To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

H8/300L SLP Series

Interfacing the H8/38024 with an IIC Serial EEPROM

Introduction

An IIC serial EEPROM is connected to the H8/38024 via the two lines of SCL (Serial Clock) and SDA (Serial Data). Data in the on-chip ROM is written to the EEPROM, and the data that was written to the EEPROM is read back to the on-chip RAM.

Target Device

H8/38024

Contents

1. Specifications

1. As shown in figure 1.1, an IIC serial EEPROM is connected to the H8/38024 via two lines, SCL (Serial Clock) and SDA (Serial Data).
2. Data in the on-chip ROM is written to the EEPROM, and the data that was written to the EEPROM is read back to the on-chip RAM.
3. Data is transferred in an LSB-first method.

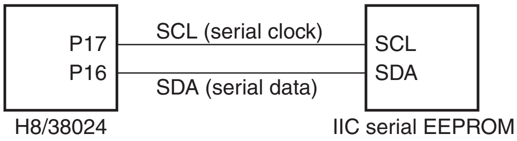


Figure 1.1 Example of EEPROM Connection

2. Concepts

1. Figure 2.1 illustrates the bus timing of the IIC serial EEPROM used for this sample task.

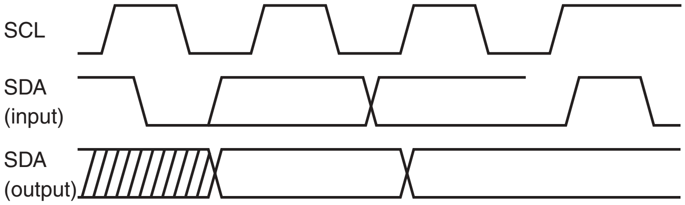


Figure 2.1 Bus Timing of IIC Serial EEPROM

2. The bus timing of the IIC serial EEPROM used is as shown in figure 2.1. In this sample task, the P17 pin is driven high and low by software to generate a serial clock, which is output to the SCL. In synchronization with this serial clock generated by software, serial data is output/input via the P16 pin to access the IIC serial EEPROM. Figure 2.2 illustrates timing waveforms of the P17 and P16 pin levels.

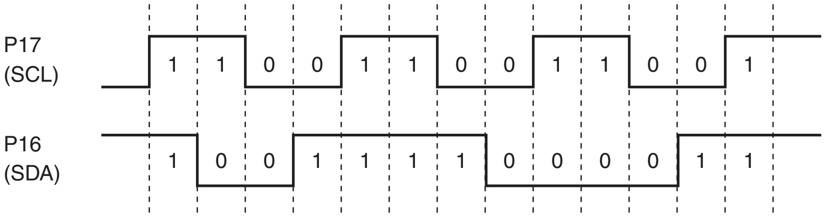


Figure 2.2 Timing Waveforms of P17 and P16 Pin Levels

3. Description of Functions

1. In this sample task, the IIC serial EEPROM is connected to the H8/38024 as shown in figure 3.1. Table 3.1 provides descriptions of the pins of the IIC serial EEPROM.

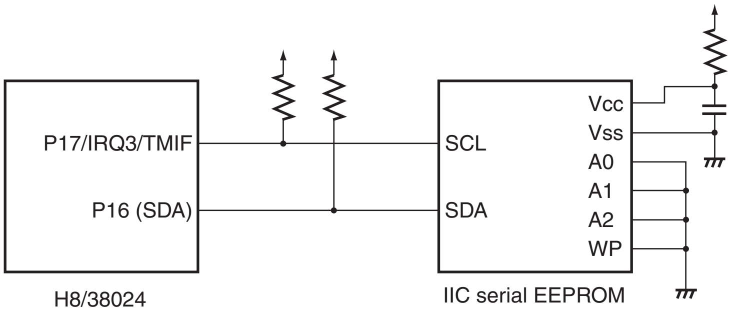


Figure 3.1 Connection between H8/38024 and IIC Serial EEPROM

Table 3.1 Pins of IIC Serial EEPROM

Pin Name	Function
A0 to A2	Device address
SCL	Serial clock input
SDA	Serial data I/O
WP	Write protection
Vcc	Vcc
Vss	GND

2. The block diagram shown in figure 3.2 illustrates how the IIC serial EEPROM is connected to the H8/38024. The H8/38024's functions used in this application are explained below.
   A. Set the P17/$\overline{\text{IRQ3}}$/TMIF pin as the P17 output pin and connect to the SCL pin of the IIC serial EEPROM for use as the serial clock output pin.
   C. Connect the P16 pin to the SDA pin of the IIC serial EEPROM for use as the serial data input/output pin.
   D. If the PCR16 bit of the port control register PCR1 is cleared to 0, the P16 pin functions as an input pin. If the PCR16 is set to 1, the P16 pin functions as an output pin.

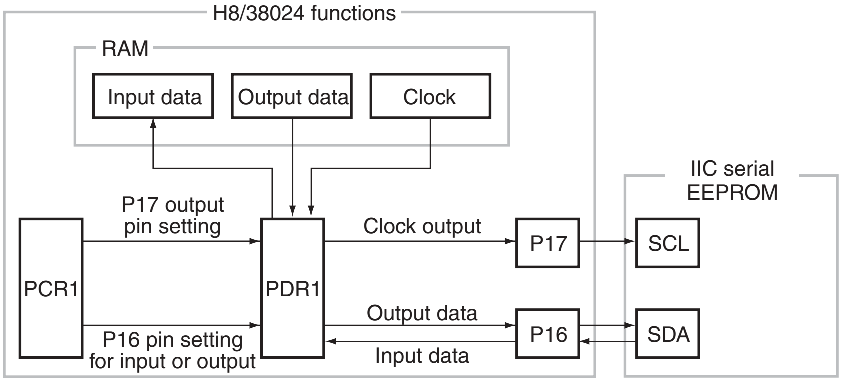


Figure 3.2 Block Diagram for Connection with IIC Serial EEPROM

3. Pin functions are assigned as below for connection with the IIC serial EEPROM.

P17 pin	Serial clock output
P16 pin	Serial data Input/output

4. The following explains specifications of the IIC serial EEPROM used for the sample task.
   A. The IIC serial EEPROM is an EEPROM (electrically erasable and programmable ROM) with a two-wire serial interface. The sample task uses a 64-kbit EEPROM (HN58X2464FPI) from Renesas technology or a 2-kbit EEPROM (FM24C03UFLM8) from Fairchild.
   B. The following lists features of the EEPROM used for the sample task.

- 64-kbit EEPROM (HN58X2464FPI) from Renesas technology

Single power supply		1.8 V to 5.5 V
Two-wire serial interface		IIC bus interface
Operating frequency		400 kHz
Current consumption	Standby state	3 μA (max.)
	Reading	1 mA (max.)
	Writing	3 mA (max.)
Page write		Page size: 32 bytes
Write cycle time		10 ms (2.7 to 5.5 V or higher)/15 ms (1.8 to 2.7 V)
Endurance		10^5 data changes (for page write)

- 2-kbit EEPROM (FM24C03UFLM8) form Fairchild

Single power supply		2.7 V to 5.5 V
Two-wire serial interface		IIC bus interface
Operating frequency		400 kHz
Current consumption	Standby state	10 μA (max.)
	Reading	1 mA (max.)
	Writing	1 mA (max.)
Page write		Page size: 16 bytes
Write cycle time		10 ms (4.5 to 5.5 V or higher)/15 ms (2.7 to 4.5 V)
Endurance		10^6 data changes (for page write)

C. To start reading and writing operation, the start condition under which the SDA input is changed from high to low while the SCL input is high is required.

Setting the SDA input from low to high while the SCL input is high makes the stop condition. In reading, reading operation is terminated by inputting the stop condition and the standby state is entered. In writing, input of rewrite data is terminated by inputting the stop condition. Standby mode is entered after the memory has been rewritten for a programming time (twc). Figure 3.3 shows timing waveforms illustrating the start and stop conditions.

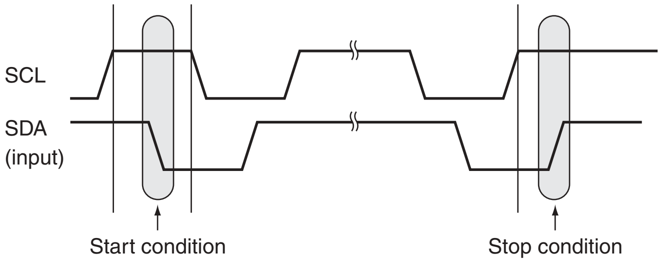


Figure 3.3 Signal Output Timing Waveforms under Start and Stop Conditions

D. Serial data including address and reading information is transferred in 8-bit units. The acknowledge signal indicates that 8-bit data has been transmitted or received successfully. The receiving side outputs 0 at the SCL's ninth clock cycle. The transmitting side releases the bus to receive the acknowledge signal at the ninth clock cycle.

As viewed from the EEPROM, data is only received in writing. Upon the completion of reception of 8-bit data, therefore, the EEPROM outputs 0 as the acknowledge signal at the ninth clock cycle. In reading, the EEPROM outputs 0 as the acknowledge signal upon reception of 8-bit data after the start condition. Then, the EEPROM outputs read data in 8-bit units. After this output, it releases the bus, waiting for 0 as the acknowledge signal to be sent from the master side. Upon detection of 0 as the acknowledge signal, the EEPROM outputs read data for the next address. If having received the stop condition without detecting 0 as the acknowledge signal, the EEPROM terminates reading operation, entering the standby state. If it has not detected 0 as the acknowledge signal and not received the stop condition, the EEPROM does not output data and stays in the bus-released state. Figure 3.4 illustrates timing waveforms for the acknowledge signal.

