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](http://www.renesas.com)

---

April 1st, 2010
Renesas Electronics Corporation

---

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

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 depend 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.

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

(Nota 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.
SH7080 Group

I²C2 Single-Master Reception (Reading from I²C-Bus EEPROM)

Introduction

This application note describes the receiving operation by the I²C bus interface 2 module (I²C2) in single-master mode. Please use this application note as a guide in designing user programs.

Although the programs given in this application note have been verified for correct operation, we strongly recommend that the user confirm correct operation before applying the programs in the actual application.

Target Device

SH7085

Contents

1. Specifications.................................................................................................................. 2
2. Conditions for Application ............................................................................................ 3
3. Summary of MCU Functions Used ................................................................................. 4
4. Operation ........................................................................................................................ 6
5. Description of Software................................................................................................. 11
6. Flowcharts .................................................................................................................... 18
7. Website .......................................................................................................................... 30
1. Specifications

- The SH7085's I\(^2\)C bus interface 2 module (I\(^2\)C2) is used to read 10 bytes of data from a two-wire serial EEPROM (HN58X2416; 16k bits; 2k words \(\times\) 8 bits).
- The connection is made in single-master configuration, in which the SH7085 is the master device.
- The device code of the connected EEPROM is B'1010.
- Before reading data from the EEPROM, the EEPROM bus state is initialized.*
- Data is read from addresses H'0000 to H'0009 in the EEPROM.
- After 1 byte of data has been received, the acknowledge bit (ACK = 0) is output. (However, ACK = 1 is output after reception of the last byte.)
- The frequency of the I\(^2\)C bus data transfer clock is 400 kHz.

Figure 1 shows an example of connection between the SH7085 and EEPROM, and table 1 shows the SH7085's I\(^2\)C2 settings. Table 2 shows the device address word of the EEPROM used in the sample task.

Note: The EEPROM's SDA bus is forcibly placed in the input state, anticipating the case when the EEPROM is unable to receive from the master device because of the EEPROM's SDA bus remaining in the output state as a result of, for example, communication having been halted while the master device is receiving data from the EEPROM.

![Figure 1 Example of Connection between SH7085 and EEPROM](image-url)
Table 1  $I^2$C2 Settings of SH7085

<table>
<thead>
<tr>
<th>Item</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating mode</td>
<td>Master reception mode</td>
</tr>
<tr>
<td>Transfer clock</td>
<td>400 kHz ($P_0 = 40$ MHz)</td>
</tr>
<tr>
<td>Data bit length</td>
<td>9 bits (including the ACK bit)</td>
</tr>
<tr>
<td>Wait cycles inserted between data and ACK</td>
<td>None</td>
</tr>
<tr>
<td>Interrupt</td>
<td>None</td>
</tr>
<tr>
<td>ACK output</td>
<td>Outputs 1 only when the last data is received.</td>
</tr>
<tr>
<td></td>
<td>Outputs 0 while data is received continuously.</td>
</tr>
</tbody>
</table>

Table 2  Device Address Word of EEPROM

<table>
<thead>
<tr>
<th>Device Code</th>
<th>Device Address Code</th>
<th>R/W Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 1 0 a10 a9 a8</td>
<td>R = 1, W = 0</td>
</tr>
</tbody>
</table>

Note: a10 to a8 are the upper 3 bits of the EEPROM address.

2. Conditions for Application

Operating frequency:
- Internal clock: 80 MHz
- Bus clock: 40 MHz
- Peripheral clock: 40 MHz
- MTU2 clock: 40 MHz
- MTU2S clock: 80 MHz

C compiler: Version 7.1.04 from Renesas Technology Corp.
3. Summary of MCU Functions Used

In this sample task, the I²C bus (Inter IC bus) is used to read data from the EEPROM.

3.1 I²C Bus Interface 2 (I²C2)

The I²C bus interface 2 (I²C2) conforms to the I²C bus interface specifications set up by Philips, and provides a subset of the I²C functions. Figure 2 shows a block diagram of the I²C2 module.

![Figure 2 Block Diagram of I²C2 Module](image-url)
• The I²C bus control register 1 (ICCR1) sets I²C2 operations.
• The I²C bus control register 2 (ICCR2) generates start and stop conditions, drives the SDA pin, monitors the SCL pin, and controls resetting of the I²C bus control circuitry.
• The I²C bus mode register (ICMR) selects MSB- or LSB-first transfer, controls the wait states in master mode, and selects the number of bits to be transferred.
• The I²C bus interrupt enable register (ICIER) enables interrupts, selects the use of the acknowledge bit, sets the acknowledge bit for transmission, and checks the received acknowledge bit.
• The I²C bus status register (ICSR) consists of various interrupt request and status flags.
• The I²C slave address register (SAR) selects a format and sets the slave address.
• The I²C bus transmit data register (ICDRT) holds data for transmission.
• The I²C bus receive data register (ICDRR) holds the received data.
• The I²C bus shift register (ICDRS) is used to transmit/receive data; this register cannot be accessed by the CPU.
4. Operation

In this sample task, the EEPROM bus state is initialized before data is read from the EEPROM.

4.1 EEPROM Bus Initialization

The EEPROM bus initialization here means to forcibly place the EEPROM's SDA bus into the input state. This initialization should be applied when the EEPROM is unable to receive from the master device because of the EEPROM's SDA bus remaining in the output state as a result of, for example, communication having been halted while the master device is receiving data from the EEPROM.

