SH7262/SH7264 Group

I^2C Bus Interface 3

Reception in Single-Master Mode (Read from EEPROM)

R01AN0061EJ0102
Rev. 1.02
Feb. 10, 2012

Summary

This application note describes an example to read data from EEPROM using the SH7262/SH7264 Microcomputers (MCUs) I^2C Bus Interface 3 (IIC3) reception in single-master mode.

Target Device

SH7264 MCU (In this document, SH7262/SH7264 are described as "SH7264".)

Contents

1. Introduction

1.1 Specifications

- Specifies the SH7264 MCU as the master device, and EEPROM as the slave device to read data from EEPROM
- Transfer rate is set to 391 kHz.

Note: Set the transfer rate to satisfy the EEPROM specifications.

1.2 Module Used

- I²C Bus Interface 3 (IIC3)

1.3 Applicable Conditions

MCU	SH7262/SH7264
Operating Frequency	Internal clock:144 MHz Bus clock: 72 MHz Peripheral clock: 36 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.03 Release 00
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo)

1.4 Related Application Notes

For more information, refer to the following application notes:

- SH7262/SH7264 Group Example of Initialization
- SH7262/SH7264 Group I²C Bus Interface 3 Transmission in Single-Master Mode (Write in EEPROM)

1.5 About Active-low Pins (Signals)

The symbol "#" suffixed to the pin (or signal) names indicates that the pins (or signals) are active-low.

2. Applications

SH7264 MCU (master device) receives data from an EEPROM (slave device) using IIC3 in the sample program.

2.1 IIC3 Operation

IIC3 is compliant to the I²C Bus (Inter IC Bus) interface specifications invented by Philips and supports subsets, however, the configuration of registers to control the I²C bus partly differs from that of Philips.

SH7264 IIC3 has the following features:

- Format options selectable, I²C bus format or clocked synchronous serial format
- Transmits or receives data continuously
  As the Shift register, Transmit data register and Receive data register are separate registers, IIC3 can transmit and receive data continuously.

Table 1 lists the features of two options of formats. Figure 1 shows the IIC3 block diagram. For details on IIC3, refer to I²C Bus Interface 3 chapter in the SH7262 Group, SH7264 Group Hardware User's Manual.

Table 1 Format Options

Format Name	Description
I²C Bus Format	• Automatically generates the START and STOP conditions in master mode • An output level of an ACK can be selected when receiving data • Automatically loads an ACK bit when transmitting data • Includes the bit synchronization/wait function IIC3 monitors the SCL status per bit in master mode to synchronize automatically. When it is not ready for transfer, it specifies the SCL to low level to wait • Six interrupt sources (1) Transmit data empty (including when slave address match) (2) Transmit end (3) Receive data full (including when slave address match) (4) Arbitration lost (5) NACK detection (6) Stop condition detection • Using the transmit data empty interrupt and the receive data full interrupt to activate the Direct Memory Access Controller (DMAC) and transfer data • Bus can be driven directly SCL and SDA pins are driven by an NMOS open-drain output when selecting the bus drive function
Clocked Synchronous Serial Format	• Four interrupt sources (1) Transmit data empty (2) Transmit end (3) Receive data full (4) Overrun error • Using the transmit data empty interrupt and the receive data full interrupt to activate the Direct Memory Access Controller (DMAC) and transfer data

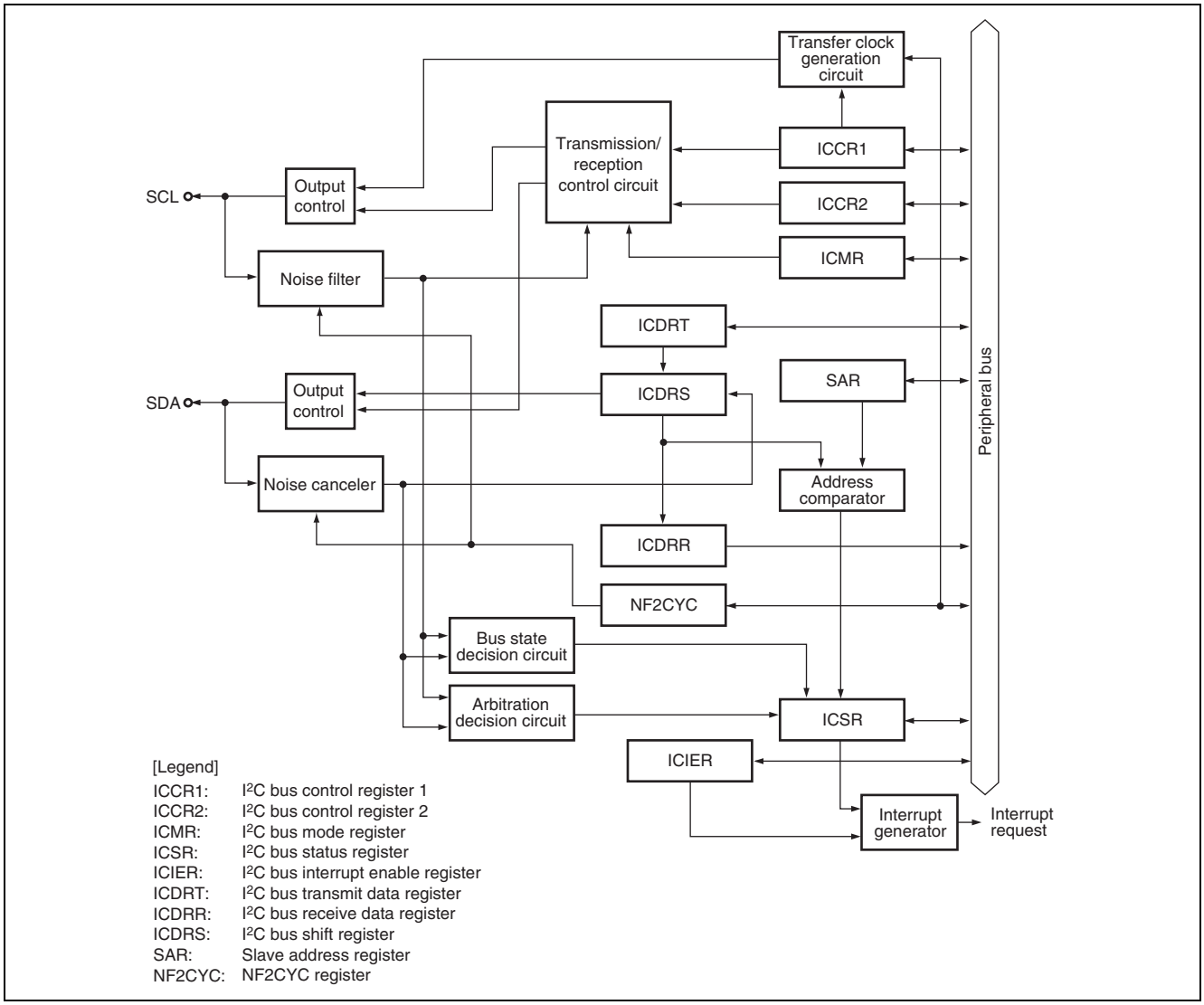

Figure 1 IIC3 Block Diagram

2.2 IIC3 Setting Procedure

This section describes how to set up IIC3. Make sure to specify the transfer rate to satisfy EEPROM electrical characteristics. Pφ/92 is specified in the sample program. Figure 2 shows the flow chart for configuring IIC3. For more information about the register setting, refer to the SH7262 Group, SH7264 Group Hardware User's Manual.

