RL78/G10

EEPROM Control by Simplified I2C Function CC-RL

Introduction

This application note describes how to control the external serial EEPROM by using Simplified I2C function of the serial array unit (SAU). Realizes reading/writing serial EEPROM through I2C bus connection using Simplified I2C function by interrupt processing.

Target Device

RL78/G10

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

Contents

1. Specification

In this application note, controls serial EEPROM which is connected to external with using Simplified I2C function of the serial array unit (SAU).

Checks a state of SW1 after end of the rest. Reads all memory areas of the serial EEPROM if SW1 is pressed. Reads and writes data of all memory areas of the serial EEPROM if SW1 is not pressed.

Access to the serial EEPROM is performed per block (data size chosen from 4/8/16 bytes). Blinks LED whenever 1 block of writing/reading processing of serial EEPROM is completed normally. If the writing/reading processing is failed, stops LED blinking. (The state of LED is lighting or extinction and subsequent processing is not performed.) When the writing/reading of serial EEPROM carry out a normal end, makes LED lighting state and moves to the waiting for SW1 keypress. If SW1 is pressed, makes LED lighting-out state and performs writing/reading all the memory areas of EEPROM.

The detail specification of this application note is shown below.

- RL78/G10 microcontroller is used as the master, and serial EEPROM is used as the slave.
- Selects a target serial EEPROM from 512K-bit (64K-byte) to 2K-bit (256-byte). (Selectable serial EEPROMs are 2K-bit, 4K-bit, 8K-bit, 16K-bit, 32K-bit, 64K-bit, 128K-bit, 256K-bit and 512K-bit. 256K-bit serial EEPROM is selected as the default.)
- Selects a communication rate either first mode (Max.384k bps*) or normal mode (Max.100 kbps).
  * As for the transfer rate of simplified I2C function, the duty ration is 50% due to use SAU operation clock. For this reason, it is required to set a transfer rate that the low level width of a SCL signal meets the specification of I2C bus (1.3 μs). And the first mode is not 400kbps but around 384kbps. (Refer to RL78/G10 User's Manual: Hardware for more information.)
- Accesses serial EEPROM per a data block unit defined beforehand. In this application note, the data size of 1 block is chosen from 4/8/16 bytes in order to avoid processing of the page boundary at the time of EEPROM writing. (Refer to “1.1 Page Boundary Processing” for more information.)
- Bus release processing is performed in consideration of a possibility that I2C bus is occupied. (Refer to “1.2 I2C Bus Release Processing” for more information.)
- Using interrupt processing, writing/reading EEPROM are performed in the background.

Table 1.1 shows the peripheral function to be used and its use. **Figure 1.1** shows the operation outline.

Table 1.1 Peripheral Function to be Used and its Use

Peripheral Function	**Use**
Serial array unit (SAU)	Using a simplified I2C function, I2C master transmission and reception are performed. (SCL00 pin and SDA00 pin are used.)
Timer array unit 0 (TAU0) Channel 1	Uses the interval timer of 100us or 400us in order to check the completion of writing at data transmission processing. (A timer speed is switched to either first mode or normal mode.)

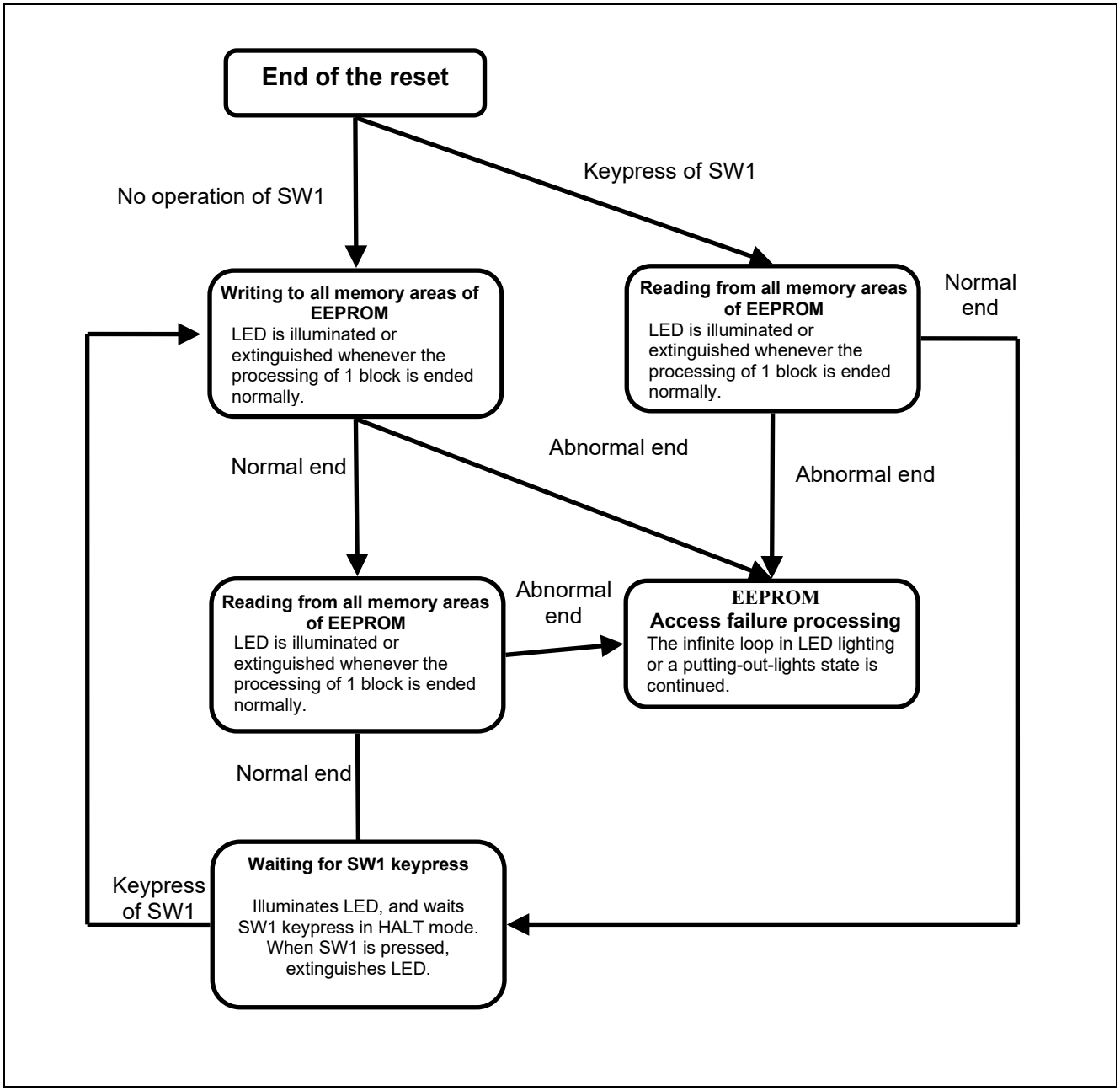


Figure 1.1 Operation Outline

1.1 Page Boundary Processing

EEPROM is divided into the page of the size which differs from 16 bytes to 128 bytes according to a kind. When reading data from EEPROM, paying attention to these page boundaries is not necessary, but following cautions are required to write data in EEPROM.
As for the writing of EEPROM, accessing to plural pages at one processing (1 block) is prohibited. Since a page will not be automatically updated, if the address for writing arrives at the final address of a page before end of the writing, it overwrites from the head address of the same page. For this reason, if it writes in being unconscious of a page boundary, there is a possibility that data overwrite which is not expected has occurred. In this application note, in order to avoid the incorrect writing by page boundary access, a data block is chosen from 4/8/16 byte in advance. Since these block sizes are enough small to write in one page, the page boundary is not need to be conscious.

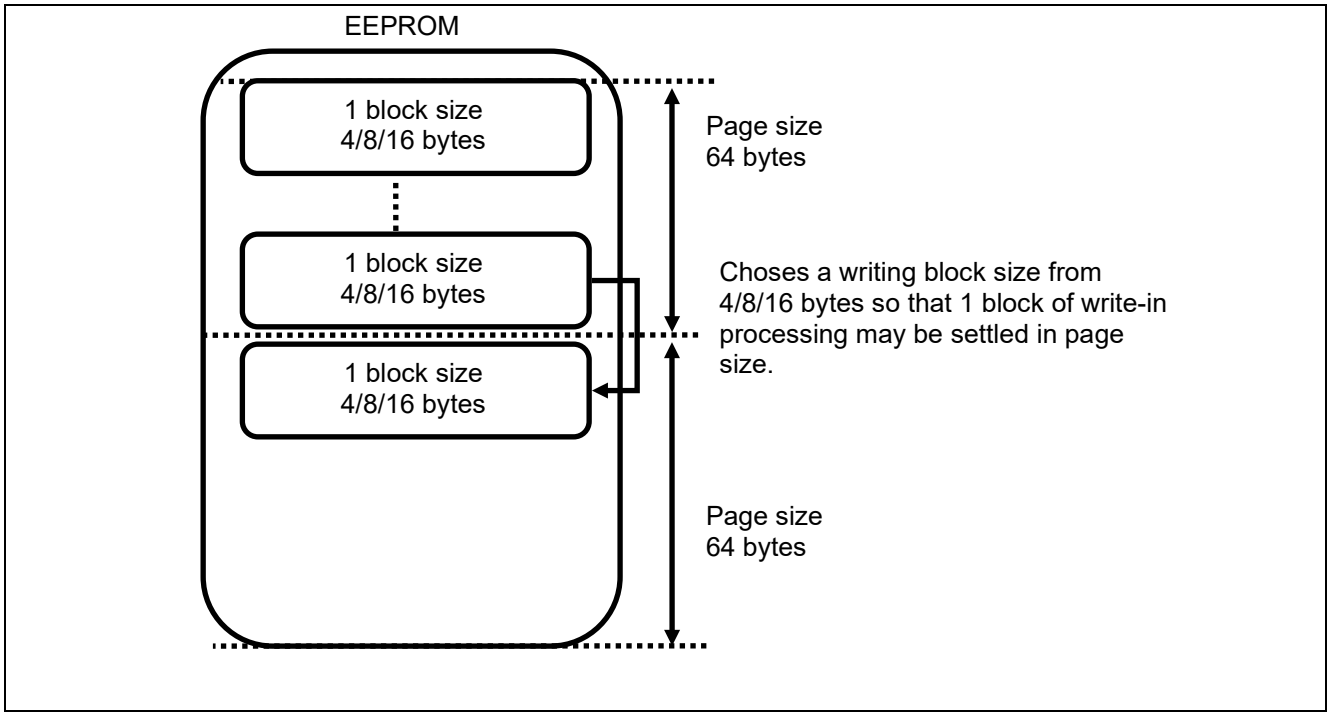


Figure 1.2 EEPROM Page Boundary Processing

1.2 I2C Bus Release Processing

When I2C bus is used, generates the stop condition first and makes the I2C bus into a released state.

However, if the slave (EEPROM) makes the SDA signal the low level, stop condition cannot be generated and I2 C bus may be unable to be opened.

For example, when read-out of EEPROM has been interrupted by the reasons of the communications processing of I2C bus not having been finished normally on the way, this abnormal condition occurs. The following two can be considered as the cause.

(1) The master was reset by factors other than a power down during read-out of EEPROM.

(2) The master answered ACK to the last data at the time of read-out of EEPROM.

(1) will be in an abnormal condition because there is no means which applies reset to EEPROM from the outside. (2) will be in an abnormal condition because it is necessary to certainly answer NACK to the received data (in the case that data reception ends normally as well) when the master of I2C bus ends reception operation.

In such a case, the master of I2C bus manipulates the SCL signal by software to generate the fake clocks (dummy clocks) of I2C bus. And waits for a SDA signal to become high-level.

If a SCL signal is generated for longer than or equal to 9 clocks, EEPROM will make a SDA signal high-level. The reason is as follows. EEPROM outputs data according to a transmission request from the master. If the data is 0 (low level), EEPROM has stopped. If SCL signal is detected, EEPROM shifts to the next data output and stops the drive of a SDA signal after 8 clocks at the latest for the ACK reception from the master. And after 9 clocks, EEPROM detects NACK and stops transmission.

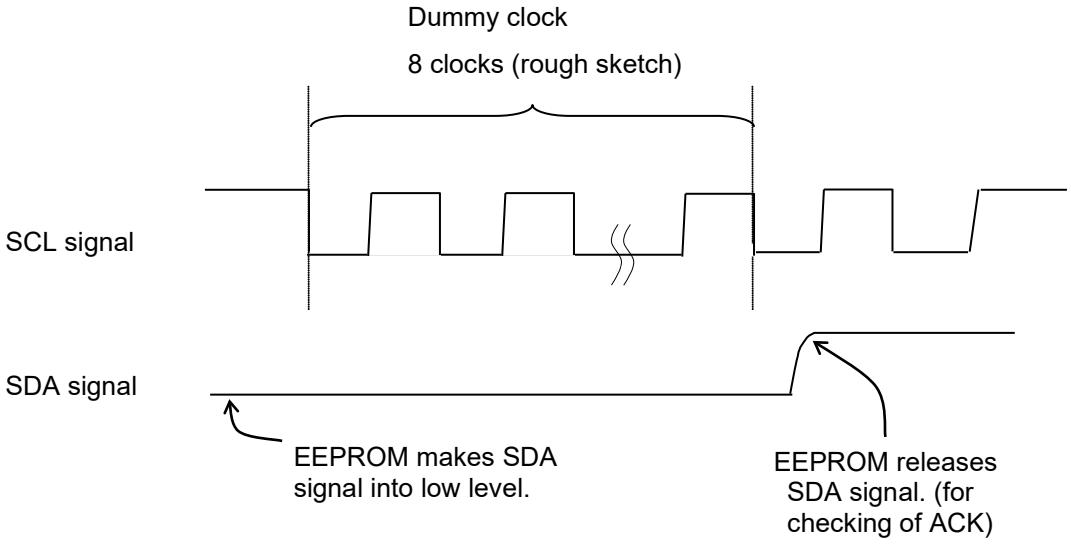


Figure 1.3 Bus Release Processing

1.3 Address Setup Processing of EEPROM

1.3.1 Targeted EEPROM Specification

In this application note, uses Serial EEPROM of Renesas Electronics. The serial EEPROM varies in an addressing method of the memory cell in the EEPROM depending on capacity size.

Table 1.2 shows the specification of serial EEPROM used with this application note made by Renesas Electronics.

Table 1.2 Addressing Method of EEPROM

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	A2	A1	A0	R/$\overline{W}$

Products	Capacity (byte)	Required address length	A2 to A0	1st byte	2nd byte
R1EX24512B	64K	16 bits (a15 to a0)	—	a15 to a8	a7 to a0
R1EX24256B Note	32K	15 bits (a14 to a0)	—	a14 to a8	a7 to a0
R1EX128B	16K	14 bits (a13 to a0)	—	a13 to a8	a7 to a0
R1EX24064A	8K	13 bits (a12 to a0)	—	a12 to a8	a7 to a0
R1EX24032A	4K	12 bits (a11 to a0)	—	a11 to a8	a7 to a0
R1EX24016A	2K	11 bits (a10 to a0)	a10 to a8	a7 to a0	—
R1EX24008A	1K	10 bits (a9 to a0)	a9, a8	a7 to a0	—
R1EX24004A	512	9 bits (a8 to a0)	A0=a8	a7 to a0	—
R1EX24002A	256	8 bits (a7 to a0)	—	a7 to a0	—

Note: It is EEPROM used for the operation confirmation of this application note.

1.3.2 Address Update

Reading or writing is performed by specifying an address, the address is updated automatically and comes to show the next address in EEPROM. Therefore, it is not necessary to specify an address each time when performing continuation writing and reading data. However, the continuation writing over a page boundary requires the described cautions "1.1 Page Boundary Processing".

1.3.3 Writing Processing of EEPROM

If a master (RL78/G10) generates stop condition, EEPROM will actually start writing data received to a memory cell. The execution time of this writing processing (for 1 page)is about 5 ms, and EEPROM does not responds ACK to a master (RL78/G10) in this 5 ms.

In order to write data continuously, it is necessary to transmit the next data after the waiting of the writing end waiting (about 5 ms). As a method of checking a writing completion waiting state, start condition is generated from a master (RL78/G10), and it transmits in the writing mode (LSB=0) of EEPROM. Although it becomes a NACK response under writing processing, if writing is completed, it becomes an ACK response and can check the state of EEPROM.

2. Operation Confirmation Conditions

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

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	RL78/G10 (R5F10Y16)
Operating frequency	・ High-speed on-chip oscillator (HOCO) clock: 20 MHz ・ CPU/peripheral hardware clock: 20 MHz
Operating voltage	3.3V(Operation is possible over a voltage range of 2.9 to 5.5 V.) SPOR Detection Voltage: Rising edge: 2.90V Falling edge: 2.84V
Integrated development environment (CS+)	CS+ for CC V3.01.00 from Renesas Electronics Corp.
Assembler (CS+)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V4.1.0.018 from Renesas Electronics Corp.
Assembler (e^2 studio)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.3 from IAR Systems.
Assembler (IAR)	IAR Assembler for Renesas RL78 V4.21.2.2420 from IAR Systems.
Board used	RL78/G10 target board (QB-R5F10Y16-TB)

Table 2.2 EEPROM Specifications

Item	Contents
EEPROM used	R1EX24256B
Operating voltage range	single power supply: 1.8V to 5.5V
Maximum operation frequency	400kHz
Capacity	256K bits
Page size	64 bytes
Rewriting time	5ms

3. Related Application Notes

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

RL78/G10 Initialization CC-RL (R01AN2668E) Application Note

RL78/G10 Timer Array Unit (Interval Timer) CC-RL (R01AN3074E) Application Note

4. Description of Hardware

4.1 Hardware Configuration Example

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

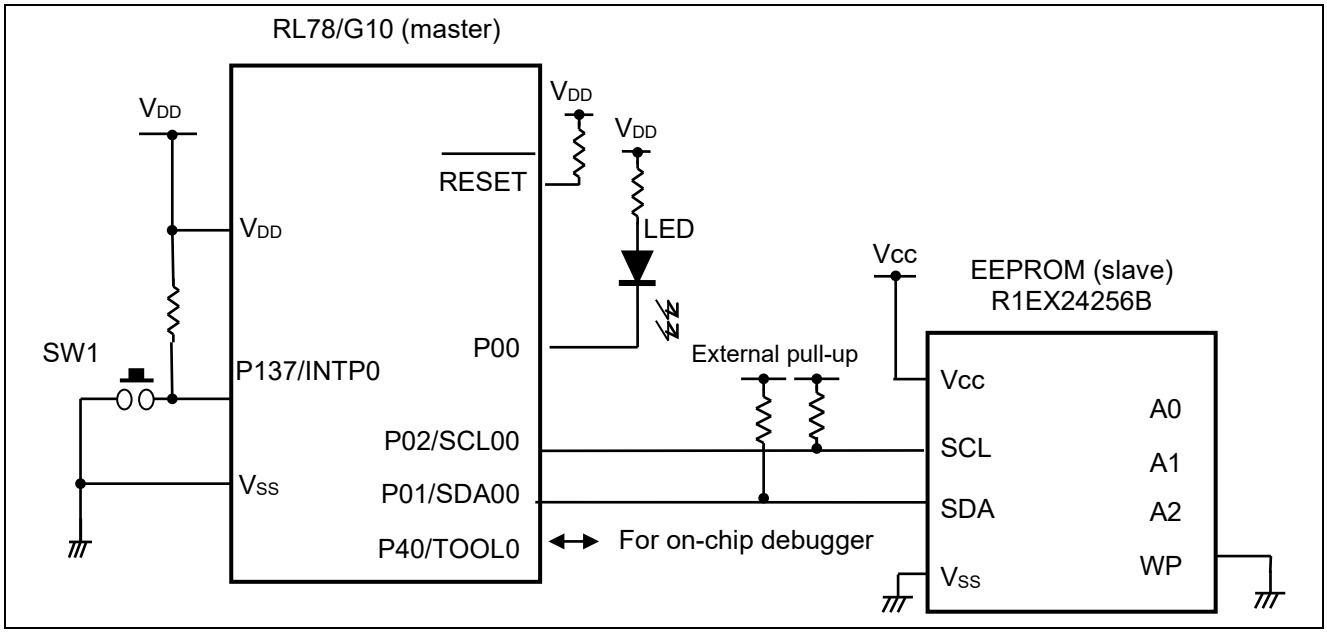


Figure 4.1 Hardware Configuration

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} orV_{SS} via a resistor).
2. V_{DD} must be held at not lower than the reset release voltage (V_{SPOR}) that is specified as SPOR.
3. Since EEPROM device address pins (A0, A1, A2) are pulled-up inside EEPROM, they are recognized un-connecting.

4.2 List of Pins to be Used

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

Table 4.1 Pins to be Used and their Functions

Pin Name	I/O	Description
P00	**Output**	LED drive port Under EEPROM read-out/writing processing: Blinking: Blinks LED whenever 1 block of writing/reading processing of serial EEPROM is completed normally. Lighting/extinction: An error is occurred at writing/reading of EEPROM. Waiting for SW1 keypress: Lighting: Waiting for SW1 keypress. Extinction: Accepts SW1 keypress and starts EEPROM writing/reading processing again.
P01/SDA00	**Input** **Output**	Simplified I2C data input/output
P02/SCL00	**Output**	Simplified I2C clock output
P137/INTP0	**Input**	Switch input (SW1) At power-on No operation: Starts EEPROM writing/reading processing. Keypress: Reads out EEPROM. Waiting for keypress No operation: Continues to wait keypress. Keypress: Starts EEPROM writing/reading processing.

5. Software

5.1 Operation Outline

This application note checks the SW1 state after the release from the reset. When SW1 is pressed, reads all memory areas of serial EEPROM. When SWI is not pressed, reads and writes all memory areas of serial EEPROM

Blinks LED whenever 1 block of writing/reading processing of serial EEPROM is completed normally. If the writing/reading processing is failed, stops LED blinking. (The state of LED is lighting or extinction and subsequent processing is not performed.) When the writing/reading of serial EEPROM carry out a normal end, makes LED lighting state and moves to the waiting for SW1 keypress. If SW1 is pressed, makes LED lighting-out state and performs writing/reading all the memory areas of EEPROM.

(1) Initializes on-chip peripheral functions.
<Setting conditions>

a) Sets I/O port.
- Sets a digital I/O to PMC0 register.
- Sets the initial value to P0 register as initializes LED connection pin (P00) is high output, SDA00 pin (P01) is low output, and SCL00 pin (P02) is low output.
- Sets the initial value to PM0 register as initializes LED connection pin (PM00) is output, SDA00 pin (PM01) is input, and SCL00 (PM02) is input.

b) Sets clock generator.
- Sets HOCODIV register to 20MHz.

c) Initializes interruption related registers.
- Sets the mask of INTP0 interrupt.
- Enables to detect falling edge of INTP0 interrupt.
- Clears the interrupt request of INTP0 interrupt.

(2) Clears the memory area used by a program.
a) Clears the number of blocks writing/reading (variable: BLOCK_NUMBER).
b) Clears data storage areas of reading and writing (variable: WRITE_BUFF, READ_BUFF, R_BUFF_END).
c) Clears I2C interrupt control status (variable: STATUS).
d) Clears the number of data I2C writing/reading processing (variable: DATACOUNT).
e) Clears the slave address of I2C (variable: SLAVEADDR).

(3) Initializes SAU as the simplified I2C function.
a) Sets SAU0EN bit of PER0 register as a clock is provided to SAU.
b) Sets SPS0 register as the frequency of operation clock CK00 and operation clock CK01 are 20MHz.
c) Sets SIR00 register as the error flag is cleared.
d) Sets SMR00H register as the operation clock CK00 is selected.
e) Sets SMR00L register as the simplified I2C mode and the transfer end interrupt are set.
f) Sets SCR00H register as the transfer mode, no parity, and type 1 are set.
g) Sets SCR00L register as MSB first is set.
h) Sets SDR00H register as the transfer clock is 384kbps at fast mode and 100kbps at normal mode.
i) Sets SO00 bit of SO0 register as SDA pin is high.
j) Sets CKO00 bit of CKO0 register as SCL pin is high.
k) Sets POM0 register as SDA pin and SCL pin are output.

(4) Sets TAU0 for confirming the Transmit data completion. Initializes TAU0 to 100μs at the fast mode and to 400μs at the normal mode.
   a) Sets TAU0EN bit of PER0 register to provide the clock to TAU.
   b) Sets TTH01 bit of TTH0 register and TT01 bit of TT0 register to stop the timer.
   c) Sets the frequency of the operation clock CK00 and the operation clock CK01 to fCLK/16 (fast mode) or fCLK/64 (normal mode) through TPS0 register.
   d) Sets TMR01H register as the operation clock CK01 is selected and The 8-bit timer operation is set.
   e) Sets TMR01L register to set the interval timer.
   f) Sets TDR01H register as 100μs at the fast mode and 400μs at the normal mode.
   g) Sets TOE01 bit of TOE0 register to set TO01 output disabled.
   h) Sets TO0 register to set Timer output value 0.
   i) Sets TMMK01H bit of MK0L register to set the interrupt request mask.
   j) Sets TMIF01H bit of IF0L register to set interrupt request clear.

