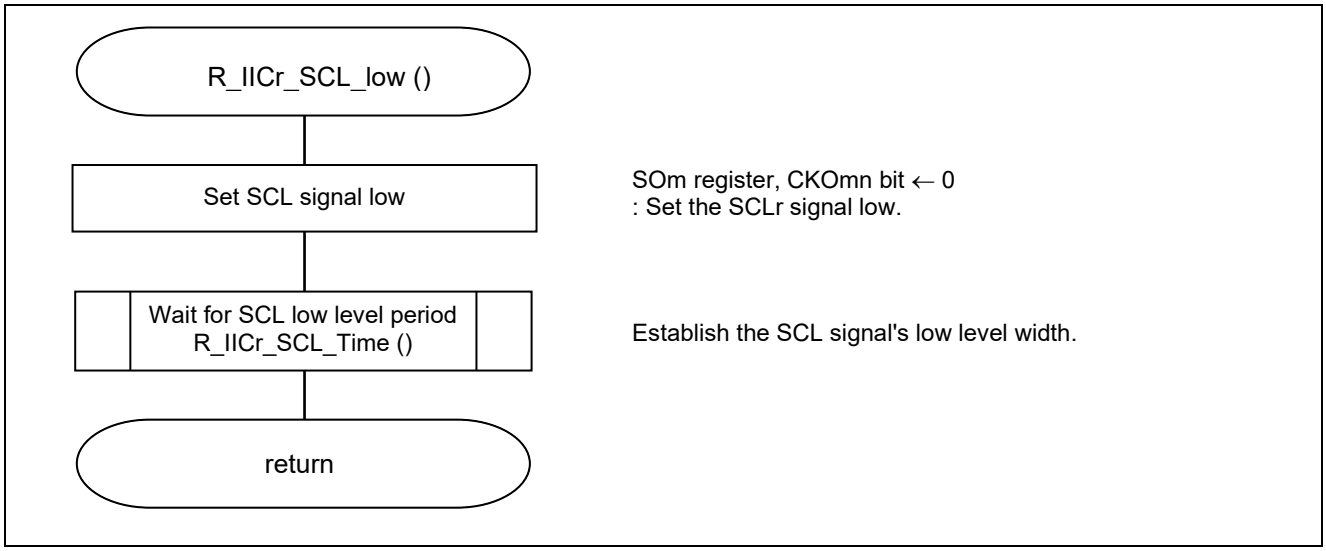


Figure 5.35 SCL Set Low Processing

5.7.26 ACK Confirmation Processing

Figure 5.36 shows the flowchart for checking the ACK response.