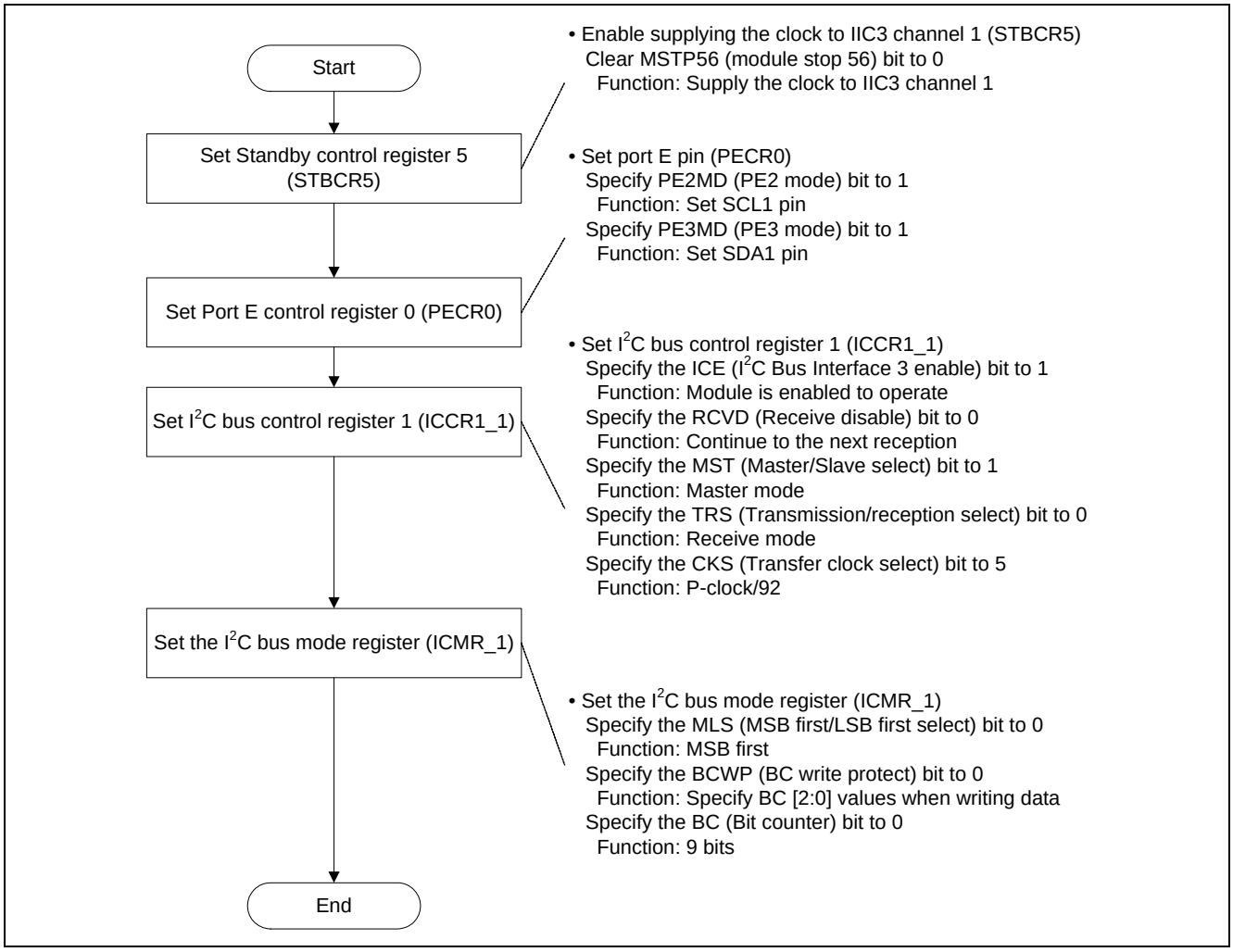

Figure 2 IIC3 Configuration Flow Chart

2.3 Sample Program Operation

The sample program specifies IIC3 in master receive mode to read 10 bytes of data sequentially (sequential read). This sample program uses "B'1010" as the device code and "B'000" as the device address. For more information about device codes and the device address, refer to the EEPROM datasheet provided by the manufacturer.

The memory address indicates the read start address on EEPROM and the address is incremented at every time the MCU reads data from EEPROM. Figure 3 shows the IIC3 sequential read operation. Figure 4 shows the sample program operating environment.

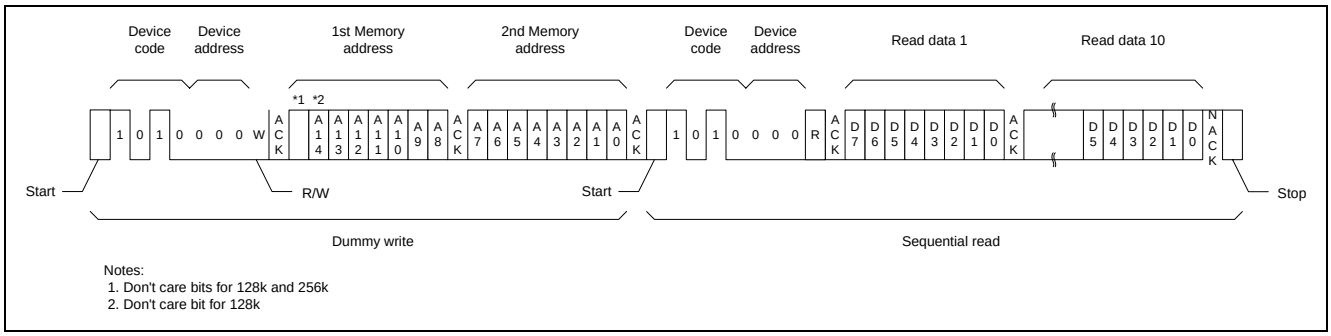

Figure 3 Sequential Read Operation

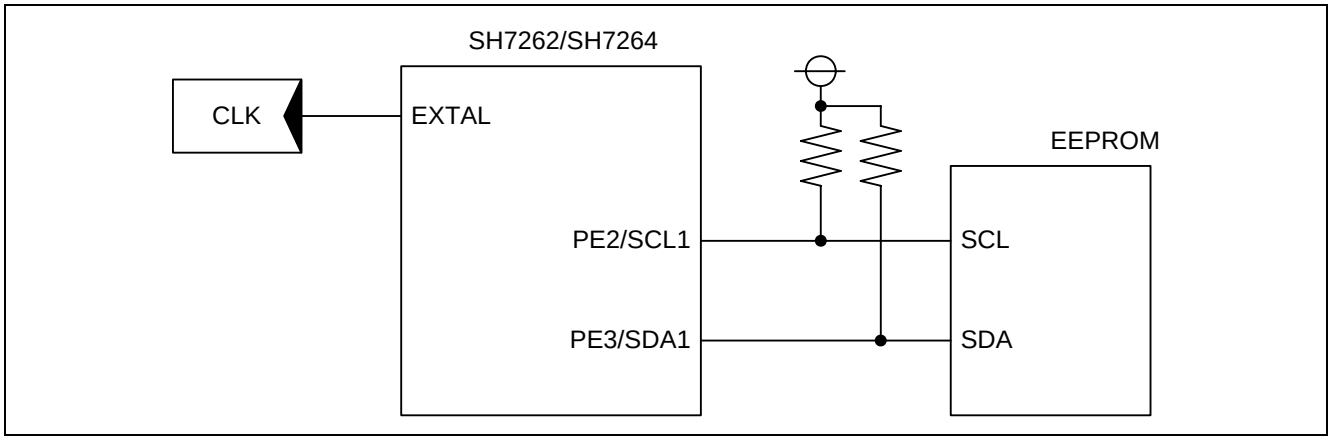

Figure 4 Sample Program Operating Environment

2.4 Sample Program Procedure

Table 2 lists the register settings in the sample program. Table 3 lists the macro definitions used in the sample program. Figure 5 to Figure 10 show flow charts of the sample program.