In the initialization processing, the SCL and SDA bus lines are driven by using the port I/O (general-purpose I/O) function. After generating a start condition, the MCU outputs dummy data (H'FF), receives acknowledge (ACK) from the EEPROM, and then generates a stop condition.

Figure 3 shows the initialization processing, and the software and hardware processing in the figure are described in table 3.

![Figure 3 EEPROM Bus Initialization Processing](image)

**Table 3 Software and Hardware Processing in Bus Initialization**

<table>
<thead>
<tr>
<th>Software Processing</th>
<th>Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processing 1</strong></td>
<td>—</td>
</tr>
<tr>
<td>Set PB2 (SCL) and PB3 (SDA) as port (general input/output) pins.</td>
<td>—</td>
</tr>
<tr>
<td>Set the output level on PB2 and PB3 to high.</td>
<td>—</td>
</tr>
<tr>
<td>Set the output level on PB3 to low to generate a start condition.</td>
<td>—</td>
</tr>
<tr>
<td>Set the output level on PB2 to low.</td>
<td>—</td>
</tr>
<tr>
<td><strong>Processing 2</strong></td>
<td>Output the transfer clock by changing the output level on PB2 with use of a software wait timer.</td>
</tr>
<tr>
<td>Fix the output level on PB3 to high to transmit dummy data (H'FF).</td>
<td>—</td>
</tr>
<tr>
<td><strong>Processing 3</strong></td>
<td>Output the transfer clock (for receiving the acknowledge bit) from PB2.</td>
</tr>
<tr>
<td>Set PB3 as an input pin.</td>
<td>—</td>
</tr>
<tr>
<td>Read the state on the PB3 pin.</td>
<td>—</td>
</tr>
<tr>
<td><strong>Processing 4</strong></td>
<td>Set the output level on PB2 and PB3 to low.</td>
</tr>
<tr>
<td>Set the output level on PB3 to high to generate a stop condition.</td>
<td>—</td>
</tr>
</tbody>
</table>
4.2 Reading Data from EEPROM

In this sample task, the address (H'0000) is specified by Random Reading, and then 10 bytes of data are continuously read by Sequential Read.

Figure 4 shows the contents of communication when data are read from the EEPROM.

After a start condition is generated, the device address word is transmitted with the R/W code set to 0 (specifying write operation). The lower byte of the EEPROM read start address is then transmitted.

A start condition is generated again to start reading. After the second generation of a start condition, the device address word is transmitted with the R/W code set to 1 (specifying read operation). This makes data to be output from the EEPROM sequentially in accordance with the I²C format.

The master outputs the acknowledge bit (ACK) on receiving one byte of data. The EEPROM outputs the next byte on receiving \( \text{ACK} = 0 \). The master outputs \( \text{ACK} = 1 \) on receiving the last byte and generates a stop condition.

![Diagram of I²C communication](image)

Figure 4 Contents of Communication When Data are Read from EEPROM

Notes:  
1. Slave address + upper 3 bits of memory address + R/W code (0): Specifies write operation.  
2. Slave address + upper 3 bits of memory address + R/W code (1): Specifies read operation.
Figure 5 shows the operation when data reading from the EEPROM is started (start condition generated again), and figure 6 shows the ending operation. The software and hardware processing in figures 5 and 6 are described in tables 4 and 5, respectively.

**Figure 5 Operation when Reading from EEPROM Starts**
### Table 4 Software and Hardware Processing when Reading from EEPROM Starts

<table>
<thead>
<tr>
<th>Processing 1</th>
<th>Software Processing</th>
<th>Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set BBSY to 1 and SCP to 0 in the ICCR2 register.</td>
<td>• Generate a start condition (second time).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Set the TDRE bit in the ICSR register to 1.</td>
</tr>
<tr>
<td>Processing 2</td>
<td>Write the device address word data to the ICDRT register (specifying read operation).</td>
<td>• Clear the TDRE bit in the ICSR register to 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Transfer the transmit data in the ICDRT register to the ICDRS register, and then output the data from the SDA pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Set the TDRE bit in the ICSR register to 1.</td>
</tr>
<tr>
<td>Processing 3</td>
<td>• Confirm that TEND in the ICSR register is 1.</td>
<td>• Set the TEND bit in the ICSR register to 1 after outputting the last bit of the transmit data.</td>
</tr>
<tr>
<td></td>
<td>• Check the received acknowledge through the ACKBR bit in the ICIER register.</td>
<td>• Output the clock for reception by dummy-reading the ICDRR register.</td>
</tr>
<tr>
<td></td>
<td>• Clear the TEND bit in the ICSR register to 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Clear the TRS bit in the ICCR1 register to 0 (master reception mode).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Clear the TDRE bit in the ICSR register to 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Dummy-read the ICDRR register (starting reception).</td>
<td></td>
</tr>
<tr>
<td>Processing 4</td>
<td>• Confirm that RDRF in the ICSR register is 1.</td>
<td>• Output the acknowledge bit after receiving the last bit.</td>
</tr>
<tr>
<td></td>
<td>• Read the ICDRR register.</td>
<td>• Transfer the received data in the ICDRS register to the ICDRR register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Set the RDRF bit in the ICSR register to 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clear the RDRF bit in the ICSR register to 0 after the ICDRR register is read.</td>
</tr>
</tbody>
</table>
**Table 5**  Software and Hardware Processing when Reading from EEPROM Ends