(5) Generates stop condition and makes I2C bus the bus release status before using I2C bus. In order to make I2C bus the bus released status, waits for SDA signal becomes high by generating spurious I2C bus clock (dummy clock) through controlling SCL00 signal by a program. After becoming high of SDA signal, generates the stop condition and releases the bus.

(6) Monitors the state of SW1, and if SW1 is pressed, performs EEPROM reading processing. If SW1 is not pressed, performs reading and writing processing of EEPROM. Refer to Chapter 5.2 for more information about EEPROM control part.

(7) Creates data to be written to EEPROM.
Creates data to be written based on selected block size (4/8/16-byte), the number of blocks of EEPROM that is destination to write, and the final 1 byte of write buffer.

The example of data to be written at the time of choosing 4 bytes as block size is shown in Table 5.1.

Table 5.1 Data to be written into EEPROM

Block number to be written	The example of write data. (In the case that the block size is 4 bytes.)
After reset releasing	0x00, 0x00, 0x00, 0x00
Write block number 0	0x01, 0x02, 0x03, 0x04 (Creates data to be written from the previous wrote data: final value 0x00.)
Write block number 1	0x05, 0x06, 0x07, 0x08 (Creates data to be written from the previous wrote data: final value 0x04.)
・・・	・・・
Write block number 16	0x41, 0x42, 0x43, 0x44
・・・	・・・
Write block number 63	0xFD, 0xFE, 0xFF, 0x00
Write block number 64	0x01, 0x02, 0x03, 0x04
・・・	・・・

Note: This application note shows control examples of EEPROM (R1EX24256B) by I2C bus using the simplified I2C functions of RL78/G10 (IIC00). If channels or EEPRO to be used are changed, conducts an extensive evaluation of the modified program.

5.2 Details of Serial EEPROM Control Program

5.2.1 Interrupt Processing Outline

In this application note, uses the interrupt request (INTIIC00) of simplified I2C function (IIC00). The communications processing with EEPROM is divided into some processing routines. Table 5.2 shows those processing.

Table 5.2 Interrupt Processing Outline

No.	Processing routine name	Processing
0	IINTIIC00	If IIC00 receives ACK, makes to branch to processing of 1 to 9 of the following. If NACK is received, it will be checked whether STATUS is AFT_TX (under EEPROM writing). If it is AFT_TX, nothing is done, but error processing is performed if it is except AFT_TX.
1	R_IIC00_Tx_addr1	Upper bytes transmitting processing of the EEPROM address after the completion of slave address transmitting. It is processing of only EEPROM of 32K to a 512K bit, and common to transmission and reception. The next processing is R_IIC00_Tx_addr2.
2	R_IIC00_Tx_addr2	Lower address transmitting processing of EEPROM. This processing is common to transmission and reception. The next processing at the time of transmission is R_IIC00_TxDataST (data transmission start). The next processing at the time of reception is R_IIC00_Rx_RST (restart).
3	R_IIC00_Rx_RST	Performs the restart condition generate and the slave address transmission in reception mode after the completion of slave address transmission at the time of reception. The next processing is R_IIC00_RxDataST (data reception start).
4	R_IIC00_RxDataST	The start of data reception processing after the completion of slave address transmission in receiving mode. The next processing is R_IIC00_RxData (data reception).
5	R_IIC00_RxData	The data reception processing. Stores received data to a buffer. The next processing as follows. The remaining data is 2 bytes or more: R_IIC00_RxData The remaining data is 1: R_IIC00_Rx_Last (receiving the final data)
6	R_IIC00_Rx_Last	Stops IIC00 by the completion of final data reception, and generates stop condition.
7	R_IIC00_TxDataST	Data transmission-start processing after the completion of slave address transmission by data transmission. The next processing is R_IIC00_TxData (data transmission).
8	R_IIC00_TxData	The completion of data transmission of 1 byte. The following data will be transmitted if there is the remaining data. The next processing is R_IIC00_TxData. If the remaining day is lost, generates stop condition (EEPROM starts writing received data to the memory cell) , sets AFT_TX (under EEPROM writing) to STATUS, and stops operation of IIC00. Activates the timer in order to confirm the completion of writing. The next processing is IINTTM01H (timer interrupt).
9	R_EEPROM_WT_END	The completion of writing processing of EEPROM (the ACK response to slave address transmission). Generates stop condition, and stops the timer which confirms the completion of writing.
10	IINTTM01H	Interval interruption in every 100 µs or 400 µs. Confirms the completion of writing at EEPROM (transmission of slave address in transmission mode). The response to this slave address transmission is INTIIC00 interrupt request. Transmission processing will be completed if the response becomes R_EEPROM_WT_END (EEPROM writing completion).

5.2.2 EEPROM Control Program State Transitions

In this application note, if the access processing to EEPROM begins, a state will change by interrupt request (INTIIC00) generating of IIC00. The change state of read-out and write-in processing is shown below.

(1) Reading processing state transition

Figure 5.1 and Figure 5.2 show the serial EEPROM reading processing state transition.

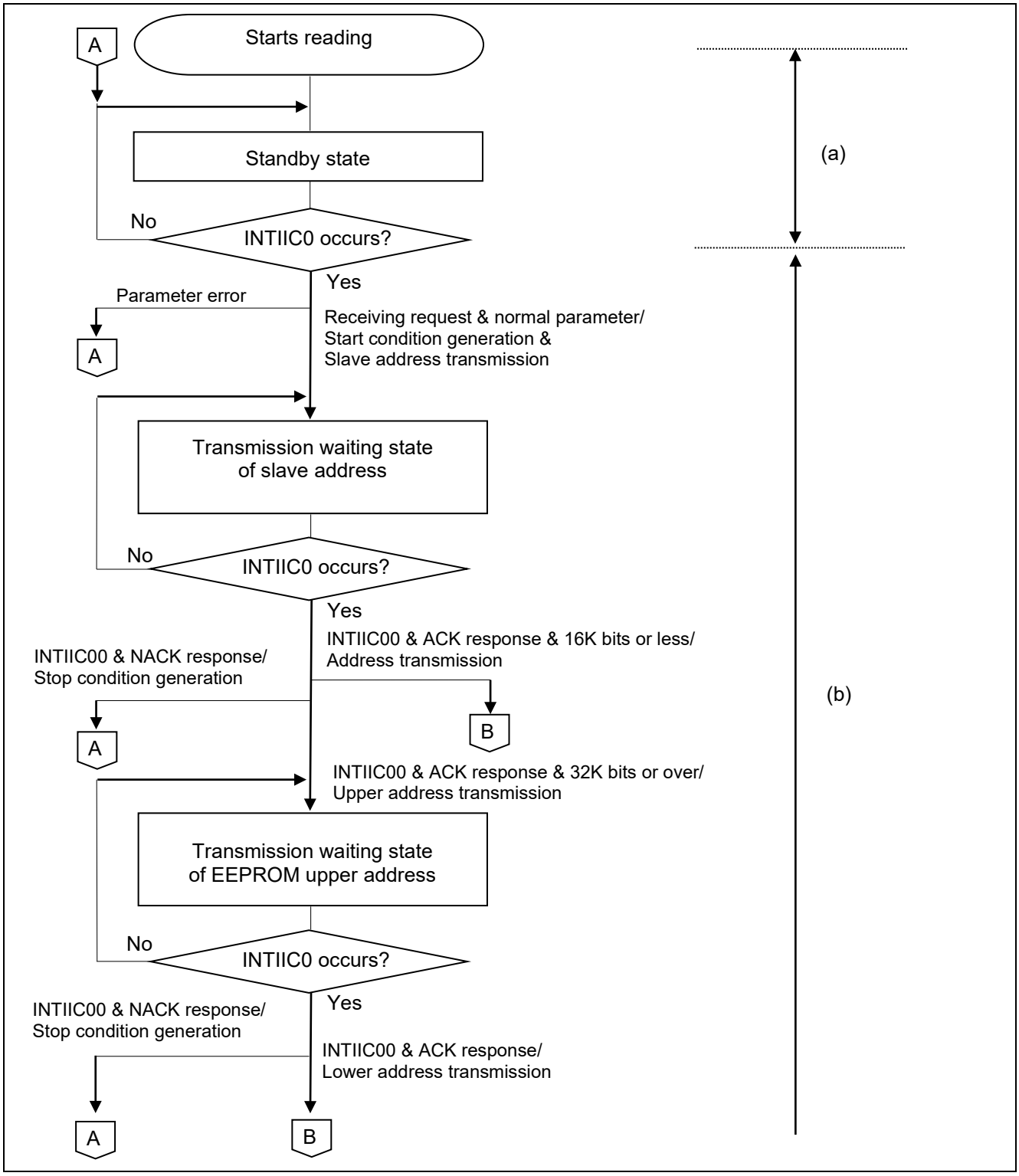


Figure 5.1 Reading Processing State Transition (1/2)

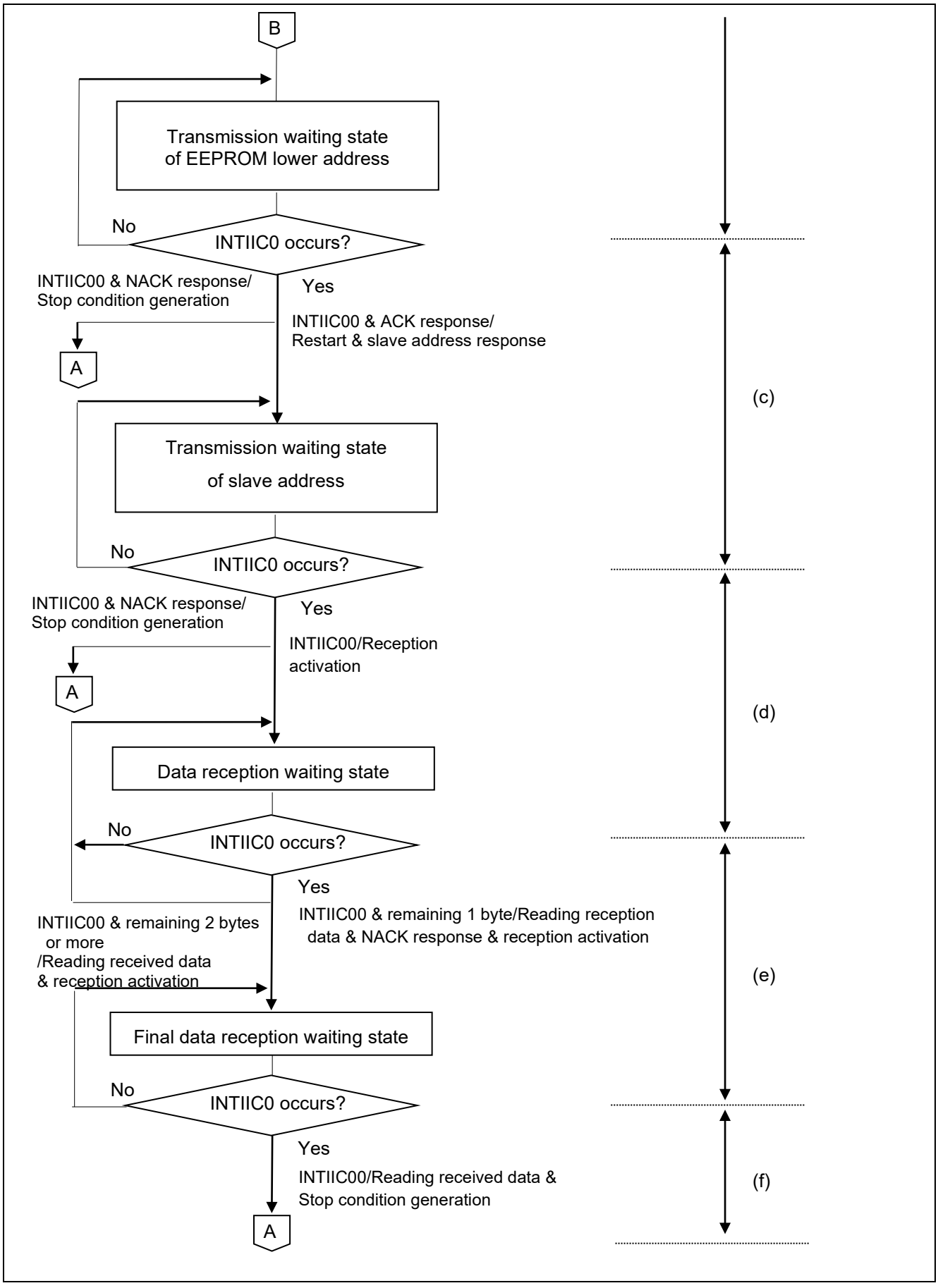


Figure 5.2 Reading Processing State Transition (2/2)

(a) If EEPROM reading processing is called under the standby state, a parameter (the target block number) will be checked. When the block number is wrong, since it is a parameter error, processing is ended. If a parameter is right, start condition will be generated and a slave address will be transmitted as LSB=0.

(b) Checks a response from a slave by a interrupt request from INTIIC00. When it is NACK response, generates a stop condition and ends the processing. When it is ACK response, transmits EEPROM address. At this time, 2 bytes of address information is transmitted to EEPROM which is 32K bits or above, and 1 byte of address information is transmitted to EEPROM which is 16K bits or less.

(c) The response from a slave is checked by the interrupt request of INTIIC00. If it is a NACK response, generates a stop condition and ends the processing. When it is ACK response, generates a restart condition and transmits a slave address as LSB=1.

(d) Checks a response from a slave by INTIIC00 interrupt request. If it is NACK response, generates stop condition and ends the processing. If it is ACK response, stops IIC00, and then changes into receiving mode and reboots. Activates the receiving operation by writing the dummy data to SIO00.

(e) Stores received data into a buffer by INTIIC00 interrupt request. If the number of remaining receiving data is 2 or more, activate the receiving operation by writing the dummy data to SIO00. If the number of remaining receiving data is 1, disables ACK response (SOE00=0) and activates the receiving operation by writing dummy data to SIO00.

(f) Stores receiving data to a buffer by INTIIC00 interrupt request. Since the receiving processing is completed, stops IIC00 operation, generates stop condition, and finishes the processing.

(2) Writing processing state transition

Figure 5.3 and Figure 5.4 show the serial EEPROM writing processing state transition.

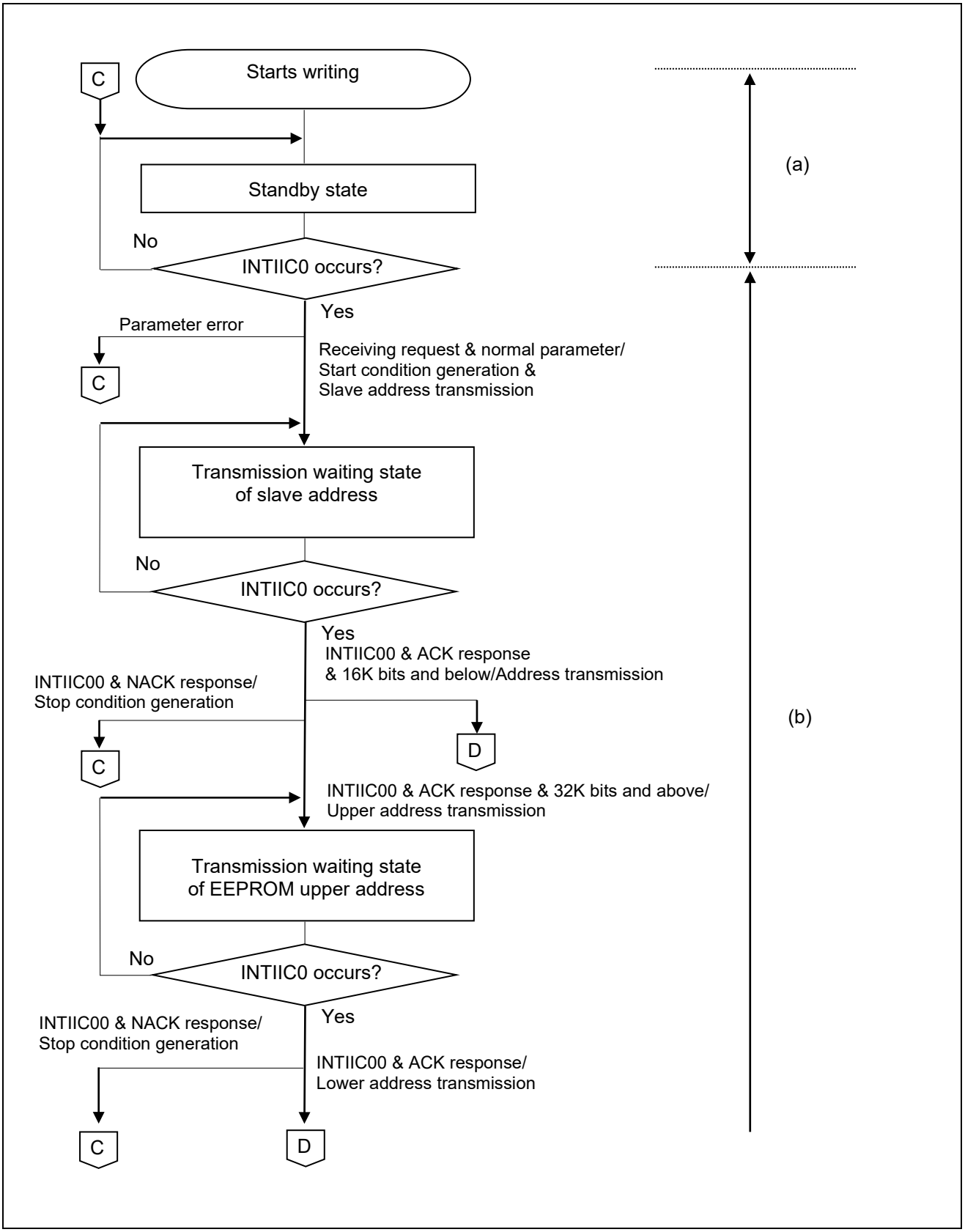


Figure 5.3 Writing Processing State Transition (1/2)

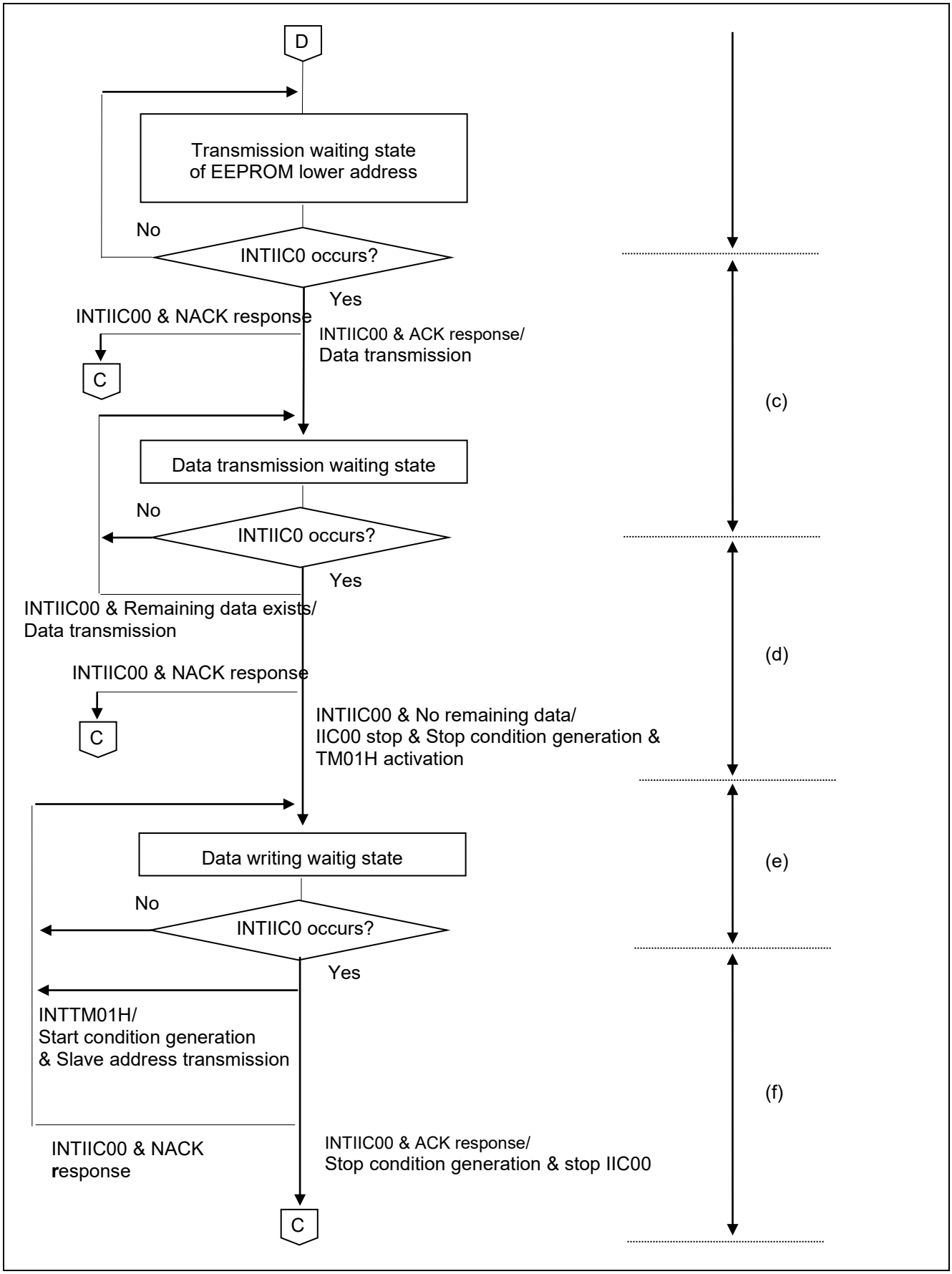


Figure 5.4 Writing Processing State Transition (2/2)

(a) If EEPROM reading processing is called under the standby state, a parameter (the target block number) will be checked. When the block number is wrong, since it is a parameter error, processing is ended. If a parameter is right, start condition will be generated and a slave address will be transmitted as LSB=0.

(b) Checks a response from a slave by a interrupt request from INTIIC00. When it is NACK response, generates a stop condition and ends the processing. When it is ACK response, transmits EEPROM address. At this time, 2 bytes of address information is transmitted to EEPROM which is 32K bits or above, and 1 byte of address information is transmitted to EEPROM which is 16K bits or less.

(c) The response from a slave is checked by the interrupt request of INTIIC00. If it is a NACK response, generates a stop condition and ends the processing. When it is ACK response, writes transmission data to SIO00 and starts transmission.

(d) Checks a response from a slave by INTIIC00 interrupt request. If it is NACK response, generates stop condition and finishes the processing. If it is ACK response, writes the next transmission data to SIO00 and starts the transmission. If there is no next data, generates stop condition, completes the transmission, and starts writing to EEPROM. In order to confirm the completion of writing, activates TM01H as an interval timer of 100μs. At this time, changes Variable STATUS into AFT_TX and sets up under the writing processing after transmission.

(e) In order to confirm the completion of writing to EEPROM by INTTM01H interrupt request, transmits start condition and slave address (LSB=0).

(f) Checks a response from a slave by INTIIC00 interrupt request. If it is NACK response, waits for the next INTTM01H interrupt request. If it is ACK response, generates stop condition, and stops IIC00 and TM01H since writing to EEPROM is completed. Variable STATUS is set to the completion of transmission.

5.3 Option Byte Settings

Table 5.3 shows the option byte setting.

Table 5.3 Option Byte Setting

Address	Value	Description
000C0H	11101110B	Watchdog timer operation stop (Stops counting after the release from the reset state.)
000C1H	11110111B	SPOR detection voltage: Falling edge: VDD < 2.84V Rising edge: VDD ≧ 2.90V
000C2H	11111001B	HOCO : 20MHz
000C3H	10000101B	Enables the on-chip debugger

5.4 Constants

Table 5.4 and Table 5.5 list the constants that are used in this sample program.

Table 5.4 Constants for the Sample Program (1/2)

Constant	**Setting**	**Description**
CLKFREQ	**20000**	It is the definition which expressed the operation clock fCLK of RL78/G10 per kHz.
FAST_MODE	**-**	It is defined at the time of the fast mode use. When undefined, it operates as a normal mode.
<Fast mode setting>		
DIVIDE	**13 * CLKFREQ / 10000**	384kbps
<Normal mode setting>		
DIVIDE	**50 * CLKFREQ / 10000**	100 kbps
<Common>		
CSDRDATA	**(DIVIDE - 1) * 2**	SDR00H
SCLLOWW	**(DIVIDE -13 + 4)/5**	SCL low time
SCLHIGHW	**(DIVIDE2 -13 + 4)/5**	SCL high time
R1EX24002A	**00H**	2K-bit EEPROM
R1EX24004A	**01H**	4K-bit EEPROM
R1EX24008A	**02H**	8K-bit EEPROM
R1EX24016A	**03H**	16K-bit EEPROM
R1EX24032A	**04H**	32K-bit EEPROM
R1EX24064A	**05H**	64K-bit EEPROM
R1EX24128B	**06H**	128K-bit EEPROM
R1EX24256B	**07H**	256K-bit EEPROM
R1EX24512B	**08H**	512K-bit EEPROM
EEPROM_MAX	**09H**	device end

Table 5.5 Constants for the Sample Program (1/2)

Constant	Setting	Description
BLKSIZE	**4**	The size of a block is defined. By default, it is 4 bytes/block. Although the block size can be changed into 8 or 16 bytes, if makes it too large, a possibility that futility will occur will become high.
SLAVE	**0A0H**	Slave address
MASK0	**00000000B**	Mask pattern
MASK2	**11111111B**	Mask pattern
EEPROM	**R1EX24256B**	Name of EEPROM. Its value is from 0 (2K bits) to 8 (512K bits).
BLKNO	32768/BLKSIZE	The number of blocks included in EEPROM. A value is decided by capacity setup of EEPROM. Here, the value at the time of choosing 256K-bit EEPROM is shown.
I2C_OK	00000000B	Normal end
PARA_ERR	01000100B	Parameter error
NO_ACK1	01000000B	No ACK response to Slave address
NO_ACK2	01000001B	Slave data error (write-protect)
BUS_ERR	01100000B	I_2C-Bus error
TRANSMIT	10000000B	Slave address transmission status
AFT_TX	TRANSMIT + 20H	Data writing status
RECEIVE	TRANSMIT + 40H	Receiving status
TRNSEND	00H	transmit completes
SVAMSK	11111110B	Mask R/W bit
RETRYCNT	9	Maximum number of dummy SCL pulses
CTXMODETxH	10000000B	Initialization of IIC register Refer to the item of the flowchart for details.
CRXMODERxH	01000000B	
CTRXMODEL	00010111B	
CSMRDATAH	00000000B	
CSMRDATAL	00100100B	

5.5 Variables

Table 5.6 lists the variables that are used in this sample program.