Table 2 Register Settings (Default)

Register Name	Address	Setting	Description
Standby control register 5 (STBCR5)	H'FFFE 0410	H'00	MSTP56 = "0": IIC3 channel 1 is operating
I²C bus control register 1 (ICCR1_1)	H'FFFE E400	H'A5	ICE = "1": SCL/SDA pins are driven by bus RCVD = "0": Continues the next reception MST = "1", TRS = "0": Master receive mode CKS = "B'0101": Transfer rate is Pφ/92
I²C bus mode register (ICMR_1)	H'FFFE E402	H'30	MLS = "0": MSB first BCWP = "0": Sets BC value when writing BC = "B'000": 9 bits

Table 3 Macro Definitions

Macro Definitions	Setting	Function
EEPROM_MEM_ADDR	H'0000	EEPROM start address
DEVICE_CODE	H'A0	Device code
DEVICE_ADDR	H'00	Device address
IIC_DATA_WR	H'00	Write code
IIC_DATA_RD	H'01	Read code
IIC3_DATA	10	Data transfer size

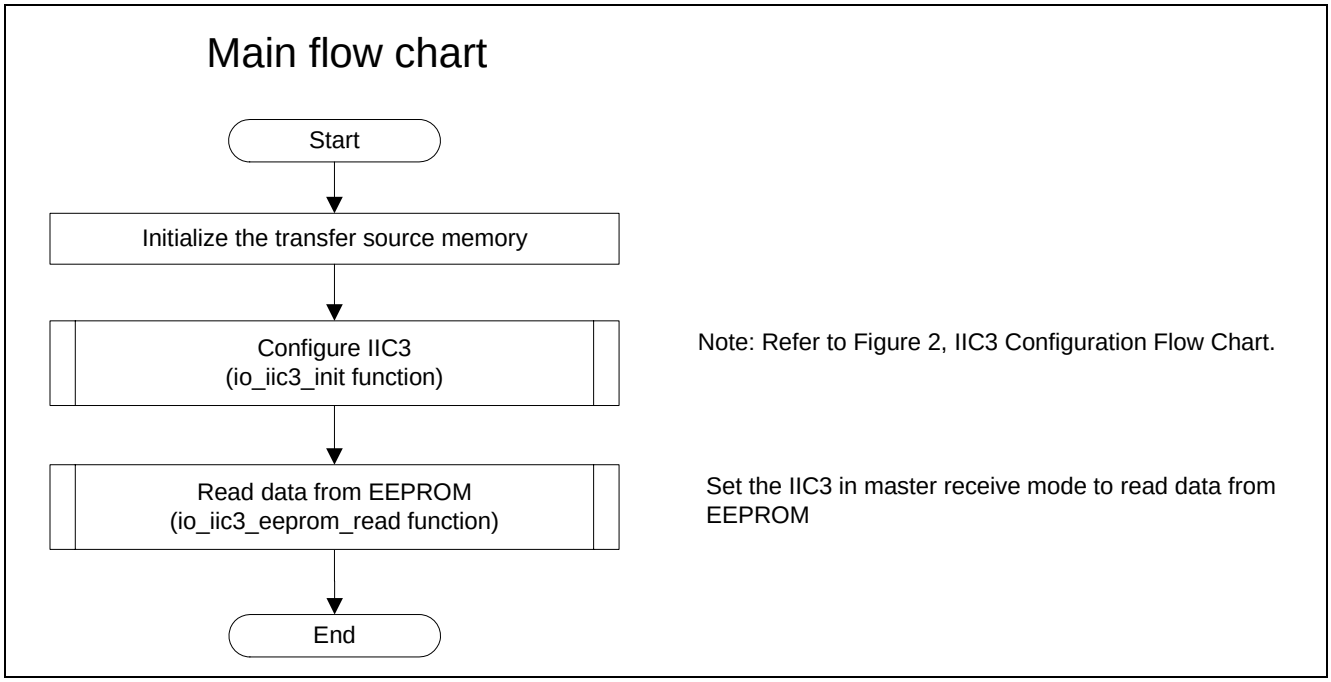

Figure 5 Sample Program Flow Chart (1/6)

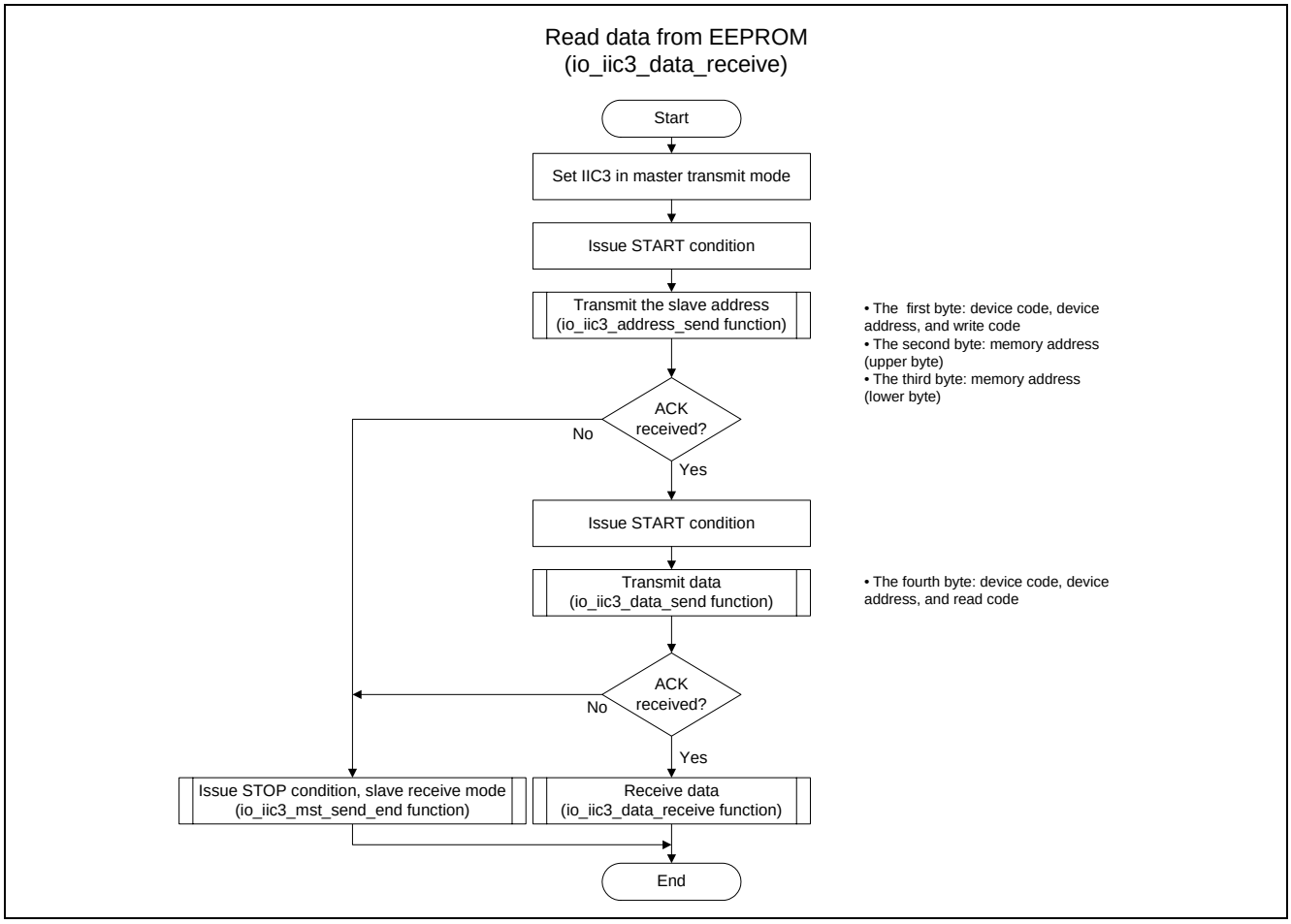

Figure 6 Sample Program Flow Chart (2/6)