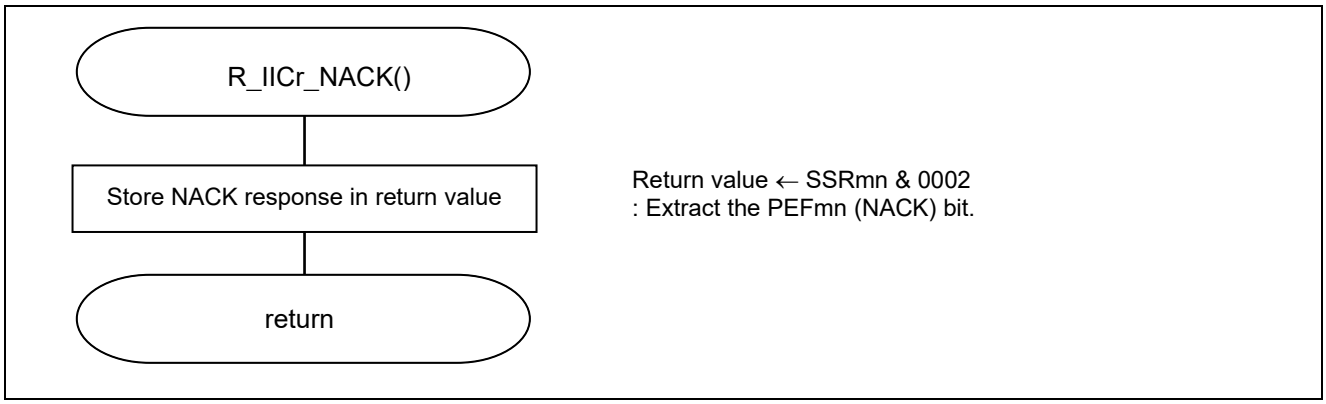


Figure 5.36 ACK Response Check Processing

5.7.27 SCL Low Level Period Wait Processing

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

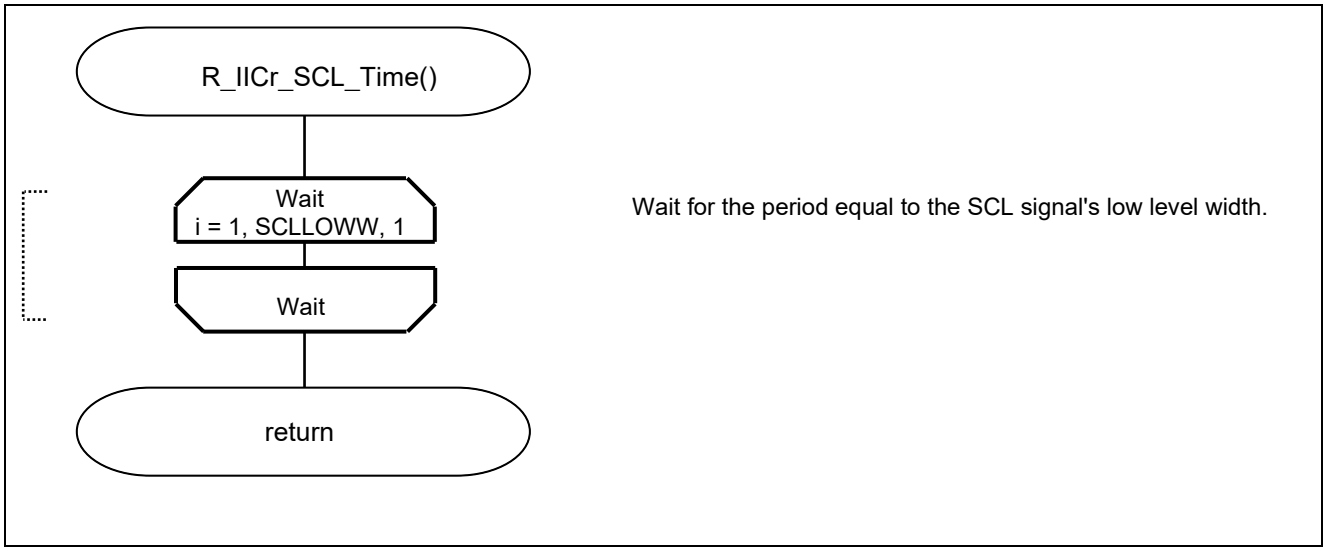


Figure 5.37 SCL Low Level Period Wait Processing

5.7.28 SCL High Level Period Wait Processing