Table 5.6 Variables for the Sample Program

Variable	Outline
NEXTADR	Stores a transfer destination address.
EEPROMADDR	Specifies EEPROM address.
BLOCK_NUMBER	Specifies the block number which EEPROM of a controlled object wants to access.
WRITE_BUFF	Sets data to write in EEPROM.
READ_BUFF	A buffer which stores read data from EEPROM.
R_BUFF_END	Stores data which is received at the end of specified size.
STATUS	Stores the status of I2C interrupt control function by function.
DATACOUNT	Stores the number of data of I2C writing or reading.
SLAVEADDR	Slave address of I2C.

5.6 Function (Subroutine) List

Table 5.7 and Table 5.8 shows the function (subroutine).

Table 5.7 Function (External Function)

Function	Outline
PUTDATA	Writing start processing to the specified block
GETDATA	Reading start processing from the specified block
PUT_ CHK	Writing to EEPROM state check processing
GET_ CHK	Reading from EEPROM state check processing
WAIT_END	Waiting processing for the completion of access to EEPROM
StopCond	Execution of stop condition to I2C bus
R_IIC00_Init	Initialization of IIC00
SINITAU	Initialization of TAU01
IINTIIC00	IIC00 interrupt handler

Table 5.8 Function (Internal Processing Function)

Function	Outline
StartCond	Generates start condition to I2C bus
R_IIC00_send_Stop	Generates stop condition to I2C bus
R_IIC00_wait_bus	Releases I2C bus
R_IIC00_SCL_pulse	Outputs SCL signal of 1 pulse
R_IIC00_SCL_high	Raises SCL signal to high
R_IIC00_SCL_low	falls SCL signal to low
R_IIC00_SCL_Time	Secures the time of the pulse width of a SCL signal
get_slave_Addr	Calculates the address in I2C bus of EEPROM

5.7 Function (Subroutine) Specifications

This section describes the specifications for the functions (subroutines) that are used in this sample program.

5.7.1 External Function

[Function Name] PUTDATA

Outline	Writing start processing to the specified block	
Explanation	Checks the state of the bus and generates start condition. Then, calculates the address of a cell from a block number and transmits a slave address.	
Argument	BLOCK_NUMBER	: Block number which data will be written in
	WRITE_BUFF	: Data to write
Return value	Variable STATUS :	
	TRANSMIT	: Processing is started normally.
	PARA_ERR	: The error in specification (The block number was too large.
	BUS_ERR	: I2 C bus is not in the state which can be used.
Remarks	Return value is stored into the global variable STATUS.	

[Function Name] GETDATA

Outline	Reading start processing from the specified block	
Explanation	Checks the state of the bus and generates start condition. Then, calculates the address of a cell from a block number and transmits a slave address.	
Argument	BLOCK_NUMBER	: Block number which data will be read from
	READ_BUFF	: Read-out data storing area
Return value	Variable STATUS :	
	RECEIVE	: Processing is started normally.
	PARA_ERR	: The error in specification (The block number was too large.
	BUS_ERR	: I2 C bus is not in the state which can be used.
Remarks	Return value is stored into the global variable STATUS.	

[Function Name] PUT_CHK

Outline	Writing to EEPROM state check processing	
Explanation	Checks the state after start writing to EEPROM. If the processing is completed, Z flag is set.	
Argument	None	
Return value	Z flag = 1	: Processing is completed.
	Variable STATUS :	
	I2C_OK	: Normal end
	BUS_ERR	: The state which cannot use a bus.
	NO_ACK1	: No ACK response from a slave.
	TRANSMIT	: Under slave address transmission.
	AFT_TX	: Data transfer is completed and it is under writing.
	Z flag = 0	: Under continuation of processing.
Remarks	If MSB of STATUS is 1, it is under processing. If MSB of STATUS is 0, it is the completion of processing.	

[Function Name] GET_CHK

Outline	Reading from EEPROM state check processing	
Explanation	Checks the state after start reading from EEPROM. If the processing is completed, Z flag is set.	
Argument	None	
Return value	Z flag = 1	: Processing is completed.
	Variable STATUS :	
	I2C_OK	: Normal end
	BUS_ERR	: The state which cannot use a bus.
	NO_ACK1	: No ACK response from a slave.
	Z flag = 0	: Under continuation of processing.
	RECEIVE	: Under slave address transmission.
	RxData	: Under receiving data.
	RxLast	: Under receiving the final data.
Remarks	If MSB of STATUS is 1, it is under processing. If MSB of STATUS is 0, it is the completion of processing.	

[Function Name] WAIT_END

Outline	Waiting processing for the completion of access to EEPROM	
Explanation	Checks the value of variable STATUS, and waits for the completion of processing.	
Argument	None	
Return value	CY flag	= 0: Normal end = 1: Abnormal end
Remarks		

[Function Name] StopCond

Outline	Execution of stop condition to I2C bus	
Explanation	Performs stop condition to I2C bus. Then checks SDA signal, and if SDA signal is low, performs the bus release processing. After finishing bus release, generates stop condition again.	
Argument	None	
Return value	CY flag	= 0: Normal end = 1: Abnormal end
Remarks	The pin is output state. IIC00 is stopped.	

[Function Name] R_IIC00_Init

Outline	Initialization of IIC00	
Explanation	Initializes the interval timer for confirming the completion of writing to EEPROM. The interval is set to 100us at the time of the fast mode, and to 400us at the time of the normal mode.	
Argument	None	
Return value	None	
Remarks	None	

[Function Name] SINITAU

Outline	Initialization of TAU01	
Explanation	Initializes the interval timer of 100μs for confirming the completion of writing to EEPROM.	
Argument	None	
Return value	None	
Remarks	None	

5.7.2 Internal Processing Function

[Function Name] StartCond

Outline	Generates start condition to I2C bus
Explanation	Generates start condition to I2C bus by manipulating CKO0 and SO0 bit, and makes IIC00 operation enabled state.
Argument	None
Return value	None
Remarks	IIC00 will be in the state of operation enabled by transmitting permission.

[Function Name] R_IIC00_send_Stop

Outline	Generates stop condition to I2C bus
Explanation	Stopping operation of IIC00 and adjusting timing with using software, generates stop condition (raises SDA signal, raises SCL signal and then raises SDA signal) to I2C bus by manipulating CKO0 and SO0 bit.
Argument	None
Return value	None
Remarks	None

[Function Name] R_IIC00_wait_bus

Outline	Releases I2C bus	
Explanation	Confirms that SDA signal is high. If SDA signal is low, checks the state of the SDA signal outputting 10 false clocks to SCL signal. Even if outputs 9 clocks, when a SDA signal does not become high, it is considered as an error.	
Argument	None	
Return value	Variable STATUS :	
	I2C_OK	: Confirmation of bus release (SDA moves to high.)
	BUS_ERR	: SDA still remains in low.
Remarks	This is a countermeasure against the process discontinuation after data output from EEPROM because of CPU reset and so on. Normally, generates stop condition after this.	

[Function Name] R_IIC00_SCL_pulse

Outline	Outputs SCL signal of 1 pulse
Explanation	Outputs low and high to SCL signal by manipulating CKO0 register. After switching SCL signal, the time for securing the low width or high width of a SCL signal is secured.
Argument	None
Return value	None
Remarks	None

[Function Name] R_IIC00_SCL_high

Outline	Raises SCL signal to high	
Explanation	In order to generate stop condition, manipulates CKO0 register to make SCL signal into high, and secures the time for securing high width.	
Argument	None	
Return value	None	
Remarks	None	

[Function Name] R_IIC00_ SCL_low

Outline	Falls SCL signal to low	
Explanation	In order to move to the communication operation after generating start condition, manipulates CKO0 register to make SCL signal into low, and secures the time for securing low width.	
Argument	None	
Return value	None	
Remarks	None	

[Function Name] R_IIC00_SCL_Time

Outline	Secures the time of the pulse width of a SCL signal	
Explanation	Secures pulse width of SCL signal (1.3us).	
Argument	None	
Return value	None	
Remarks	None	

[Function Name] get_slave _Addr

Outline	Calculates the address in I2C bus of EEPROM	
Explanation	Calculates the information to incorporate a memory cell address of EEPROM into a slave address of I2C bus from the information which is set in variable area for internal EEPROM control. And calculates a slave address which is used at actual I2C bus.	
Argument	None	
Return value	CY flag :	= 0: The block number is normal. = 1: The block number is error.
Remarks	None	

5.8 Flowcharts

Figure 5.5 shows the overall flowchart of the sample program described in this application note.

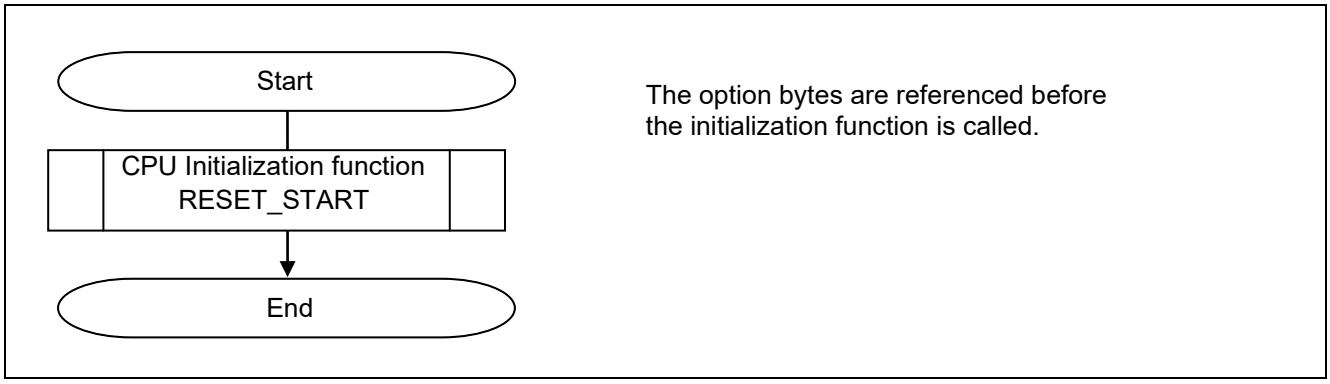


Figure 5.5 Overall Flowchart

5.8.1 CPU Initialization Function

Figure 5.6 shows the flowchart for the CPU initialization function.

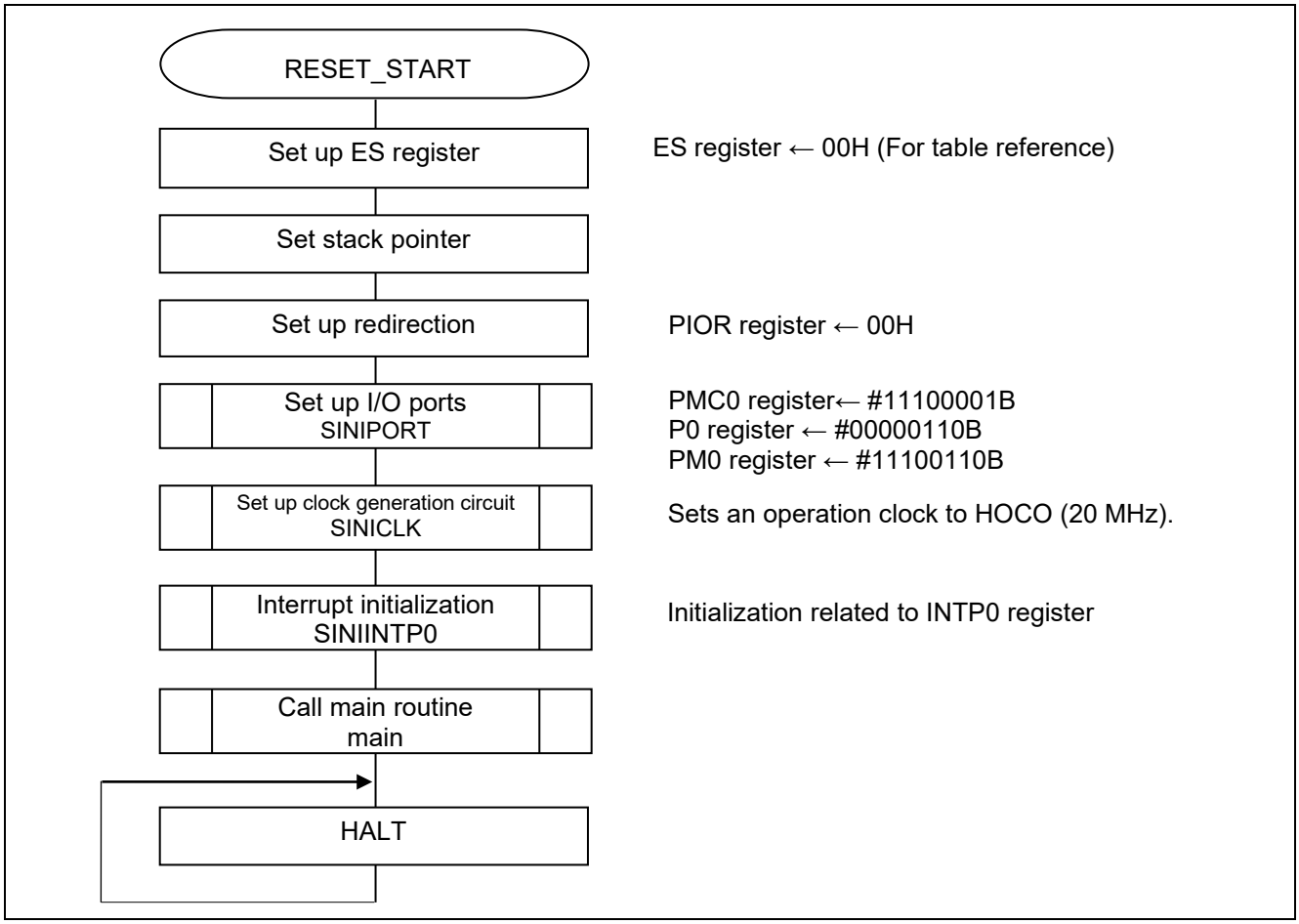


Figure 5.6 CPU Initialization Function

5.8.2 I/O Port Setup

Figure 5.7 shows the flowchart for I/O port setup.

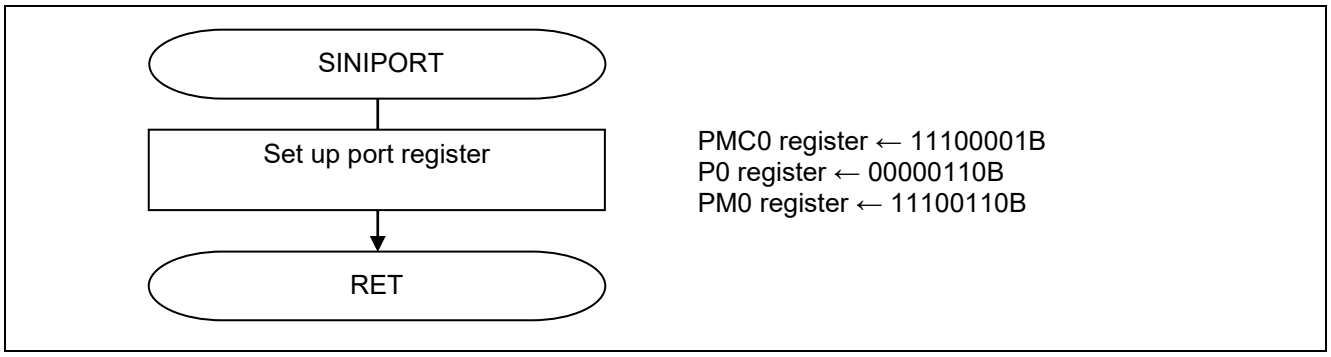


Figure 5.7 I/O Port Setup Function

Note: Refer to the section entitled "Flowcharts" in RL78/G10 Initialization Application Note (R01AN1454E) for the configuration of the unused ports.

Port setting

- Port mode control register 0 (PMC0)

  Switching between analog input and digital I/O

- Port registers 0 (P0)

  Setup of an output latch of each port

- Port mode registers 0 (PM0)

  Selection of the I/O mode of each port

Symbol：PMC0

7	6	5	4	3	2	1	0
1	1	1	PMC04	PMC03	PMC02	PMC01	1
1	1	1	**0**	**0**	**0**	**0**	1

Bits 4 to 1

PMC0n	P0n pin digital I/O/analog input selection
0	**Digital I/O (alternate function other than analog input)**
1	Analog input

Symbol : P0

7	6	5	4	3	2	1	0
0	0	0	P04	P03	P02	P01	P00
0	0	0	0	0	**1**	**1**	**0**

Bits 2 and 1

P0n	P0n pin output data control (in output mode)
0	Output mode (output buffer on)
1	**Input mode (output buffer off)**

Bit 0

P00	P00 pin output data control (in output mode)
0	**Output mode (output buffer on)**
1	Input mode (output buffer off)

Symbol : PM0

7	6	5	4	3	2	1	0
1	1	1	PM04	PM03	PM02	PM01	PM00
1	1	1	0	0	**1**	**1**	**0**

Bits 2 and 1

PM0n	P0n pin I/O mode selection
0	Output mode (output buffer on)
1	**Input mode (output buffer off)**

Bit 0

PM0n	P00 pin I/O mode selection
0	**Output mode (output buffer on)**
1	Input mode (output buffer off)

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.3 Clock Generation Circuit

Figure 5.8 shows the flowchart for clock generation circuit setup.

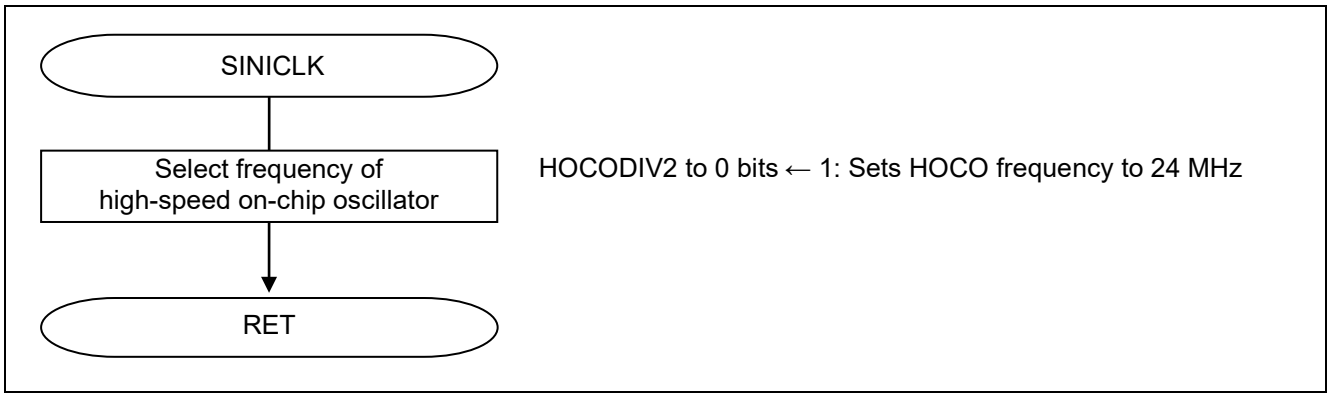


Figure 5.8 Clock Generation Circuit Setup

Selection of high-speed on-chip oscillator frequency

・High-speed on-chip oscillator frequency selection register (HOCODIV)
Selects the frequency of high-speed on-chip oscillator.

Symbol：HOCODIV

7	6	5	4	3	2	1	0
0	0	0	0	0	HOCODIV2	HOCODIV1	HOCODIV0
0	0	0	0	0	**0**	**0**	**1**

Bits 2 to 0

HOCODIV 2	HOCODIV 1	HOCODIV 0	High-speed on-chip oscillator clock frequency selection
0	**0**	**1**	**20MHz**
0	1	0	10MHz
0	1	1	5MHz
1	0	0	2.5MHz
1	0	1	1.25MHz
Other than above			Setting prohibited

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.4 Interrupt Setup

Figure 5.9 shows the flowchart for setting up the interrupt.

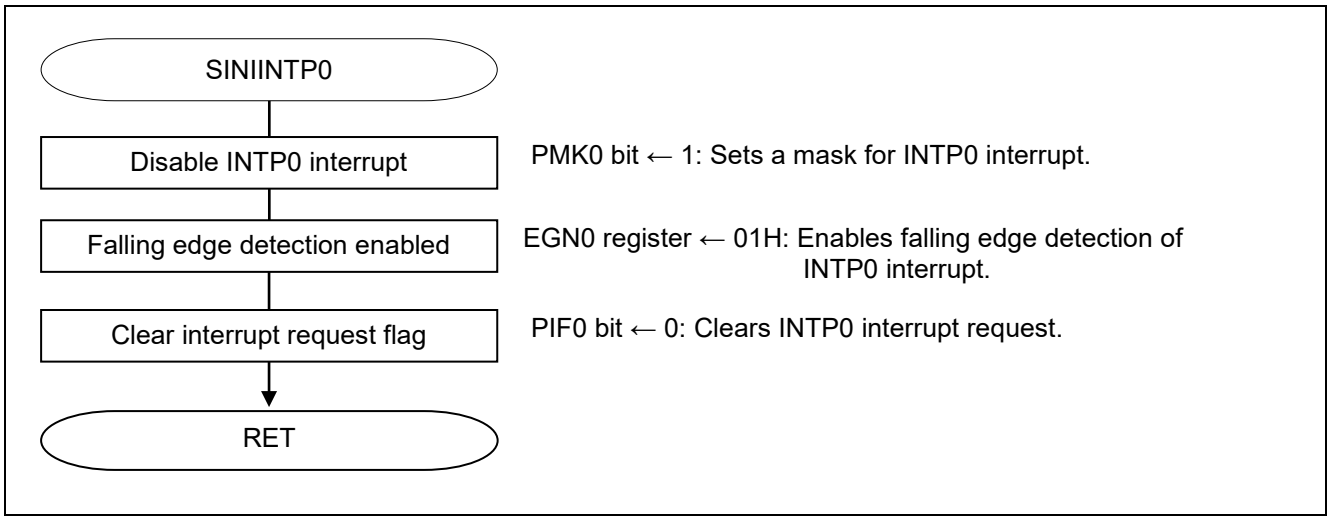


Figure 5.9 Interrupt Setup

Setup INTP0 pin edge detection

・Interrupt mask flag registers (MK0L)

Sets up the interrupt mask flag.

・External interrupt falling edge enable register 0 (EGN0)

This register specifies the valid edge for INTP0

・Interrupt request flag registers (IF0L)

Clears interrupt request flag.

Symbol：MK0L

7	6	5	4	3	2	1	0
TMMK00	TMMK01H	SREMK0	SRMK0	STMK0 CSIMK00 IICMK00	PMK1	PMK0	WDTIMK
x	x	x	x	x	x	**1**	x

Bit 1

PMK0	Interrupt servicing control
0	Interrupt servicing enabled
1	**Interrupt servicing disabled**

Symbol : EGN0

7	6	5	4	3	2	1	0
0	0	0	0	EGN3[Note]	EGN2[Note]	EGN1	EGN0
0	0	0	0	x	x	x	**1**

EGP0	EGN0	INTP0 pin valid edge selection
0	0	Edge detection disabled
0	**1**	**Falling edge**
1	0	Rising edge
1	1	Both rising and falling edges

Note 16-pin products only.