<table>
<thead>
<tr>
<th>Processing 1</th>
<th>Software Processing</th>
<th>Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Confirm that RDRF in the ICSR register is 1.</td>
<td>• Transfer the received data in the ICDRS register to the ICDRR register.</td>
</tr>
<tr>
<td></td>
<td>• Set the ACKBT bit in the ICIER register to 1.</td>
<td>• Set the RDRF bit in the ICSR register to 1.</td>
</tr>
<tr>
<td></td>
<td>• Set the RCVD bit in the ICCR1 register to 1.</td>
<td>• Clear the RDRF bit in the ICSR register to 0 after the ICDRR register is read.</td>
</tr>
<tr>
<td></td>
<td>• Read the ICDRR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing 2</th>
<th>Software Processing</th>
<th>Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Confirm that RDRF in the ICSR register is 1.</td>
<td>• Transfer the received data in the ICDRS register to the ICDRR register.</td>
</tr>
<tr>
<td></td>
<td>• Clear the STOP bit in the ICSR register to 0.</td>
<td>• Set the RDRF bit in the ICSR register to 1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing 3</th>
<th>Software Processing</th>
<th>Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Set BBSY and SCP in the ICCR2 register to 0.</td>
<td>• Generate a stop condition.</td>
</tr>
<tr>
<td></td>
<td>• Confirm that STOP in the ICSR register is 1.</td>
<td>• Set the STOP bit in the ICSR register to 1.</td>
</tr>
<tr>
<td></td>
<td>• Read the ICDRR register (last data).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Clear the RCVD bit in the ICCR1 register to 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Set MST and TRS in the ICCR1 register to 0 (slave reception mode).</td>
<td></td>
</tr>
</tbody>
</table>
5. Description of Software

5.1 Modules

Table 6 describes the modules used in the sample application.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main function</td>
<td>main()</td>
<td>Sets the EEPROM read start address, and calls the data read function.</td>
</tr>
<tr>
<td>I²C2 initialization function</td>
<td>init_iic()</td>
<td>Cancels module standby mode, and configures the PFC and I²C2.</td>
</tr>
<tr>
<td>Data read function</td>
<td>read_EEPROM()</td>
<td>Generates a start condition, reads data from the EEPROM, and generates a stop condition.</td>
</tr>
<tr>
<td>EEPROM address transmission function</td>
<td>set_addr_EEPROM()</td>
<td>Generates a start condition and a slave address, and sets the EEPROM address.</td>
</tr>
<tr>
<td>EEPROM bus initialization function</td>
<td>init_EEPROM()</td>
<td>Forcibly initializes the SDA bus that has hanged up due to a communication error or other reason.</td>
</tr>
<tr>
<td>I²C2 reset function</td>
<td>reset_iic()</td>
<td>Resets and halts the I²C2 module.</td>
</tr>
<tr>
<td>Start condition generation function</td>
<td>iic_start()</td>
<td>Generates a start condition through port processing.</td>
</tr>
<tr>
<td>Output level set function</td>
<td>iic_sda_out()</td>
<td>Sets the output level of the bus line.</td>
</tr>
<tr>
<td>Transmit data set function</td>
<td>iic_set()</td>
<td>Outputs the clock and transmit data through port processing.</td>
</tr>
<tr>
<td>1-byte transmission function</td>
<td>iic_bytesend()</td>
<td>Transmits 1 byte of data through port processing.</td>
</tr>
<tr>
<td>Acknowledge reception function</td>
<td>iic_ackck()</td>
<td>Receives the acknowledge from the EEPROM through port processing.</td>
</tr>
<tr>
<td>Stop condition generation function</td>
<td>iic_stop()</td>
<td>Generates a stop condition through port processing.</td>
</tr>
<tr>
<td>Software wait function</td>
<td>wait_timer()</td>
<td>Wait timer using a software counter.</td>
</tr>
</tbody>
</table>
## 5.2 Variables

Table 7 lists the variables used in the sample application.

<table>
<thead>
<tr>
<th>Label Name of Variable</th>
<th>Function</th>
<th>Used In</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_data[0:9]</td>
<td>Array for storing read data</td>
<td>Main function</td>
</tr>
<tr>
<td>address</td>
<td>Read start address of the EEPROM</td>
<td>Main function</td>
</tr>
<tr>
<td>addr</td>
<td>Copy of read start address of the EEPROM</td>
<td>Data read function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EEPROM address transmission function</td>
</tr>
<tr>
<td>*r_data</td>
<td>Pointer variable to the array for storing read data</td>
<td>Data read function</td>
</tr>
<tr>
<td>num</td>
<td>Number of received data bytes</td>
<td>Data read function</td>
</tr>
<tr>
<td>ack</td>
<td>Acknowledge determination flag</td>
<td>Data read function</td>
</tr>
<tr>
<td>count</td>
<td>Counter for continuous data reading</td>
<td>Data read function</td>
</tr>
<tr>
<td>dummy</td>
<td>Variable for dummy reading</td>
<td>Data read function</td>
</tr>
<tr>
<td>data</td>
<td>Information of output levels on the PB2 (SCL) and PB3 (SDA) pins</td>
<td>Output level set function</td>
</tr>
<tr>
<td>scl</td>
<td>Information of output levels on the SCL line</td>
<td>Transmit data set function</td>
</tr>
<tr>
<td>sda</td>
<td>Information of output levels on the SDA line</td>
<td>Transmit data set function</td>
</tr>
<tr>
<td>tx_data</td>
<td>One byte of data for transmission through port processing</td>
<td>1-byte transmission function</td>
</tr>
<tr>
<td>ck_bit</td>
<td>Clock information for data transmission through port processing</td>
<td>1-byte transmission function</td>
</tr>
<tr>
<td>bit_data</td>
<td>One bit of data for transmission through port processing</td>
<td>1-byte transmission function</td>
</tr>
<tr>
<td>ack_flag</td>
<td>Acknowledge bit received through port processing</td>
<td>Acknowledge reception function</td>
</tr>
<tr>
<td>wait_cnt</td>
<td>Count value for software timer</td>
<td>Software wait function</td>
</tr>
<tr>
<td>cnt</td>
<td>Variable for software counter</td>
<td>Software wait function</td>
</tr>
</tbody>
</table>
5.3 Register Settings