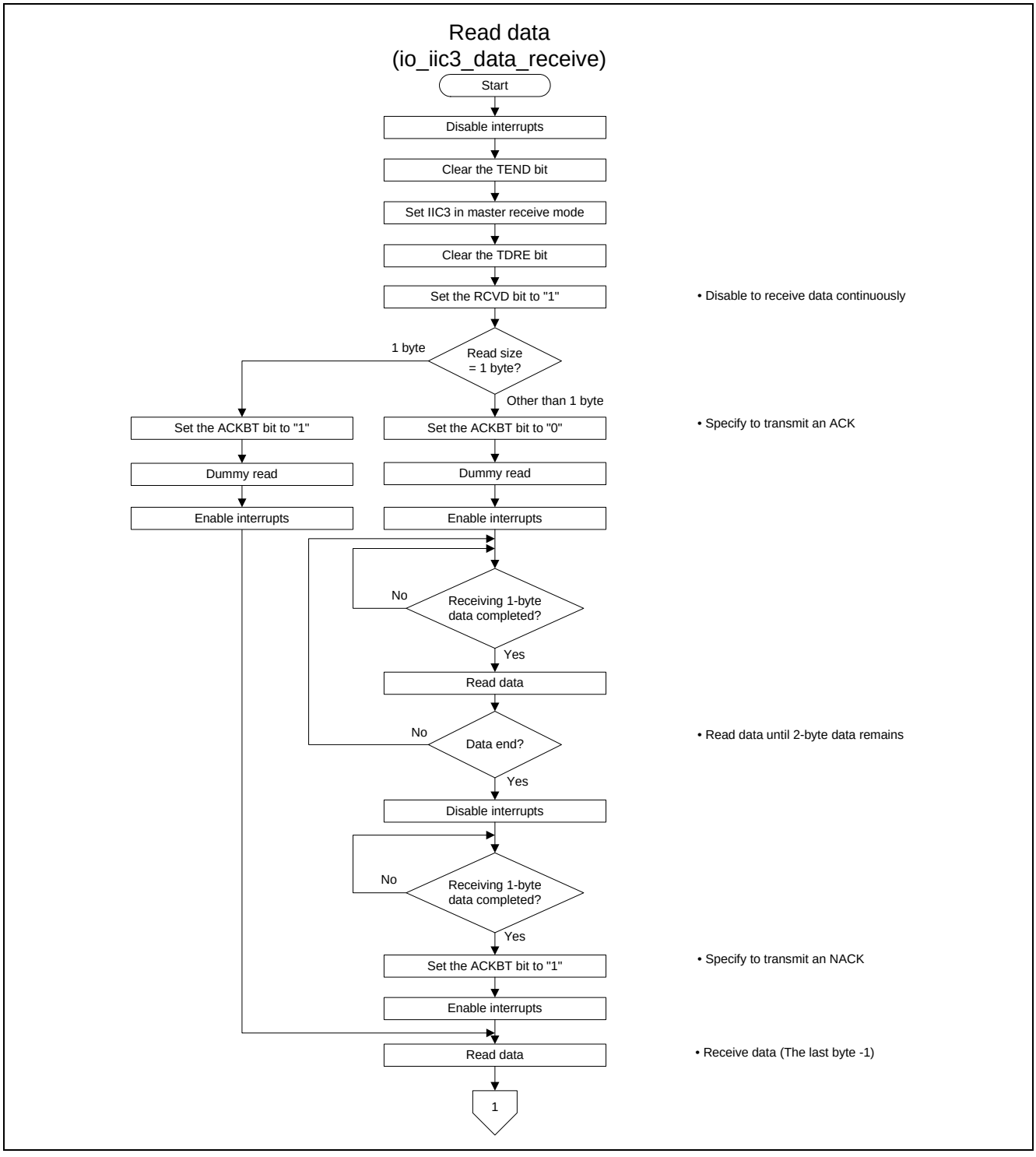

Figure 7 Sample Program Flow Chart (3/6)

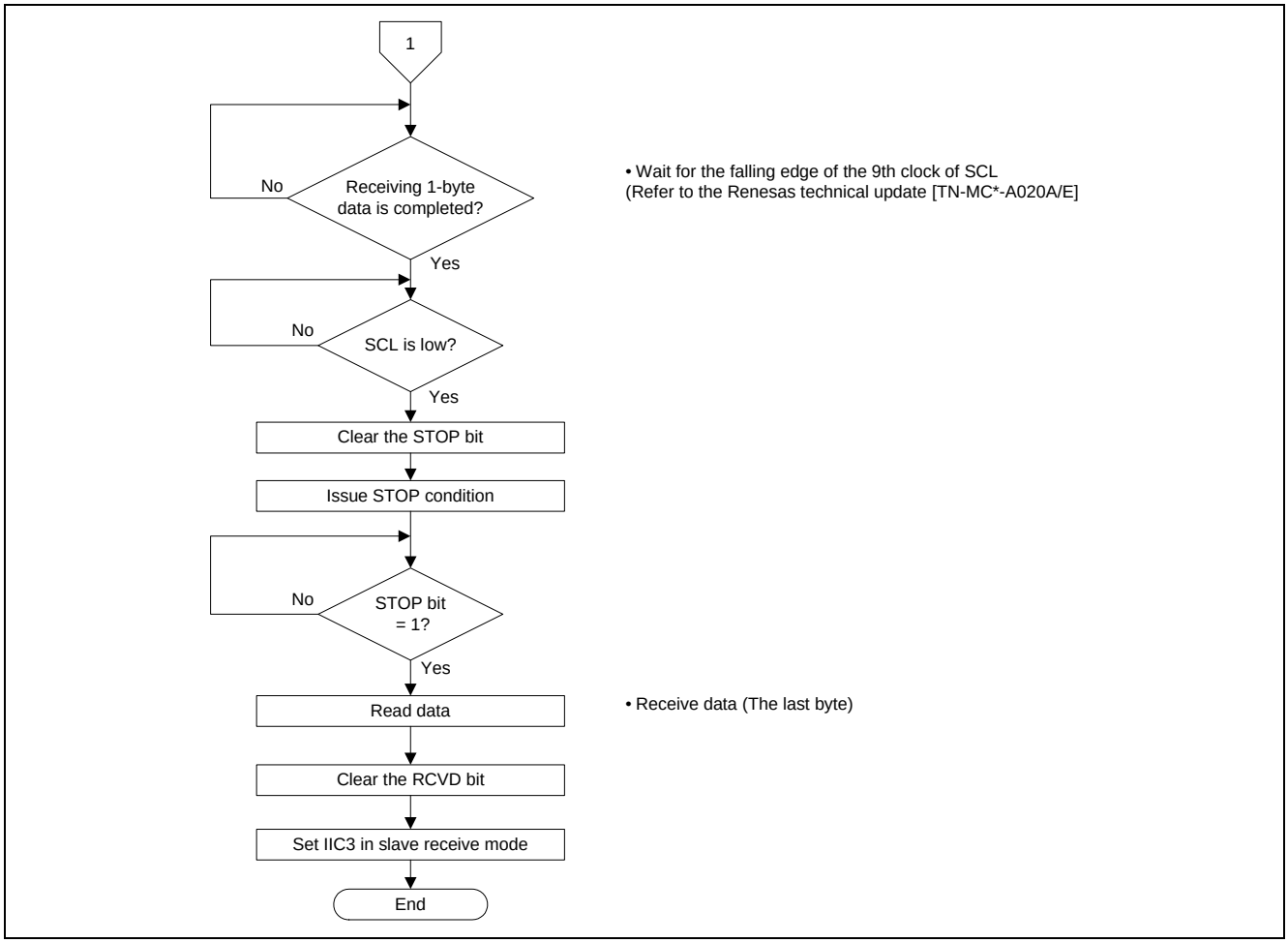

Figure 8 Sample Program Flow Chart (4/6)