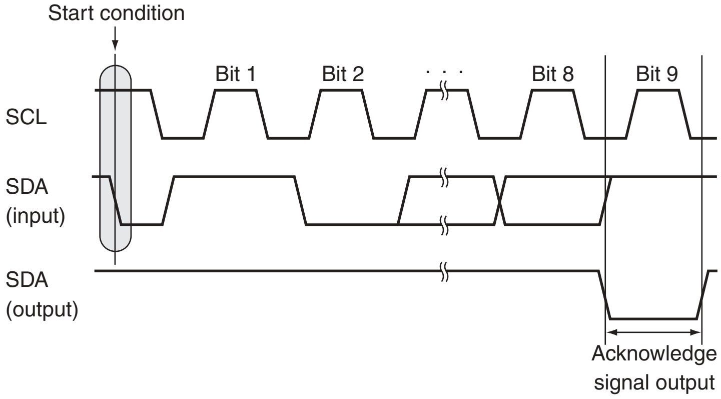


Figure 3.4 Timing Waveforms for Acknowledge Signal Output

E. Following the start condition, input an 8-bit device address word. By this input, the device starts reading and writing operation. The device address word consists of a 4-bit device code, a 3-bit device address code, and a one-bit read/write code.
The upper four bits of the device address word provide the device code used for device type identification. The EEPROM used for the sample task has the device code fixed at 1010.
Following the device code, input a 3-bit device address code in the order of A2, A1, and A0. The device address code is used to determine which of up to eight devices connected to the bus is to be selected. In the sample task, the device address code for the EEPROM is set to 000.
Bit 8 of the device address word is an R/W code. If 0 is input, the EEPROM is set for writing operation. If 1 is input, the EEPROM is set for reading operation. If the device code is not 1010 or if the device address code does not match, the EEPROM enters standby mode rather than reading/writing operation. Figure 3.5 shows the device address word.

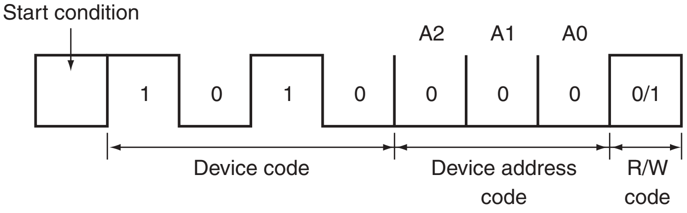


Figure 3.5 Device Address Word

F. In this sample task, the page write function that rewrites up to 32 (or 16) bytes at a time is used for writing operation. Input the start condition, the device address word, the memory address (n), and write data (Dn) in that order while checking 0 output of the acknowledge signal every nine bits. If write data (Dn+1) is input, the EEPROM will enter page-write mode. When write data (Dn+1) is input, the in-page address (a0 to a4) is incremented automatically to (n+1). So, write data can be input continuously, with the in-page address incremented. Up to 32 (or 16) bytes of write data can be input. When the in-page address (a0 to a4) reaches the last address in the page, the address rolls over to the start address. In this case, write data is input to the same address twice or more times, but the last input write data is valid. If the stop condition is input, the EEPROM terminates write data input, entering rewriting operation. Figure 3.6 illustrates page-write operation.

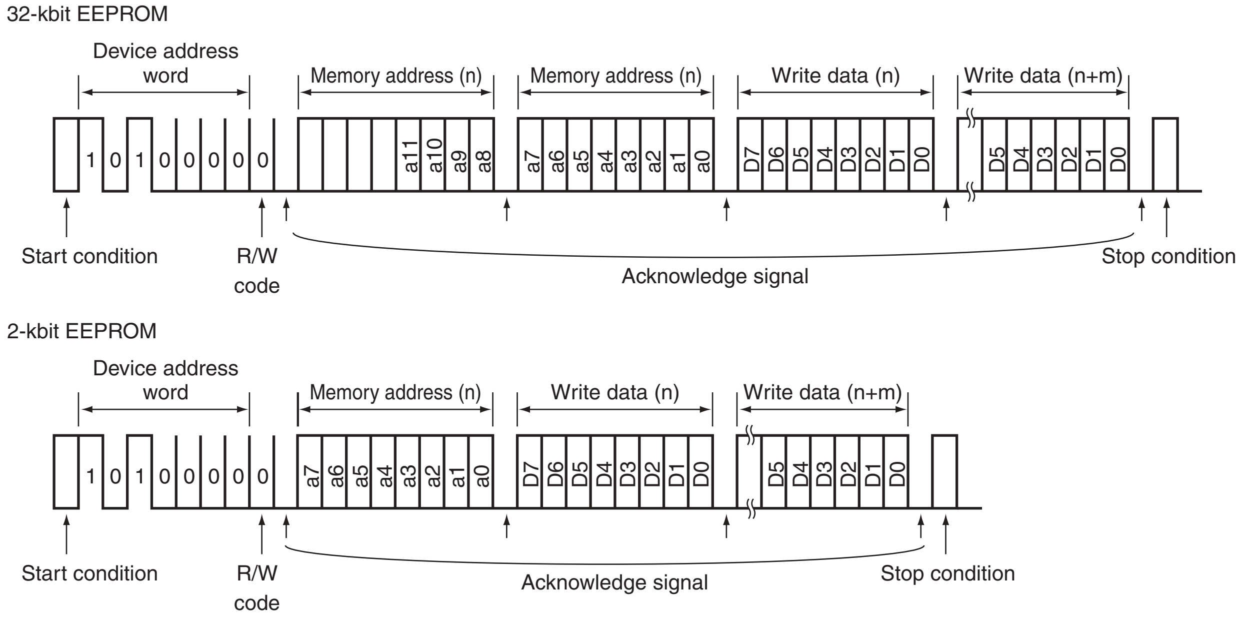


Figure 3.6 Page-Write Operation

G. An acknowledge polling function is provided to determine whether the EEPROM is being rewritten. During rewriting, input the start condition, then an 8-bit device address word. When using the acknowledge polling function, the read/write code may be either 1 or 0. Whether the EEPROM is being rewritten is determined by the acknowledge signal at the ninth bit. An acknowledge signal of 1 indicates that the EEPROM is being rewritten, while an acknowledge signal of 0 indicates that the rewriting has ended. The acknowledge polling function becomes active upon stop condition input after write data is input.

H. In this sample task, sequential read mode, in which data are sequentially read, is used for reading. Input the start address for data to be read in dummy write mode. Inputting an acknowledge signal of 0 after the output of 8-bit data increments the address, with next 8-bit data output. If an acknowledge signal of 0 is input continuously after the data output, the EEPROM sequentially outputs data while incrementing the address. The address, if it has reached the last one, rolls over to 0. In that case as well, sequential read mode can be used. To terminate the operation, input an acknowledge signal of 1 (or release the bus without inputting an acknowledge signal) and then input the stop condition. Figure 3.7 illustrates sequential read operation.

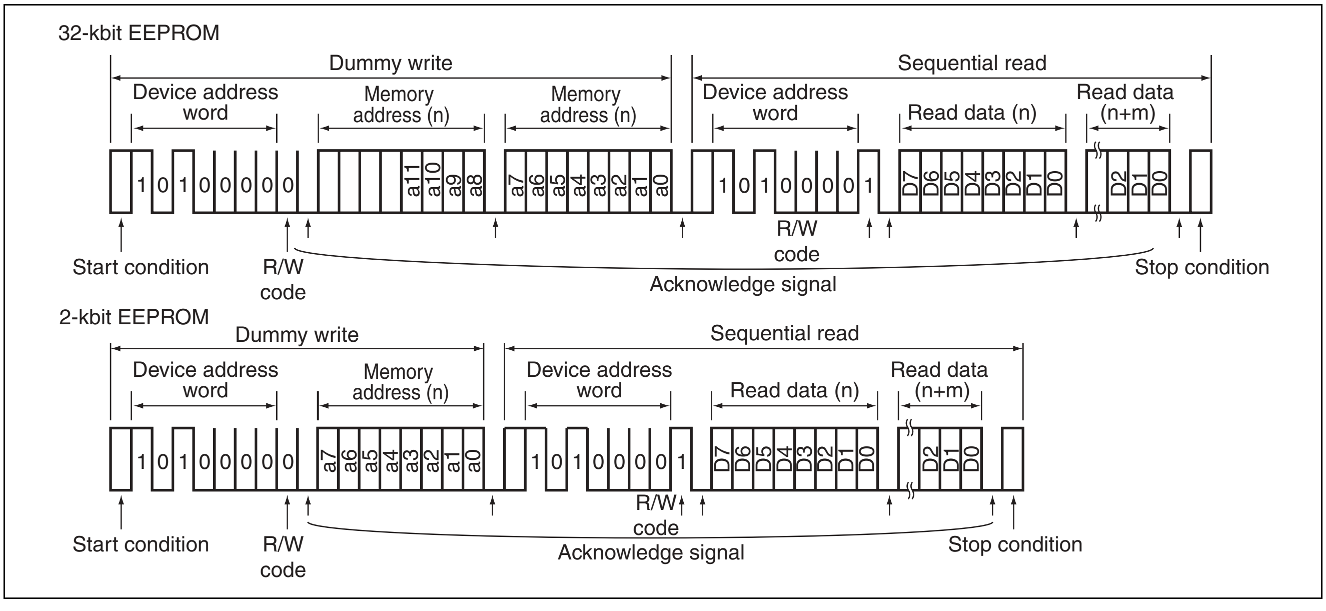


Figure 3.7 Sequential Reading

I. To enter reading operation after the end of writing operation, the IIC serial EEPROM must wait for the write cycle time (twc), which is up to 10 ms when a voltage of 5 volts is applied. Figure 3.8 shows write cycle timing waveforms.