Symbol : IF0L

7	6	5	4	3	2	1	0
TMIF00	TMIF01H	SREIF0	SRIF0	STIF0 CSIIF00 IICIF00	PIF1	PIF0	WDTIIF
x	x	x	x	x	x	**0**	x

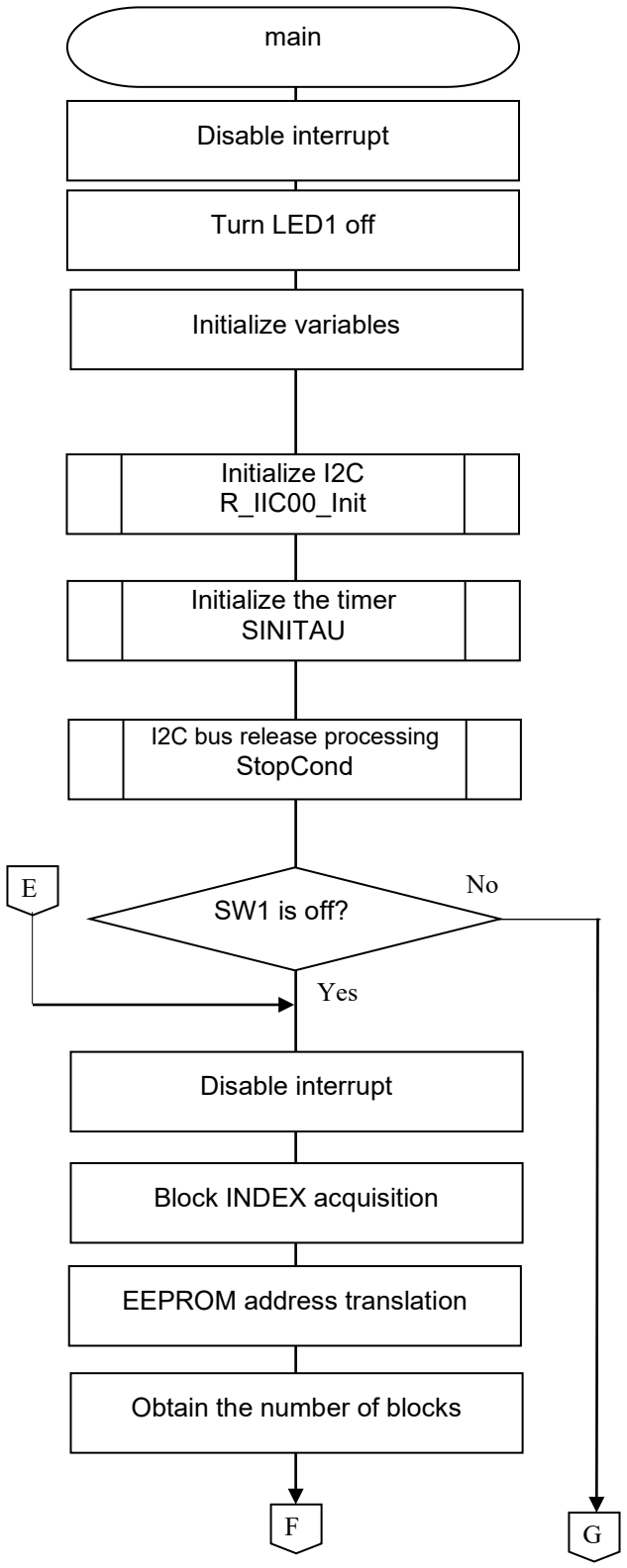
Bit 1

PIF0	Interrupt request flag
0	**No interrupt request signal is generated**
1	Interrupt request is generated, interrupt request status

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.5 Main Processing

Figure 5.10, Figure 5.11, Figure 5.12, and Figure 5.13 show the flowchart for main processing.



DI bit ← 1

Turns LED1 off.
0 bit of P0 register ← 1

Clears using memory areas.
Variable: BLOCK_NUMBER←0
Variable: WRITE_BUFF←0 Variable: STATUS←0
Variable: READ_BUFF←0 Variable: DATACOUNT←0
Variable: R_BUFF_END←0 Variable: SLAVEADDR←0

Initializes the timer with the following setup values.
Fast mode: 100us
Normal mode: 400us

Performs the I2C bus release processing.

When SW1 is off, performs writing processing to EEPROM.
When SW1 is pressed, moves to No judgment.
When SW1 is pressed, performs reading processing from EEPROM.

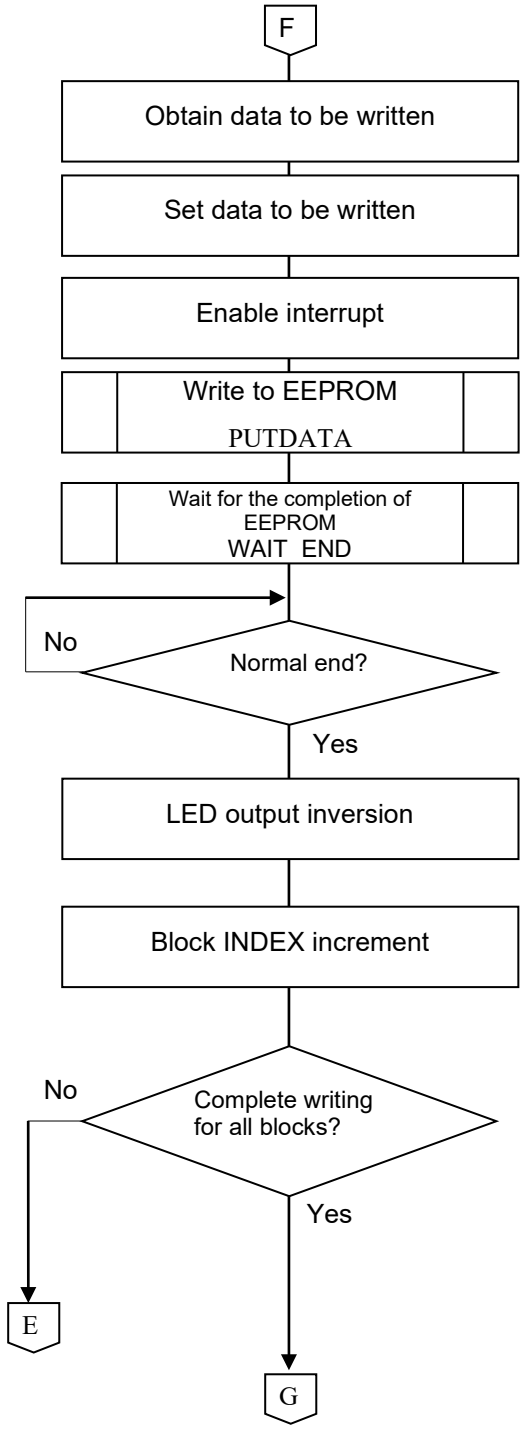
IE bit ← 0

AX register ← Variable: BLOCKNUMBER

Shifts AX register to the 2 to 4-bit left according to block size.

B register ←Constant: BLKSIZE

Figure 5.10 Main Processing (1/4)



Write processing to EEPROM.
A register ← A register + variable: READ_BUFF-1
A register ← A register + B register

Sets data which to be written to EEPROM to the memory.

IE bit ← 1

Calls writing function to EEPROM, and

starts interrupt processing.

Waits for the completion of writing to EEPROM.

An infinite loop is carried out when the processing ends in an error.

When CY flag is set as the return value of WAIT_END, moves to the No judgment.

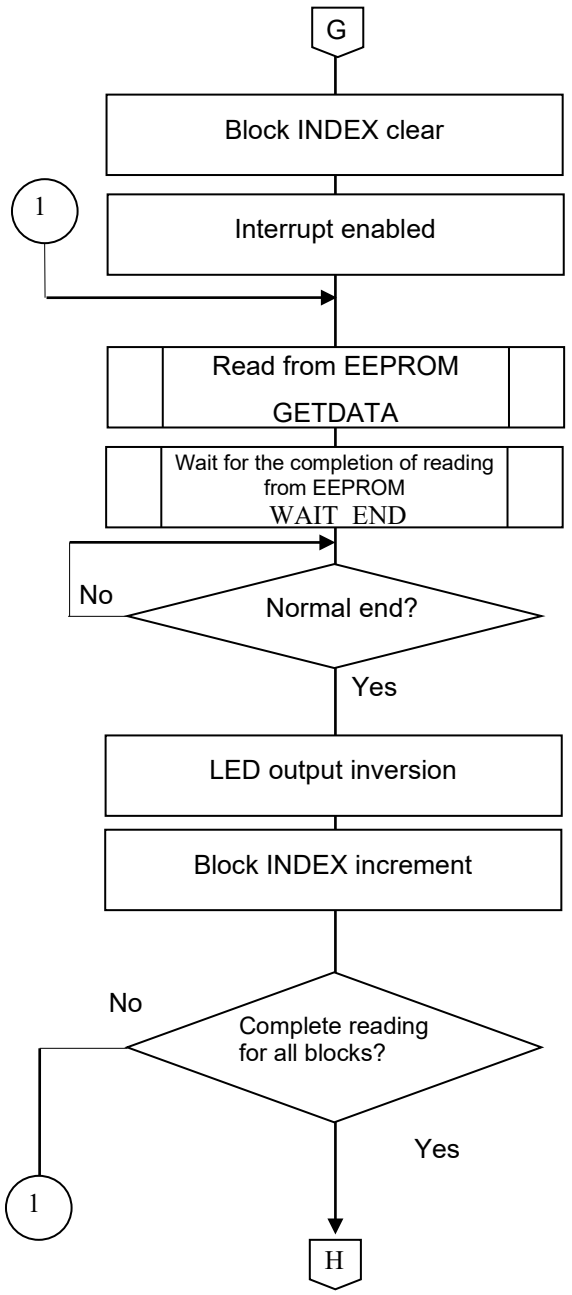
P0 register ← P0 register ^ 0x01

Variable: BLOCKNUMBER ← BLOCKNUMBER+1

AX register ← Variable: BLOCKNUMBER

Repeats the write processing for the number of EEPROM blocks.
Compares AX register and the constant: BLKNO.
If CY flag is set (constant: BLKNO is larger than AX register value) moves to the No judgment.

Figure 5.11 Main Processing (2/4)



Reads from EEPROM.

Variable: BLOCKNUMBER←0

IE bit ← 1

Calls reading function to EEPROM, and

starts interrupt processing.

Waits for the completion of reading from EEPROM.

An infinite loop is carried out when the processing ends in an error.

When CY flag is set as the return value of WAIT_END, moves to the No judgment.

P0 register ← P0 register ^ 0x01

Variable: BLOCKNUMBER←BLOCKNUMBER+1

AX register← Variable: BLOCKNUMBER

Repeats the read processing for the number of EEPROM blocks.

Compares AX register and the constant: BLKNO. If CY flag is set (constant: BLKNO is larger than AX register value) moves to the No judgment.

Figure 5.12 Main Processing (3/4)

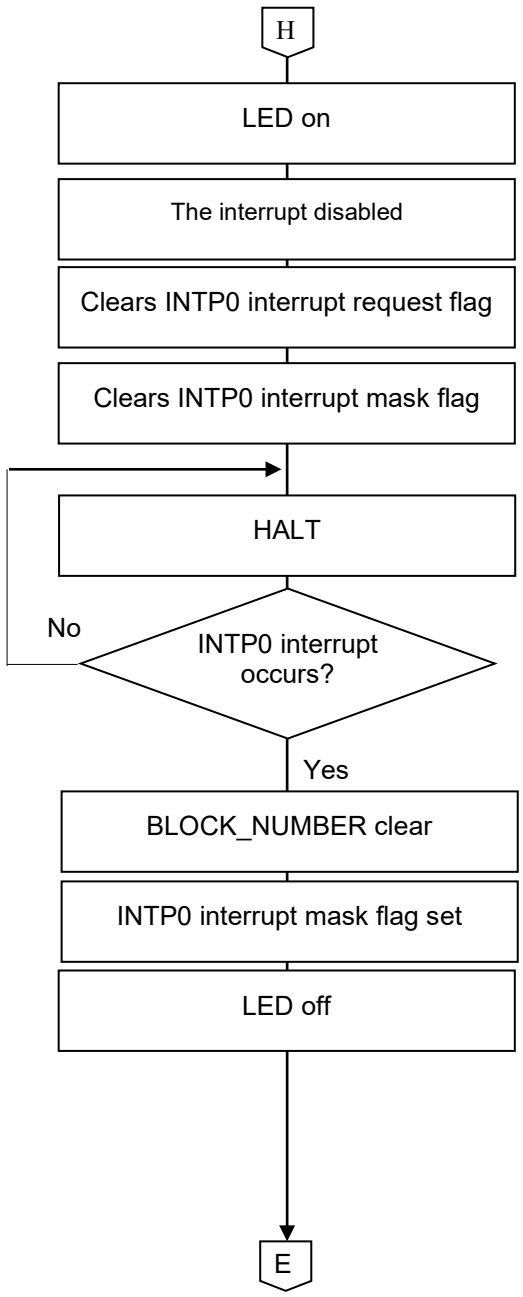


Figure 5.13 Main Processing (4/4)

5.8.6 IIC00 Initialization

Figure 5.14 shows the IIC00 initialization (R_IIC00_Init). Chanel 0 of SAU0 is set to IIC00 at IIC initialization. Fast mode is selected for IIC00 setup. In the simplified I2C function of SAU, since duty of a transfer clock cannot be set up freely, it is necessary to set up a transmission clock so that the low level width of a SCL signal may satisfy the specification value (1.3 ms) of I2 C bus. Therefore, fast mode becomes a transfer rate of about 384k bps instead of 400k bps.

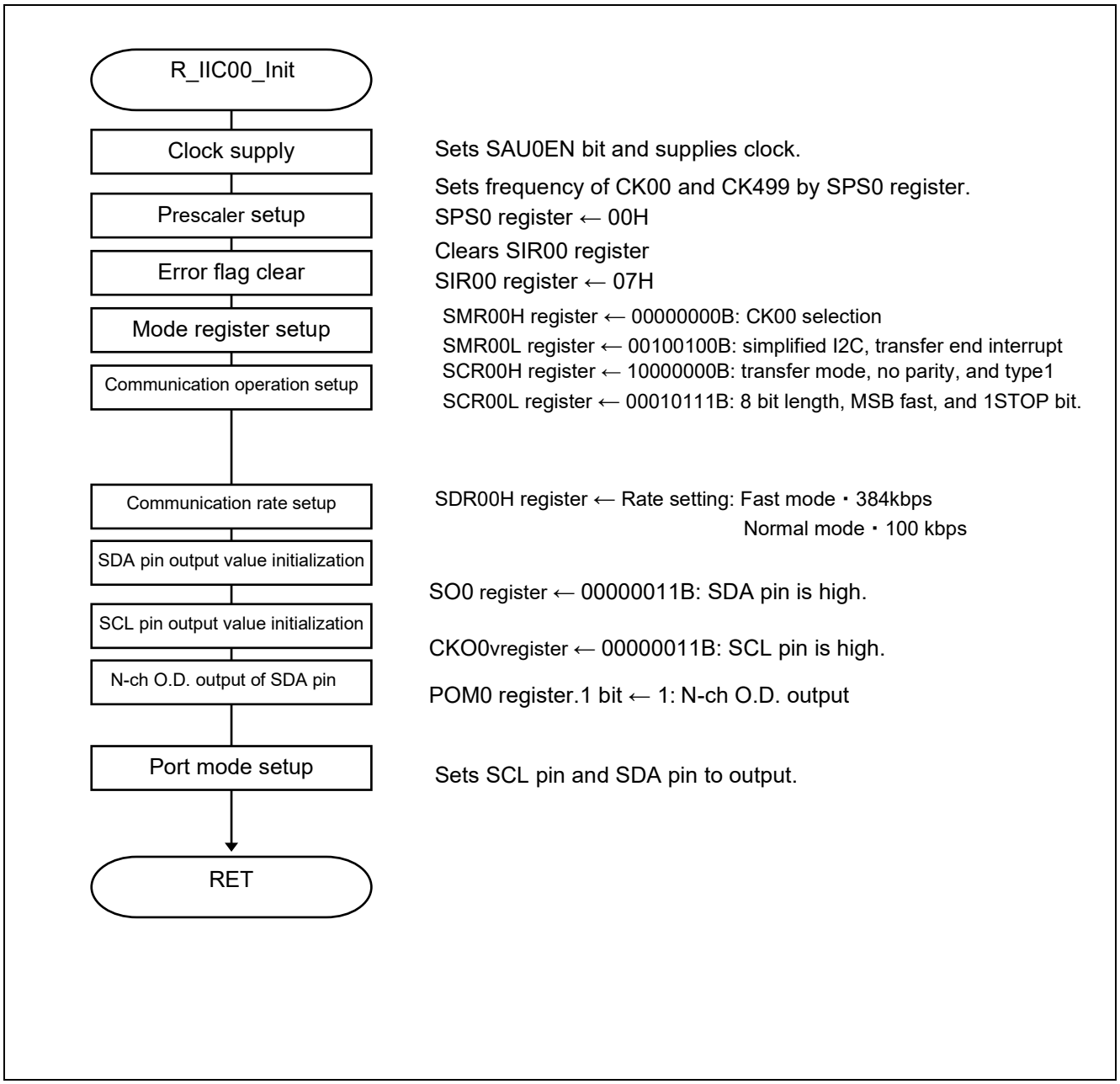


Figure 5.14 IIC00 Initialization

Start supplying clock to the serial array unit 0.

・Peripheral enable register 0 (PER0)

Starts supplying clock to the serial array unit 0.

Symbol：PER0

7	6	5	4	3	2	1	0
TMKAEN Note	CMPEN Note	ADCEN	IICA0EN Note	0	SAU0EN	0	TAU0EN
x	x	x	x	0	**1**	0	X

Note 16-pin products only.

Bit 2

SAU0EN	Control of serial array unit 0 input clock supply
0	Stops supply of input clock. · SFR used by serial array unit 0 cannot be written. · Serial array unit 0 is in the reset status.
1	Enables input clock supply. · SFR used by serial array unit 0 can be read/written.

Setup of the serial clock frequency.

・**Serial clock select register 0** (SPS0)

Selects the operation clock of serial array unit 0.

Symbol：SPS0

7	6	5	4	3	2	1	0
PRS013	PRS012	PRS011	PRS010	PRS003	PRS002	PRS001	PRS000
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits 7 to 0

PRS 0n3	PRS 0n2	PRS 0n1	PRS 0n0	Section of operation clock (CKn) (n = 0、 1)					
					f_{CLK} = 1.25MHz	f_{CLK} = 2.5MHz	f_{CLK} = 5MHz	f_{CLK} = 10MHz	f_{CLK} = 20MHz
0	**0**	**0**	**0**	**f_{CLK}**	1.25 MHz	2.5 MHz	5 MHz	10 MHz	**20 MHz**
0	0	0	1	$f_{CLK}/2$	625 kHz	1.25 MHz	2.5 MHz	5 MHz	10 MHz
0	0	1	0	$f_{CLK}/2^2$	313 kHz	625 kHz	1.25 MHz	2.5 MHz	5 MHz
0	0	1	1	$f_{CLK}/2^3$	156 kHz	313 kHz	625 kHz	1.25 MHz	2.5 MHz
0	1	0	0	$f_{CLK}/2^4$	78 kHz	156 kHz	313 kHz	625 kHz	1.25 MHz
0	1	0	1	$f_{CLK}/2^5$	39 kHz	78 kHz	156 kHz	313 kHz	625 kHz
0	1	1	0	$f_{CLK}/2^6$	19.5 kHz	39 kHz	78 kHz	156 kHz	313 kHz
0	1	1	1	$f_{CLK}/2^7$	9.8 kHz	19.5 kHz	39 kHz	78 kHz	156 kHz
1	0	0	0	$f_{CLK}/2^8$	4.9 kHz	9.8 kHz	19.5 kHz	39 kHz	78 kHz
1	0	0	1	$f_{CLK}/2^9$	2.5 kHz	4.9 kHz	9.8 kHz	19.5 kHz	39 kHz
1	0	1	0	$f_{CLK}/2^{10}$	1.22 kHz	2.5 kHz	4.9 kHz	9.8 kHz	19.5 kHz
1	0	1	1	$f_{CLK}/2^{11}$	625 Hz	1.22 kHz	2.5 kHz	4.9 kHz	9.8 kHz
1	1	0	0	$f_{CLK}/2^{12}$	313 Hz	625 Hz	1.22 kHz	2.5 kHz	4.9 kHz
1	1	0	1	$f_{CLK}/2^{13}$	152 Hz	313 Hz	625 Hz	1.22 kHz	2.5 kHz
1	1	1	0	$f_{CLK}/2^{14}$	78Hz	152 Hz	313 Hz	625 Hz	1.22 kHz
1	1	1	1	$f_{CLK}/2^{15}$	39Hz	78Hz	152 Hz	313 Hz	625 Hz

Clears Serial flag clear trigger register 0

・Serial flag clear trigger register 0n (SIR0n)

Clears the error flag of SAU

Symbol：SIR0n

7	6	5	4	3	2	1	0
0	0	0	0	0	FECT0n Note	PECT0n	OVCT0n
0	0	0	0	0	**1**	**1**	**1**

Note Provided in the SIR01 register only.

Bit 2

FECT0n	**Clear trigger of framing error of channel n**
0	Not cleared
1	**Clears the FEF0n bit of the SSR0n register to 0.**

Bit 1

PECT0n	**Clear trigger of parity error flag of channel n**
0	Not cleared
1	**Clears the PEF0n bit of the SSR0n register to 0.**

Bit 0

OVCT0n	**Clear trigger of overrun error flag of channel n**
0	Not cleared
1	**Clears the OVF0n bit of the SSR0n register to 0.**

Setup of the transmission channel operation

・Serial mode register 00 (SMR00H, SMR00L)

Interrupt source

Operation mode

Selection of the transfer clock.

Selection of fMCK.

Symbol：SMR00H

7	6	5	4	3	2	1	0
CKS00	CCS00	0	0	0	0	0	0
0	**0**	0	0	0	0	0	0

SMR00L

7	6	5	4	3	2	1	0
0	0	1	0	0	MD002	MD001	MD000
0	0	1	0	0	**1**	**0**	**0**

Bit 7(SMR00H)

CKS00	Selection of operation clock (fMCK) of channel 0
0	**Operation clock CK00 set by the SPS0 register**
1	Operation clock CK01 set by the SPS0 register

Bit 6(SMR00H)

CCS00	Selection of transfer clock (fTCLK) of channel 0
0	**Divided operation clock fMCK specified by the CKS00 bit**
1	Clock input fSCK from the SCKp pin (slave transfer in CSI mode)

Bits 2 and 1(SMR00L)

MD002	MD001	Setting of operation mode of channel 0
0	0	CSI mode
0	1	UART mode
1	**0**	**Simplified I2C mode**
1	1	Setting prohibited

Bit 0(SMR00L)

MD000	Selection of interrupt source of channel 0
0	**Transfer end interrupt**
1	Buffer empty interrupt (Occurs when data is transferred from the SDR00L register to the shift register.)

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of communication operation setting of transmission channel

・Serial communication operation setting register 00 (SCR00H, SCR00L)

Sets data length, data transfer sequence, and operation data.

Symbol：SCR00H

7	6	5	4	3	2	1	0
TXE00	RXE00	DAP00	CKP00	0	EOC00	PTC001	PTC000
1	**0**	0	0	0	**0**	0	0

Bits 7 and 6

TXE00	RXE00	Setting of operation mode of channel 0
0	0	Disable communication
0	1	Reception only
1	**0**	**Transmission only**
1	1	Transmission/reception

Bit 2

EOC00	Selection of masking of error interrupt signal (INTSRE0)
0	**Disables generation of error interrupt INTSRE0 (INTSR0 is generated).**
1	Enables generation of error interrupt INTSRE0 (INTSR0 is not generated if an error occurs).

Setup of transmission channel transfer clock

・Serial data register 00 (SDR00H, SDR00L)

Transfer clock frequency: Undefined

Symbol：SDR00H (Division ratio setting register)

7	6	5	4	3	2	1	0
0/1	**0/1**	**0/1**	**0/1**	**0/1**	**0/1**	**0/1**	0

SDR00L (Transmit/receive buffer register)

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

Bits 7 to 1(SDR00H)

SDR00H[7:1]							Transfer clock setting by dividing the operating clock (fMCK)
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	$f_{MCK}/8$
・ ・	・ ・	・ ・	・ ・	・ ・	・ ・	・ ・	・ ・
1	1	1	1	1	1	0	$f_{MCK}/254$
1	1	1	1	1	1	1	$f_{MCK}/256$

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of initial output level of transmission channel

・Serial output register 0 (SO0)
Setup of initial output level

Symbol：SO0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SO01 Note	SO00
0	0	0	0	0	**0**	**1**	**1**

Note 16-pin products only.

Bits 1 and 0

SO01	SO00	**Serial data output of channel 0**
1	**1**	**Serial data output value is “1”.**
0	0	Serial data output value is “0”.

Setup of serial clock output of transmission channel

・Serial clock output register (CKO0)
Setup of serial clock output

Symbol：CKO0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	CKO01	CKO00
0	**0**	0	0	0	0	**1**	**1**

Bits 1 and 0

CKO01	CKO00	**Serial clock output of channel 0**
1	**1**	**Serial clock output value is “1”.**
0	0	Serial clock output value is “0”.

