SH7216 Group

Reading/Writing EEPROM Using I²C Bus Interface 3

R01AN0062EJ0101
Rev. 1.01
Feb. 10, 2012

Summary

This application note describes examples of reading/writing EEPROM using the SH7216 Microcomputers (MCUs) I²C Bus Interface 3 (IIC3) transmission and reception in master mode.

Target Device

SH7216 MCU

Contents

1. Introduction

1.1 Specifications

- Specifies the SH7216 MCU as the master device, and EEPROM as the slave device to write data to EEPROM
- Specifies the SH7216 MCU as the master device, and EEPROM as the slave device to read data from EEPROM
- The transfer rate is set to 195 kHz

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

1.2 Modules Used

- I^2C Bus Interface (IIC3)

1.3 Applicable Conditions

MCU	SH7216
Operating Frequency	Internal clock: 200 MHz Bus clock: 50 MHz Peripheral clock: 50 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 -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

1.4 Related Application Note

For more information, refer to the following application note:

- SH7216 Group Example of Initialization

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

The SH7216 (master device) writes data to an EEPROM (slave device) using the IIC3 and then receives data from the EEPROM.

2.1 IIC3 Operation

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

The SH7216 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 format options. Figure 1 shows the IIC3 block diagram. For details on IIC3, refer to I²C Bus Interface 3 chapter in the SH7214 Group, SH7216 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