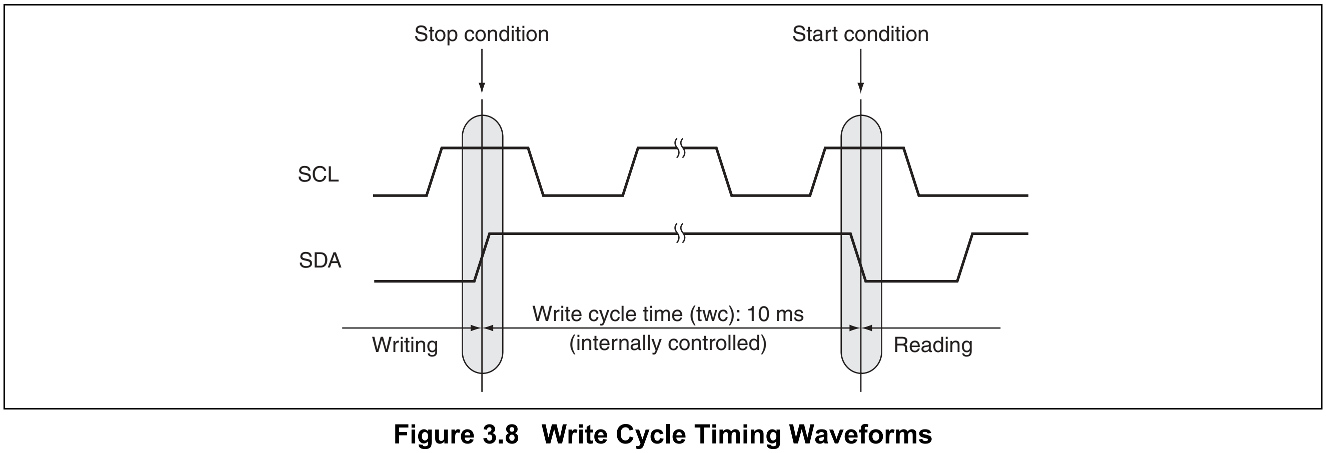
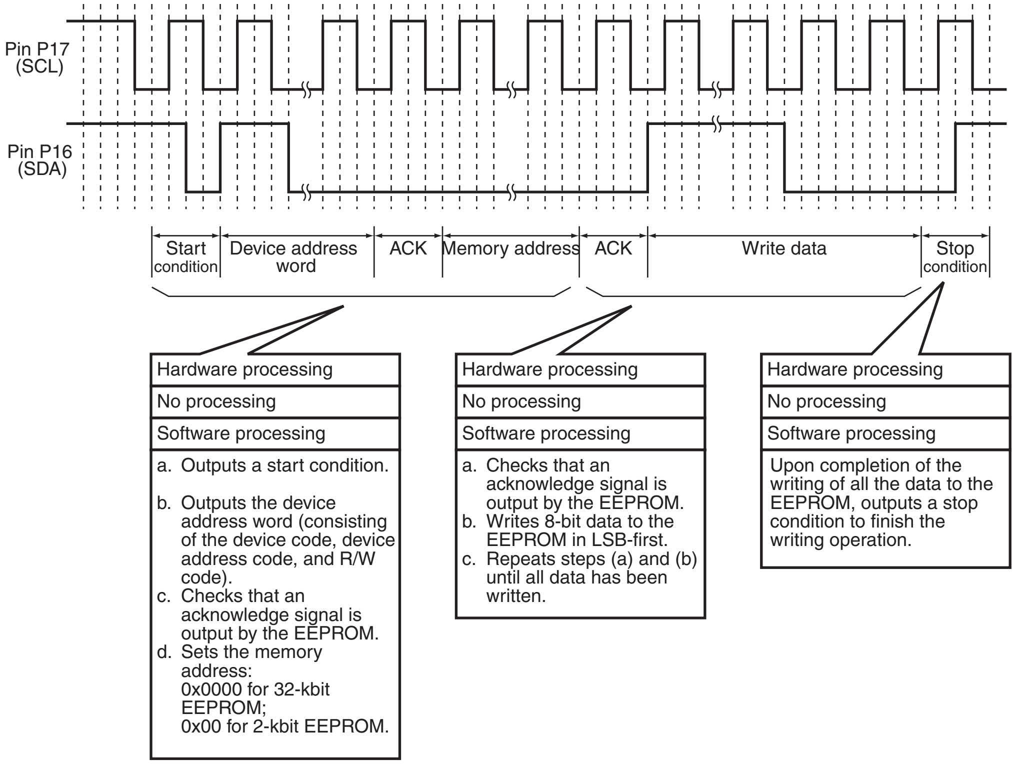


Figure 3.8 Write Cycle Timing Waveforms

4. Principle of Operation

1. Figure 4.1 illustrates the principle of operation of this sample task. As shown here, the H8/38024's hardware and software processing allows data to be written (transmitted) to the EEPROM.



Hardware processing

No processing

Software processing

a. Outputs a start condition.
b. Outputs the device address word (consisting of the device code, device address code, and R/W code).
c. Checks that an acknowledge signal is output by the EEPROM.
d. Sets the memory address:
0x0000 for 32-kbit EEPROM;
0x00 for 2-kbit EEPROM.

Hardware processing

No processing

Software processing

a. Checks that an acknowledge signal is output by the EEPROM.
b. Writes 8-bit data to the EEPROM in LSB-first.
c. Repeats steps (a) and (b) until all data has been written.

Hardware processing

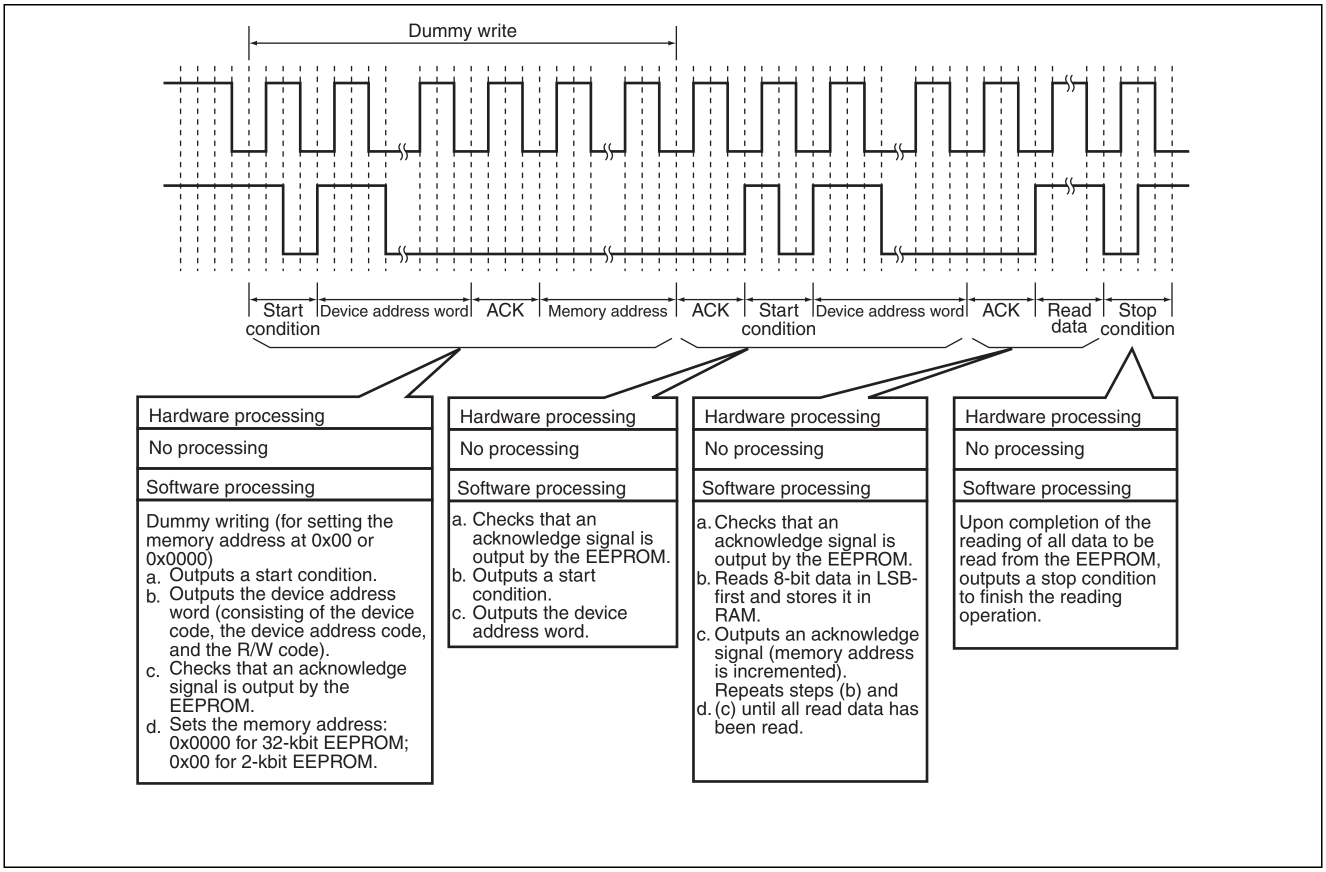
No processing

Software processing

Upon completion of the writing of all the data to the EEPROM, outputs a stop condition to finish the writing operation.

Figure 4.1 Operation Principle of EEPROM Writing

2. Figure 4.2 illustrates the operation principle for reading (reception). As shown here, the H8/38024's hardware and software processing allows data to be read (received) from the EEPROM.



Hardware processing

No processing

Software processing

Dummy writing (for setting the memory address at 0x00 or 0x0000)

a. Outputs a start condition.
b. Outputs the device address word (consisting of the device code, the device address code, and the R/W code).
c. Checks that an acknowledge signal is output by the EEPROM.
d. Sets the memory address: 0x0000 for 32-kbit EEPROM; 0x00 for 2-kbit EEPROM.

Hardware processing

No processing

Software processing

a. Checks that an acknowledge signal is output by the EEPROM.
b. Outputs a start condition.
c. Outputs the device address word.

Hardware processing

No processing

Software processing

a. Checks that an acknowledge signal is output by the EEPROM.
b. Reads 8-bit data in LSB-first and stores it in RAM.
c. Outputs an acknowledge signal (memory address is incremented).
d. Repeats steps (b) and (c) until all read data has been read.

Hardware processing

No processing

Software processing

Upon completion of the reading of all data to be read from the EEPROM, outputs a stop condition to finish the reading operation.

Figure 4.2 Operation Principle for Reading from EEPROM

3. Table 4.1 lists the operation of input/output to/from port 1 used for the sample task. Settings listed in table 4.1 are used to output the serial clock, and input/output serial data.