Setup of port output mode register

・Port output mode register 0 (POM0)
Sets a N-ch open-drain output (VDD tolerant) mode for transmitting data

Symbol：POM0

7	6	5	4	3	2	1	0
POM07 Note	POM06 Note	0	0	0	0	POM01	POM00
x	x	0	0	0	**0**	**1**	x

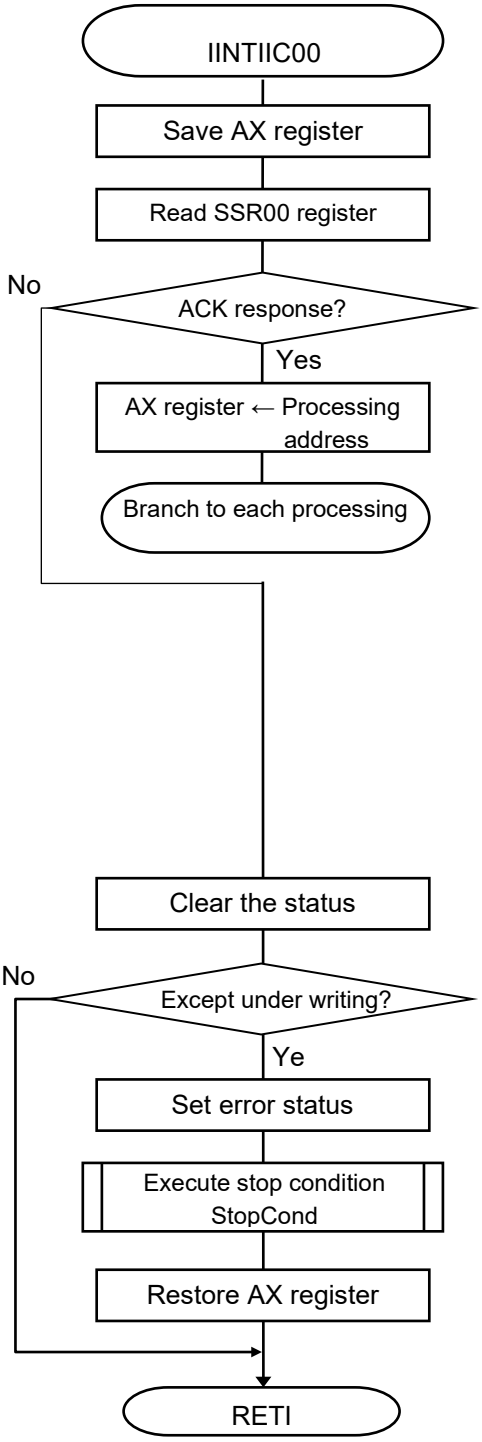
Note 16-pin products only.

Bit 1

POM01	**P01 pin output mode selection**
0	Normal output mode
1	**N-ch open-drain output (VDD tolerant) mode**

5.8.7 INTIIC00 Interrupt Entry Processing

Figure 5.15 shows INTIIC00 interrupt entry processing (IINTIIC00). Confirms a response from EEPROM by INTIIC00 interrupt processing. If it is ACK response, branches to the address (address of the routine which actually processes) of the value of the variable NEXTADR stored in advance. If it is NACK response, checks the state of writing of EEPROM. When it is under writing (Variable STATUS is AFT_TX), processing is ended as it is, and in the case of others, error processing is performed.



The contents of the AX register used by processing are saved to a stack area.

In order to check the state of I2C communication, the contents of SSR00

If it is NACK response, move to communication state check.
If 1 bit of SSR00 register is set, it is No judging.
Variable: Reads the value of NEXTADR into AX register

Branches to the address saved in AX register.
Branch destinations are indicated to the following flow charts.

- R_IIC00_Tx_addr1 : Upper address transmitting
- R_IIC00_Tx_addr2 : Lower address transmitting
- R_IIC00_TxDataST : Start of data reception processing
- R_IIC00_TxData : Data transmission
- R_EEPROM_WT_END : Completion of data writing
- R_IIC00_Rx_RST : Restart
- R_IIC00_RxDataST : Start of I2C data reception
- R_IIC00_RxData : I2C data reception
- R_IIC00_Rx_Last : Final data reception

Clears the error status of SSR00 register.

If it is under writing at EEPROM (variable: STATUS==constant: AFT_TX), the processing is ended as a No judging.

Variable: Sets the error status to STATUS.

Executes stop condition to I2C bus.

Restores the contents of AX register from a stack area.

Figure 5.15 INTIIC00 Interrupt Entry Processing

5.8.8 EEPROM Upper Address Transmitting

Figure 5.16 shows EEPROM upper address transmitting (R_IIC00_Tx_addr1). It is a processing routine which is used at EEPROM not less than 32K bits. In control of EEPROM of 16K bit or less, by $IF sentence, it will be in the state where anything does not have processing, and EEPROM lower address transmitting processing (R_IIC00_Tx_addr2) will start directly.

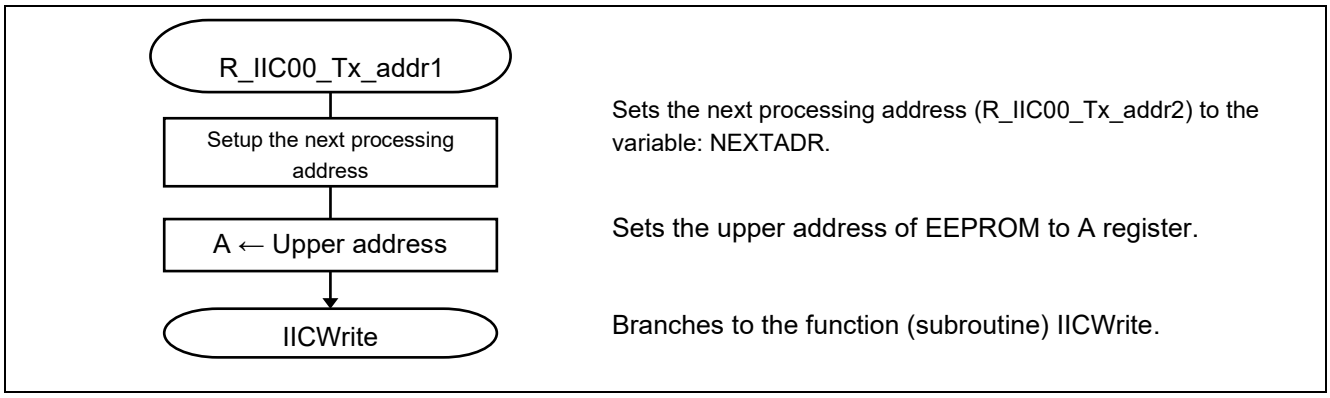


Figure 5.16 EEPROM Upper Address Transmitting (R_IIC00_Tx_addr1)

5.8.9 EEPROM Lower Address Transmitting

Figure 5.17 shows EEPROM lower address transmitting (R_IIC00_Tx_addr2). The memory cell to EEPROM selected in the slave address is addressed. In 32K bits or more EEPROM, it becomes transmission (specified) of a lower address.

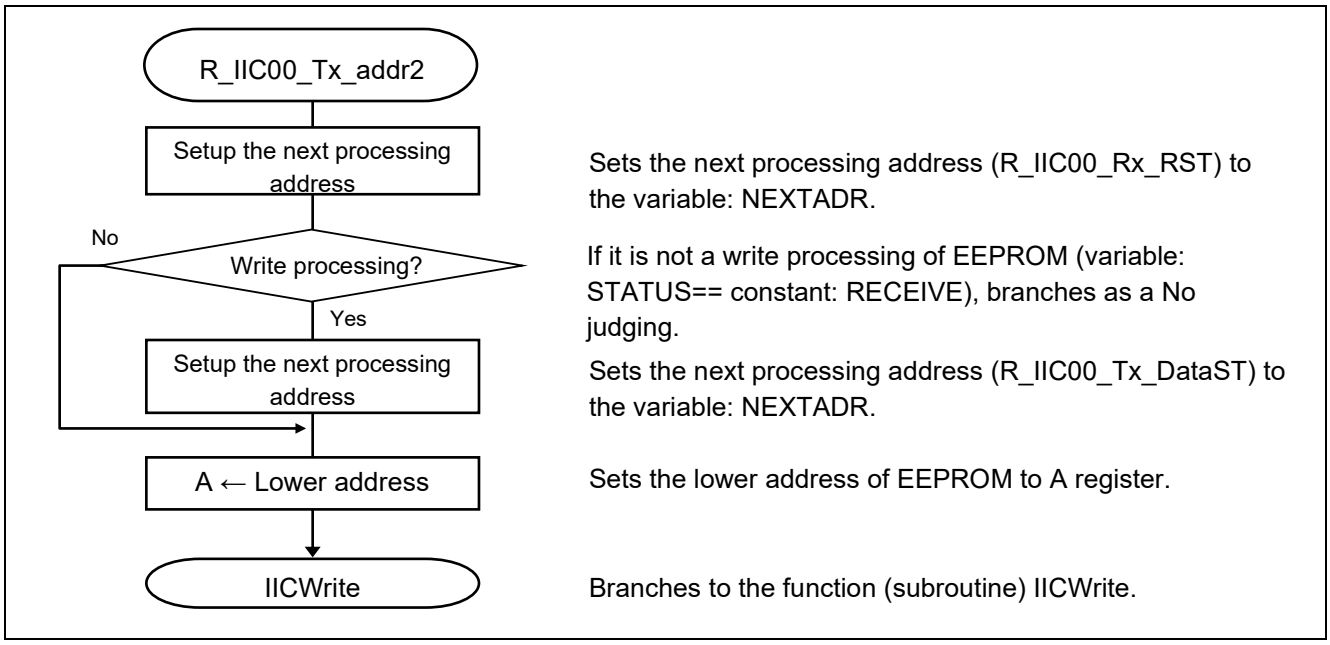


Figure 5.17 EEPROM Lower Address Transmitting (R_IIC00_Tx_addr2)

5.8.10 Restart Processing Setup

Figure 5.18 shows the restart processing (R_IIC00_Rx_RST). After the completion of address transmission of EEPROM, in order to read data from EEPROM, the communication direction of I2C bus is changed to master reception.

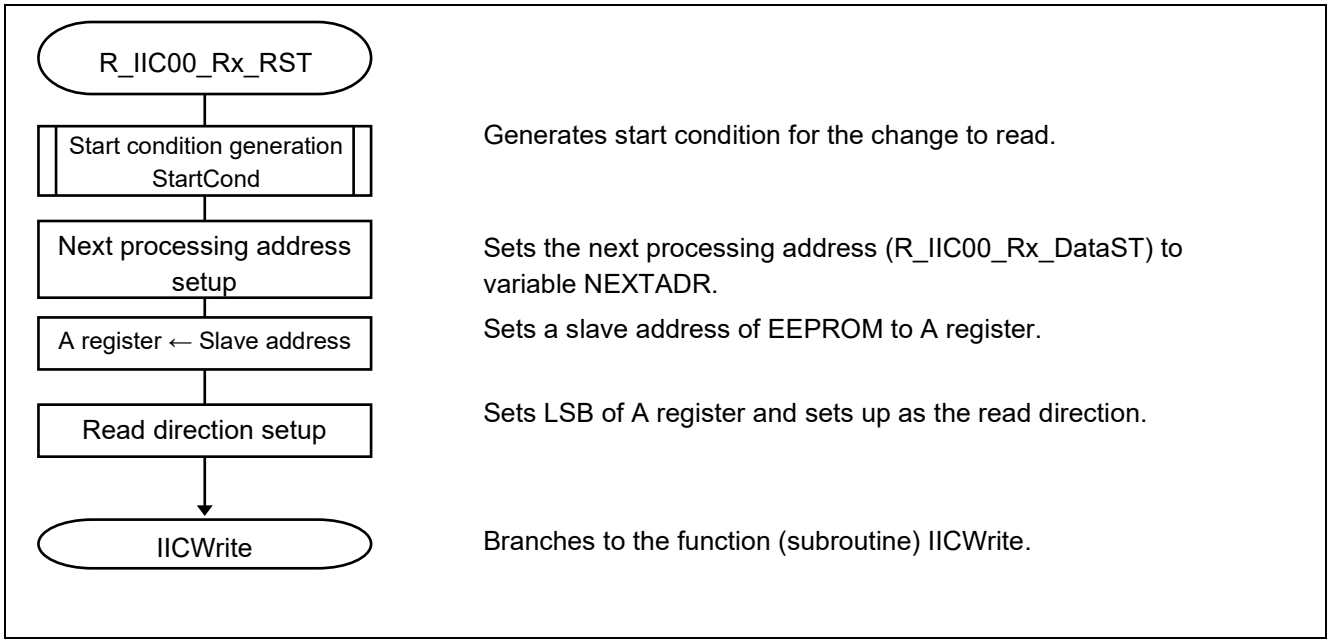


Figure 5.18 Restart Processing

5.8.11 I2C Write Processing

Figure 5.19 shows I2C write processing (IICWrite). This processing is a common routine for escaping from interrupt processing, after transmitting the data of A register to I2C bus.

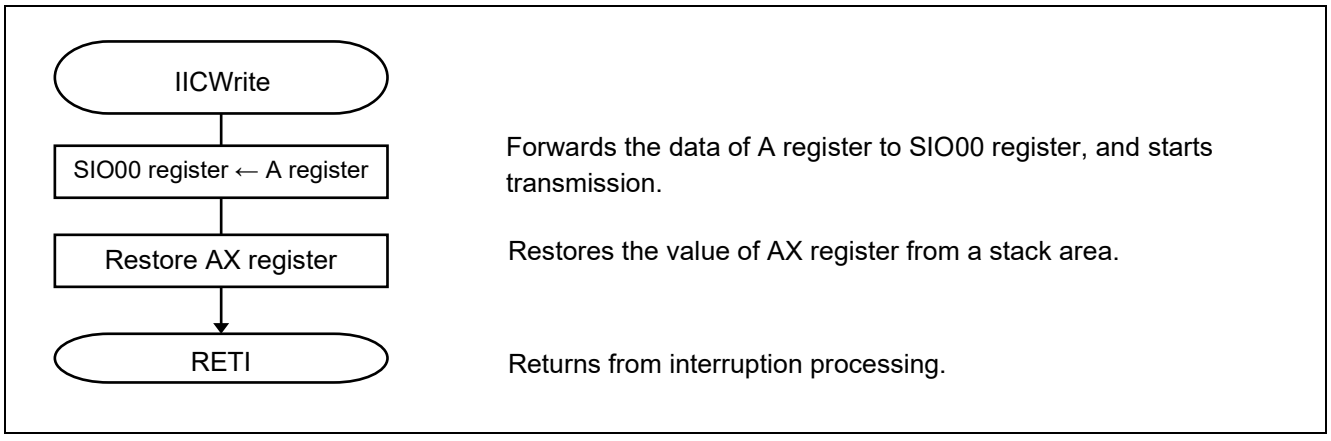


Figure 5.19 I2C Write Processing

5.8.12 I2C Data Reception Start Processing

Figure 5.20 shows the I2C data reception start processing (R_IIC00_RxDataST). This is a processing routine of the completion of slave address transmission in master reception after restarting. Stops IIC00, changes the mode of IIC00 from transmission to reception, and reboots IIC00. Writes dummy data to SIO00 and starts receiving processing.

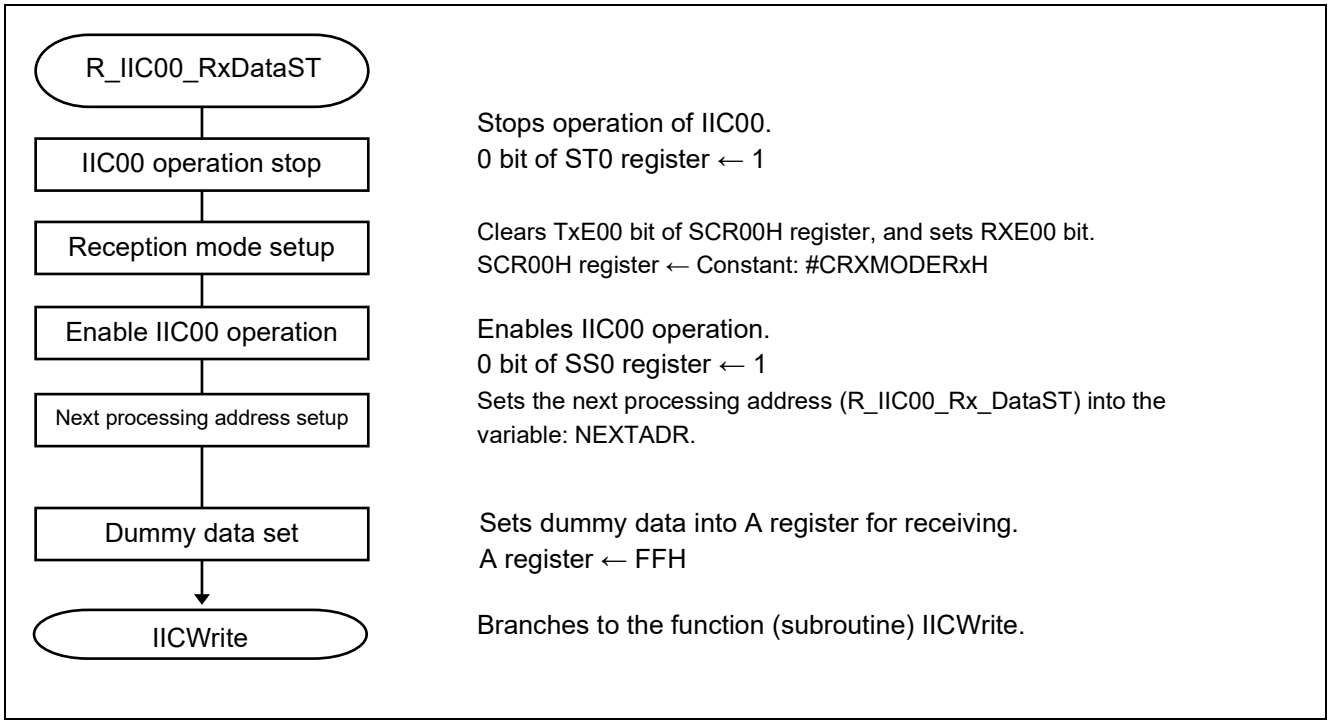


Figure 5.20 I2C Data Reception Start Processing

5.8.13 I2C Data Reception Processing

Figure 5.21 shows I2C data reception processing (R_IIC00_RxData). Stores received data to a buffer. When the next received data are the final data, in order that a master responds a NACK, disables the data output of IIC00 and starts reception.

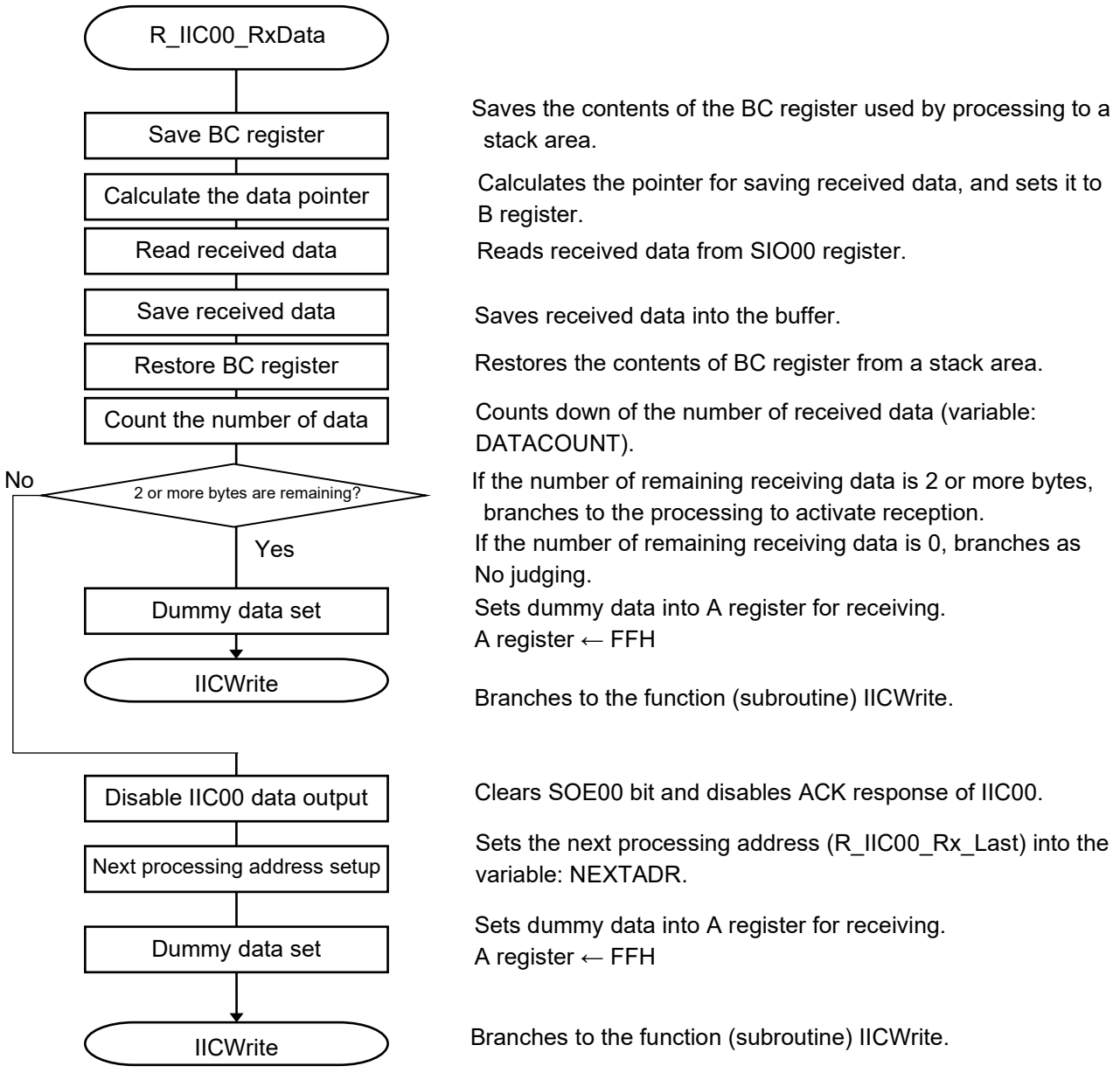


Figure 5.21 I2C Data Reception Processing

5.8.14 Final Data Reception Processing

Figure 5.22 shows the final data reception processing (R_IIC00_RxLast). Stores the final received data to a buffer, and stops receiving operation of IIC00. Generates stop condition and sets a status (variable STATUS) to the completion of receiving.

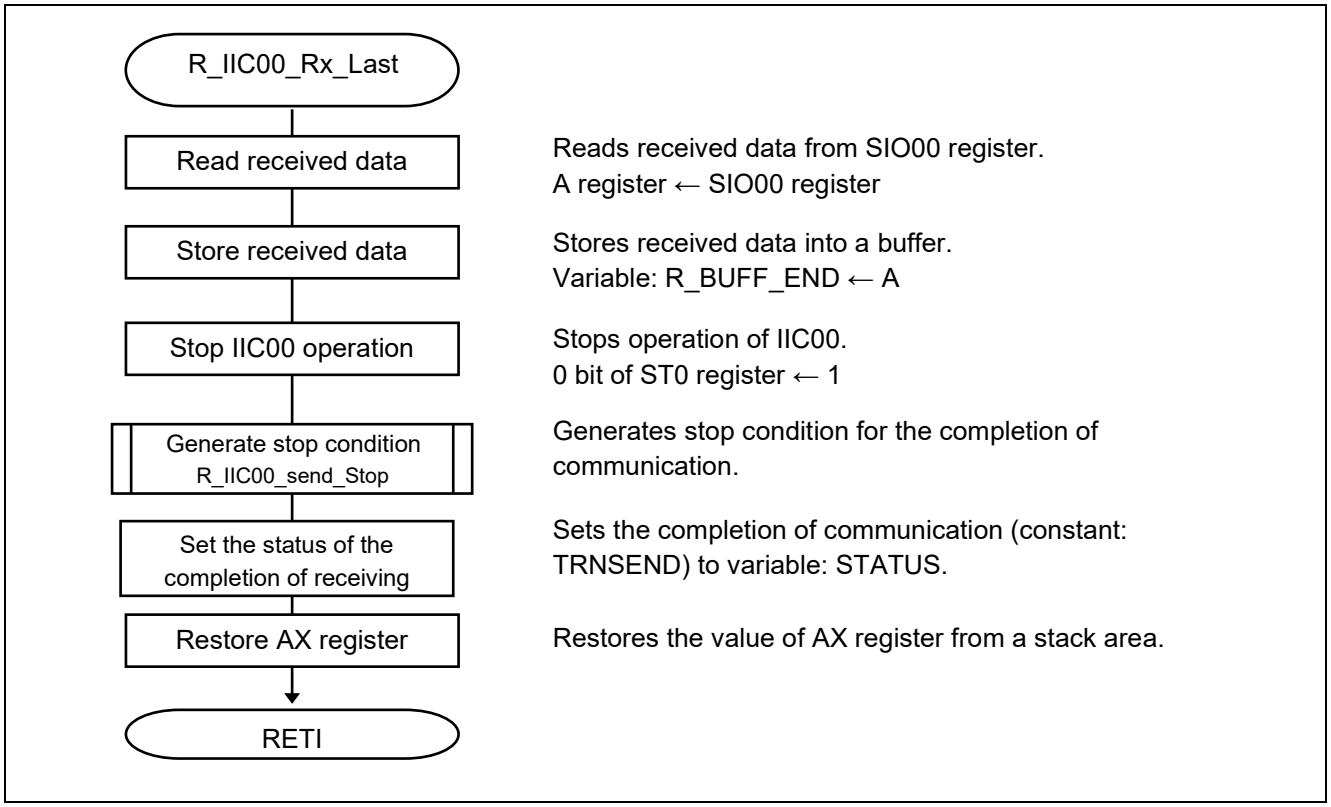


Figure 5.22 Final Data Reception Processing

5.8.15 Data Transmission Start Processing

Figure 5.23 shows the data transmission start processing (R_IIC00_TxDataST). This is the data transmission to EEPROM start processing. Sets the next processing address (R_IIC00_TxData) into the variable: NEXTADR.