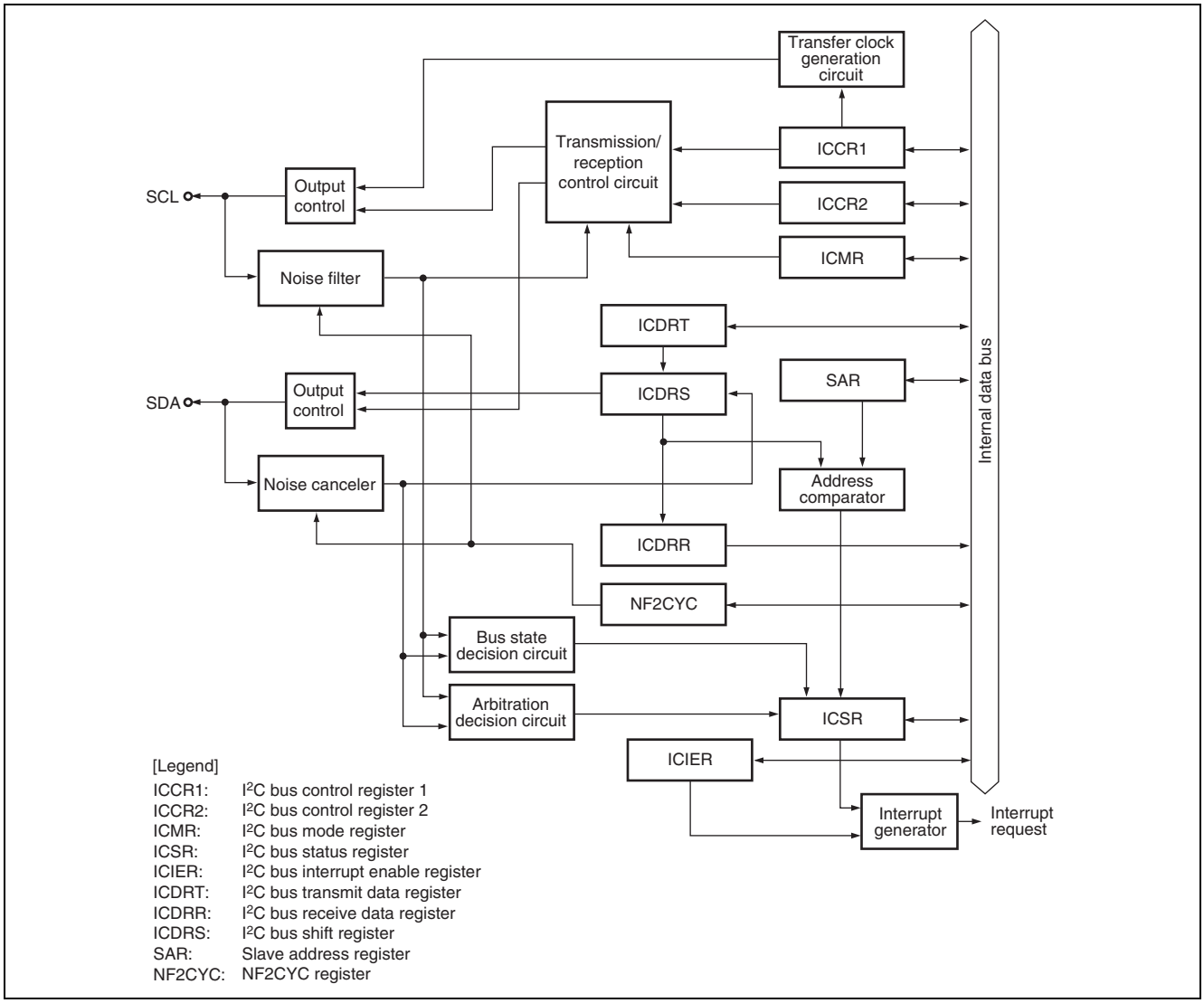


Figure 1 IIC3 Block Diagram

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/256 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 SH7214 Group, SH7216 Group Hardware User's Manual.