The register settings used in the sample application are described below. Note that the set values are specifically used in the sample task and that they are different from the initial values.

5.3.1 Settings for the Clock Pulse Generator (CPG)

- Frequency Control Register (FRQCR)
  - Set value: H'0241
  - Function: Specifies the frequency division ratio.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>—</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>14 to 12</td>
<td>IFC[2] to IFC[0]</td>
<td>000</td>
<td>Frequency division ratio for internal clock (Iφ) 000: ×1 (80 MHz when input clock is 10 MHz)</td>
</tr>
<tr>
<td>11 to 9</td>
<td>BFC[2] to BFC[0]</td>
<td>001</td>
<td>Frequency division ratio for bus clock (Bφ) 001: ×1/2 (40 MHz when input clock is 10 MHz)</td>
</tr>
<tr>
<td>8 to 6</td>
<td>PFC[2] to PFC[0]</td>
<td>001</td>
<td>Frequency division ratio for peripheral clock (Pφ) 001: ×1/2 (40 MHz when input clock is 10 MHz)</td>
</tr>
<tr>
<td>5 to 3</td>
<td>MIFC[2] to MIFC[0]</td>
<td>000</td>
<td>Frequency division ratio for MTU2S clock (MIφ) 000: ×1 (80 MHz when input clock is 10 MHz)</td>
</tr>
<tr>
<td>2 to 0</td>
<td>MPFC[2] to MPFC[0]</td>
<td>001</td>
<td>Frequency division ratio for MTU2 clock (MPφ) 001: ×1/2 (40 MHz when input clock is 10 MHz)</td>
</tr>
</tbody>
</table>

5.3.2 Settings for Power-Down Modes

- Standby Control Register 3 (STBCR3)
  - Set value: H'7F
  - Function: Controls the operation of individual modules in power-down modes.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MSTP15</td>
<td>0</td>
<td>0: The I²C2 runs.</td>
</tr>
<tr>
<td>6</td>
<td>MSTP14</td>
<td>1</td>
<td>1: Stops supply of the clock signal to the SCIF.</td>
</tr>
<tr>
<td>5</td>
<td>MSTP13</td>
<td>1</td>
<td>1: Stops supply of the clock signal to SCI_2.</td>
</tr>
<tr>
<td>4</td>
<td>MSTP12</td>
<td>1</td>
<td>1: Stops supply of the clock signal to SCI_1.</td>
</tr>
<tr>
<td>3</td>
<td>MSTP11</td>
<td>1</td>
<td>1: Stops supply of the clock signal to SCI_0.</td>
</tr>
<tr>
<td>2</td>
<td>MSTP10</td>
<td>1</td>
<td>1: Stops supply of the clock signal to the SSU.</td>
</tr>
<tr>
<td>1 and 0</td>
<td>—</td>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
5.3.3 Settings for I^2C Bus Interface 2 (I^2C2)

- I^2C Bus Control Register 1 (ICCR1)
  - Set value: H’B5
  - Function: Selects the operating mode and transfer clock of the I^2C2.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ICE</td>
<td>1</td>
<td>1: Enables transfer operation (the SCL/SDA buses are driven).</td>
</tr>
<tr>
<td>6</td>
<td>RCVD</td>
<td>0</td>
<td>0: Performs the next reception when ICDRR is read.</td>
</tr>
<tr>
<td>5</td>
<td>MST</td>
<td>1</td>
<td>MST: Master/Slave Select</td>
</tr>
<tr>
<td>4</td>
<td>TRS</td>
<td>1*</td>
<td>TRS: Transmit/Receive Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10: Master reception mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11: Master transmission mode.</td>
</tr>
<tr>
<td>3 to 0</td>
<td>CKS[3] to CKS[0]</td>
<td>0101</td>
<td>Transfer Clock Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101:</td>
<td>Sets the transfer rate to 400 kHz (P_φ = 40 MHz).</td>
</tr>
</tbody>
</table>

Note: * Changed to 0 (reception) after an address is transmitted to the EEPROM.