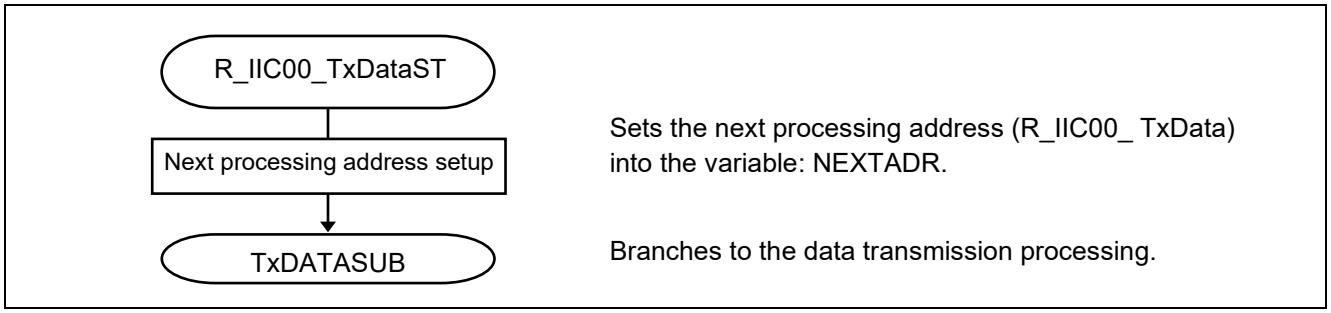


Figure 5.23 Data Transmission Start Processing

5.8.16 Data Transmission Processing

Figure 5.24 shows the data transmission processing (R_IIC00_TxData). This is the completion of data transmission processing of 1-byte. If there is remaining data, continues to transmit data read from buffer. When there is no data, generates stop condition (start to write data to memory cell). After that, in order to check the completion of writing to EEPROM, starts the interval timer for 100 ms (it is 400 ms at the time of a normal mode) in the fast mode.

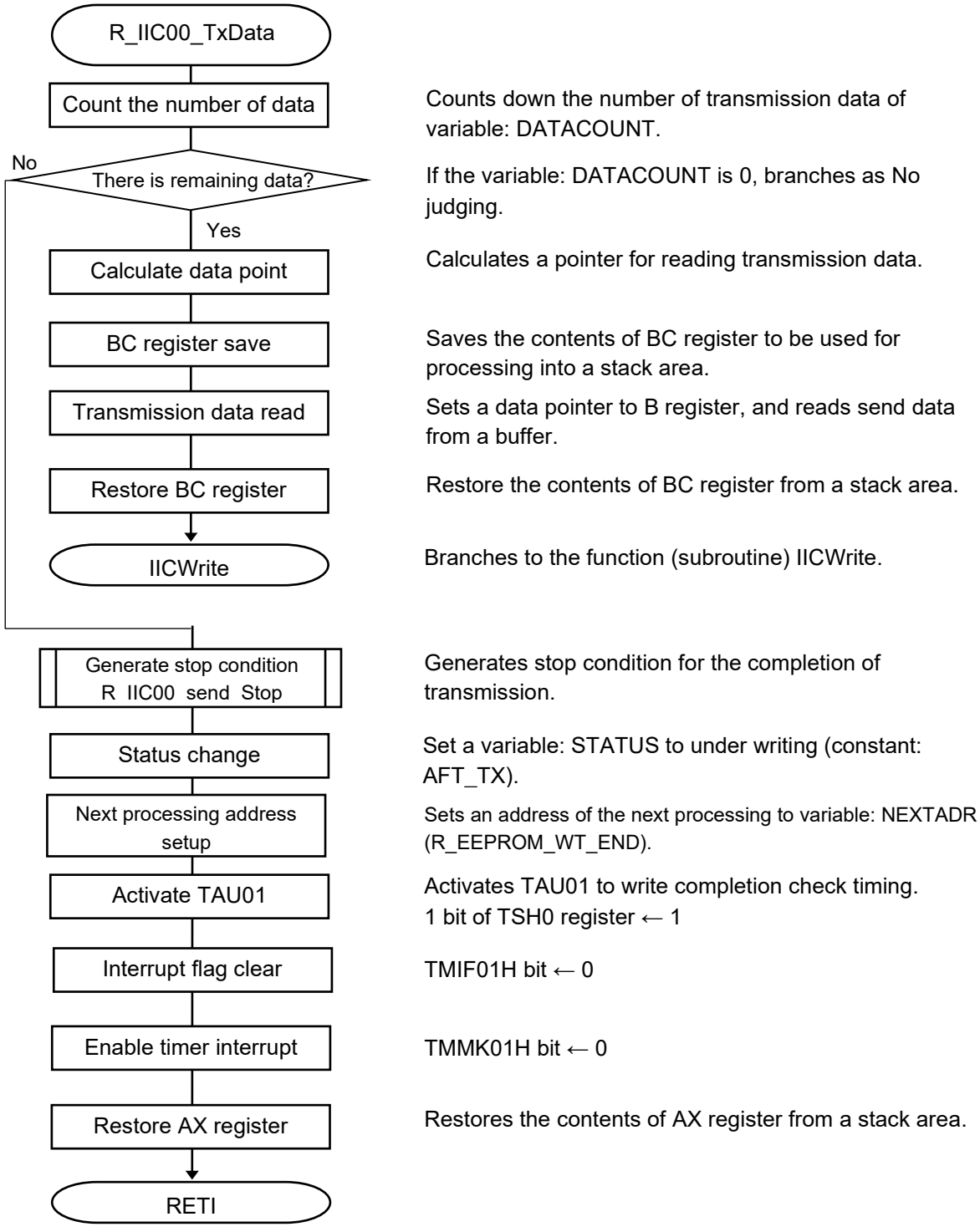
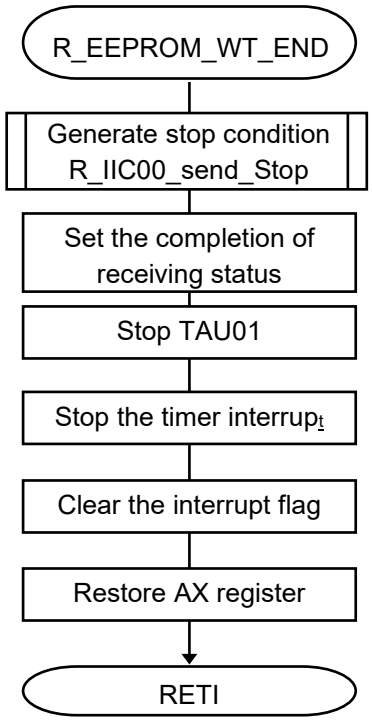


Figure 5.24 Data Transmission Processing

5.8.17 Completion of Data Writing

Figure 5.25 shows the completion of data writing (R_EEPROM_WT_END). This flowchart describes a processing that the data to be written is received by EEPROM successfully. The writing to EEPROM is completed by this processing.



Generates stop condition for the completion of communication.

Sets the completion of communication (constant: TRNSEND) to variable: STATUS.

Stops TAU01 which is used for check of the completion of writing.
1 bit of TTH0 register ← 1
TMMK01H bit ← 1

TMIF01H bit ←0

Restores the contents of AX register from a stack area.

Figure 5.25 Completion of Data Writing

5.8.18 Check Processing for Data Write Completion

Figure 5.26 shows the check processing for data writing completion (IINTTM01H). Starts transmission of a slave address in transmission mode to EEPROM by the interval interrupt. The response from EEPROM to slave address is processed by IINTIIC00.

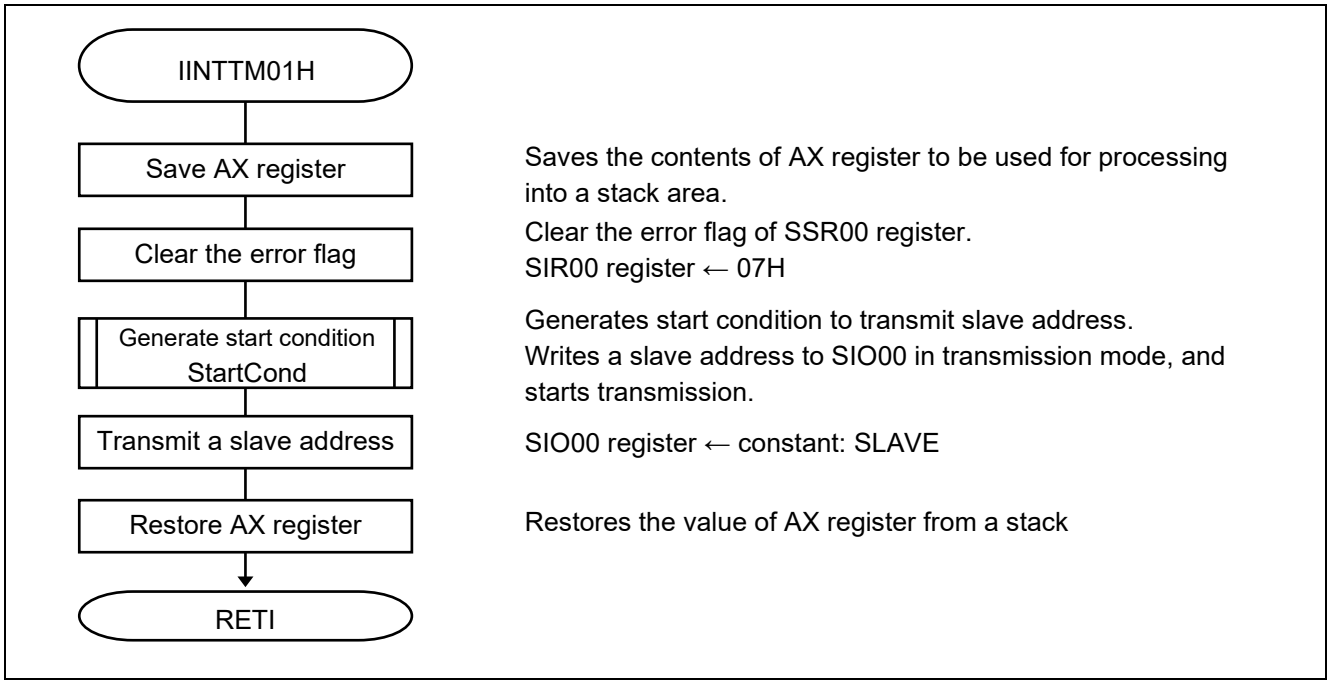
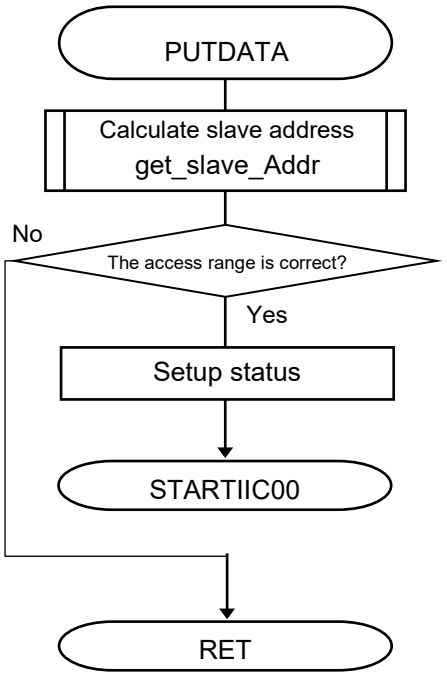


Figure 5.26 Check Processing for Data Write Completion

5.8.19 Write Start Processing to Specified Block

Figure 5.27 shows the write start processing to specified block (PUTDATA).



Calculates a slave address from specified block number, and if the block number is out of EEPROM's range, ends the processing as an error.

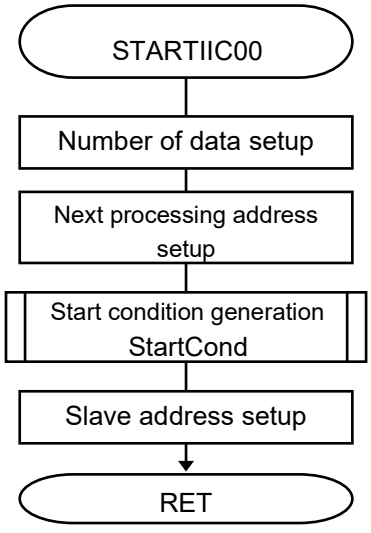
When CY flag is set as a return value of get_slave_Addr, moves to No judging.

Sets the status to transmission.
Variable: STATUS ← Constant: #TRANSMIT

Figure 5.27 Write Start Processing to Specified Block

5.8.20 I2C Bus Access Start

Figure 5.28 shows the I2C bus access start (STARTIIC00). This processing is a common routine for starting I2C bus access.



Sets the size of a block as the number of transmission data.
Variable: DATACOUNT ← Constant: #BLKSIZE

Sets the next processing address (R_IIC00_Tx_addr1) into the variable: NEXTADR.

Generates start condition to I2C bus.

Writes slave address into SIO00 register.
A ← Variable: SLAVEADDR
SIO00 register ← A

Figure 5.28 I2C Bus Access Start

5.8.21 Specified Block Read-out Start Processing

Figure 5.29 shows the specified block read-out start processing (GETDATA).

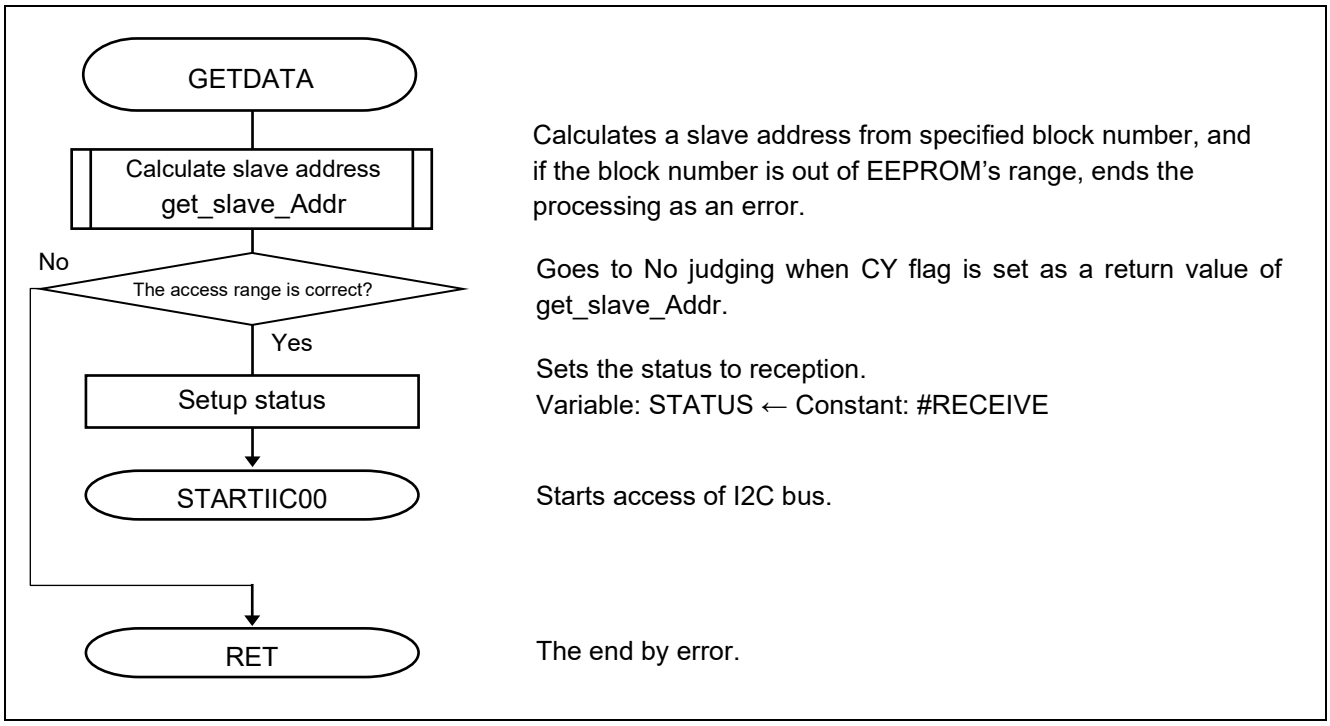


Figure 5.29 Specified Block Read-out Start Processing (GETDATA)

5.8.22 Read Situation Check Processing

Figure 5.3 shows the read situation check processing (GET_CHK). The write situation check processing (PUT_CHK) and the read situation check processing (GET_CHK) are the same processings.

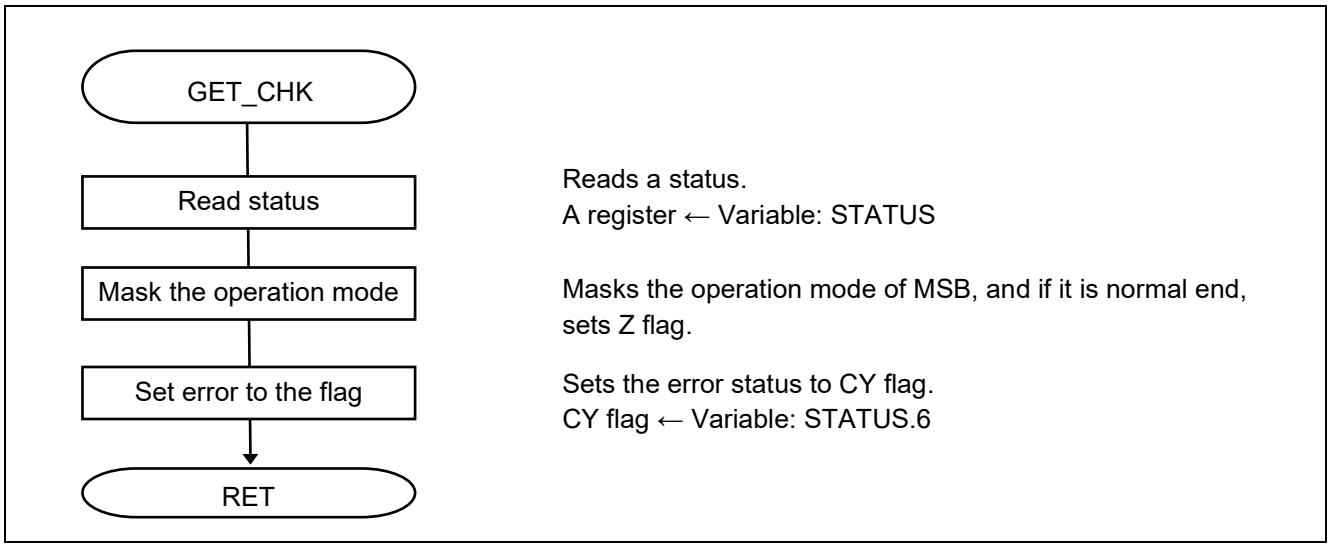


Figure 5.30 Read Situation Check Processing

5.8.23 Write/Read Completion Waiting Processing

Figure 5.31 shows the write/read completion waiting processing (WAIT_END).

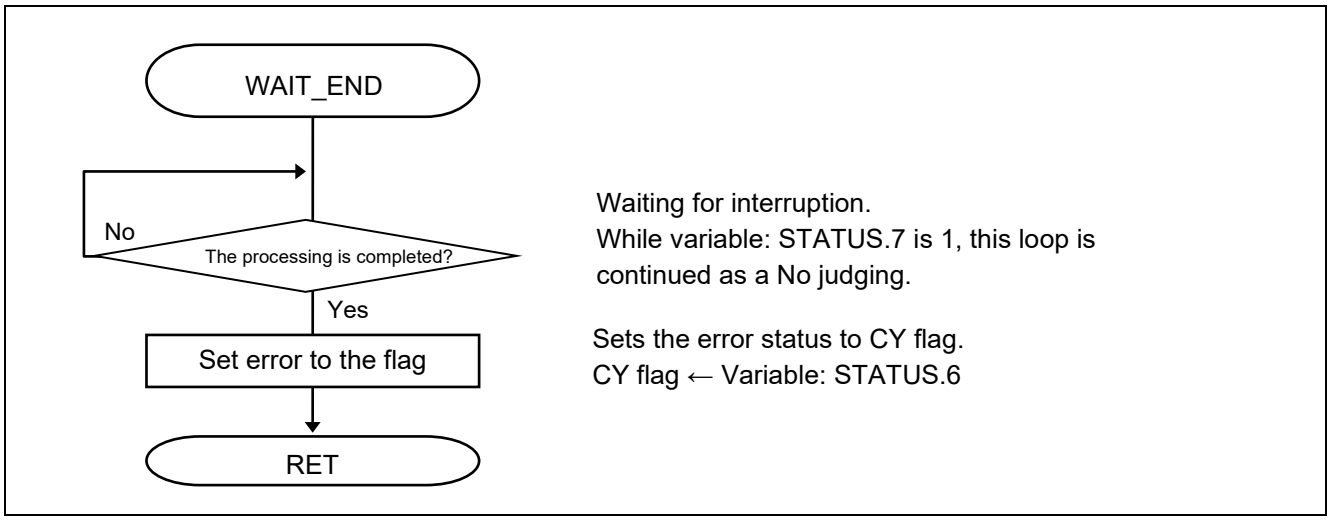


Figure 5.31 Write/Read Completion Waiting Processing

5.8.24 Stop Condition Execution

Figure 5.32 shows the stop condition execution (StopCond).

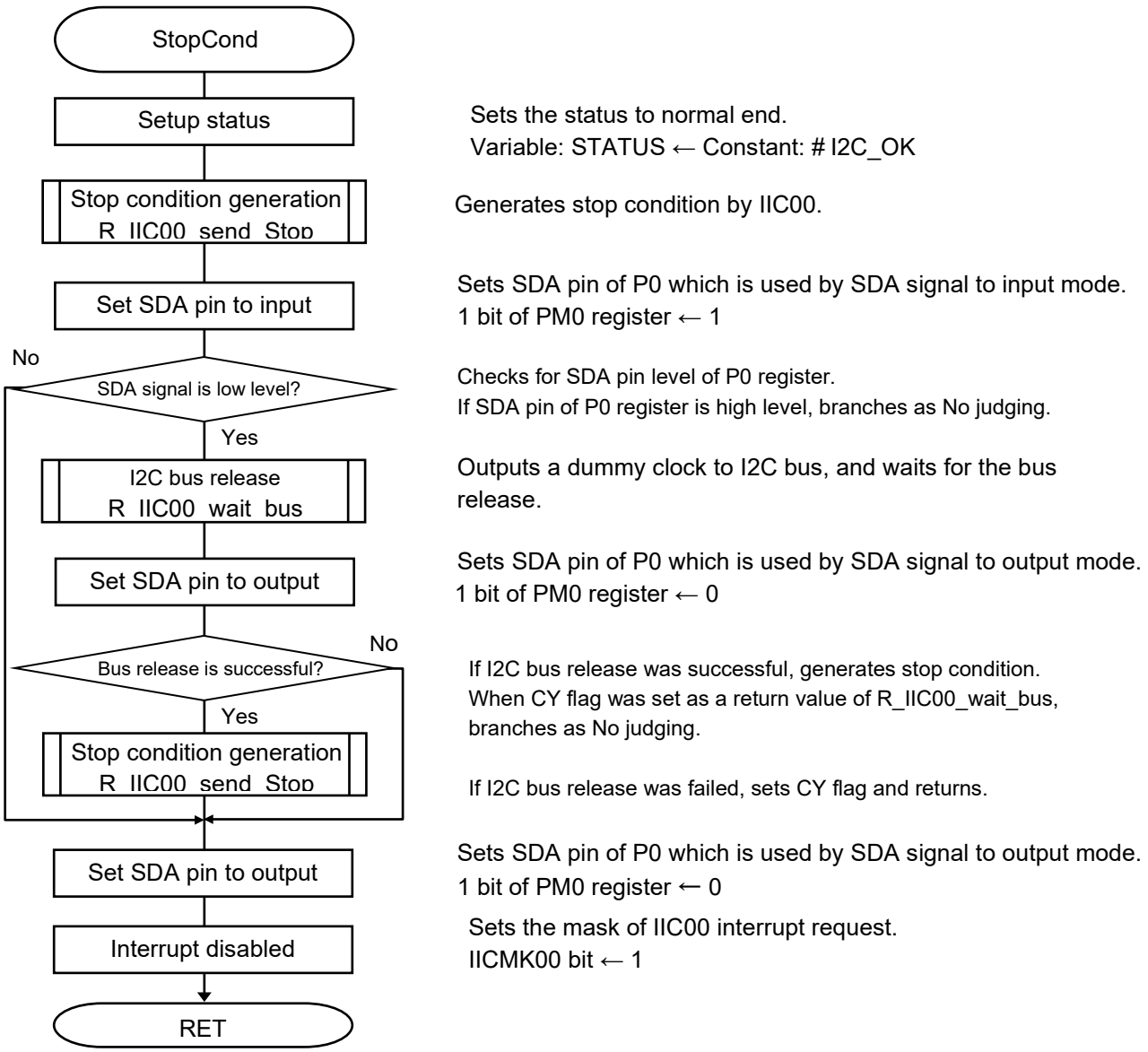


Figure 5.32 Stop Condition Execution

5.8.25 Timer Initialization

Figure 5.33 shows the timer initialization (SINITAU).