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

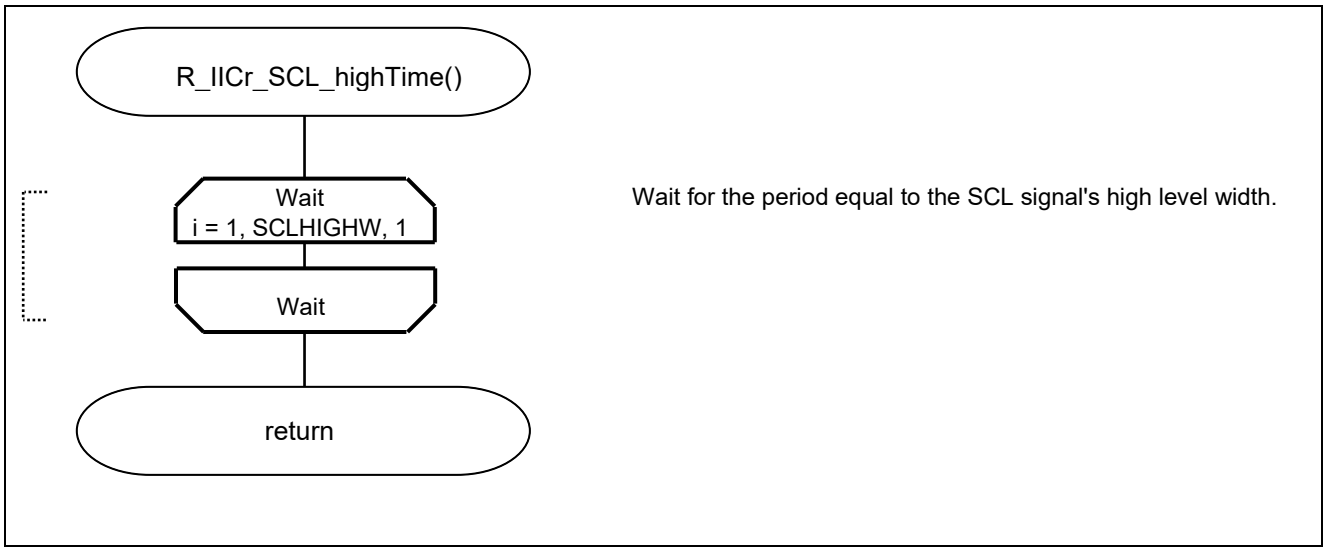


Figure 5.38 SCL High Level Period Wait Processing

5.7.29 INTIICr Interrupt Processing

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

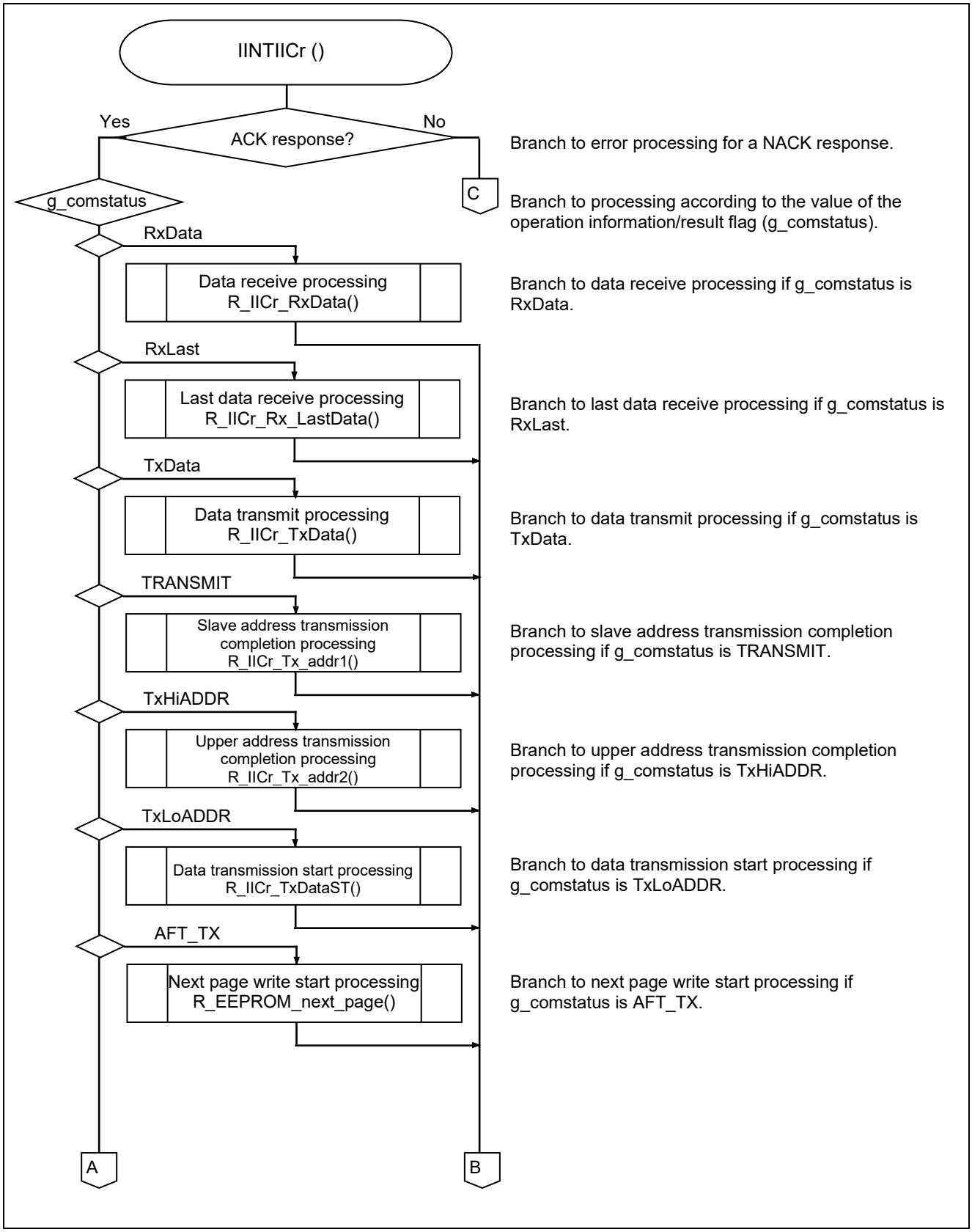


Figure 5.39 INTIICr Interrupt Processing (1/3)