- I^2C Bus Control Register 2 (ICCR2)
  - Setting value: H’7D
  - Function: Generates start and stop conditions, drives the SDA pin, monitors the SCL pin, and controls resetting of the I^2C bus control circuitry.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>BBSY</td>
<td>0*</td>
<td>Indicates whether the I^2C bus is occupied or released, and generates start and stop conditions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: I^2C bus is released.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SCP</td>
<td>1*</td>
<td>Controls generation of start and stop conditions.</td>
</tr>
<tr>
<td>5</td>
<td>SDAO</td>
<td>1</td>
<td>1: The output on the SDA pin is high (read-only bit).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changes the output on the SDA pin to Hi-Z (when written).</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SDAOP</td>
<td>1</td>
<td>SDAO Write Protect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write 0 to this bit when writing to SDAO.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SCLO</td>
<td>1</td>
<td>1: The output on the SCL pin is high (read-only bit).</td>
</tr>
<tr>
<td>2</td>
<td>—</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>IICRST</td>
<td>0</td>
<td>I^2C Control Unit Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resets the control circuitry of the I^2C when 1 is written.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>—</td>
<td>1</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Note: * To generate a start condition, write b’10 to BBSY and SCP.
      To generate a stop condition, write b’00 to BBSY and SCP.
• **I²C Bus Mode Register (ICMR)**  
  — Setting value: H'38  
  — Function: Selects MSB- or LSB-first transfer, controls the wait states in master mode, and selects the number of bits to be transferred.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MLS</td>
<td>0*1</td>
<td>0: MSB-first</td>
</tr>
<tr>
<td>6</td>
<td>WAIT</td>
<td>0*2</td>
<td>0: Data and acknowledge bit are transferred continuously.</td>
</tr>
<tr>
<td>5 and 4</td>
<td>—</td>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 3   | BCWP     | 1     | BC Write Protect  
  Write 0 to this bit when writing to the BC[2] to BC[0] bits. |
| 2 to 0 | BC[2] to BC[0] | 000 000 | 9 bits (data and acknowledge bit) are transferred. |

Notes: 1. This bit must be set to 0 (MSB-first) when the I²C bus format is used.  
  2. This bit must be set to 0.

• **I²C Bus Interrupt Enable Register (ICIER)**  
  — Setting value: H'04  
  — Function: Enables or disables interrupt sources and controls the acknowledge bit.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TIE</td>
<td>0</td>
<td>0: Disables the transmit data empty interrupt request (IITXI).</td>
</tr>
<tr>
<td>6</td>
<td>TEIE</td>
<td>0</td>
<td>0: Disables the transmit end interrupt request (IITEI).</td>
</tr>
<tr>
<td>5</td>
<td>RIE</td>
<td>0</td>
<td>0: Disables the receive data full interrupt request (IIRXI).</td>
</tr>
<tr>
<td>4</td>
<td>NAKIE</td>
<td>0</td>
<td>0: Disables the NACK receive interrupt request (IINAKI).</td>
</tr>
<tr>
<td>3</td>
<td>STIE</td>
<td>0</td>
<td>0: Disables the stop condition detection interrupt request (IISTPI).</td>
</tr>
<tr>
<td>2</td>
<td>ACKE</td>
<td>1</td>
<td>1: Halts data transfer when the received acknowledge bit is 1.</td>
</tr>
<tr>
<td>1</td>
<td>ACKBR</td>
<td>0</td>
<td>Holds the acknowledge data in transmission mode (read-only bit).</td>
</tr>
<tr>
<td>0</td>
<td>ACKBT</td>
<td>0</td>
<td>0: Transmits ACK = 0 in reception mode.</td>
</tr>
</tbody>
</table>

• **I²C Bus Status Register (ICSR)**  
  — Setting value: H'00  
  — Function: Provides various interrupt request and status flags.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TDRE</td>
<td>0</td>
<td>Transmit data register empty flag</td>
</tr>
<tr>
<td>6</td>
<td>TEND</td>
<td>0</td>
<td>Transmit end flag</td>
</tr>
<tr>
<td>5</td>
<td>RDRF</td>
<td>0</td>
<td>Receive data register full flag</td>
</tr>
<tr>
<td>4</td>
<td>NACKF</td>
<td>0</td>
<td>No acknowledge detection flag</td>
</tr>
<tr>
<td>3</td>
<td>STOP</td>
<td>0</td>
<td>Stop condition detection flag</td>
</tr>
<tr>
<td>2</td>
<td>AL/OVE</td>
<td>0</td>
<td>Arbitration lost flag/overrun error flag</td>
</tr>
<tr>
<td>1</td>
<td>AAS</td>
<td>0</td>
<td>Slave address recognition flag</td>
</tr>
<tr>
<td>0</td>
<td>ADZ</td>
<td>0</td>
<td>General call address recognition flag</td>
</tr>
</tbody>
</table>
• **I²C Bus Transmit Data Register (ICDRT)**
  — Setting value: H'FF (initial value)
  — Function: Holds data for transmission.

• **I²C Bus Receive Data Register (ICDRR)**
  — Setting value: H'FF
  — Function: Holds the received data. (Read only)

5.3.4 **Settings for the Pin Function Controller (PFC)**

• **Port B Control Register L1 (PBCRL1)**
  — Setting value: H'4400
  — Function: Selects the functions of the multiplexed pins on port B (PB3 to PB0).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>—</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>14 to 12</td>
<td>PB3MD[2] to PB3MD[0]</td>
<td>100*</td>
<td>PB3 mode 100: SDA I/O (I²C2)</td>
</tr>
<tr>
<td>11</td>
<td>—</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>10 to 8</td>
<td>PB2MD[2] to PB2MD[0]</td>
<td>100*</td>
<td>PB2 mode 100: SCL I/O (I²C2)</td>
</tr>
<tr>
<td>7</td>
<td>—</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>6 to 4</td>
<td>PB1MD[2] to PB1MD[0]</td>
<td>000</td>
<td>PB1 mode 000: PB1 I/O (port)</td>
</tr>
<tr>
<td>3</td>
<td>—</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>2 to 0</td>
<td>PB0MD[2] to PB0MD[0]</td>
<td>000</td>
<td>PB0 mode 000: PB1 I/O (port)</td>
</tr>
</tbody>
</table>