Table 4.1 Input/Output Operation

Register		Transmission to IIC Serial EEPROM	Reception from IIC Serial EEPROM
PCR1	PCR17	1	1
	PCR16	1	0
PDR1	P17	Transmits the clock	Transmits the clock
	P16	Transmit data	Receive data

4. Figure 4.3 shows the memory map of the H8/38024 used for the sample task.

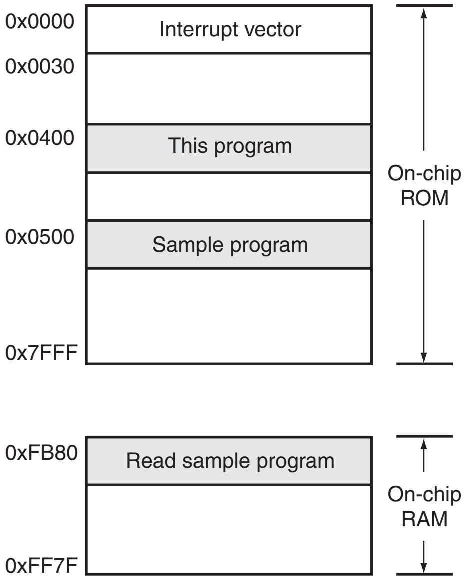


Figure 4.3 Memory Map of H8/38024 Used for Sample Task

5. Description of Software

5.1 Modules

Table 5.1 lists the modules used for the sample task.

Table 5.1 Description of Modules

Module	Label	Function
Main routine	main	Disables an interrupt, waits for a write cycle time, and controls writing/reading to/from the EEPROM.
Data transmission on IIC bus	send_code	Converts specified data into the signal for the IIC bus to be transmitted to the EEPROM.
SCL and SDA data setting	send_p1a	Sets data on P17 and P16 connected to the SCL and SDA pins of the EEPROM.
Start condition code output	send_start	Outputs the start condition code.
Stop condition code output	send_stop	Outputs the stop condition code.
Bit 1 output	send_bit1	Outputs bit “1” for the IIC bus.
Bit 0 output	send_bit0	Outputs bit “0” for the IIC bus.
Data reception on IIC bus	rcv_code	Reads data from the EEPROM and stores it in the RAM.
Acknowledge signal reception	rcv_ack	Receives the acknowledge signal.
Received sample program	splpgm	Sample program received from the EEPROM

5.2 Arguments

Table 5.2 Description of Arguments

Argument	Function	Label	Data Length	Input/ Output
*inpdt	Start address of the data to be transmitted	send_code	1 byte	Input
SIZE	Number of data bytes to be transmitted	send_code	1 byte	Input
*outpdt	Start address of the area where to store received data	rcv_code	1 byte	Output
SIZE	Number of data bytes to be received	rcv_code	1 byte	Input
dt	P17 and P16 settings	send_p1a	1 byte	Input

5.3 Internal Registers

Table 5.3 lists the internal registers used for the sample task.

Table 5.3 Description of Internal Registers

Register		Function	Address	Setting
PDR1	P17	Port data register 1 (port data register 17) When P17 = 0, the output level on pin P17 (SCL) is low. When P17 = 1, the output level on pin P17 (SCL) is high.	0xFFD4 Bit 7	—
	P16	Port data register 1 (port data register 16) When P16 = 0, the output level on pin P16 (SDA) is low. When P16 = 1, the output level on pin P16 (SDA) is high.	0xFFD4 Bit 6	—
PCR1	PCR17	Port control register 1 (port control register 17) When PCR17 = 0, the P17 (SCL) is set as the output pin. When PCR17 = 1, the P17 (SCL) is set as the input pin.	0xFFE4 Bit 7	1
	PCR16	Port control register 1 (port control register 16) When PCR16 = 0, the P16 (SDA) is set as the output pin. When PCR16 = 1, the P16 (SDA) is set as the input pin.	0xFFE4 Bit 6	—

5.4 Description of RAM

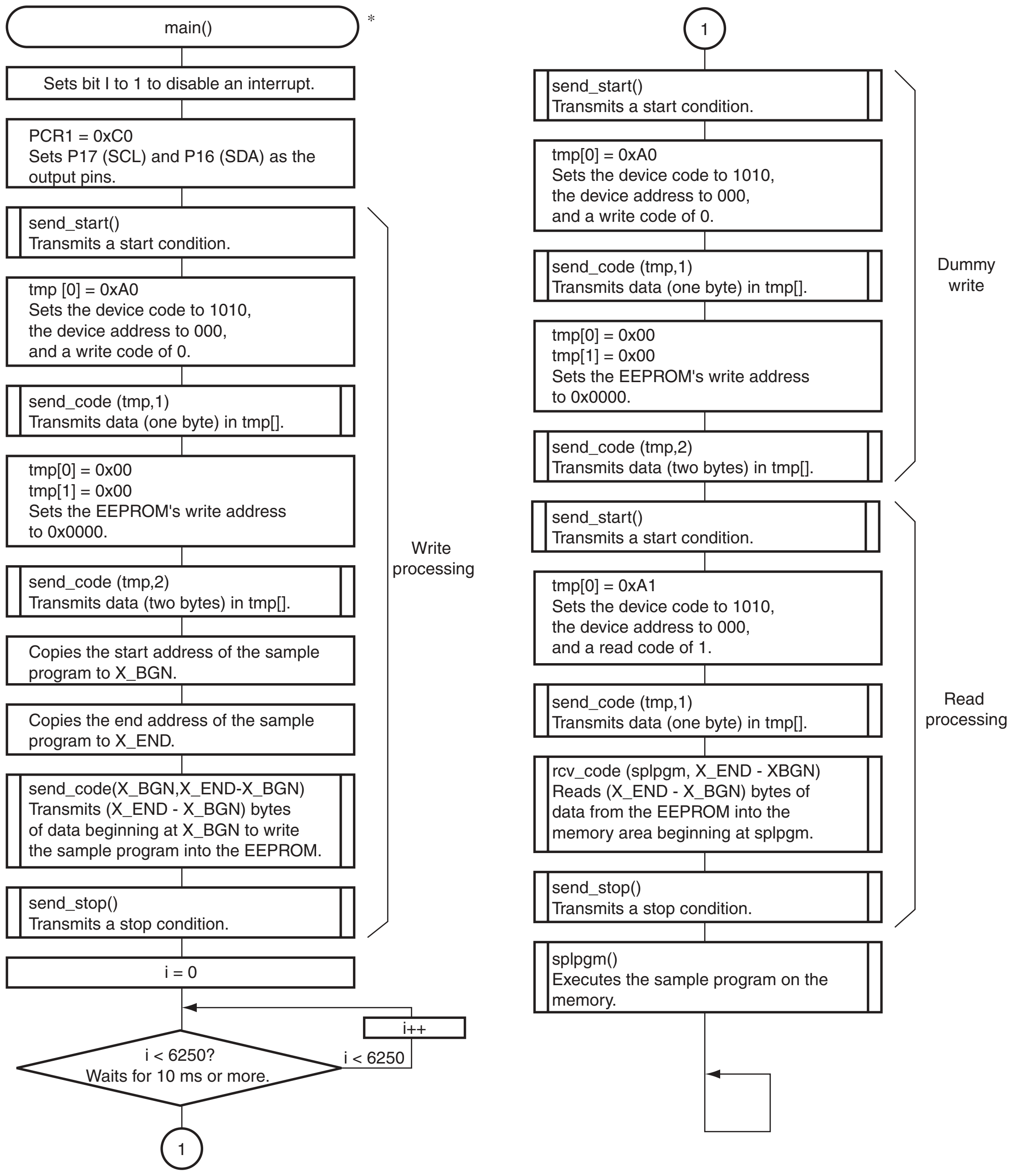
Table 5.4 lists the RAM used for the sample task

Table 5.4 Description of RAM

Label	Function	Address	Used in
splpgm	Stores a sample program read from the EEPROM.	6 bytes	main