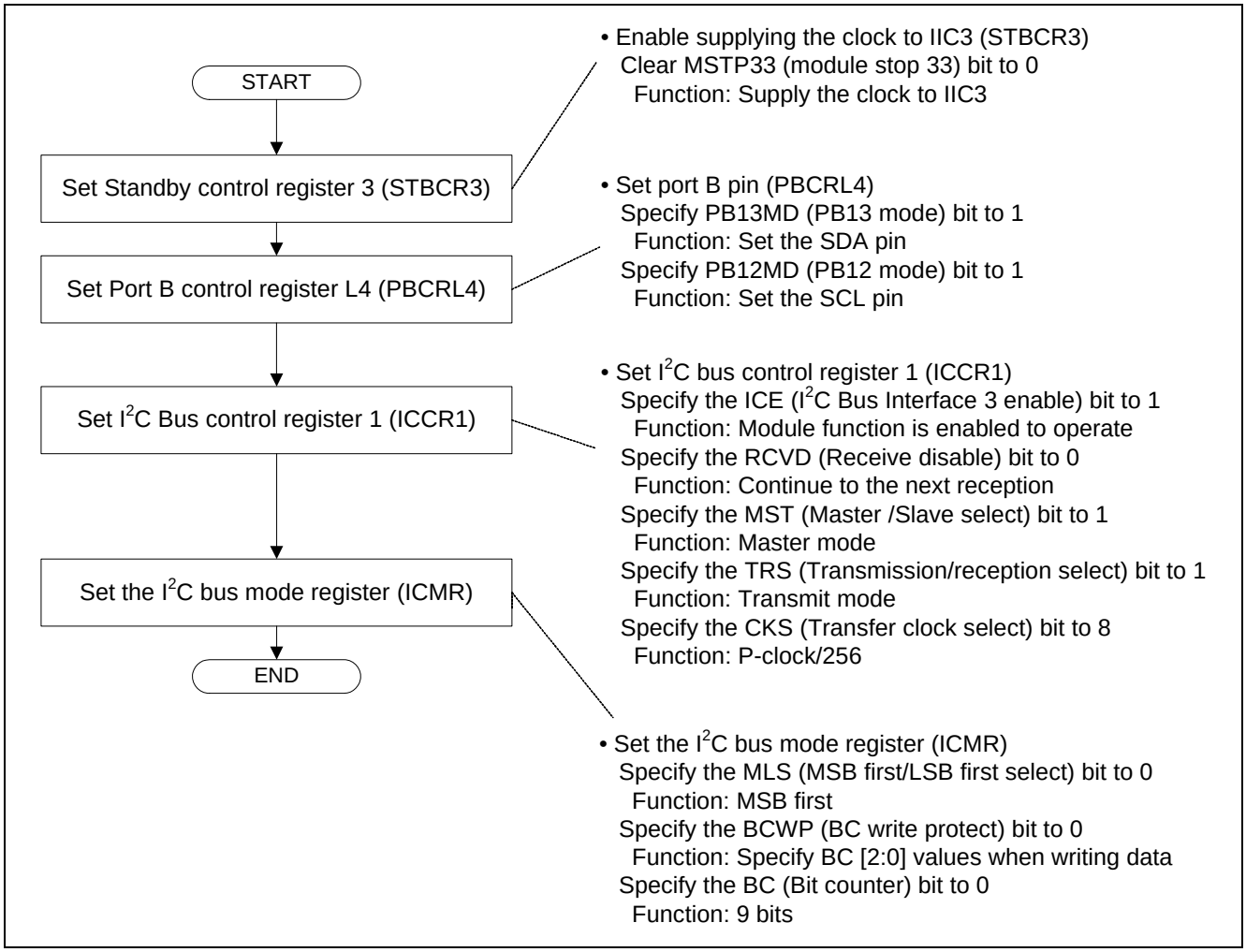


Figure 2 IIC3 Configuration Flow Chart

2.3 Sample Program Operation

The sample program specifies IIC3 in master transmit mode to write 10-byte data in pages (page write). Then, it specifies IIC3 in master receive mode to read 10-byte data sequentially (sequential read).

For device codes, refer to the EEPROM data sheet provided by the manufacturer. The sample program uses the device code "B'1010".

The sample program uses the device address "B'000". For more information, refer to the EEPROM data sheet provided by the manufacturer.

The memory address indicates the write start address or read start address, and the address is incremented at every time writing or reading to/from EEPROM. Figure 3 shows the page write operation. Figure 4 shows the sequential read operation. Figure 5 shows the operating environment of the sample program.

The sample program is tested with the EEPROM (part number: R1EX24064ASA00A, Renesas Electronics).