**Note:** *In EEPROM bus initialization, these bits are set to 000 to use the multiplexed pins as general I/O (port) pins.*
- Port B IO Register (PBIORL)
  - Set value: H'000C
  - Function: Selects the signal directions on the port B pins.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 to 10</td>
<td>All 0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PB9IOR</td>
<td>0</td>
<td>0: PB9 is input.</td>
</tr>
<tr>
<td>8</td>
<td>PB8IOR</td>
<td>0</td>
<td>0: PB8 is input.</td>
</tr>
<tr>
<td>7</td>
<td>PB7IOR</td>
<td>0</td>
<td>0: PB7 is input.</td>
</tr>
<tr>
<td>6</td>
<td>PB6IOR</td>
<td>0</td>
<td>0: PB6 is input.</td>
</tr>
<tr>
<td>5</td>
<td>PB5IOR</td>
<td>0</td>
<td>0: PB5 is input.</td>
</tr>
<tr>
<td>4</td>
<td>PB4IOR</td>
<td>0</td>
<td>0: PB4 is input.</td>
</tr>
<tr>
<td>3</td>
<td>PB3IOR</td>
<td>1</td>
<td>1: PB3 is output (ignored when SDA is selected by PBCRL1).</td>
</tr>
<tr>
<td>2</td>
<td>PB2IOR</td>
<td>1</td>
<td>1: PB2 is output (ignored when SCL is selected by PBCRL1).</td>
</tr>
<tr>
<td>1</td>
<td>PB1IOR</td>
<td>0</td>
<td>0: PB1 is input.</td>
</tr>
<tr>
<td>0</td>
<td>PB0IOR</td>
<td>0</td>
<td>0: PB0 is input.</td>
</tr>
</tbody>
</table>

Note: This setting is used in EEPROM bus initialization.

5.3.5 Settings for I/O Ports
- Port B Data Register L (PBDRL)
  - Set value: H'000C
  - Function: Holds the data for port B.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 to 10</td>
<td>All 0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PB9DR</td>
<td>0</td>
<td>0: Low level on PB9</td>
</tr>
<tr>
<td>8</td>
<td>PB8DR</td>
<td>0</td>
<td>0: Low level on PB8</td>
</tr>
<tr>
<td>7</td>
<td>PB7DR</td>
<td>0</td>
<td>0: Low level on PB7</td>
</tr>
<tr>
<td>6</td>
<td>PB6DR</td>
<td>0</td>
<td>0: Low level on PB6</td>
</tr>
<tr>
<td>5</td>
<td>PB5DR</td>
<td>0</td>
<td>0: Low level on PB5</td>
</tr>
<tr>
<td>4</td>
<td>PB4DR</td>
<td>0</td>
<td>0: Low level on PB4</td>
</tr>
<tr>
<td>3</td>
<td>PB3DR</td>
<td>1</td>
<td>1: High level on PB3 (Produces a transmit data signal in EEPROM bus initialization.)</td>
</tr>
<tr>
<td>2</td>
<td>PB2DR</td>
<td>1</td>
<td>1: High level on PB2 (Produces a clock signal in EEPROM bus initialization.)</td>
</tr>
<tr>
<td>1</td>
<td>PB1DR</td>
<td>0</td>
<td>0: Low level on PB1</td>
</tr>
<tr>
<td>0</td>
<td>PB0DR</td>
<td>0</td>
<td>0: Low level on PB0</td>
</tr>
</tbody>
</table>

Note: This setting is used in EEPROM bus initialization.
6. Flowcharts

6.1 Main Function

- **Main()**
  - Set the frequency control register (FRQCR)
  - Initialize the EEPROM bus
  - Set the EEPROM address
  - Initialize the I²C2 module
  - Read data from the EEPROM

**< Operating frequency setting >**
Internal clock: 80 MHz, bus clock: 40 MHz, peripheral clock: 40 MHz, MTU2S clock: 80 MHz, MTU2 clock: 40 MHz

- Initialize the EEPROM bus
  - Call the EEPROM bus initialization function.

- Set the EEPROM address
  - Set the EEPROM read start address to H'0000.

- Initialize the I²C2 module
  - Call the I²C2 initialization function.

- Read data from the EEPROM
  - Call the EEPROM data read function.
6.2 EEPROM Bus State Initialization Function

init_EEPROM()

Set the standby control register 3 (STBCR3)
- Cancel \( \text{I}^2\text{C} \) module standby mode.

Set the port B control register L1 (PBCRL1)
- Select PB2 and PB3 (general I/O) pin functions.

Set the port B data register L (PBDRL)
- Set the level of PB2 and PB3 to high.

Set the port B I/O register (PBIORL)
- Set PB2 and PB3 as outputs.

Reset the \( \text{I}^2\text{C} \) module
- Call the \( \text{I}^2\text{C} \) reset function.

Generate a start condition
- Call the start condition generating function.

Software wait
- Call the software wait function.

Transmit dummy data
- Call the 1-byte transmission function.

Check the acknowledge bit
- Call the acknowledge reception function.

Generate a stop condition
- Call the stop condition generating function.

Software wait
- Call the software wait function.