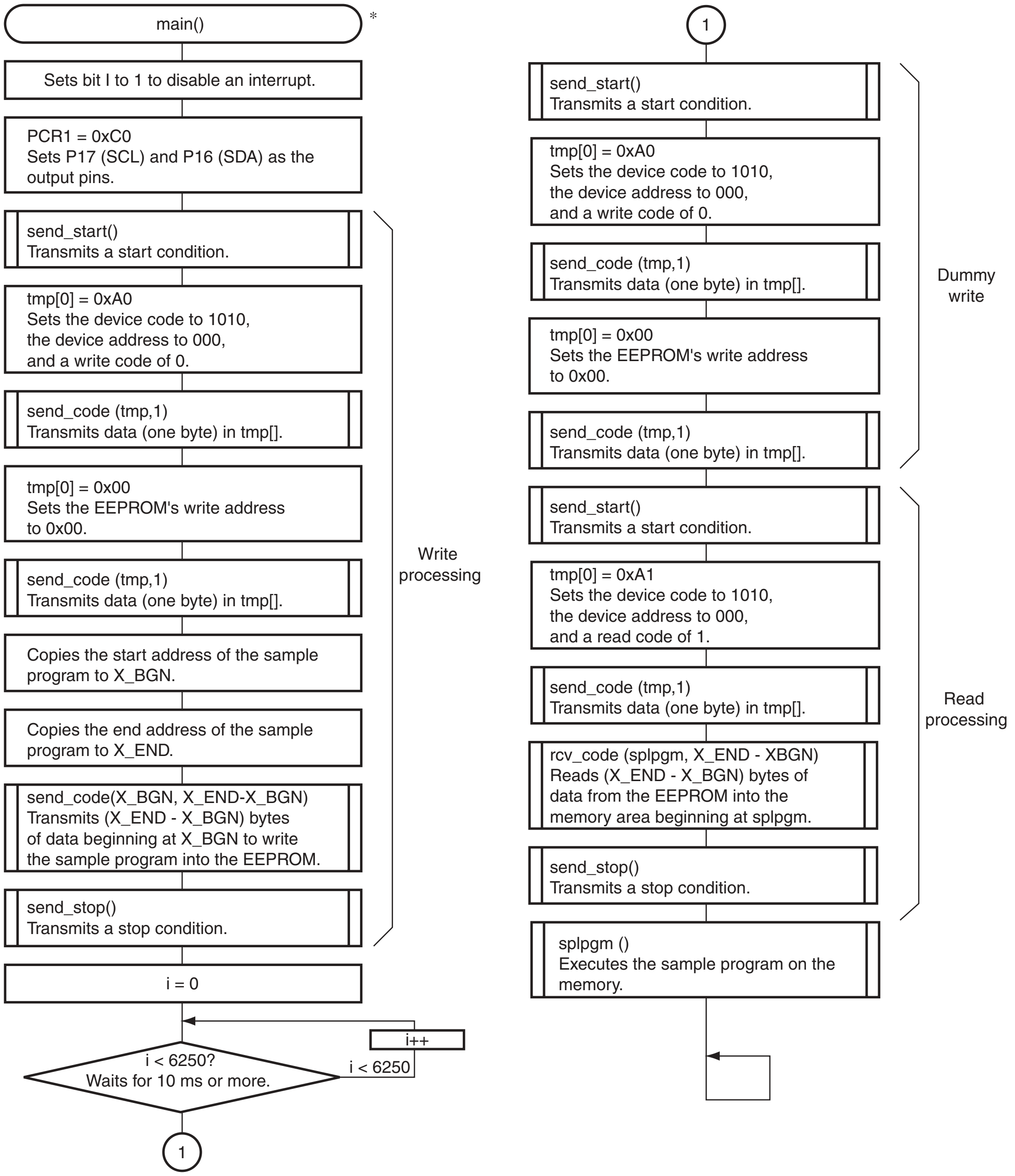
6. Flowchart

1. Main routine in 64-kbit EEPROM


main()
*
Sets bit I to 1 to disable an interrupt.
PCR1 = 0xC0
Sets P17 (SCL) and P16 (SDA) as the output pins.
send_start()
Transmits a start condition.
tmp [0] = 0xA0
Sets the device code to 1010, the device address to 000, and a write code of 0.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
tmp[0] = 0x00
tmp[1] = 0x00
Sets the EEPROM's write address to 0x0000.
send_code (tmp,2)
Transmits data (two bytes) in tmp[].
Copies the start address of the sample program to X_BGN.
Copies the end address of the sample program to X_END.
send_code(X_BGN,X_END-X_BGN)
Transmits (X_END - X_BGN) bytes of data beginning at X_BGN to write the sample program into the EEPROM.
send_stop()
Transmits a stop condition.
Write processing
i = 0
i++
i < 6250? Waits for 10 ms or more.
i < 6250
1
1
send_start()
Transmits a start condition.
tmp[0] = 0xA0
Sets the device code to 1010, the device address to 000, and a write code of 0.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
tmp[0] = 0x00
tmp[1] = 0x00
Sets the EEPROM's write address to 0x0000.
send_code (tmp,2)
Transmits data (two bytes) in tmp[].
Dummy write
send_start()
Transmits a start condition.
tmp[0] = 0xA1
Sets the device code to 1010, the device address to 000, and a read code of 1.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
rcv_code (splpgm, X_END - XBGN)
Reads (X_END - X_BGN) bytes of data from the EEPROM into the memory area beginning at splpgm.
send_stop()
Transmits a stop condition.
Read processing
splpgm()
Executes the sample program on the memory.

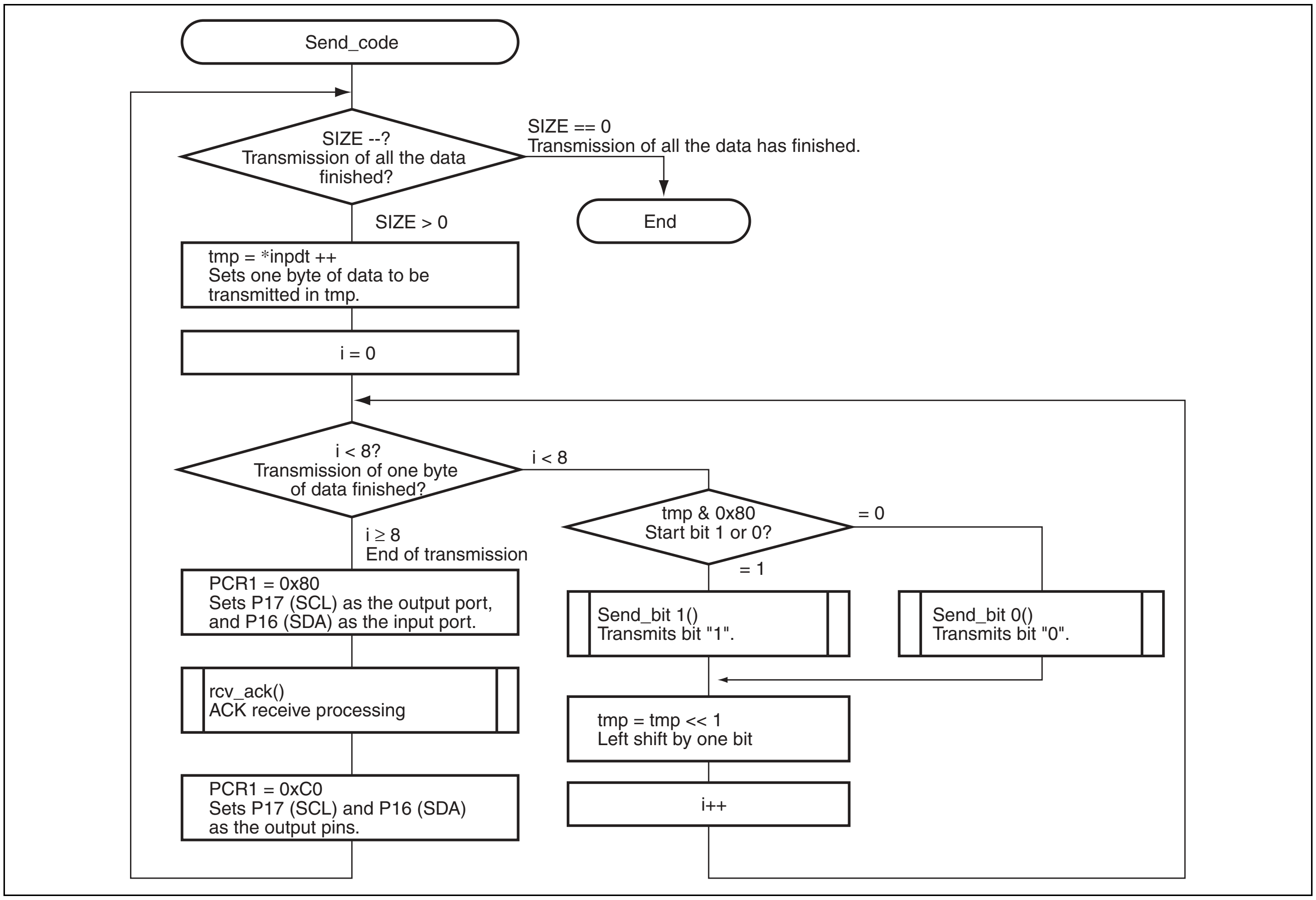
Note: * In the sample task, the stack pointer is set in INIT.SRC (assembly language).

2. Main routine in 2-kbit EEPROM

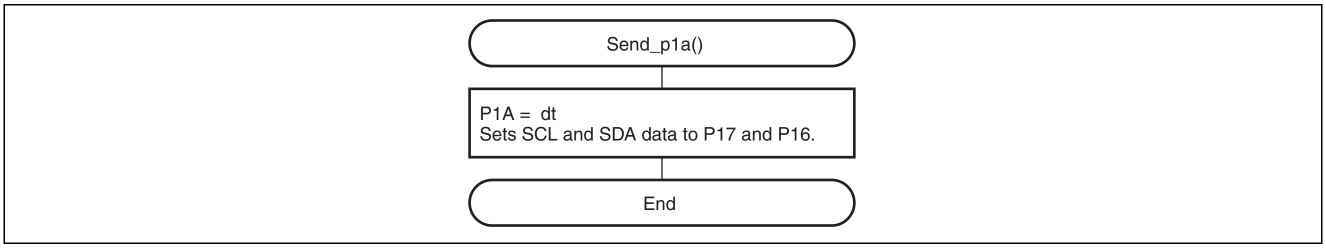

main()
*
Sets bit I to 1 to disable an interrupt.
PCR1 = 0xC0
Sets P17 (SCL) and P16 (SDA) as the output pins.
send_start()
Transmits a start condition.
tmp[0] = 0xA0
Sets the device code to 1010, the device address to 000, and a write code of 0.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
tmp[0] = 0x00
Sets the EEPROM's write address to 0x00.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
Copies the start address of the sample program to X_BGN.
Copies the end address of the sample program to X_END.
send_code(X_BGN, X_END-X_BGN)
Transmits (X_END - X_BGN) bytes of data beginning at X_BGN to write the sample program into the EEPROM.
send_stop()
Transmits a stop condition.
Write processing
i = 0
i++
i < 6250?
Waits for 10 ms or more.
i < 6250
1
1
send_start()
Transmits a start condition.
tmp[0] = 0xA0
Sets the device code to 1010, the device address to 000, and a write code of 0.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
tmp[0] = 0x00
Sets the EEPROM's write address to 0x00.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
Dummy write
send_start()
Transmits a start condition.
tmp[0] = 0xA1
Sets the device code to 1010, the device address to 000, and a read code of 1.
send_code (tmp,1)
Transmits data (one byte) in tmp[].
rcv_code (splpgm, X_END - XBGN)
Reads (X_END - X_BGN) bytes of data from the EEPROM into the memory area beginning at splpgm.
send_stop()
Transmits a stop condition.
Read processing
splpgm ()
Executes the sample program on the memory.

Note: * In the sample task, the stack pointer is set in INIT.SRC (assembly language).