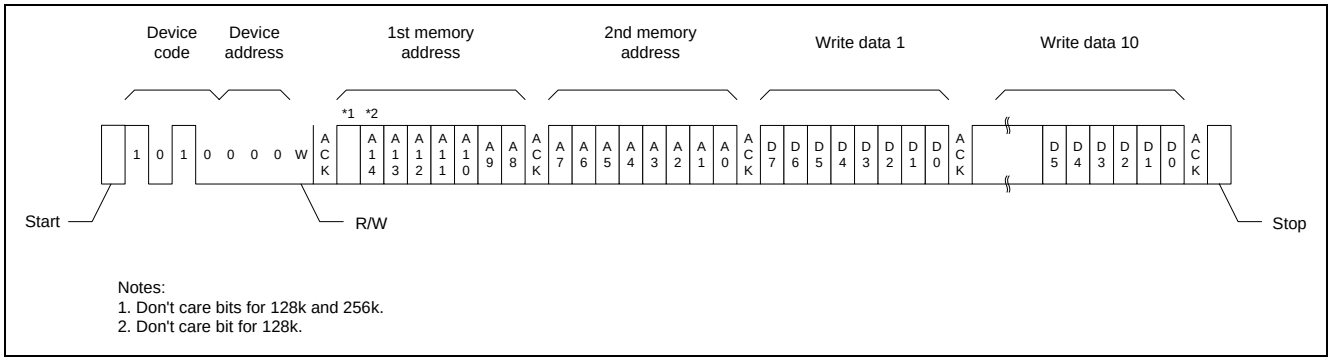


Figure 3 Page Write Operation

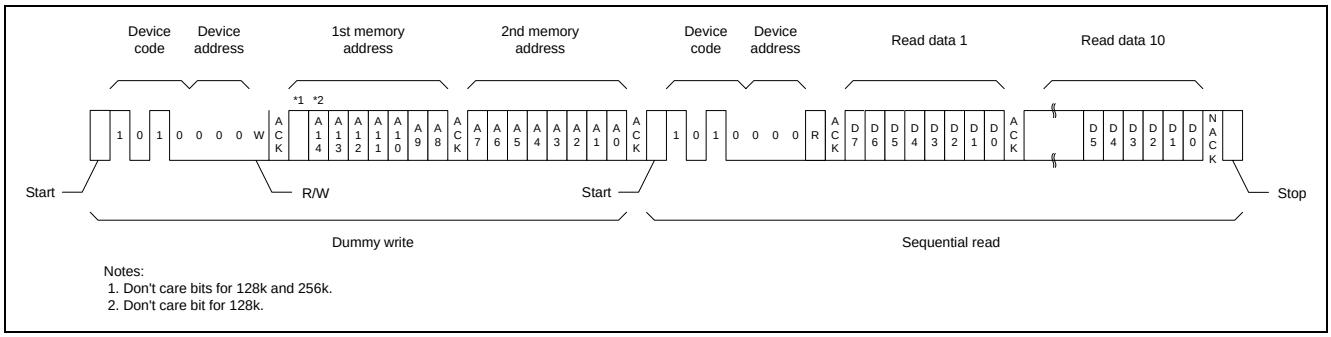


Figure 4 Sequential Read Operation

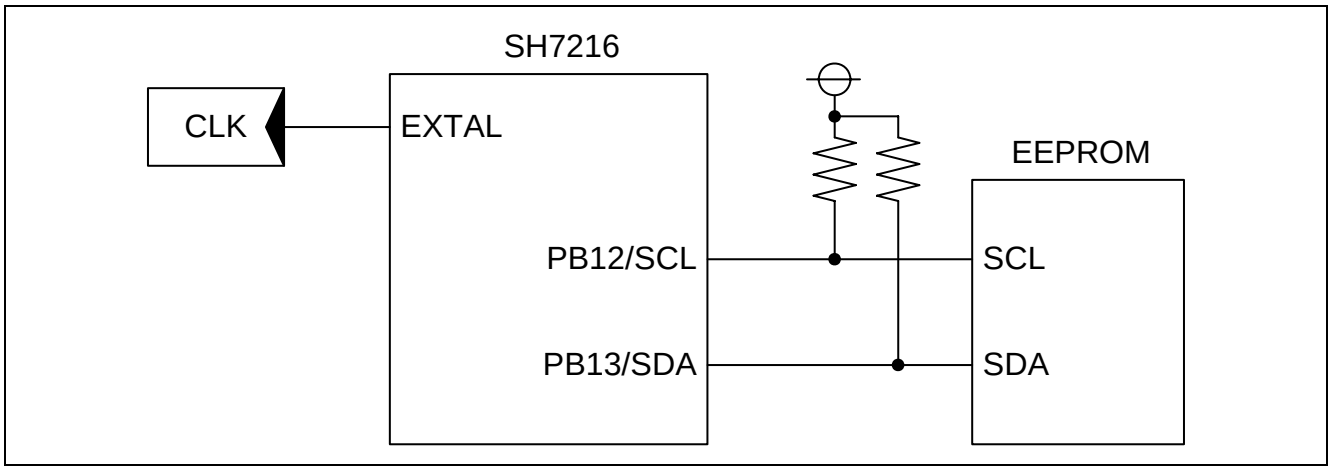


Figure 5 Sample Program Operating Environment

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 6 to Figure 13 show flow charts of the sample program.