Return
6.2.1 \textit{I}^2\textit{C}2 Reset Function

- \texttt{reset\_iic()}
  - Reset the control circuitry of the \textit{I}^2\textit{C}2 module. Reset the control circuitry of the \textit{I}^2\textit{C}2.
  - Halt the \textit{I}^2\textit{C}2 module. Release the \textit{I}^2\textit{C} bus.

6.2.2 Start Condition Generation Function

- \texttt{iic\_start()}
  - Set the output level on the PB2 (SCL) and PB3 (SDA) pins.
    - Control the output level on the PB2 (SCL) and PB3 (SDA) pins.
      - Call the output level set function. (Drive PB2 (SCL) and PB3 (SDA) high.)
      - Call the transmit data set function. (Drive PB2 (SCL) and PB3 (SDA) high.)
      - Call the transmit data set function. (Drive PB2 (SCL) high and PB3 (SDA) low.)
      - Call the transmit data set function. (Drive PB2 (SCL) and PB3 (SDA) low.)

6.2.3 Output Level Set Function

- \texttt{iic\_sda\_out()}
  - Argument:\ data
    - Set the output level of the PB2 (SCL) and PB3 (SDA) pins.
    - Set the PB2 (SCL) and PB3 (SDA) pins as outputs.
      - Set the output levels according to the information (data) from the routine that called this function.
### 6.2.4 Transmit Data Set Function

**Arguments:** `scl`, `sda`

- **Set the output level information of the PB2 (SCL) and PB3 (SDA) pins**
- **Set the default output level data.** (Low level for PB2 (SCL) and PB3 (SDA))
- **Check SDA data information**
  - `sda == 1`?
    - Yes: **Set the output level of PB3 (SDA) to high**
    - No: **Check SCL data information**
      - `scl == 1`?
        - Yes: **Set the output level of PB2 (SCL) to high**
          - **Reflect the resulting output level data on the port pins.**
        - No: **Set the output level of PB3 (SDA) according to the information (sda) from the routine that called this function.**
      - No: **Set the output level of PB3 (SDA) according to the information (sda) from the routine that called this function.**
  - No: **Set the output level of PB3 (SDA) to high**
- **Set the port B data register L (PBDRL)**
- **Return**
### 6.2.5 1-Byte Transmission Function

**Function:** `iic_bytesend()`

Argument: `tx_data`

- Set the output level on the PB2 (SCL) and PB3 (SDA) pins

  ```
  Set the output level on the PB2 (SCL) and PB3 (SDA) pins
  ```

- Call the output level set function.
  (Drive PB2 (SCL) and PB3 (SDA) low.)

- Initialize the transfer counter.

- Repeat until transmission of 1 byte of data is complete.

- Create 1-bit transmit data information according to the information (tx_data) from the routine that called this function.

- Set the output level of SDA based on the created transmit data information.

- Call the transmit data set function.
  (Drive PB2 (SCL) low and PB3 (SDA) to the bit_data level.)

- Call the software wait function.

- Set the transmit data to 1

- Control the output level on the PB2 (SCL) and PB3 (SDA) pins

- Call the transmit data set function.
  (Drive PB2 (SCL) high and PB3 (SDA) to the bit_data level.)

- Call the software wait function.

- Set the transmit data to 0

- Control the output level on the PB2 (SCL) and PB3 (SDA) pins

- Call the transmit data set function.
  (Drive PB2 (SCL) low and PB3 (SDA) to the bit_data level.)

- Call the software wait function.

- Check the transmit bit count `ckbit > 0`?

- Yes

  - Create transmit data (1 bit) information

  ```
  Create transmit data (1 bit) information
  ```

  - Call the output level set function.
    (Drive PB2 (SCL) and PB3 (SDA) low.)

  - Initialize the transfer counter.

  - Repeat until transmission of 1 byte of data is complete.

  - Call the transmit data set function.
    (Drive PB2 (SCL) low and PB3 (SDA) to the bit_data level.)

  - Call the software wait function.

- No

  - Check the transmit data information `bit_data != 0`?

    - Yes

      - Set the transmit data to 1

      - Control the output level on the PB2 (SCL) and PB3 (SDA) pins

      - Call the transmit data set function.
        (Drive PB2 (SCL) high and PB3 (SDA) to the bit_data level.)

      - Call the software wait function.

    - No

      - Set the transmit data to 0

      - Control the output level on the PB2 (SCL) and PB3 (SDA) pins

      - Call the transmit data set function.
        (Drive PB2 (SCL) low and PB3 (SDA) to the bit_data level.)

      - Call the software wait function.

- Return
### 6.2.6 Acknowledge Reception Function

- **iic_ackck()**
  - Set PB3 (SDA) as an input pin
  - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
    - Wait for a half clock cycle
    - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
    - Wait for a half clock cycle
  - Check the ACK bit
    - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
    - Wait for a half clock cycle
  - Return (ack_flag)

**Description:**
- Call the transmit data set function. (Drive PB2 (SCL) low; PB3 (SDA) is input.)
- Call the software wait function.
- Call the transmit data set function. (Drive PB2 (SCL) high; PB3 (SDA) is input.)
- Call the software wait function.
- Read the state on the PB3 (SDA) pin.
- Call the transmit data set function. (Drive PB2 (SCL) low; PB3 (SDA) is input.)
- Call the software wait function.