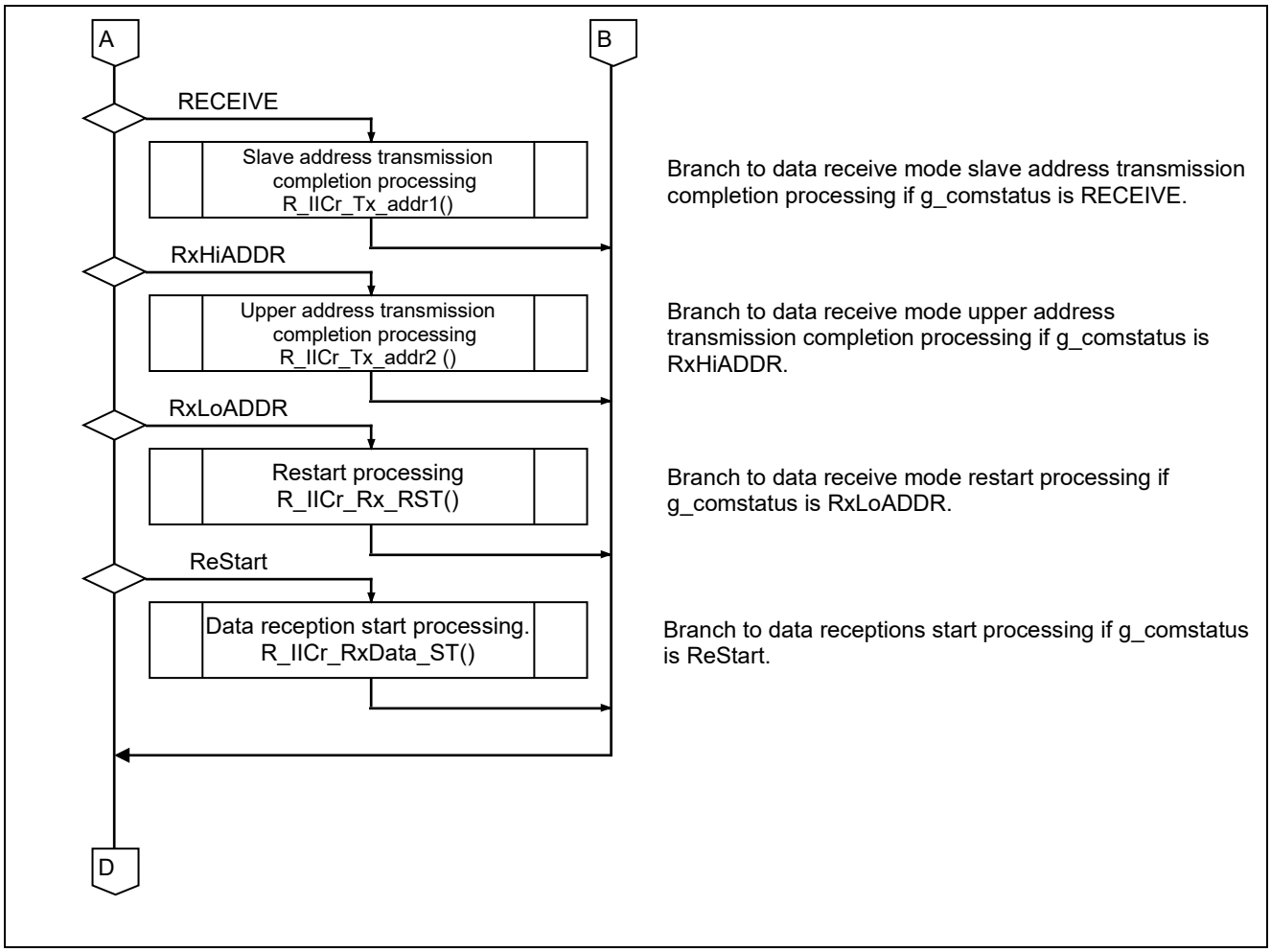


Figure 5.40 INTIICr Interrupt Processing (2/3)