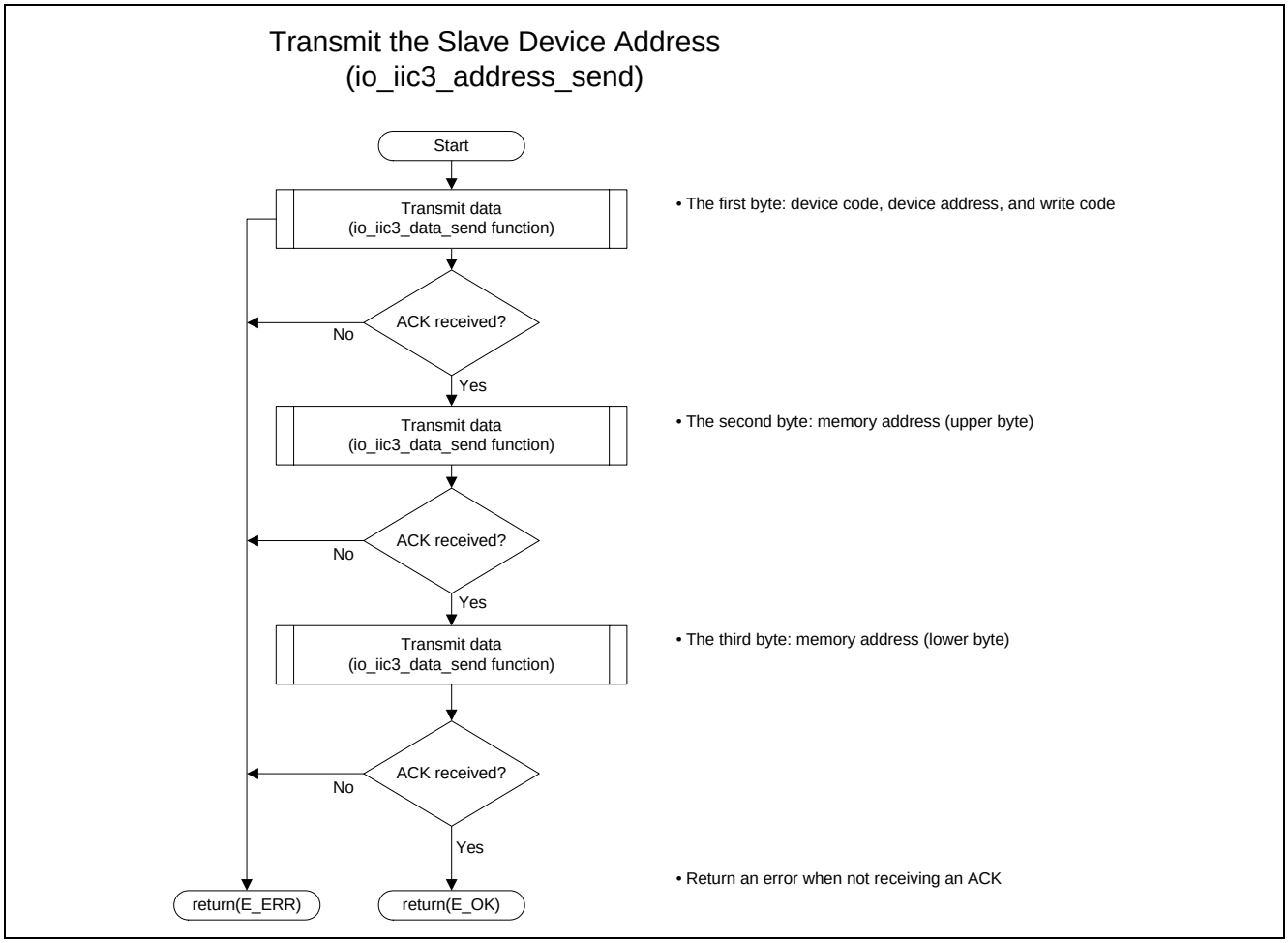

Figure 9 Sample Program Flow Chart (5/6)