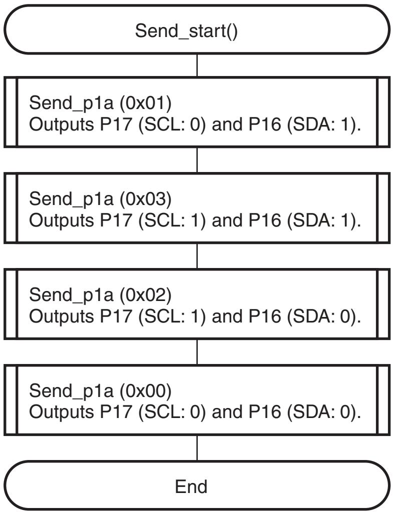
3. Data transmission on IIC bus


Send_code
SIZE --?
Transmission of all the data finished?
SIZE == 0
Transmission of all the data has finished.
End
SIZE > 0
tmp = *inpdt ++
Sets one byte of data to be transmitted in tmp.
i = 0
i < 8?
Transmission of one byte of data finished?
i < 8
tmp & 0x80
Start bit 1 or 0?
= 0
= 1
i ≥ 8
End of transmission
PCR1 = 0x80
Sets P17 (SCL) as the output port, and P16 (SDA) as the input port.
Send_bit 1()
Transmits bit "1".
Send_bit 0()
Transmits bit "0".
rcv_ack()
ACK receive processing
tmp = tmp << 1
Left shift by one bit
PCR1 = 0xC0
Sets P17 (SCL) and P16 (SDA) as the output pins.
i++

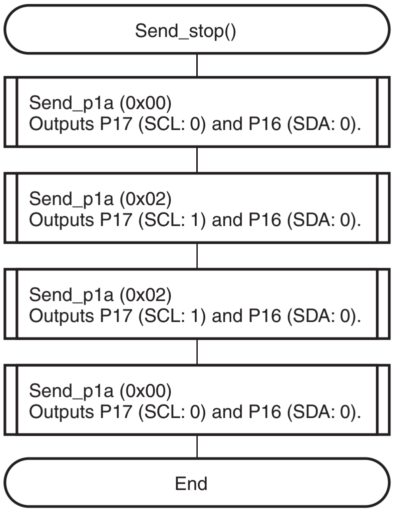
4. SCL and SDA data setting



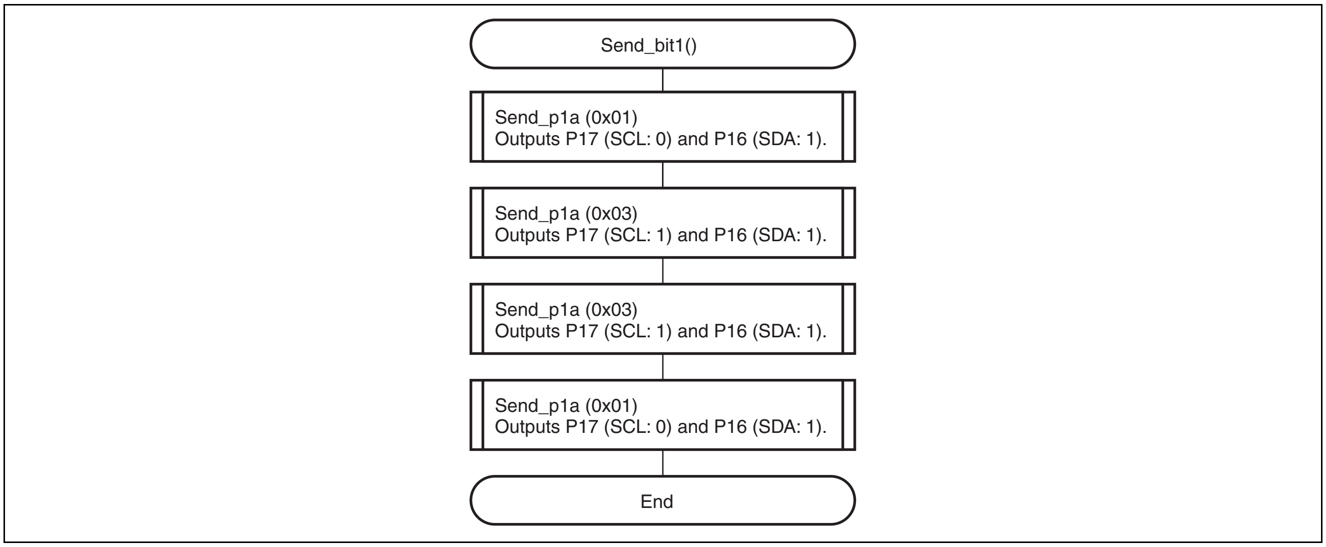
5. Start condition code output


Send_start()
Send_p1a (0x01)
Outputs P17 (SCL: 0) and P16 (SDA: 1).
Send_p1a (0x03)
Outputs P17 (SCL: 1) and P16 (SDA: 1).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
End