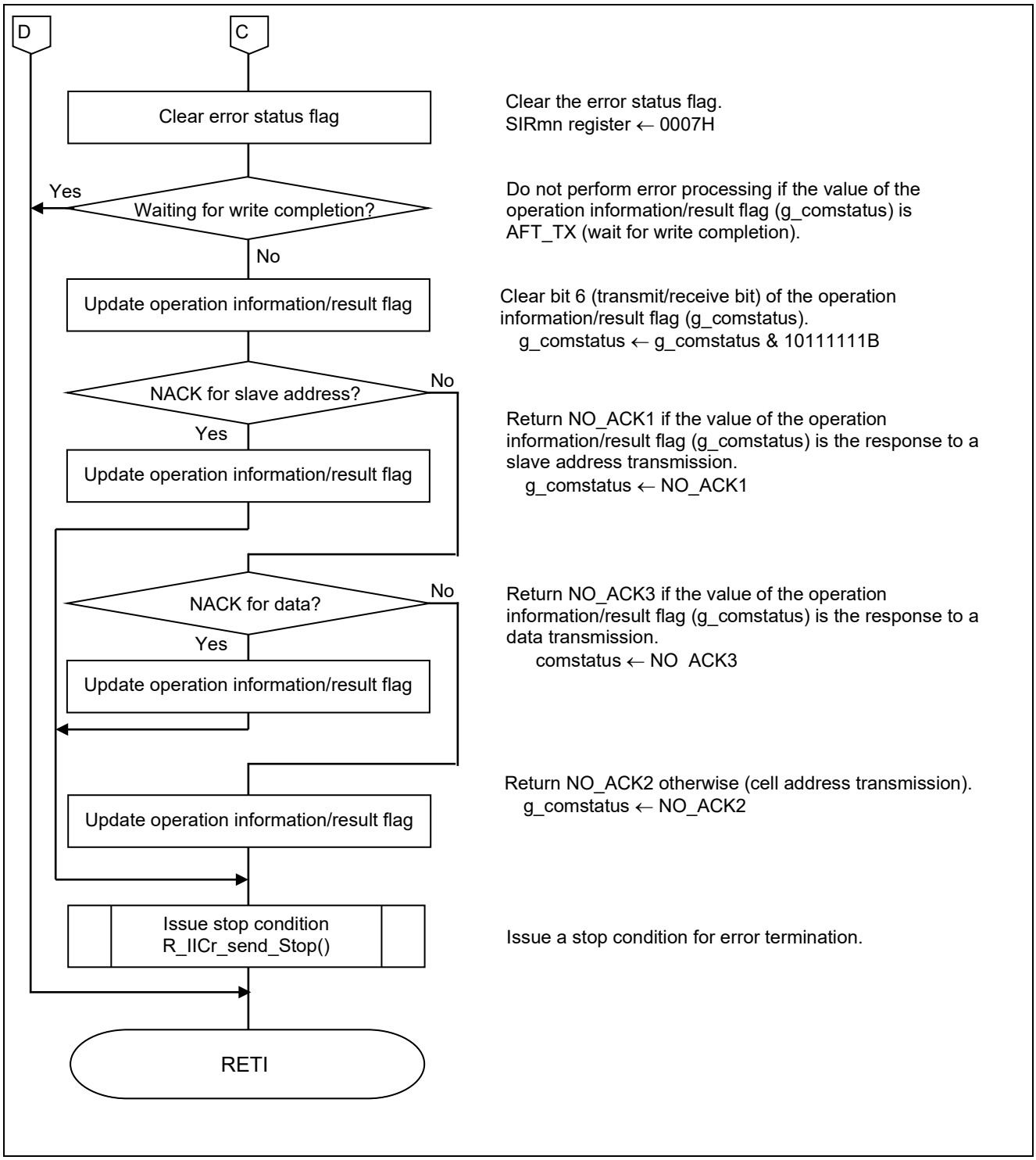


Figure 5.41 INTIICr Interrupt Processing (3/3)

5.7.30 INTTM02 Interrupt Processing

Figure 5.42 shows the flowchart for the INTTM02 interrupt processing.

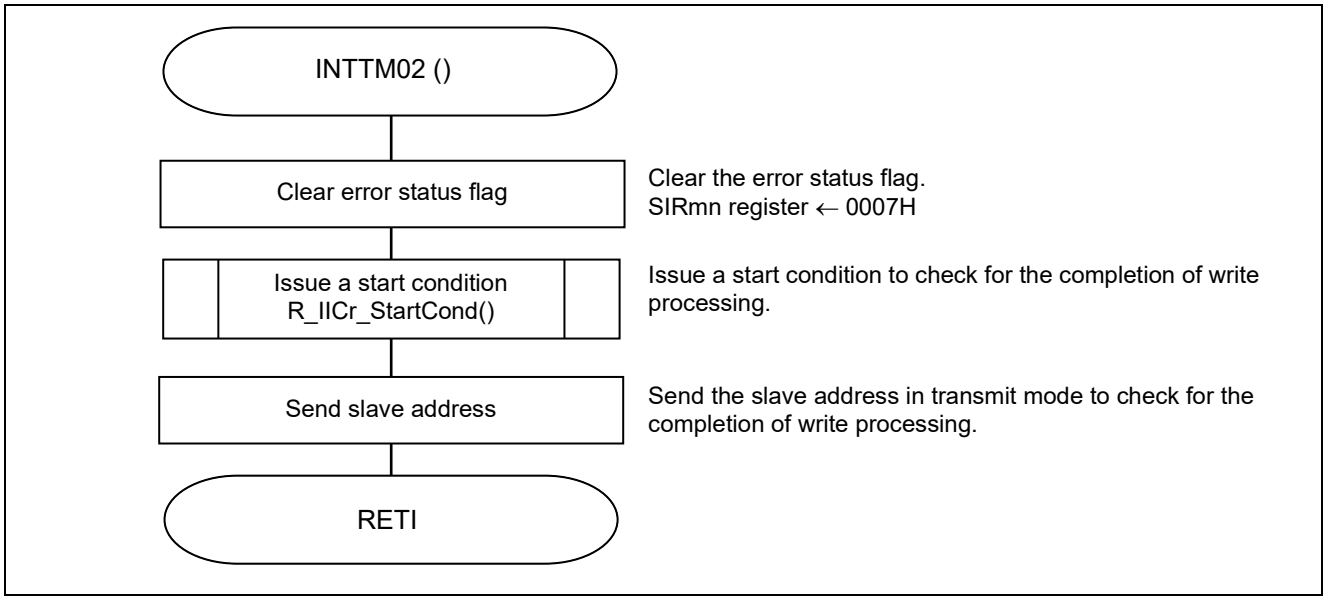


Figure 5.42 INTTM02 Interrupt Processing

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G23 User's Manual: Hardware (R01UH0896E)

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.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.11.25	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

- A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

- The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

- Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

- Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the 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.

5. Clock signals

- After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

- Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

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

8. Differences between products

- Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.