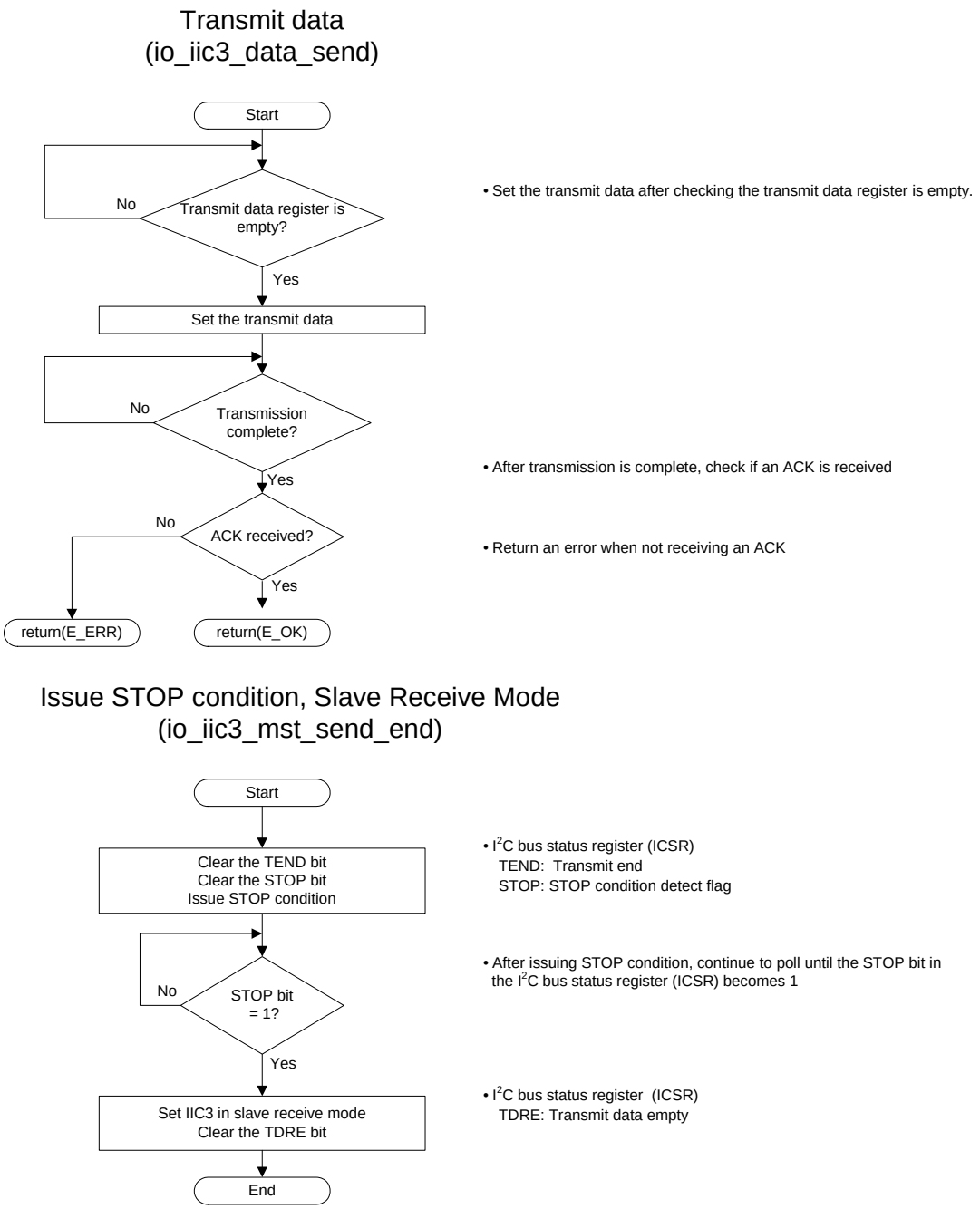

Figure 10 Sample Program Flow Chart (6/6)

2.5 Notes for Master Receive Mode

When reading the I²C bus receive data register (ICDRR) at the falling edge of around the 8th clock, the receive data may not be retrieved.

When the receive buffer is full and specifying the receive disable bit (RCVD) in the I²C bus controller register at the falling edge of around the 8th clock, STOP condition may not be issued. To solve these problems, follow either of the procedures listed below. The sample program sets the RCVD bit to 1 to transmit or receive data in units of one byte.

1. Read the ICDRR bit in master receive mode before the rising edge of the 8th clock.
2. Set the RCVD bit in master receive mode and transmit or receive data in units of one byte.

2.6 Notes for Setting the ACKBT in Master Receive Mode

When IIC3 is in master receive mode, set the ACKBT bit before rising the 8th SCL signal of the last data continuously transferred. Otherwise, a slave device may overrun.

As the sample program sets the RCVD bit to 1 to transmit or receive data in units of one byte, this is not applicable.

2.7 Notes for Issuing the STOP Condition or START Condition Again in Master Receive Mode

When issuing the STOP condition or START condition again at the falling edge of the SCL 9th clock, an additional cycle is output after the 9th clock. Make sure to issue the STOP condition or START condition again after receiving data in master receive mode, and the falling edge of the SCL 9th clock.

How to make sure the falling edge of the SCL 9th clock:

- Check the RDRF (receive data register full) bit in the ICSR register is set to 1, and then check the SCLO bit (SCL monitor) in ICCR2 register is set to 0 (SCL pin is low).

For more information, refer to the Renesas Technical Update (document number: TN-MC*-A020A/E).

2.8 Notes for Using the IICRST Bit

When writing 0 to the ICE bit in ICCR1 register or writing 1 to the ICCRST bit in ICCR2 register while I²C bus is operating, the BBSY bit in ICCR2 register and STOP bit in the ICSR register are not defined.

For more information, refer to the Renesas Technical Update (document number: TN-MC*-A022A/E).

3. Sample Program Listing

3.1 Supplement to the Sample Program

As the capacity of the SH7264 large-capacity internal RAM varies as 1 MB or 640 KB, depending on the MCU type, the section alignment and register setting must be partly altered. To support both MCU types, this application note provides two types of sample programs (workspaces) for 1-MB RAM and 640-KB RAM.

As the MCU with 640-KB RAM must be write-enabled before writing data in the data-retention RAM, the System control register 5 (SYSCR5) is set to write-enable the RAM in the sample program for 640-KB RAM.

Review your product and use the appropriate workspace.

3.2 Sample Program Listing "main.c" (1/11)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corp. and is only
5   *   intended for use with Renesas products.  No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corp. and is protected under
8   *   all applicable laws, including copyright laws.
9   *
10  *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11  *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12  *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13  *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14  *   DISCLAIMED.
15  *
16  *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17  *   ELECTRONICS CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18  *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19  *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20  *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21  *
22  *   Renesas reserves the right, without notice, to make changes to this
23  *   software and to discontinue the availability of this software.
24  *   By using this software, you agree to the additional terms and
25  *   conditions found by accessing the following link:
26  *   http://www.renesas.com/disclaimer
27  ************************************************************************
28  * Copyright(C) 2009(2010) Renesas Electronics Corporation. All rights reserved.
29  *""FILE COMMENT""*********** Technical reference data *********************
30  *   System Name : SH7264 Sample Program
31  *   File Name   : main.c
32  *   Abstract    : IIC3 master receive mode sample program
33  *   Version     : 1.01.00
34  *   Device      : SH7264/SH7262
35  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36  *               : C/C++ compiler package for the SuperH RISC engine family
37  *               :                             (Ver.9.03 Release00).
38  *   OS          : None
39  *   H/W Platform: M3A-HS64G50(CPU board)
40  *   Description :
41  ************************************************************************
42  *   History     : Jan.23,2009 Ver.1.00.00
43  *               : Jul.06,2010 Ver.1.01.00 Modified to support TU (TN-MC*-A020A/E)
44  *""FILE COMMENT END""*****************************************************/
45  #include <machine.h>
46  #include "iodefine.h"       /* SH7264 iodefine  */
47
```

3.3 Sample Program Listing "main.c" (2/11)

```
48    /* ==== symbol definition ==== */
49    #define EEPROM_MEM_ADDR 0x0000
50    #define DEVICE_CODE 0xA0  /* EEPROM device code   :b'1010       */
51    #define DEVICE_ADDR 0x00  /* EEPROM device address:b'000    */
52    #define IIC_DATA_WR 0x00  /* Data write code      :b'0      */
53    #define IIC_DATA_RD 0x01  /* Data read code       :b'1      */
54    #define IIC3_DATA 10
55
56    #define E_OK 0
57    #define E_ERR -1
58
59    /* ==== RAM allocation variable declaration ==== */
60    unsigned char ReadData[IIC3_DATA];
61
62    /* ==== prototype declaration ==== */
63    void main(void);
64    void io_iic3_mst_send_end(void);
65    int io_iic3_init(void);
66    int io_iic3_eeprom_read(unsigned char d_code,unsigned char d_adr,unsigned short r_adr,
67                            unsigned int r_size,unsigned char* r_buf);
68    int io_iic3_data_receive(unsigned char* r_buf,unsigned int r_size);
69    int io_iic3_data_send(unsigned char data);
70    int io_iic3_address_send(unsigned char* data);
71
```

3.4 Sample Program Listing "main.c" (3/11)

```
72   /*""FUNC COMMENT""*****************************************************************
73    * ID           :
74    * Outline      : Sample program main
75    *-------------------------------------------------------------------------------
76    * Include      :
77    *-------------------------------------------------------------------------------
78    * Declaration  : void main(void);
79    *-------------------------------------------------------------------------------
80    * Description  : Receives data from EEPROM using IIC3.
81    *-------------------------------------------------------------------------------
82    * Argument     : void
83    *-------------------------------------------------------------------------------
84    * Return Value : void
85    *-------------------------------------------------------------------------------
86    * Note         : None
87    *""FUNC COMMENT END""*************************************************************/
88   void main(void)
89   {
90     int i;
91
92     /* ==== Clears the buffer storing data ==== */
93     for(i=0;i<IIC3_DATA;i++){
94       ReadData[i] = 0x00;
95     }
96
97     /* ==== Configures IIC3 ==== */
98     io_iic3_init();
99
100    /* ==== Receives data in IIC3 master receive mode ==== */
101    io_iic3_eeprom_read(DEVICE_CODE,        /* Device code         */
102                        DEVICE_ADDR,       /* Device address      */
103                        0x0000,            /* Read start address  */
104                        sizeof(ReadData),  /* Read data size      */
105                        ReadData);         /* Buffer storing data */
106
107    while(1){
108      /* Loop */
109    }
110  }
111
```

3.5 Sample Program Listing "main.c" (4/11)