### 6.2.7 Stop Condition Generation Function

- **iic_stop()**
  - Set the output level on the PB2 (SCL) and PB3 (SDA) pins
  - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
  - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
  - Control the output level on the PB2 (SCL) and PB3 (SDA) pins
  - Return

**Description:**
- Call the output level set function. (Drive PB2 (SCL) and PB3 (SDA) low.)
- Call the transmit data set function. (Drive PB2 (SCL) and PB3 (SDA) low.)
- Call the transmit data set function. (Drive PB2 (SCL) high and PB3 (SDA) low.)
- Call the transmit data set function. (Drive PB2 (SCL) and PB3 (SDA) high.)
### 6.2.8 Software Wait Function

**wait_timer()**

- **Argument:** `wait_cnt`

- **cnt = 0**
  - **Initialize the timer counter.**

- **Check the timer counter**
  - `cnt < wait_cnt?`

  - **Yes:**
    - **cnt ++**
    - **Increment the timer counter.**

  - **No:**
    - **Software timer**

- **Return**
6.3  \textit{I}^2\textit{C}2 Initialization Function

\begin{itemize}
  \item \texttt{init\_iic()}
    \begin{itemize}
      \item Set the standby control register 3 (STBCR3)
        \begin{itemize}
          \item Cancel \textit{I}^2\textit{C}2 module standby mode.
        \end{itemize}
      \item Set the port B control register L1 (PBCRL1)
        \begin{itemize}
          \item Set the \textit{I}^2\textit{C} bus control register 1 (ICCR1)
        \end{itemize}
      \item Select determination by the acknowledge bit
        \begin{itemize}
          \item Transmission is halted when ACK = 1 is received.
        \end{itemize}
    \end{itemize}
  \item Return
    \begin{itemize}
      \item <Setting contents>
        \begin{itemize}
          \item Enables \textit{I}^2\textit{C}2 for transfer operation (bus is driven).
          \item Allows the next reception when ICDRR is read.
          \item Master transmission mode
          \item Transfer clock: 400 kHz
        \end{itemize}
    \end{itemize}
\end{itemize}
6.4 Data Read Function

read_EEPROM()

Arguments: addr, *r_data, num

Set the device code and read start address

Call the EEPROM address transmission function.

Check ACK from EEPROM

ack == 1?

Yes

Generate a start condition

Generate a start condition again (BBSY = 1, SCP = 0).

Set transmit data in ICDRT

Set the device address word data.

Check the TEND flag

TEND == 0?

Yes

Confirm that 1-byte data has been transmitted.

No

Check ACK from EEPROM

ACKBR != 0?

Yes

Check the ACKBR bit in the ICIER register.

No

ack = 0

Assign 0 to the ack flag variable when no ACK is detected (high level is received as ACK).

1
1

3

No

Check ACK from EEPROM

ack == 1?

Yes

TEND = 0

Change the operating mode to master reception mode.

TRS = 0

Set the ACK value to be transmitted when data is received

TDRE = 0

Transmit ACK = 0 when data is received successfully.

No

Check the number of received data bytes

num > 1?

Yes

Dummy read

Start reception by dummy-reading ICDRR.

No

Check the RDRF flag

RDRF == 0?

Yes

count = 2

Set the initial value in the transfer counter.

No

Check the receive data counter

count < num?

Yes

Read ICDRR

Store the received data in the array.

r_data ++

Increment the pointer for storing received data.

No

Check the RDRF flag

RDRF == 0?

Yes

No

1

2
Dummy read

Clear the STOP flag to 0 in advance to detect a stop condition.

Generate a stop condition (BBSY = 0, SCP = 0).

Check the STOP flag
STOP == 0?

Yes

Confirm that a stop condition has been generated.

No

Read ICDRR

Clear the RCVD bit to 0

Change the operating mode

Return

Check the number of received data bytes num == 1?

Yes

When num == 1:
Start reception by dummy-reading ICDRR.
When num != 1:
Read the (num - 1)th data.
Increment the pointer for storing received data.

No

Check the RDRF flag
RDRF == 0?

Yes

Confirm that 1 byte of data has been received.

No

Read ICDRR

r_data ++

Check the number of received data bytes
num == 1?

ACKBT = 1

RCVD = 1

Transmit ACK = 1 to EEPROM on receiving the next data.

Disable reception after the next data is received.
### 6.4.1 EEPROM Address Transmission Function

#### set_addr_EEPROM()

**Return value:**
- 1: Setting completed
- 0: Setting error

**Argument:** addr

- **Check the I²C bus state**
  - Bus released state when BBSY = 0

- **Generate a start condition**
  - Generate a start condition (BBSY = 1, SCP = 0).

- **Set transmit data in ICDRT**
  - Set the device address word data.

- **Check the TEND flag (TEND == 0)**
  - Confirm that 1 byte of data has been transmitted.

  - **Check ACK from EEPROM (ACKBR != 0?)**
    - **Yes**
      - Check the ACKBR bit in the ICIER register. If no ACK is detected, return with a return value of 0.
    - **No**

  - **Return (0)**

- **Set transmit data in ICDRT**
  - Set the EEPROM address (the lower 1 byte).

- **Check the TDRE flag (TDRE == 0?)**
  - Confirm that 1 byte of data has been transmitted.

  - **Check ACK from EEPROM (ACKBR != 0?)**
    - **Yes**
      - Check the ACKBR bit in the ICIER register. If no ACK is detected, return with a return value of 0.
    - **No**

  - **Return (0)**

- **Return (1)**
7. Website

- Website of Renesas Technology Corp.
  http://www.renesas.com/
## Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep.14.05</td>
<td></td>
<td></td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
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.