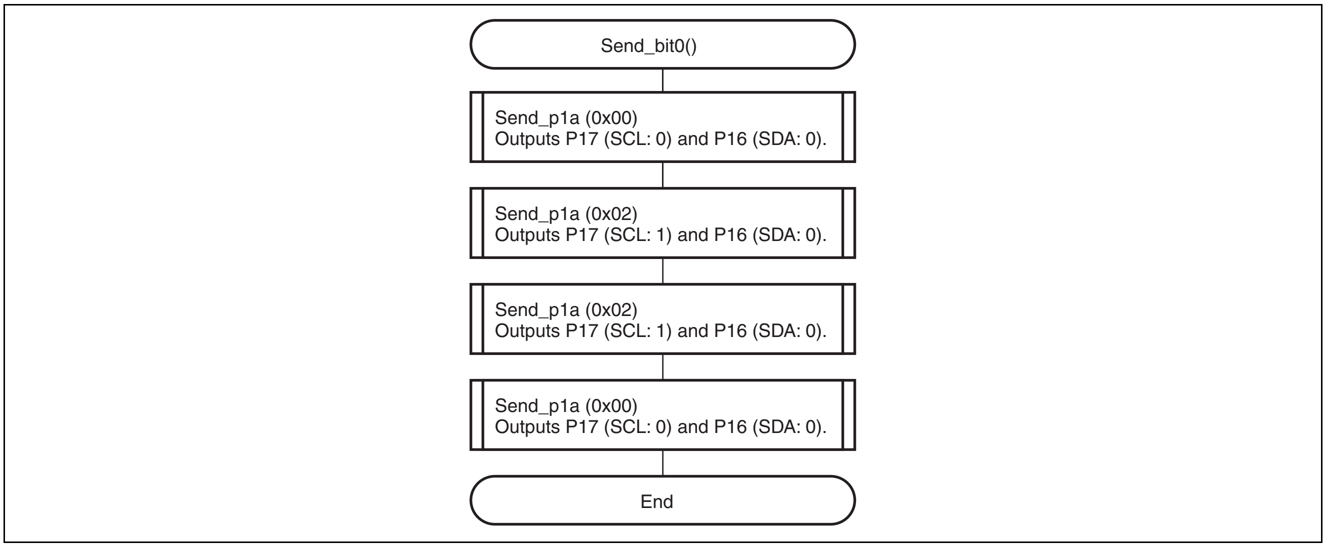
6. Stop condition code output


Send_stop()
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
End

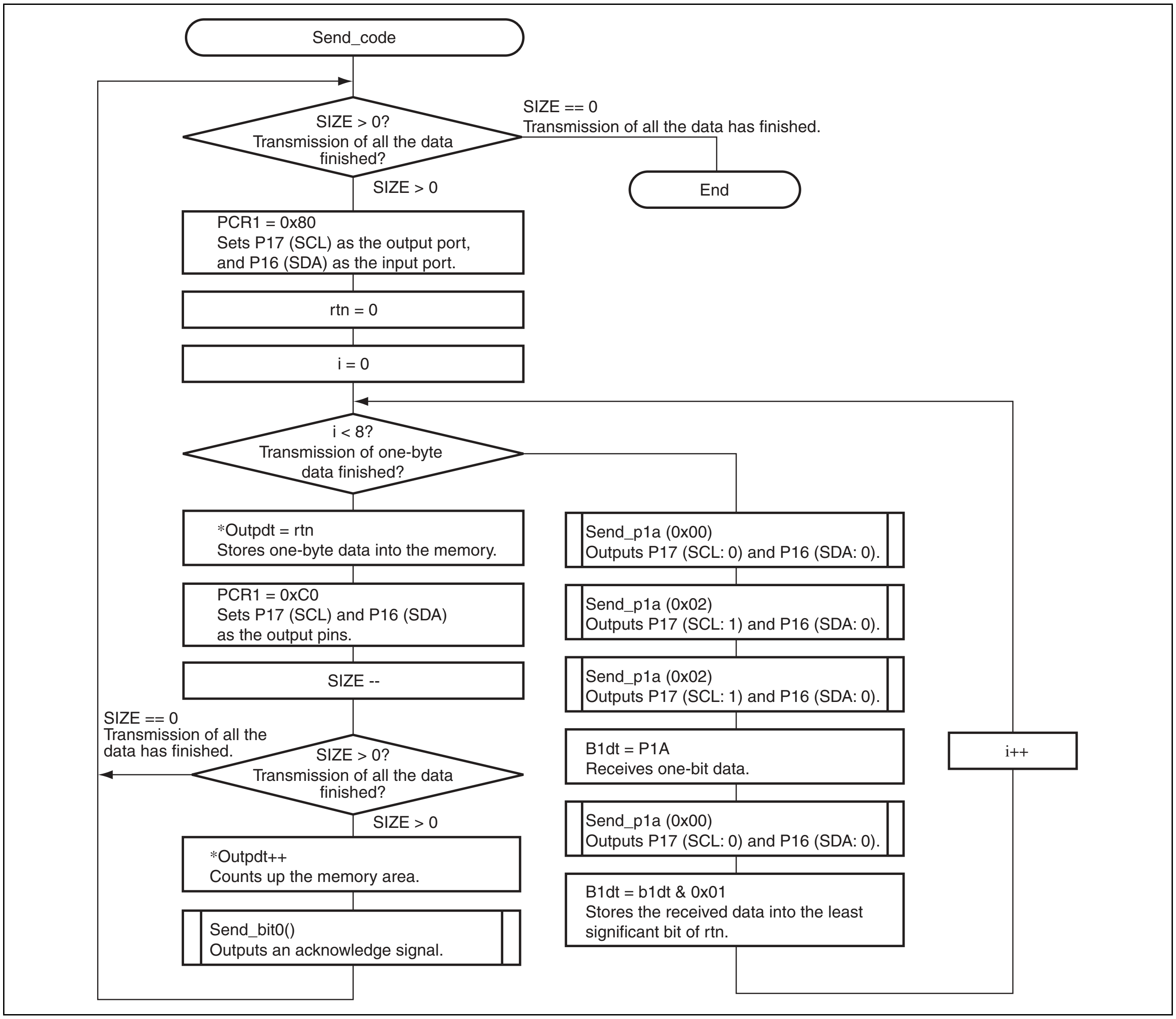
7. Bit 1 output


Send_bit1()
Send_p1a (0x01)
Outputs P17 (SCL: 0) and P16 (SDA: 1).
Send_p1a (0x03)
Outputs P17 (SCL: 1) and P16 (SDA: 1).
Send_p1a (0x03)
Outputs P17 (SCL: 1) and P16 (SDA: 1).
Send_p1a (0x01)
Outputs P17 (SCL: 0) and P16 (SDA: 1).
End

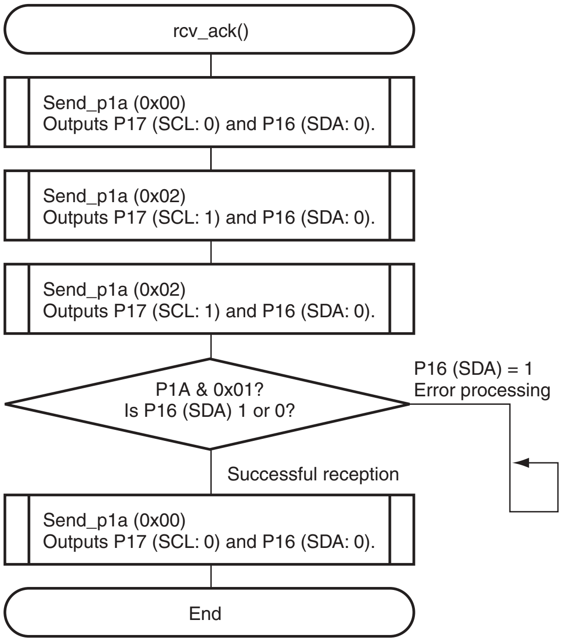
8. Bit 0 output


Send_bit0()
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
End

9. Data reception on IIC bus


Send_code
SIZE > 0?
Transmission of all the data finished?
SIZE == 0
Transmission of all the data has finished.
End
SIZE > 0
PCR1 = 0x80
Sets P17 (SCL) as the output port, and P16 (SDA) as the input port.
rtn = 0
i = 0
i < 8?
Transmission of one-byte data finished?
*Outpdt = rtn
Stores one-byte data into the memory.
PCR1 = 0xC0
Sets P17 (SCL) and P16 (SDA) as the output pins.
SIZE --
SIZE == 0
Transmission of all the data has finished.
SIZE > 0?
Transmission of all the data finished?
SIZE > 0
*Outpdt++
Counts up the memory area.
Send_bit0()
Outputs an acknowledge signal.
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
B1dt = P1A
Receives one-bit data.
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
B1dt = b1dt & 0x01
Stores the received data into the least significant bit of rtn.
i++

10. Acknowledge signal reception


rcv_ack()
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
Send_p1a (0x02)
Outputs P17 (SCL: 1) and P16 (SDA: 0).
P1A & 0x01?
Is P16 (SDA) 1 or 0?
P16 (SDA) = 1
Error processing
Successful reception
Send_p1a (0x00)
Outputs P17 (SCL: 0) and P16 (SDA: 0).
End

7. Program Listing

INIT.SRC (Program listing)

```
    .EXPORT   _INIT
    .IMPORT   _main
 ;
    .SECTION  P,CODE
 _INIT:
    MOV.W     #H'FF80,R7
    LDC.B     #B'10000000,CCR
    JMP       @_main
 ;
    .END
```

7.1 Program list for 64-kbit EEPROM

```
/*************************************************************************************************************/
/*                                                                                                         */
/*  H8/300L Super Low Power Series                                                                         */
/*      -H8/38024 Series-                                                                                  */
/*  Application Note                                                                                       */
/*                                                                                                         */
/*  '64kbit EEPROM Write & Read Control'                                                                   */
/*                                                                                                         */
/*  Function                                                                                               */
/*  : I/O Port Base                                                                                        */
/*                                                                                                         */
/*  External Clock : 10MHz                                                                                 */
/*  Internal Clock : 5MHz                                                                                  */
/*  Sub Clock      : 32.768kHz                                                                             */
/*                                                                                                         */
/*************************************************************************************************************/

#include    <machine.h>

/*************************************************************************************************************/
/*  Symbol Definition                                                                                      */
/*************************************************************************************************************/
struct BIT {
    unsigned char   b7:1;       /* bit7  */
    unsigned char   b6:1;       /* bit6  */
    unsigned char   b5:1;       /* bit5  */
    unsigned char   b4:1;       /* bit4  */
    unsigned char   b3:1;       /* bit3  */
    unsigned char   b2:1;       /* bit2  */
    unsigned char   b1:1;       /* bit1  */
    unsigned char   b0:1;       /* bit0  */
};

struct BIT2 {
    unsigned char   ba:2;       /* bit7,6 */
    unsigned char     :5;       /* other  */
};

#define     PDR1        *(volatile unsigned char *)0xFFD4        /* Port Data Register 1                     */
#define     PDR1_BIT2   (*(struct BIT2 *)0xFFD4)                 /* Port Data Register 1                     */
#define     P1A         PDR1_BIT2.ba                             /* P17, P16                                 */
#define     PCR1        *(volatile unsigned char *)0xFFE4        /* Port Control Register 1                  */
```

```
/**************************************************************************************************************/
/*  Function define                                                                                           */
/**************************************************************************************************************/
extern void INIT( void );                                         /* SP Set                                   */
extern void splpgm( void );

void main( void );
void send_code( unsigned char *inpdt, unsigned char SIZE );
void send_p1a( unsigned char dt );
void send_start( void );
void send_stop( void );
void send_bit1( void );
void send_bit0( void );
void rcv_code( unsigned char *outpdt, unsigned char SIZE );
void rcv_ack( void );

/**************************************************************************************************************/
/*  Vector Address                                                                                            */
/**************************************************************************************************************/
#pragma section     V1                                            /* Vector Section Set                       */
void (*const VEC_TBL1[])(void) = {                                /* 0x0000 - 0x000F                          */
    INIT                                                          /* 0x0000 Reset Vector                      */
};

#pragma section                                                   /* P                                        */
/**************************************************************************************************************/
/*  Main Program                                                                                              */
/**************************************************************************************************************/
void main( void )
{
    unsigned char   tmp[2];
    unsigned char   *X_BGN;
    unsigned char   *X_END;
    unsigned short  i;

    set_imask_ccr(1);                                             /* Interrupt Disable                        */

    PCR1 = 0xC0;                                                  /* P17, P16 Set Output Port                 */
    send_start();                                                 /* Send Start Condition                     */
    tmp[0] = 0xA0;                                                /* Set Write Code                           */
    send_code(tmp,1);                                             /* Send 1byte                               */
    tmp[0] = 0x00;                                                /* Set EEPROM Write Address                 */
    tmp[1] = 0x00;
    send_code(tmp,2);                                             /* Send 2byte                               */

    X_BGN = __sectop("SPLPG");                                    /* Sample Program Top address               */
    X_END = __secend("SPLEND");                                   /* Sample Program End address               */
    send_code(X_BGN,X_END-X_BGN);                                 /* Send and Write Sample Program            */
    send_stop();                                                  /* Send Stop Condition                      */

    for(i = 0; i < 6250; i++);                                    /* Need to wait 10 msec                     */

    send_start();                                                 /* Dummy Write                              */
    tmp[0] = 0xA0;
    send_code(tmp,1);
    tmp[0] = 0x00;
    tmp[1] = 0x00;
    send_code(tmp,2);
    send_start();                                               /* Send Start Condition                  */
    tmp[0] = 0xA1;                                              /* Set Read Code                         */
    send_code(tmp,1);                                           /* Send 1byte                            */
    rcv_code((unsigned char*)splpgm,X_END-X_BGN);               /* Read EEPROM --> RAM (splpgm)          */
    send_stop();                                                /* Send Stop Condition                   */

    splpgm();                                                   /* Go to Sample Program                  */

    while(1);
}

/*****************************************************************************************************************/
/*  Send IIC Communication Data                                                                                  */
/*****************************************************************************************************************/
void send_code( unsigned char *inpdt, unsigned char SIZE )
{
    unsigned char i,tmp;

    while(SIZE--){
        tmp = *inpdt++;                                         /* Set 1 byte                            */
        for(i = 0; i < 8; i++){                                 /* 1 byte Send Finish?                   */
            if(tmp & 0x80)
                send_bit1();                                    /* Send bit"1"                           */
            else
                send_bit0();                                    /* Send bit"0"                           */

            tmp = tmp<<1;                                       /* Next bit                              */
        }
        PCR1 = 0x80;                                            /* P17: Output Port, P16: Input Port     */
        rcv_ack();                                              /* Receive Acknowledge                   */
        PCR1 = 0xC0;                                            /* P17, P16 Set Output Port              */
    }
}

/*****************************************************************************************************************/
/*  Set SCL,SDA                                                                                                  */
/*****************************************************************************************************************/
void send_p1a( unsigned char dt )
{
    P1A = dt;
}

/*****************************************************************************************************************/
/*  Start Condition                                                                                              */
/*****************************************************************************************************************/
void send_start( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x02);
    send_p1a(0x00);
}
```

```
/**************************************************************************************************************/
/*  Stop Condition
/**************************************************************************************************************/
void send_stop( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x03);
    send_p1a(0x01);
}

/**************************************************************************************************************/
/*  Send bit "1"                                                                                              */
/**************************************************************************************************************/
void send_bit1( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x03);
    send_p1a(0x01);
}

/**************************************************************************************************************/
/*  Send bit "0"                                                                                              */
/**************************************************************************************************************/
void send_bit0( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    send_p1a(0x00);
}

/**************************************************************************************************************/
/*  Receive IIC Communication Data                                                                            */
/**************************************************************************************************************/
void rcv_code( unsigned char *outpdt, unsigned char SIZE )
{
    unsigned char i,b1dt,rtn;

    while(SIZE > 0){
        PCR1 = 0x80;                                      /* P17: Output Port   P16:Input Port        */
        rtn = 0;
        for(i = 0; i < 8; i++){                           /* 1 byte Receive Finish?                   */
            rtn = rtn<<1;

            send_p1a(0x00);
            send_p1a(0x02);
            send_p1a(0x02);
            b1dt = P1A;                                   /* Receive 1 bit                            */
            send_p1a(0x00);
            b1dt = b1dt & 0x01;

            rtn = rtn|b1dt;
        }
        *outpdt = rtn;                                    /* Set Receive Data                         */

        PCR1 = 0xC0;                                      /* P17, P16 Set Output Port                 */
        SIZE--;
        if(SIZE > 0){
            *outpdt++;                                           /* Ram Address Count up                      */
            send_bit0();                                         /* Output Acknowledge                        */
        }
    }
}

/**************************************************************************************************************/
/*  Receive Acknowledge                                                                                       */
/**************************************************************************************************************/
void rcv_ack( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    if(P1A & 0x01)                                               /* Receive Check Acknowledge                 */
        while(1);
    send_p1a(0x00);
}
```