Table 2 Register Settings (Default)

Register Name	Address	Setting	Description
Standby control register 3 (STBCR3)	H'FFFE 0408	H'00	MSTP33 = "0": IIC3 is operating
I²C bus control register 1 (ICCR1)	H'FFFE E000	H'B8	ICE = "1": SCL/SDA pins are driven by bus RCVD = "0": Following reception is enabled MST = "1", TRS = "1": Master transmit mode CKS = "B'1000": Transfer rate is Pφ/256
I²C bus mode register (ICMR)	H'FFFE E002	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
E_OK	0	Normal end
E_ERR	−1	Error end

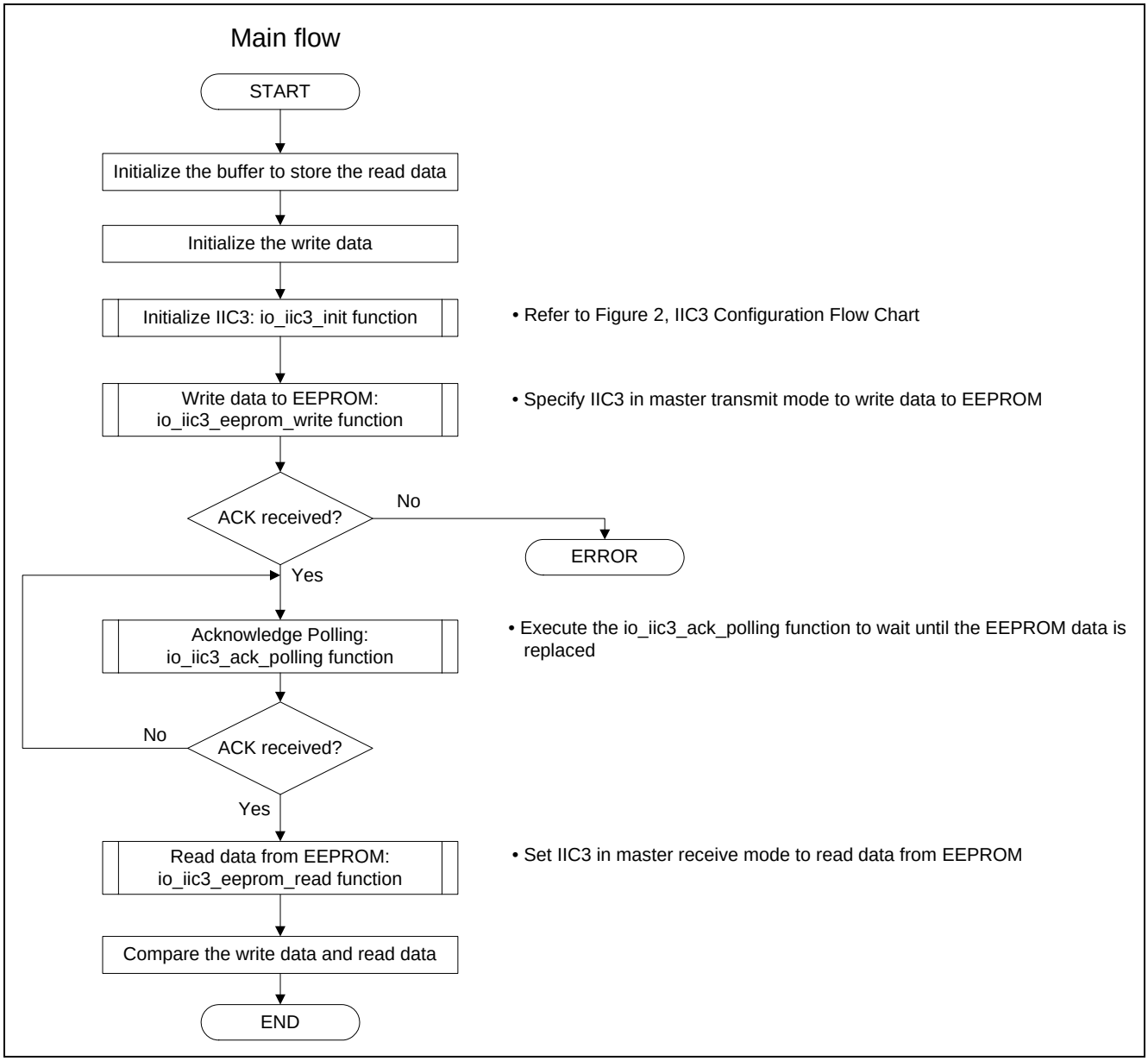


Figure 6 Sample Program Flow Chart (1/8)