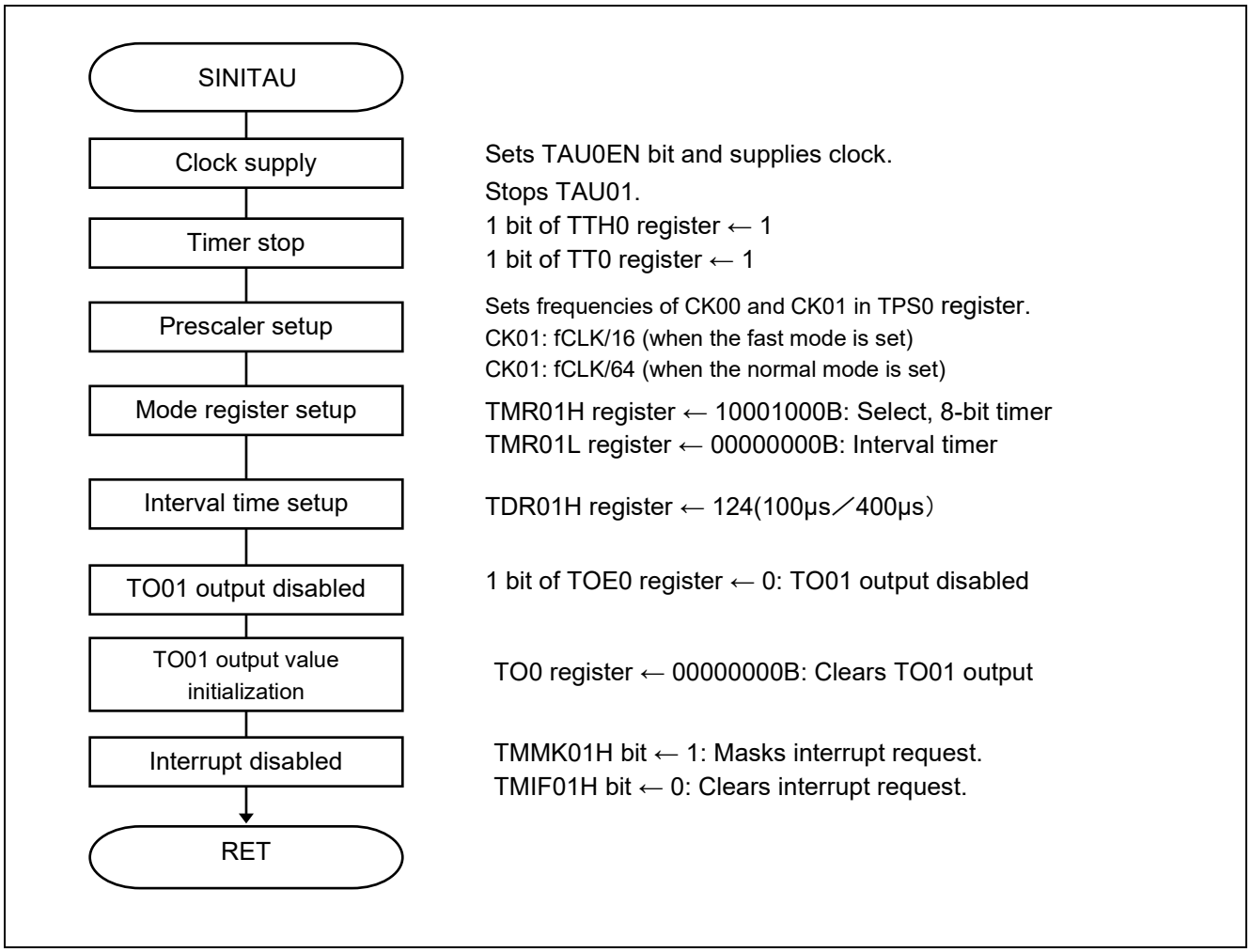


Figure 5.33 Timer Initialization

Starting clock supply to the timer array unit

・Peripheral enable register 0 (PER0)

Starts clock supply to the timer array unit 0.

Symbol：PER0

7	6	5	4	3	2	1	0
TMKAEN Note	CMPEN Note	ADCEN	IICA0EN Note	0	SAU0EN	0	TAU0EN
x	x	x	x	0	x	0	**1**

Note 16-pin products only.

Bit 0

TAU0EN	Control of timer array unit input clock
0	Stops supply of input clock. ・ SFR used by the timer array unit cannot be written. ・ The timer array unit is in the reset status.
1	**Supplies input clock.** **・SFR used by the timer array unit can be read/written.**

Timer operation stop

・Timer channel stop register 0 (TT0)

Counting operation is stopped for TAU.

・Timer channel stop register 0 (TTH0)

Counting operation is stopped for TAU.

Symbol：TT0

7	6	5	4	3	2	1	0
0	0	0	0	TT03 Note	TT02 Note	TT01	TT00
0	0	0	0	x	x	x	**1**

Note 16-pin products only.

Bits 3 to 0

TT0n	Operation stop trigger of channel n (n = 0 to 3)
0	No trigger operation
1	**TE0n is cleared to 0, and counting operation is stopped.** **TT01 and TT03 bits are the trigger to stop operation of the lower 8-bit timer when channels 1 and 3 are in the 8-bit timer mode.**

Symbol：TTH0

7	6	5	4	3	2	1	0
0	0	0	0	TTH03 Note	0	TTH01	0
0	0	0	0	x	0	**1**	0

Note 16-pin products only.

Bits 3 and 1

TTH0n	Operation stop trigger of channel n (n = 1, 3)
0	No trigger operation
1	**TEH0n is cleared to 0, and counting operation is stopped (stop trigger is generated).**

* This bit is the trigger to stop operation of the higher 8-bit timer when channels 1 and 3 are used in the 8-bit timer mode.

Setup of the timer clock frequency

・Timer clock select register 0 (TPS0)

Selects the operation clock of timer array unit 0.

Symbol：TPS0

7	6	5	4	3	2	1	0
PRS013	PRS012	PRS011	PRS010	PRS003	PRS002	PRS001	PRS000
x	**x**	**x**	**x**	**0**	**0**	**0**	**0**

Bits 7 to 0

PRS 0k3	PRS 0k2	PRS 0k1	PRS 0k0	Selection of operation clock (CK0k) Note (k = 0, 1)					
					f_{CLK}= 1.25MHz	f_{CLK}= 2.5MHz	f_{CLK}= 5MHz	f_{CLK}= 10MHz	f_{CLK}= 20MHz
0	**0**	**0**	**0**	**f_{CLK}**	1.25 MHz	2.5 MHz	5 MHz	10 MHz	**20 MHz**
0	0	0	1	$f_{CLK}/2$	625 kHz	1.25 MHz	2.5 MHz	5 MHz	10 MHz
0	0	1	0	$f_{CLK}/2^2$	313 kHz	625 kHz	1.25 MHz	2.5 MHz	5 MHz
0	0	1	1	$f_{CLK}/2^3$	156 kHz	313 kHz	625 kHz	1.25 MHz	2.5 MHz
0	**1**	**0**	**0**	**$f_{CLK}/2^4$**	78 kHz	156 kHz	313 kHz	625 kHz	**1.25 MHz**
0	1	0	1	$f_{CLK}/2^5$	39 kHz	78 kHz	156 kHz	313 kHz	625 kHz
0	**1**	**1**	**0**	**$f_{CLK}/2^6$**	19.5 kHz	39 kHz	78 kHz	156 kHz	**313 kHz**
0	1	1	1	$f_{CLK}/2^7$	9.8 kHz	19.5 kHz	39 kHz	78 kHz	156 kHz
1	0	0	0	$f_{CLK}/2^8$	4.9 kHz	9.8 kHz	19.5 kHz	39 kHz	78 kHz
1	0	0	1	$f_{CLK}/2^9$	2.5 kHz	4.9 kHz	9.8 kHz	19.5 kHz	39 kHz
1	0	1	0	$f_{CLK}/2^{10}$	1.22 kHz	2.5 kHz	4.9 kHz	9.8 kHz	19.5 kHz
1	0	1	1	$f_{CLK}/2^{11}$	625 Hz	1.22 kHz	2.5 kHz	4.9 kHz	9.8 kHz
1	1	0	0	$f_{CLK}/2^{12}$	313 Hz	625 Hz	1.22 kHz	2.5 kHz	4.9 kHz
1	1	0	1	$f_{CLK}/2^{13}$	152 Hz	313 Hz	625 Hz	1.22 kHz	2.5 kHz
1	1	1	0	$f_{CLK}/2^{14}$	78Hz	152 Hz	313 Hz	625 Hz	1.22 kHz
1	1	1	1	$f_{CLK}/2^{15}$	39Hz	78Hz	152 Hz	313 Hz	625 Hz

Note When changing the clock selected for fCLK (by changing the system clock control register (CKC) value), stop timer array unit (TT0 = 0FH, TTH0 = 0AH).

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of channel 0 operation mode

・Timer mode register 0n (TMR0nH、TMR0nL)
Selection of the operation clock (fMCK).
Selection of the count clock.
Select the 16 or 8-bit timer.
Specifying the start trigger and capture trigger.
Selection of the valid edge of the timer input.
Setting of the operation mode.

Symbol：TMR0nH

7	6	5	4	3	2	1	0
CKS0n1	0	0	CCS0n	SPLIT0n	STS0n2	STS0n1	STS0n0
1	0	0	**0**	1	**0**	**0**	**0**

Bit 7

CKS0n1	Selection of operation clock (fMCK) of channel n
0	Operation clock CK00 set by timer clock select register 0 (TPS0)
1	**Operation clock CK01 set by timer clock select register 0 (TPS0)**

Bit 4

CCS0n	Selection of count clock (fTCLK) of channel n
0	**Operation clock (fMCK) specified by the CKS0n1 bit**
1	Valid edge of input signal input from the TI0n pin

Bit 3

SPLIT0n	Selection of 8 or 16-bit timer operation for channels 1 and 3 (n = 1, 3)
0	Operates as 16-bit timer.
1	**Operates as 8-bit timer**

Bits 2 to 0

STS002	STS001	STS000	Setting of start trigger or capture trigger of channel n (n = 0, 1 (for 10-pin products); n = 0 to 3 (for 16-pin products))
0	**0**	**0**	**Only software trigger start is valid (other trigger sources are unselected).**
0	0	1	Valid edge of the TI0n pin input is used as the start trigger and capture trigger.
0	1	0	Both the edges of the TI0n pin input are used as a start trigger and a capture trigger.
1	0	0	When the channel is used as a slave channel with the one-shot pulse output, PWM output function, or multiple PWM output function: The interrupt request signal of the master channel (INTTM0n) is used as the start trigger.
1	1	0	When the channel is used as a slave channel in two-channel input with one-shot pulse output function: The interrupt request signal of the master channel (INTTM0n) is used as the start trigger. A valid edge of the TI03 pin input of the slave channel is used as the end trigger
Other than above			Setting prohibited

Symbol：TMR0nL

7	6	5	4	3	2	1	0
CIS0n1	CIS0n0	0	0	MD0n3	MD0n2	MD0n1	MD0n0
0	**0**	0	0	**0**	**0**	**0**	**0**

Bits 7 and 6

CIS0n1	CIS0n0	Selection of TI0n pin input valid edge
0	**0**	**Falling edge**
0	1	Rising edge
1	0	Both edges (when low-level width is measured) Start trigger: Falling edge, Capture trigger: Rising edge
1	1	Both edges (when high-level width is measured) Start trigger: Rising edge, Capture trigger: Falling edge

Bits 3 to 1

MD 0n3	MD 0n2	MD 0n1	Setting of operation mode of channel n	Corresponding function	Count operation of TCR
0	**0**	**0**	**Interval timer mode**	**Interval timer/Square wave output/Divider function/PWM output (master)**	**Down count**
0	1	0	Capture mode	Input pulse interval measurement/Twochannel input with one-shot pulse output function (slave)	Up count
0	1	1	Event counter mode	External event counter	Down count
1	0	0	One-count mode	Delay counter/One-shot pulse output/Twochannel input with one-shot pulse output function (master)/PWM output (slave)	Down count
1	1	0	Capture & one-count mode	Measurement of high-/low-level width of input signal	Up count
Other than above			Setting prohibited		
The operation of each mode changes depending on the operation of MD0n0 bit (refer to the table below).					

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Bit 0

Operation mode (Value set by the MD0n3 to MD0n1 bits)	MD 000	Setting of starting counting and interrupt
・Interval timer mode (0, 0, 0) **・Capture mode (0, 1, 0)**	**0**	**Timer interrupt is not generated when counting is started (timer output does not change, either).**
	1	Timer interrupt is generated when counting is started (timer output also changes).
・Event counter mode (0, 1, 1)	0	Timer interrupt is not generated when counting is started (timer output does not change, either).
・One-count mode Note 1 (1, 0, 0)	0	Start trigger is invalid during counting operation. At that time, a timer interrupt is not generated.
	1	Start trigger is valid during counting operation . At that time, a timer interrupt is not generated.
・Capture & one-count mode (1, 1, 0)	0	Timer interrupt is not generated when counting is started (timer output does not change, either). Start trigger is invalid during counting operation. At that time, a timer interrupt is not generated.
Other than above		Setting prohibited

Setup of Timer data register 0

・Timer data register 0n (TDR0nH、TDR0nL)

Symbol：**TDR0nH**

7	6	5	4	3	2	1	0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

TDR0nL

7	6	5	4	3	2	1	0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of disabling timer output

・Timer output enable register 0 (TOE0)
Sets enabling/disabling timer output of each channel.

Symbol：TOE0

7	6	5	4	3	2	1	0
0	0	0	0	TOE03 Note	TOE02 Note	TOE01	TOE00
0	0	0	0	x	x	**0**	x

Note 16-pin products only.

Bit 1

TOE01	Timer output enable/disable of channel 0
0	**Disable output of timer. Without reflecting on TO0n bit timer operation, to fixed the output. Writing to the TO0n bit is enabled and the level set in the TO0n bit is output from the TO0n pin.**
1	Enable output of timer. Reflected in the TO0n bit timer operation, to generate the output waveform. Writing to the TO0n bit is disabled (writing is ignored).

Setup of output value of timer output pin

・Timer output register 0 (TO0)
Sets the output value of timer output pin of each channel.

Symbol：TO0

7	6	5	4	3	2	1	0
0	0	0	0	TO03 Note	TO02 Note	TO01	TO00
0	0	0	0	x	x	x	**0**

Note 16-pin products only.

Bit 0

TO00	Timer output of channel 0
0	**Timer output value is “0”.**
1	Timer output value is “1”.

Setup of the timer capture completion interrupt

- Interrupt mask flag registers (MK0L)
  Clears the interrupt request flag.
- Interrupt request flag registers (IF0L)
  Sets the interrupt mask.

Symbol：MK0L

7	6	5	4	3	2	1	0
TMMK00	TMMK01H	SREMK0	SRMK0	STMK0 CSIMK00 IICMK00	PMK1	PMK0	WDTIMK
x	**1**	x	x	x	x	x	x

Bit 6

TMMK01	Interrupt servicing control
0	Interrupt servicing enabled
1	**Interrupt servicing disabled**

Symbol：IF0L

7	6	5	4	3	2	1	0
TMIF00	TMIF01H	SREIF0	SRIF0	STIF0 CSIIF00 IICIF00	PIF1	PIF0	WDTIIF
x	**0**	x	x	x	x	x	x

Bit 6

TMIF00	Interrupt request flag
0	**No interrupt request signal is generated**
1	Interrupt request is generated, interrupt request status

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.26 Start Condition Generation

Figure 5.34 shows the start condition generation (StartCond).

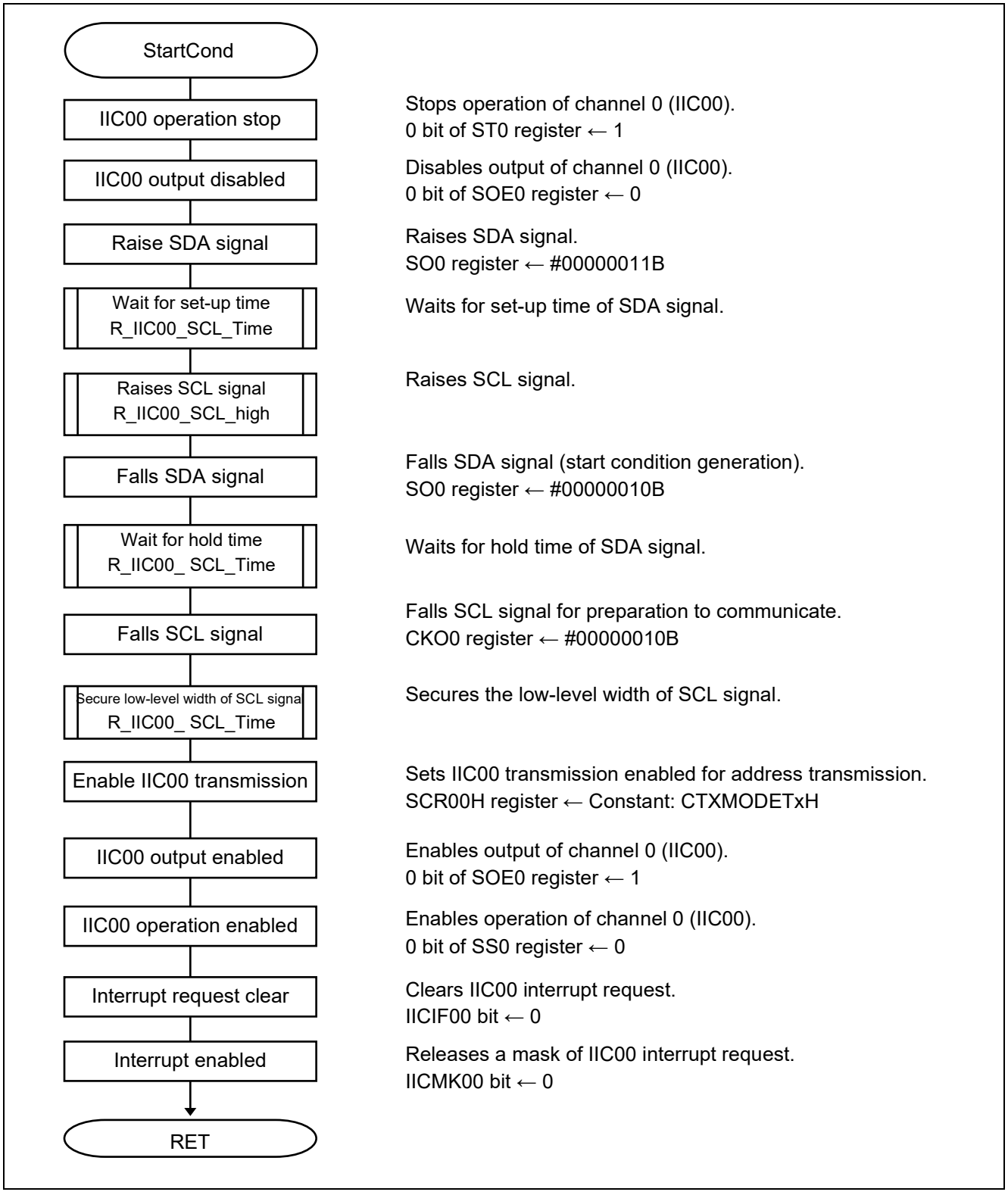


Figure 5.34 Start Condition Generation (StartCond)

Stops the serial communication

・Serial channel stop register 0 (ST0)

Stops the serial communication operation for IIC00

・Serial output enable register 0 (SOE0)

Disables output

Symbol：ST0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	ST01	ST00
0	0	0	0	0	0	x	**1**

Bit 0

ST00	Operation stop trigger of channel 0
0	No trigger operation
1	**Clears the SE00 bit to 0 and stops the communication operation**

Symbol：SOE0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SOE01	SOE00
0	0	0	0	0	0	x	**0**

Bit 0

SOE00	Serial output enable/disable of channel 0
0	**Disables output by serial communication operation**
1	Enables output by serial communication operation

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of initial output level

・Serial output register 0 (SO0)

Setting of initial output level

Symbol：SO0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SO01 Note	SO00
0	0	0	0	0	0	x	**1/0**

Note 16-pin products only.

Bit 0

SO00	Serial data output of channel 0
1	**Serial data output value is “1”.**
0	**Serial data output value is “0”.**

Setup of the serial click output for translation channel

・Serial clock output register (CKO0)
Setting of the serial clock output level for starting the communication

Symbol：CKO0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	CKO01	CKO00
0	0	0	0	0	0	x	**0/1**

Bit 0

CKO00	**Serial clock output of channel 0**
1	**Serial clock output value is “1”.**
0	**Serial clock output value is “0”.**

Setup of the communication operation for translation channel

・Serial communication operation setting register 00 (SCR00H)
Setting of operation mode

Symbol：SCR00H

7	6	5	4	3	2	1	0
TXE00	RXE00	DAP00	CKP00	0	EOC00	PTC001	PTC000
1	**0**	**0**	**0**	0	**0**	**0**	**0**

Bits 7 and 6

TXE00	**RXE00**	**Setting of operation mode of channel 0**
0	0	Disable communication.
0	1	Reception only
1	**0**	**Transmission only**
1	1	Transmission/reception

Bits 5 and 4

DAP00	**CKP00**	**Selection of data and clock phase in CSI mode**
0	**0**	**Be sure to set DAP0n, CKP0n = 0, 0 in the UART mode and simplified I2C mode.**

Bit 2

EOC00	**Selection of masking of error interrupt signal (INTSRE0)**
0	**Disables generation of error interrupt INTSRE0 (INTSR0 is generated).**
1	Enables generation of error interrupt INTSRE0 (INTSR0 is not generated if an error occurs).

Bits 1 and 0

PTC001	**PTC000**	**Setting of parity bit in UART mode**	
		Transmission	**Reception**
0	**0**	**Does not output the parity bit.**	Receives without parity
Be sure to set PTC0n1, PTC0n0 = 0, 0 in the CSI mode and simplified I2C mode.			

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Enables output of target channel for the serial communication operation

・Serial output enable register 0 (SOE0)
Enables output

Symbol：SOE0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SOE01	SOE00
0	0	0	0	0	0	x	**1**

Bit 0

SOE00	Serial output enable/disable of channel 0
0	Disables output by serial communication operation.
1	**Enables output by serial communication operation.**

Entering the communication

・Serial channel start register 0 (SS0)
Starts the operation

Symbol： SS0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SS01	SS00
0	0	**0**	**0**	**0**	0	0	**1**

Bit 0

SS00	Operation start trigger of channel 0
0	No trigger operation
1	**Sets the SE00 bit to 1 and enters the communication wait status**

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

Setup of the timer capture completion interrupt

- Interrupt request flag registers (IF0L)
  Clears the interrupt request flag.
- Interrupt mask flag registers (MK0L)
  Sets the interrupt mask.

Symbol：IF0L

7	6	5	4	3	2	1	0
TMIF00	TMIF01H	SREIF0	SRIF0	STIF0 CSIIF00 **IICIF00**	PIF1	PIF0	WDTIIF
x	x	x	x	**0**	x	x	x

Bit 3

IICIF00	**Interrupt request flag**
0	**No interrupt request signal is generated**
1	Interrupt request is generated, interrupt request status

Symbol：MK0L

7	6	5	4	3	2	1	0
TMMK00	TMMK01H	SREMK0	SRMK0	STMK0 CSIMK00 **IICMK00**	PMK1	PMK0	WDTIMK
X	x	x	x	**0**	x	x	x

Bit 3

IICMK00	**Interrupt servicing control**
0	**Interrupt servicing enabled**
1	Interrupt servicing disabled

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.27 Stop Condition Generation

Figure 5.35 shows the stop condition generation (R_IIC00_send_Stop).