Link address specifications

Section Name	Address
CV1	0x0000
P	0x0100
SPLPG,PSPLPG,SPLEND	0x0500
PRAM	0xFB80

7.2 Program list for 2-kbit EEPROM

```
/**************************************************************************************************************/
/*                                                                                                            */
/*  H8/300L Super Low Power Series                                                                            */
/*      -H8/38024 Series-                                                                                     */
/*  Application Note                                                                                          */
/*                                                                                                            */
/*  '2kbit EEPROM Write & Read Control'                                                                       */
/*                                                                                                            */
/*  Function                                                                                                  */
/*  : I/O Port Base                                                                                           */
/*                                                                                                            */
/*  External Clock : 10MHz                                                                                    */
/*  Internal Clock : 5MHz                                                                                     */
/*  Sub Clock      : 32.768kHz                                                                                */
/*                                                                                                            */
/**************************************************************************************************************/

#include    <machine.h>

/**************************************************************************************************************/
/*  Symbol Definition                                                                                         */
/**************************************************************************************************************/
struct BIT {
    unsigned char   b7:1;       /* bit7  */
    unsigned char   b6:1;       /* bit6  */
    unsigned char   b5:1;       /* bit5  */
    unsigned char   b4:1;       /* bit4  */
    unsigned char   b3:1;       /* bit3  */
    unsigned char   b2:1;       /* bit2  */
    unsigned char   b1:1;       /* bit1  */
    unsigned char   b0:1;       /* bit0  */
};

struct BIT2 {
    unsigned char   ba:2;       /* bit7,6 */
    unsigned char     :5;       /* other  */
};

#define     PDR1        *(volatile unsigned char *)0xFFD4        /* Port Data Register 1                      */
#define     PDR1_BIT2   (*(struct BIT2 *)0xFFD4)                 /* Port Data Register 1                      */
#define     P1A         PDR1_BIT2.ba                             /* P17, P16                                  */
#define     PCR1        *(volatile unsigned char *)0xFFE4        /* Port Control Register 1                   */

/**************************************************************************************************************/
/*  Function define                                                                                           */
/**************************************************************************************************************/
extern void INIT( void );                                        /* SP Set                                    */
extern void splpgm( void );

void main( void );
void send_code( unsigned char *inpdt, unsigned char SIZE );
void send_p1a( unsigned char dt );
void send_start( void );
void send_stop( void );
void send_bit1( void );
void send_bit0( void );
void rcv_code( unsigned char *outpdt, unsigned char SIZE );
void rcv_ack( void );

/***************************************************************************************************************/
/*  Vector Address                                                                                             */
/***************************************************************************************************************/
#pragma section     V1                                            /* Vector Section Set                        */
void (*const VEC_TBL1[])(void) = {                                /* 0x0000 - 0x000F                           */
    INIT                                                          /* 0x0000 Reset Vector                       */
};

#pragma section                                                   /* P                                         */
/***************************************************************************************************************/
/*  Main Program                                                                                               */
/***************************************************************************************************************/
void main( void )
{
    unsigned char   tmp[2];
    unsigned char   *X_BGN;
    unsigned char   *X_END;
    unsigned short  i;

    set_imask_ccr(1);                                             /* Interrupt Disable                         */

    PCR1 = 0xC0;                                                  /* P17, P16 Set Output Port                  */
    send_start();                                                 /* Send Start Condition                      */
    tmp[0] = 0xA0;                                                /* Set Write Code                            */
    send_code(tmp,1);                                             /* Send 1byte                                */
    tmp[0] = 0x00;                                                /* Set EEPROM Write Address                  */
    send_code(tmp,1);                                             /* Send 1byte                                */

    X_BGN = __sectop("SPLPG");                                    /* Sample Program Top address                */
    X_END = __secend("SPLEND");                                   /* Sample Program End address                */
    send_code(X_BGN,X_END-X_BGN);                                 /* Send and Write Sample Program             */
    send_stop();                                                  /* Send Stop Condition                       */

    for(i = 0; i < 6250; i++);                                    /* Need to wait 10 msec                      */

send_start();                                                     /* Dummy Write                               */
    tmp[0] = 0xA0;
    send_code(tmp,1);
    tmp[0] = 0x00;
    send_code(tmp,1);

    send_start();                                                 /* Send Start Condition                      */
    tmp[0] = 0xA1;                                                /* Set Read Code                             */
    send_code(tmp,1);                                             /* Send 1byte                                */
    rcv_code((unsigned char*)splpgm,X_END-X_BGN);                 /* Read EEPROM --> RAM(splpgm)               */
    send_stop();                                                  /* Send Stop Condition                       */

    splpgm();                                                     /* Go to Sample Program                      */

    while(1);
}
```

```
/*************************************************************************************************************/
/*  Send IIC Communication Data                                                                              */
/*************************************************************************************************************/
void send_code( unsigned char *inpdt, unsigned char SIZE )
{
    unsigned char i,tmp;

    while(SIZE--){
        tmp = *inpdt++;
/* Set 1 byte                       */
        for(i = 0; i < 8; i++){                                  /* 1 byte Send Finish?                      */
            if(tmp & 0x80)
                send_bit1();                                     /* Send bit"1"                              */
            else
                send_bit0();                                     /* Send bit"0"                              */

            tmp = tmp<<1;                                        /* Next bit                                 */
        }
        PCR1 = 0x80;                                             /* P17: Output Port , P16: Input Port       */
        rcv_ack();                                               /* Receive Acknowledge                      */
        PCR1 = 0xC0;                                             /* P17, P16 Set Output Port                 */
    }
}

/*************************************************************************************************************/
/*  Set SCL,SDA                                                                                              */
/*************************************************************************************************************/
void send_p1a( unsigned char dt )
{
    P1A = dt;
}

/*************************************************************************************************************/
/*  Start Condition                                                                                          */
/*************************************************************************************************************/
void send_start( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x02);
    send_p1a(0x00);
}

/*************************************************************************************************************/
/*  Stop Condition
/*************************************************************************************************************/
void send_stop( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x03);
    send_p1a(0x01);
}
```

```
/**************************************************************************************************************/
/*  Send bit "1"                                                                                              */
/**************************************************************************************************************/
void send_bit1( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x03);
    send_p1a(0x01);
}

/**************************************************************************************************************/
/*  Send bit "0"                                                                                              */
/**************************************************************************************************************/
void send_bit0( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    send_p1a(0x00);
}

/**************************************************************************************************************/
/*  Receive IIC Communication Data                                                                            */
/**************************************************************************************************************/
void rcv_code( unsigned char *outpdt, unsigned char SIZE )
{
    unsigned char i,b1dt,rtn;

    while(SIZE > 0){
        PCR1 = 0x80;                                             /* P17: Output Port, P16: Input Port         */
        rtn = 0;
        for(i = 0; i < 8; i++){                                  /* 1 byte Receive Finish?                    */
            rtn = rtn<<1;

            send_p1a(0x00);
            send_p1a(0x02);
            send_p1a(0x02);
            b1dt = P1A;                                          /* Receive 1 bit                             */
            send_p1a(0x00);
            b1dt = b1dt & 0x01;

            rtn = rtn|b1dt;
        }
        *outpdt = rtn;                                           /* Set Receive Data                          */

        PCR1 = 0xC0;                                             /* P17, P16 Set Output Port                  */
        SIZE--;
        if(SIZE > 0){
            *outpdt++;                                           /* Ram Address Count-up                      */
            send_bit0();                                         /* Output Acknowledge                        */
        }
    }
}
```

```
/**************************************************************************************************************/
/*  Receive Acknowledge                                                                                       */
/**************************************************************************************************************/
void rcv_ack( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    if(P1A & 0x01)                                               /* Receive Check Acknowledge                 */
        while(1);
    send_p1a(0x00);
}
```

Link address specifications

Section Name	Address
CV1	0x0000
P	0x0100
SPLPG, PSPLPG, SPLEND	0x0500
PRAM	0xFB80

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec.19.03	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.