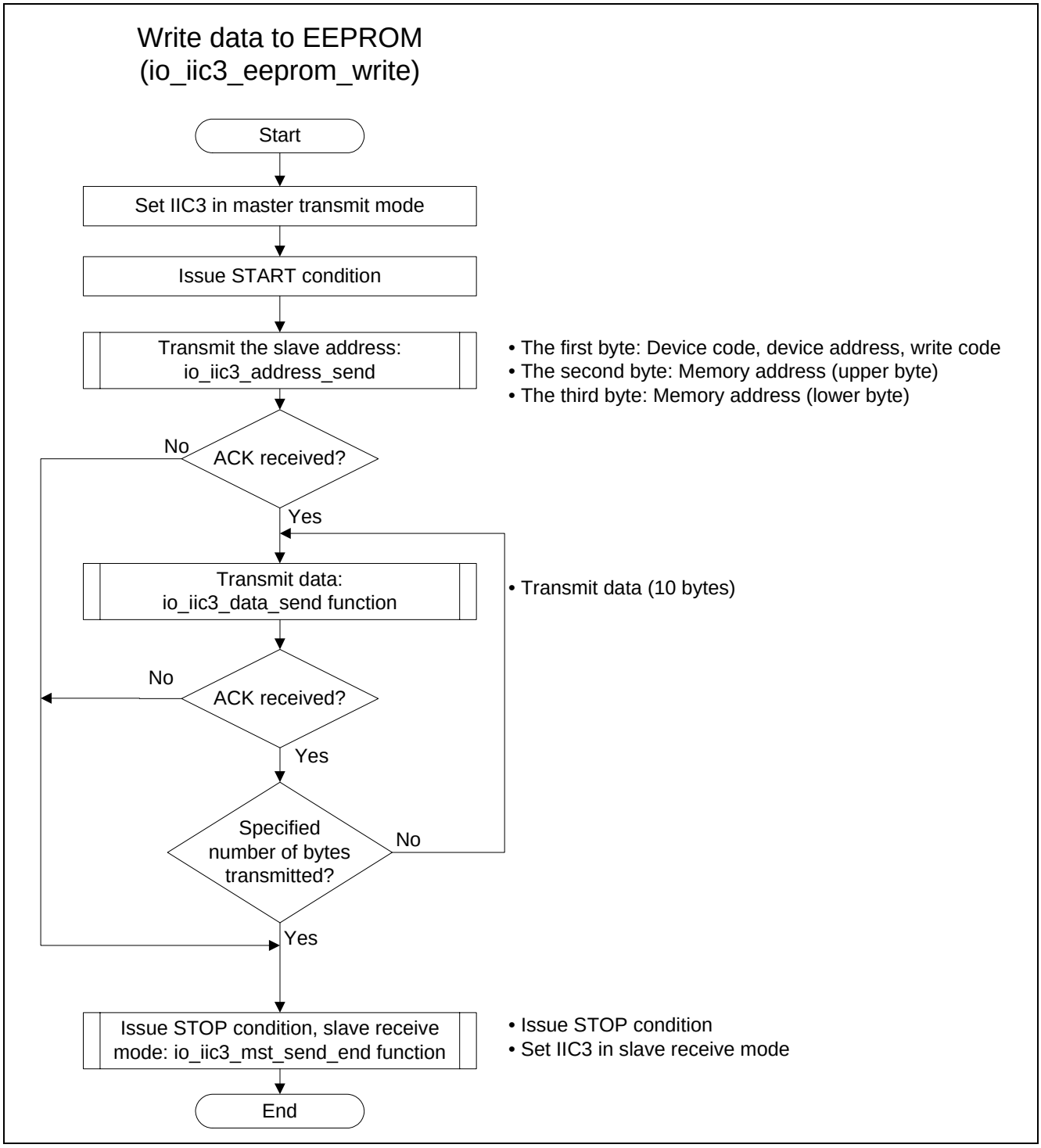


Figure 7 Sample Program Flow Chart (2/8)