```
112  /*""FUNC COMMENT""*****************************************************************
113   * ID           :
114   * Outline      : IIC3 module configuration
115   *-------------------------------------------------------------------------------
116   * Include      : iodefine.h
117   *-------------------------------------------------------------------------------
118   * Declaration  : int io_iic3_init(void);
119   *-------------------------------------------------------------------------------
120   * Description  : Configures IIC3 channel 1.
121   *-------------------------------------------------------------------------------
122   * Argument     : void
123   *-------------------------------------------------------------------------------
124   * Return Value : E_OK
125   *-------------------------------------------------------------------------------
126   * Note         : None
127   *""FUNC COMMENT END""*************************************************************/
128  int io_iic3_init(void)
129  {
130
131    /* ---- STBCR5 ---- */
132    CPG.STBCR5.BIT.MSTP56   = 0;    /* IIC3 channel 1 is operating   */
133
134    /* ---- PORT ---- */
135    PORT.PECR0.BIT.PE2MD = 0x01;/* SCL1 select */
136    PORT.PECR0.BIT.PE3MD = 0x01;/* SDA1 select */
137
138
139    /* ----IIC31 module operation enabled ---- */
140    IIC3_1.ICCR1.BIT.ICE = 1u;      /* IIC3 module is operation-enabled */
141    IIC3_1.ICCR1.BIT.RCVD = 0u;     /* Continues to the next reception */
142    IIC3_1.ICCR1.BIT.MST = 1u;      /* Selects the master mode  */
143    IIC3_1.ICCR1.BIT.TRS = 0u;      /* Selects the receive mode */
144    IIC3_1.ICCR1.BIT.CKS = 5u;      /* Transfer clock rate: P-clock/92 (391 kHz) */
145    /*  ---IIC bus mode register (ICMR) setting --- */
146    IIC3_1.ICMR.BYTE = 0x30u;
147             /*
148                   bit 7      : MLS:0 ----------------------   MSB first
149                   bits 6 to 4: Reserve:1 -----------------   Reserve bit
150                   bit 3      : BCWP:0----------------------   Not set
151                   bits 2 to 0: BC0:0, BC1:0,BC0:0---------   IIC format 9-bit
152             */
153
154    return(E_OK);
155  }
156
```

3.6 Sample Program Listing "main.c" (5/11)

```
157   /*""FUNC COMMENT""*****************************************************************
158    * ID          :
159    * Outline     : Read data from EEPROM
160    *-------------------------------------------------------------------------------
161    * Include     : iodefine.h
162    *-------------------------------------------------------------------------------
163    * Declaration : int io_iic3_eeprom_read(unsigned char d_code,
164    *             :                         unsigned char d_adr,
165    *             :                         unsigned short r_adr,
166    *             :                         unsigned int r_size,
167    *             :                         unsigned char* r_buf);
168    *-------------------------------------------------------------------------------
169    * Description : Reads the r_size bytes of data from the EEPROM specified by the
170    *             : device code (d_code), device address (d_adr), and stores the
171    *             : read data in the buffer specified by the r_buf. Specify the memory
172    *             : address of EEPROM by the r_adr.
173    *-------------------------------------------------------------------------------
174    * Argument    : unsigned char d_code : Device code
175    *             : unsigned char d_adr  : Device address
176    *             : unsigned short r_adr : Read start address
177    *             : unsigned int r_size  : Read data size
178    *             : unsigned char* r_buf : Buffer storing the read data
179    *-------------------------------------------------------------------------------
180    * Return Value: ACK received: E_OK
181    *             : ACK not received: E_ERR
182    *""FUNC COMMENT END""*************************************************************/
183   int io_iic3_eeprom_read(unsigned char d_code,unsigned char d_adr,unsigned short r_adr,
184                           unsigned int r_size,unsigned char* r_buf)
185   {
186     int ack = E_OK;
187     unsigned char send[4];
188
189     send[0] = (unsigned char)(d_code|((d_adr & 0x7)<<1)|IIC_DATA_WR);
190     send[1] = (unsigned char)((r_adr>>8) & 0x00ff);
191     send[2] = (unsigned char)(r_adr & 0x00ff);
192     send[3] = (unsigned char)(d_code|((d_adr & 0x7)<<1)|IIC_DATA_RD);
193
194     while(IIC3_1.ICCR2.BIT.BBSY == 1u){
195       /* Waits until the bus is released */
196     }
197     IIC3_1.ICCR1.BYTE |= 0x30u;                /* Sets IIC3 in master transmit mode */
198     IIC3_1.ICCR2.BYTE = ((IIC3_1.ICCR2.BYTE & 0xbfu)|0x80u);   /* Issues START condition */
199
200     ack = io_iic3_address_send(send);          /* Transmits the 1st, 2nd, and 3rd bytes */
201
```

3.7 Sample Program Listing "main.c" (6/11)

```
202     if(ack == E_OK){
203       /* ACK received from the specified device */
204       IIC3_1.ICCR2.BYTE=((IIC3_1.ICCR2.BYTE & 0xbfu) | 0x80u);/* Issues START condition */
205       ack = io_iic3_data_send(send[3]);                    /* Transmits the 4th byte */
206       if(ack == E_OK){
207           io_iic3_data_receive(r_buf,r_size);                 /* Receives data */
208       }
209       else{
210           io_iic3_mst_send_end();
211       }
212     }
213     else{
214       /* ACK not received from the specified device */
215       io_iic3_mst_send_end();
216     }
217     return(ack);
218   }
219
```

3.8 Sample Program Listing "main.c" (7/11)

```
220  /*""FUNC COMMENT""*****************************************************************
221   * ID          :
222   * Outline     : Master receive mode
223   *-------------------------------------------------------------------------------
224   * Include     : iodefine.h
225   *-------------------------------------------------------------------------------
226   * Declaration : int io_iic3_data_receive(unsigned char* r_buf,
227   *             :                          unsigned int r_size);
228   *-------------------------------------------------------------------------------
229   * Description : Sets IIC3 in master receive mode to receive the r_size bytes of
230   *             : data, and stores the receive data in the r_buf.
231   *             : After te specified number of bytes received, switches in slave
232   *             : receive mode.
233   *-------------------------------------------------------------------------------
234   * Argument    : unsigned char* r_buf : Buffer to store the read data
235   *             : unsigned int r_size  : Read data size
236   *-------------------------------------------------------------------------------
237   * Return Value: Always returns the E_OK
238   *""FUNC COMMENT END""*************************************************************/
239  int io_iic3_data_receive(unsigned char* r_buf,unsigned int r_size)
240  {
241    int cnt, mask;
242    unsigned char dummy;
243
244    mask = get_imask();
245    set_imask(15);                    /* Interrupts are disabled */
246
247    /* ==== Sets IIC3 in master receive mode (non-contiguous reception) ==== */
248    IIC3_1.ICSR.BIT.TEND = 0u;        /* Clears the TEND bit */
249    IIC3_1.ICCR1.BIT.MST = 1u;        /* Master mode */
250    IIC3_1.ICCR1.BIT.TRS = 0u;        /* Receive mode */
251    IIC3_1.ICSR.BIT.TDRE = 0u;        /* Clears the TDRE bit */
252    IIC3_1.ICCR1.BIT.RCVD = 1u;       /* Disables to receive data continuously */
253
254    /* ==== Starts receiving data (only single byte) ==== */
255    if(r_size == 1){                  /* When receiving a single byte */
256      IIC3_1.ICIER.BIT.ACKBT = 1u;    /* Sets the ACK bit to high */
257      dummy = IIC3_1.ICDRR;           /* Dummy read */
258      set_imask(mask);                /* Interrupts are enabled */
259    }
260    /* ==== Starts receiving data (more than 2 bytes) ==== */
261    else{
262      IIC3_1.ICIER.BIT.ACKBT = 0u;    /* Sets the ACK bit to low */
263      dummy = IIC3_1.ICDRR;           /* Dummy read */
264      set_imask(mask);                /* Interrupts are enabled */
265
```

3.9 Sample Program Listing "main.c" (8/11)