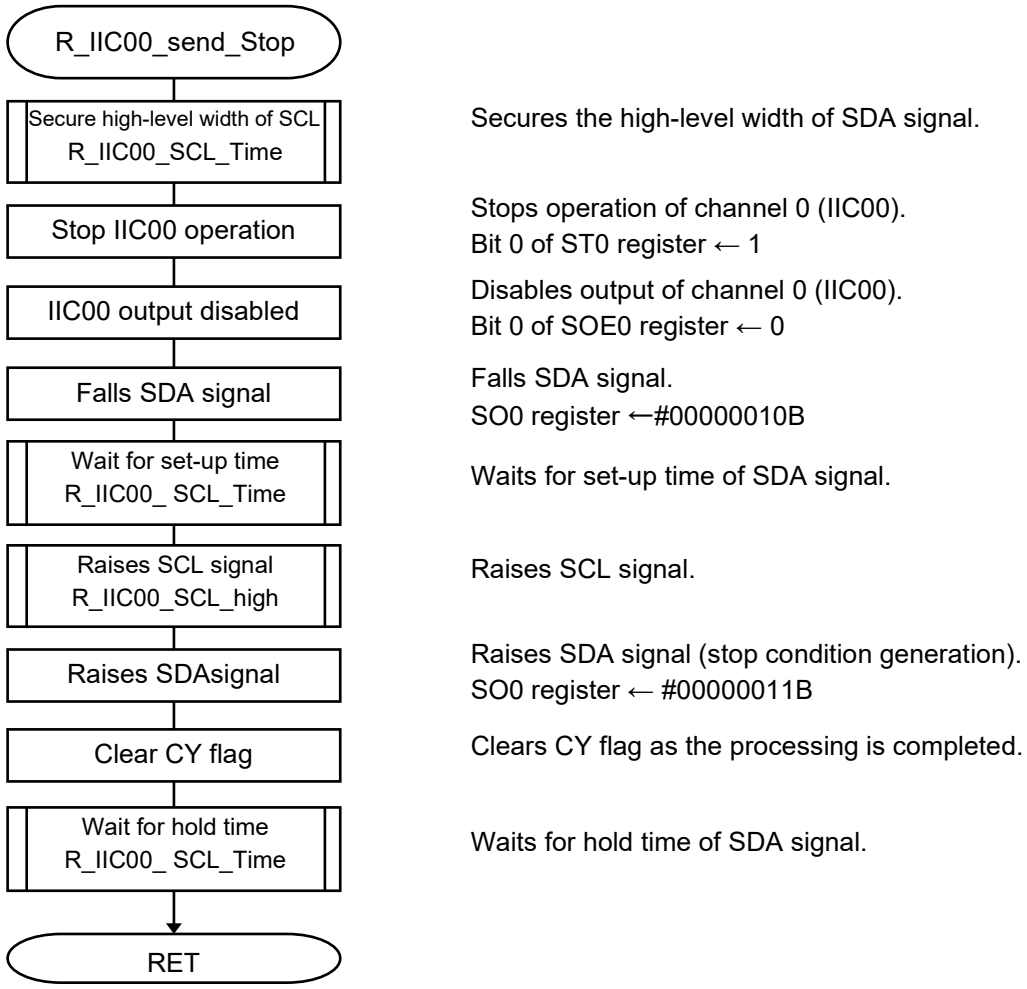


Figure 5.35 Stop Condition Generation

Stops the serial communication

- Serial channel stop register 0 (ST0)
  Stops IIC00 operation
- Serial output enable register 0 (SOE0)
  Disables output

Symbol：ST0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	ST01	ST00
0	0	0	0	0	0	x	**1**

Bit 0

ST00	Operation stop trigger of channel 0
0	No trigger operation
1	**Clears the SE00 bit to 0 and stops the communication operation**

Symbol：SOE0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SOE01	SOE00
0	0	0	0	0	0	x	**0**

Bit 0

SOE00	Serial output enable/disable of channel 0
0	**Disables output by serial communication operation.**
1	Enables output by serial communication operation.

Setup of initial output level

- Serial output register 0 (SO0)
  Setting of the serial output level

Symbol：SO0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SO01 Note	SO00
0	0	0	0	0	0	x	**1/0**

Note 16-pin products only.

Bit 0

SO00	Serial data output of channel 0
1	**Serial data output value is “1”.**
0	**Serial data output value is “0”.**

Note: Refer to ‘RL78/G10 User's Manual: Hardware’ for more information about the register setting method.

5.8.28 I2C Bus Release Processing

Figure 5.36 shows I2C bus release processing (R_IIC00_wait_bus).

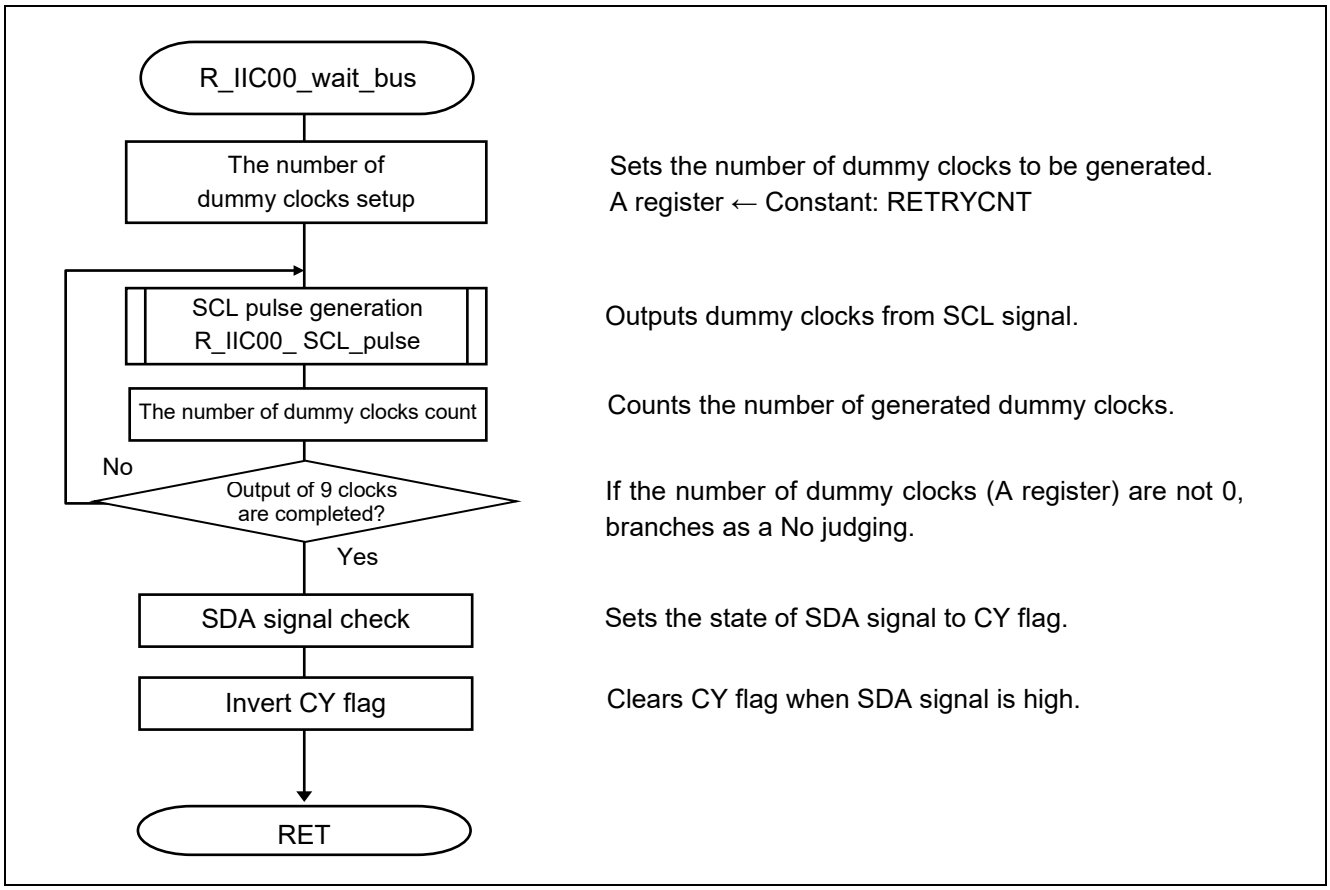


Figure 5.36 I2C Bus Release Processing

5.8.29 SCL Pulse Generation

Figure 5.37 shows the SCL pulse generation (R_IIC00_SCL_pulse).

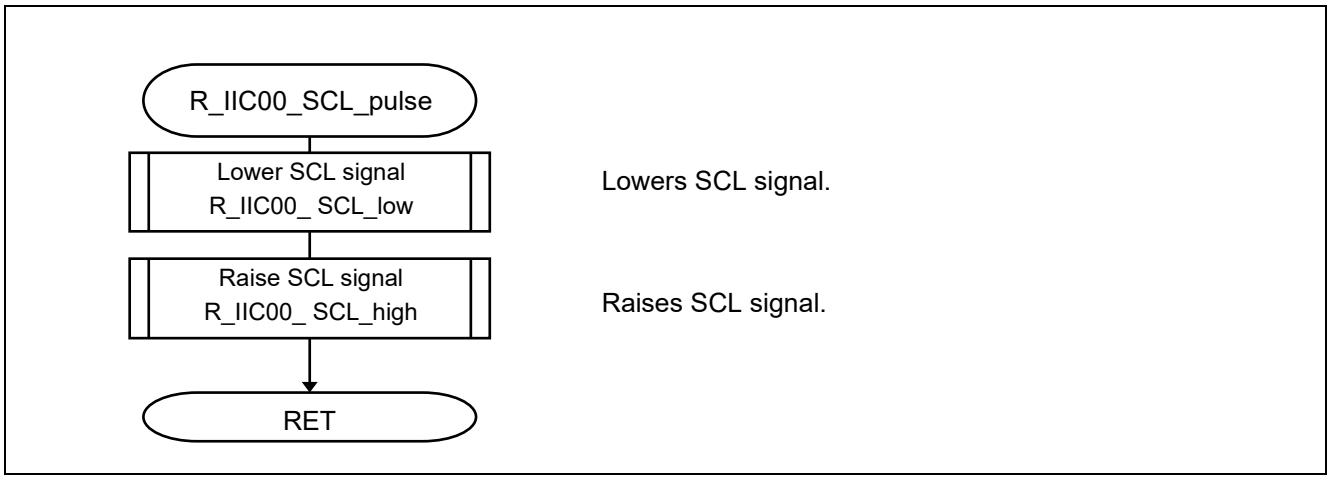


Figure 5.37 SCL Pulse Generation

5.8.30 SCL Signal Raising

Figure 5.38 shows the SCL signal raising (R_IIC00_SCL_high).

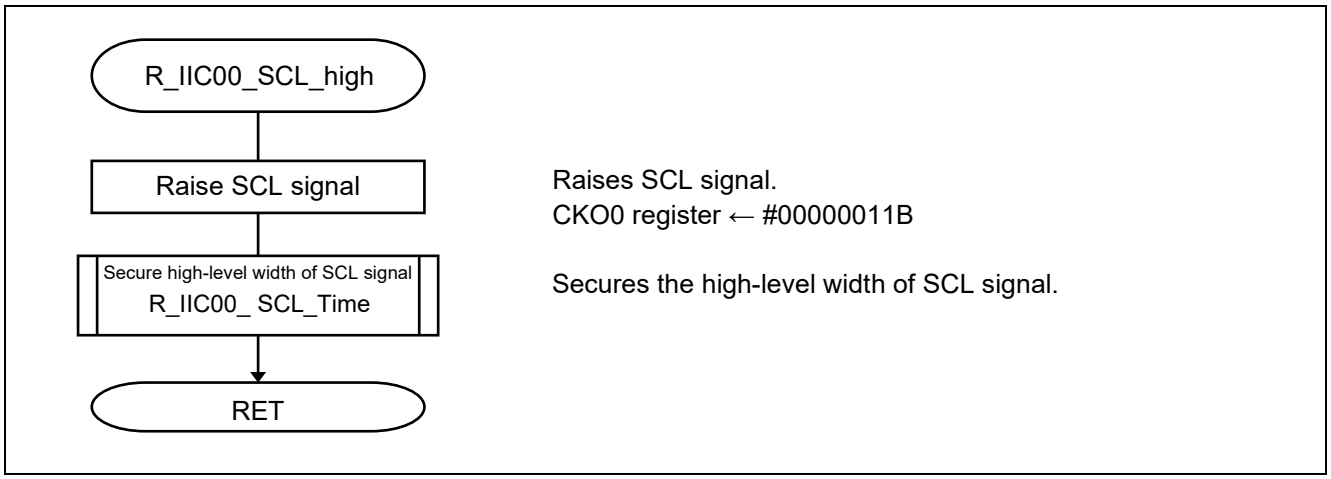


Figure 5.38 SCL Signal Raising

5.8.31 SCL Signal Lowering

Figure 5.39 shows the SCL signal lowering (R_IIC00_SCL_low).

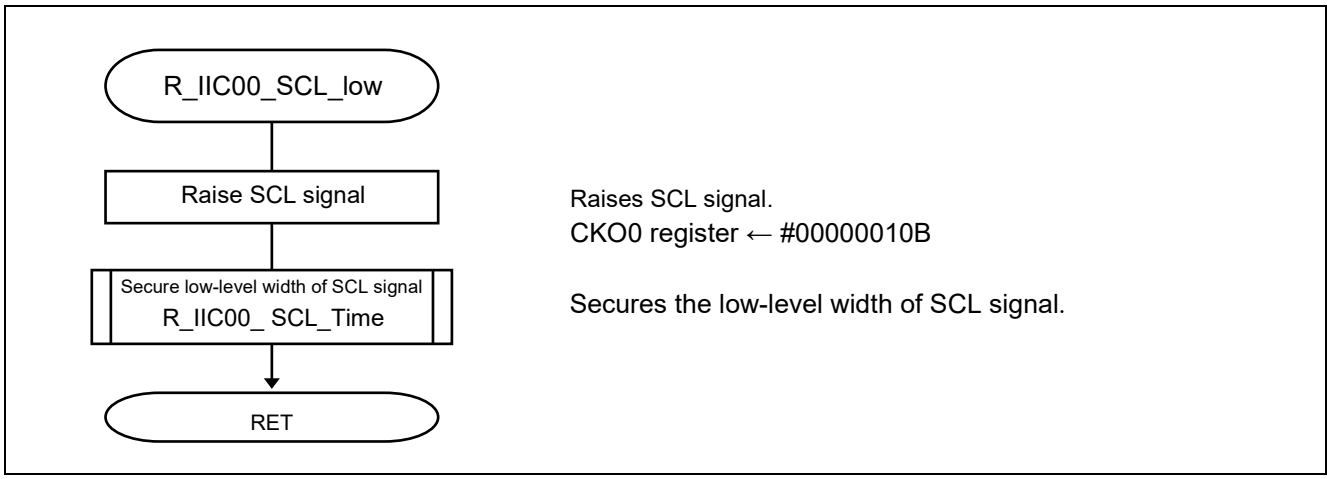


Figure 5.39 SCL Signal Lowering

5.8.32 SCL Signal Width Securing

Figure 5.40 shows the SCL signal width securing (R_IIC00_SCL_Time).

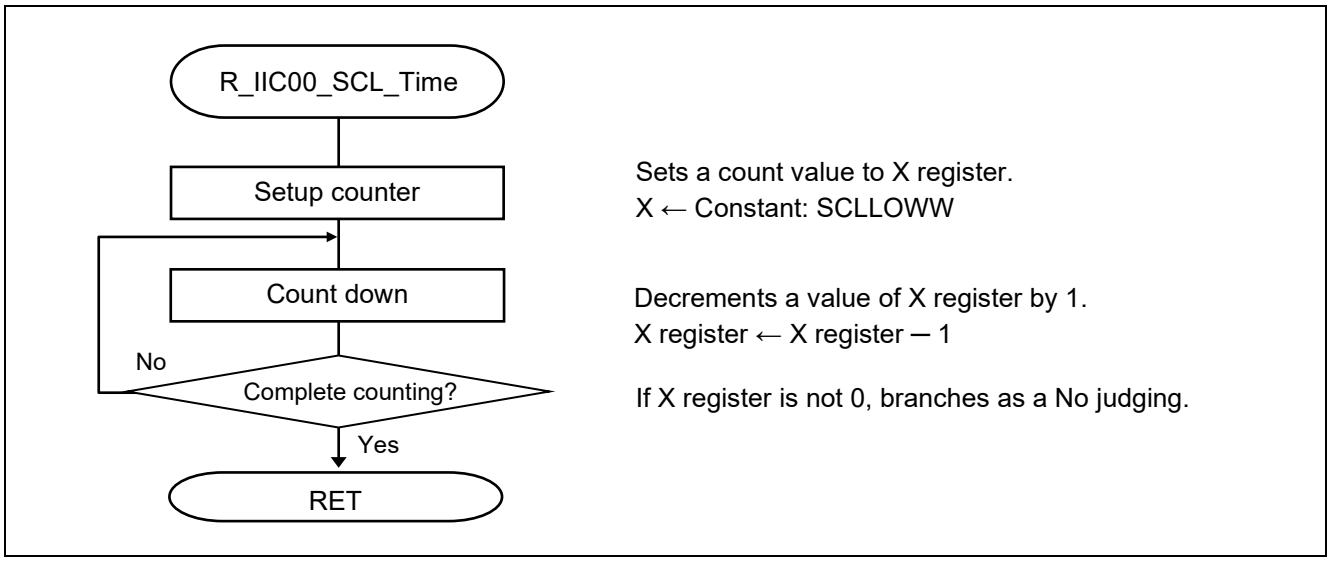
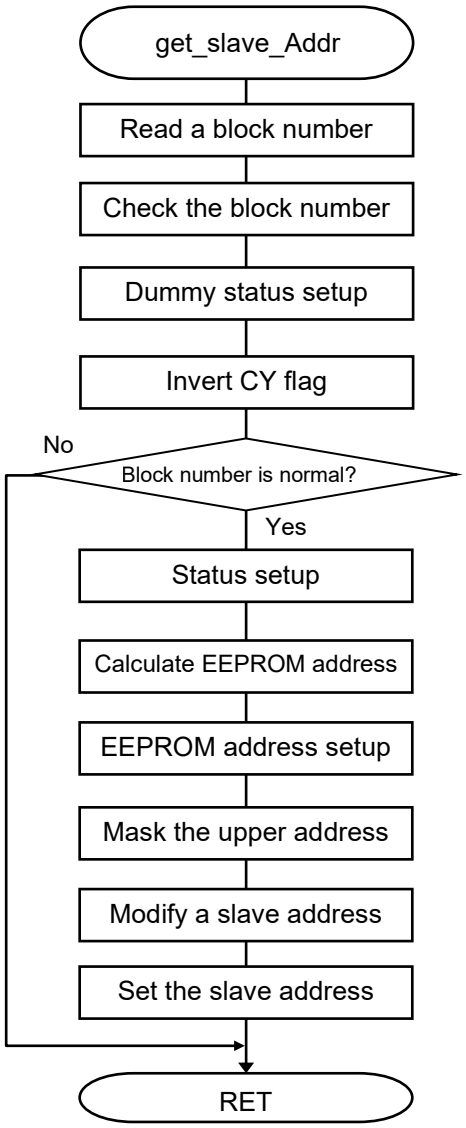


Figure 5.40 SCL Signal Width Securing

5.8.33 Calculation of Slave Address

Figure 5.41 shows the calculation of slave address (get_slave_Addr).



Sets a value of variable: BLOCK_NUMBER to AX register.

Compares the value of AX register with constant BLKNO.

Sets constant: PARA_ERR (parameter error) to variable: STATUS.

Sets CY flag when the block number specification error is occurred. If CY flag is set, branches as No judging.

Sets constant: I2C_OK (normal end) to variable: STATUS.

Converts block number into EEPROM address (×4/×8/×16)

Sets EEPROM address to variable: EEPROMADDR.

Extracts the information to be reflected in a slave address from EEPROM upper address.

Reflects extracted information in a slave address.

Sets the created slave address to variable: SLAVEADDR.

Figure 5.41 Calculation of Slave Address

5.9 Sample Code Setup

5.9.1 The Way of Sample Code Setting

The setup method for controlling serial EEPROM by sample code is shown below.
In the sample code, the definition related to setup is described by the header file (DEV&EEPROM.inc). The device to be used defines the following.

(1) Control targets

Target EEPROM for controlling is only one that is from 512K bits (64K bytes) to 2K bits (256 bytes), and defined in the header file according to EEPROM to be used. 256K-bit EEPROM is chosen by the default.

```
PROM2K      . SET   0
PROM4K      . SET   0
PROM8K      . SET   0
PROM16K     . SET   0
PROM32K     . SET   0
PROM64K     . SET   0
PROM128K    . SET   0
PROM256K    .SET    1
PROM512K    . SET   0
```

Figure 5.42 Definition of the target EEPROM

(2) Block information

It is the definition which shows the size of a block. By default, it is 4 bytes/block.

```
BLKSIZE     .SET    4       ; 4bytes/block
;BLKSIZE    .SET    8       ; 8bytes/block
;BLKSIZE    .SET    16      ; 16bytes/block
```

Figure 5.43 Block Information

Although the block size can be changed into 8 or 16 bytes, if makes it too large, a possibility that futility will occur will become high. In order to change, delete ";" of the head of a line to change.

(3) Control parameter

A parameter required for control of each EEPROM is defined below. If the block information on EEPROM is specified, required information will be set to a control parameter. A parameter when 256K-bit EEPROM is specified is shown below.

```
$ELSEIF( PROM256K )
EEPROM      .SET    R1EX24256B          ; 7
BLKNO       .SET    32768/BLKSIZE       ; 8192 blocks/device
```

Figure 5.44 Control Parameter

The meaning of each parameter is as follows.

a) EEPROM: Name of EEPROM. It is a value of 0 (2K bits) to 8 (512K bits).
b) BLKNO: The number of blocks included in EEPROM.
c) MASK: The upper bits of the cell address of EEPROM are shown. It is used when specifying the slave address of I2 C bus. It is 0x00, 0x01, 0x03, or 0x07.

(4) Transfer rate parameter

The transfer rate of I2C bus is the fast mode or normal mode. By the default, it has set to the fast mode. Initial setup is performed according to this mode. Since it is necessary to set a transfer rate to also satisfy the specifications about the low level width of SCL signal, a definition is given as follows. If this value is set to SDR00H, the wished transmission speed can be set up.

```
$IF( FAST_MODE )
DIVIDE      .EQU   13 * CLKFREQ ／ 10000        ; fast mode(384kbps)
$ELSE
DIVIDE      .EQU   50 * CLKFREQ ／ 10000        ; normal mode(100kbps)
$ENDIF
```

This value is determined by the specifications of low-level width (1.3μs).

This is the low-level width determined by the transfer rate (100kbps).

Figure 5.45 Management Information by I2C Bus to be Used

(5) Dummy clock setup parameter

The number of the dummy clocks for making a bus release is defined as follows.

```
RETRYCNT        .EQU    9               ; max. dummy SCL pulse number
```

Figure 5.46 To Specify the Number of Dummy Clocks

(6) Control information

In this program, uses global variables described in Figure 5.51 to access EEPROM. The size of the program is suppressed by using the control information fixed in this way.

```
BLOCK_NUMBER:   .DS      2              ; index block number to access
WRITE_BUFF:     .DS      BLKSIZE        ; write data buffer
READ_BUFF:      .DS      BLKSIZE-1      ; read data buffer
STATUS:         .DS      1              ; result of operation
```

Figure 5.47 EEPROM Control Information

BLOCK_NUMBER:
The block number which control target EEPROM wants to access is specified.

WRITE_BUFF:
Data to write in EEPROM is set.

READ_BUFF:
A buffer where the data read from EEPROM is stored.

5.9.2 Processing in the Sample Code

The processing in the sample code is described below.

(1) EEPROM control information clear

In Figure 5.48, clears EEPROM control information of internal memory.

```
;
;       buffer area initilize
;
        MOV     B,      #BLKSIZE*2+5    ; set data number
        CLRB    A
MAIN_LOOP1:
        MOV     BLOCK_NUMBER-1[B], A    ; clear memory
        DEC     B                       ; count down data number

        BNZ     $MAIN_LOOP1
```

Figure 5.48 EEPROM Control Information Clear

(2) Hardware initialization

In Figure 5.49, initializes the hardware to be used. And then generates stop condition and makes I2C bus an initial state.

```
;
;       IIC00 and timer initialize
;
        CALL    !R_IIC00_Init           ; initialize IIC00 function
        CALL    SINITAU                 ; initialize 100us timer
        CALL    !StopCond               ; IIC bus initialize(bus free)
```

Figure 5.49 Hardware Initialization

(3) Write to EEPROM processing

In Figure 5.50, as a preprocessing to write data in EEPROM, data to write in is set as memory area WRITE_BUFF, and the block number of EEPROM to write in variable BLOCK_NUMBER is set. Then, a control function is called.

Activates the write to EEPROM processing by “CALL !PUTDATA” and starts the write to EEPROM processing. Since all processing is interrupt processing in the background, shortly after starting, processing will return. It is possible to perform other processing simultaneously.

Here, waits for the completion of "CALL !WAIT_END" processing without performing any others. CY flag is set and returned if any errors occurred. In this example, if an error is detected, an infinite loop will be carried out at ERRORLOOP1.

```
        EI
        CALL    !PUTDATA                ; write data to EEPROM
        NOP
        CALL    !WAIT_END               ; wait for complete
ERRORLOOP1:
        BC      $ERRORLOOP1             ; loop if error
```

Figure 5.50 Write to EEPROM Processing

(4) Read from EEPROM processing

In Figure 5.51, as a read processing from EEPROM, a block number is set to variable BLOCK_NUMBER and a read-out routine is called. Here, processing returns only by the read processing from EEPROM starting as well. Waits for the completion of processing by "CALL !WAIT_END" like the write processing.

```
        CALL    !GETDATA
        CALL    !WAIT_END               ; wait for complete

ERRORLOOP2:
        BC      $ERRORLOOP2             ; loop if error
```

Figure 5.51 Read from EEPROM Processing

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G10 User's Manual: Hardware (R01UH0384E)
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 < RL78/G10 EEPROM Control by Simplified I2C Function CC-RL>

		Description	
Rev.	**Date**	**Page**	**Summary**
Rev. 1.00	Feb. 03, 2016	—	First edition issued
Rev. 1.10	June. 24, 2022	9	Operation check condition is updated.

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.