```
266       /* ==== Reads data until 2-byte data remains ==== */
267       cnt = r_size;
268       while( cnt > 2 ){
269           /* ---- Waits until receiving a single byte data is completed ---- */
270           while(IIC3_1.ICSR.BIT.RDRF == 0u){
271               /* wait */
272           }
273           /* ---- Reads data ---- */
274           *r_buf++ = IIC3_1.ICDRR;
275           cnt--;
276       }
277
278       set_imask(15);                /* Interrupts are disabled */
279       /* ==== Waits until receiving data (the last byte -1) is completed ==== */
280       while(IIC3_1.ICSR.BIT.RDRF == 0u){
281           /* wait */
282       }
283       /* ==== Sets the acknowledge to high
284               before receiving the last byte data is received ==== */
285       IIC3_1.ICIER.BIT.ACKBT = 1u;
286       set_imask(mask);              /* Interrupts are enabled */
287
288       /* ==== Starts receiving the last byte data (read the last byte -1 data) ==== */
289       *r_buf++ = IIC3_1.ICDRR;
290     }
291
292     /* ==== Waits until receiving the last byte is completed ==== */
293     while(IIC3_1.ICSR.BIT.RDRF == 0u){
294       /* wait */
295     }
296
297     /* ==== Issues the STOP condition ==== */
298     /* ---- Waits for the falling edge of the 9th clock of SCL ---- */
299     while(IIC3_1.ICCR2.BIT.SCLO == 1u){ /* Technical Update [TN-MC*-A020A/E] */
300       /* wait */
301     }
302     /* ---- Starts issuing conditions ---- */
303     IIC3_1.ICSR.BIT.STOP = 0u;          /* Clears the STOP flag */
304     IIC3_1.ICCR2.BYTE &= 0x3fu;         /* Issues the STOP condition */
305
306     /* ---- Waits until issuing the STOP condition is completed ---- */
307     while(IIC3_1.ICSR.BIT.STOP == 0u){
308       /* wait */
309     }
310
311     /* ==== Reads the last byte data ==== */
312     *r_buf = IIC3_1.ICDRR;              /* The last byte */
313
314     /* ==== Switches back to slave receive mode ==== */
315     IIC3_1.ICCR1.BIT.RCVD = 0u;         /* Clear the RCVD bit */
316     IIC3_1.ICCR1.BYTE &= 0xcfu;         /* Slave receive mode */
317
318     return(E_OK);
319   }
320
```

3.10 Sample Program Listing "main.c" (9/11)

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Transmit the slave device address
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : int io_iic3_address_send(unsigned char* data);
 *-----------------------------------------------------------------------------
 * Description  : Transmits the address of the slave device (1 byte) and the memory
 *              : address (2 bytes) specified by the argument data.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char* data ; I : Transmit data
 *-----------------------------------------------------------------------------
 * Return Value : ACK received: E_OK
 *              : ACK not received: E_ERR
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
int io_iic3_address_send(unsigned char* data)
{
  int ack;

  ack = io_iic3_data_send(*data++);      /* Slave device address */
  if(ack == E_ERR){
    return(ack);
  }
  ack = io_iic3_data_send(*data++);      /* 1st memory address */
  if(ack == E_ERR){
    return(ack);
  }
  ack = io_iic3_data_send(*data);        /* 2nd  memory address */
  if(ack == E_ERR){
    return(ack);
  }
  return(ack);
}

```

3.11 Sample Program Listing "main.c" (10/11)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Transmit one byte of data
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : int io_iic3_data_send(unsigned char data);
 *-----------------------------------------------------------------------------
 * Description  : Transmits the data as the following steps:
 *              : 1. Waits for the ICDRT empty
 *              : 2. Sets the transmit data
 *              : 3. Confirms the transmission is completed
 *              : 4. Confirms an ACK is received
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char data : Transmit data
 *-----------------------------------------------------------------------------
 * Return Value : ACK received: E_OK
 *              : ACK not received: E_ERR
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
int io_iic3_data_send(unsigned char data)
{
  int ack;

  while(IIC3_1.ICSR.BIT.TDRE == 0u){
    /* Waits for the ICDRT empty */
  }
  IIC3_1.ICDRT = data;
  while(IIC3_1.ICSR.BIT.TEND == 0u){
    /* Waits until the transmission is completed */
  }
  if(IIC3_1.ICIER.BIT.ACKBR == 0u){
    ack = E_OK;
  }
  else{
    ack = E_ERR;
  }
  return(ack);
}

```

3.12 Sample Program Listing "main.c" (11/11)

```
399   /*""FUNC COMMENT""*****************************************************************
400    * ID           :
401    * Outline      : Issue STOP condition
402    *-------------------------------------------------------------------------------
403    * Include      : iodefine.h
404    *-------------------------------------------------------------------------------
405    * Declaration  : void io_iic3_mst_send_end(void);
406    *-------------------------------------------------------------------------------
407    * Description  : Issues STOP condition, and switches the mode to slave receive mode.
408    *-------------------------------------------------------------------------------
409    * Argument     : void
410    *-------------------------------------------------------------------------------
411    * Return Value : void
412    *-------------------------------------------------------------------------------
413    * Note         : None
414    *""FUNC COMMENT END""*************************************************************/
415   void io_iic3_mst_send_end(void)
416   {
417     IIC3_1.ICSR.BIT.TEND = 0u;      /* Clears the TEND flag */
418     IIC3_1.ICSR.BIT.STOP = 0u;      /* Clears the STOP flag */
419     IIC3_1.ICCR2.BYTE &= 0x3fu;     /* Issues STOP condition */
420
421     while(IIC3_1.ICSR.BIT.STOP == 0u){
422       /* Waits until the bus is released */
423     }
424
425     IIC3_1.ICCR1.BYTE &= 0xcfu;     /* Slave receive mode */
426     IIC3_1.ICSR.BIT.TDRE = 0u;      /* Clears the TDRE bit */
427   }
428
429   /* End of File */
```

4. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev. 3.00
  The latest version of the software manual can be downloaded from the Renesas Electronics website.

- Hardware Manual
  SH7262 Group, SH7264 Group Hardware User's Manual Rev. 2.00
  The latest version of the hardware manual can be downloaded from the Renesas Electronics website.

Website and Support

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

Inquiries
http://www.renesas.com/inquiry

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.14.09	—	First edition issued
1.01	Dec.27.10	—	Updated to support the Renesas Technical Update (TN-MC*-A020A/E) Changed the company name
1.02	Feb.10.12	14	Description amended 2.5 Notes for Master Receive Mode 1. Read the ICDRR bit in master receive mode before the ***rising*** edge of the 8th clock

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

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.

RENESAS

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

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

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

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

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

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

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

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

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1