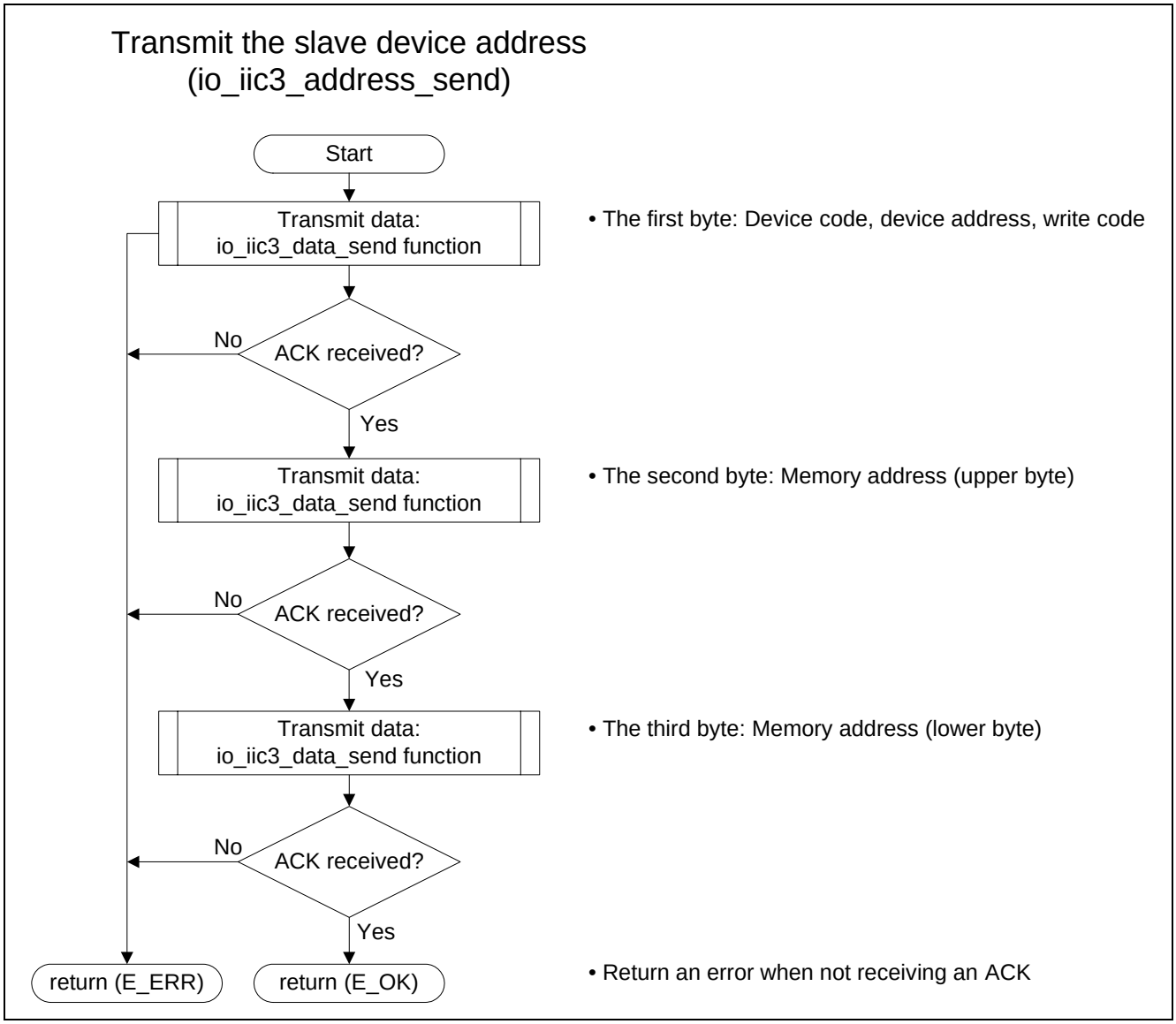
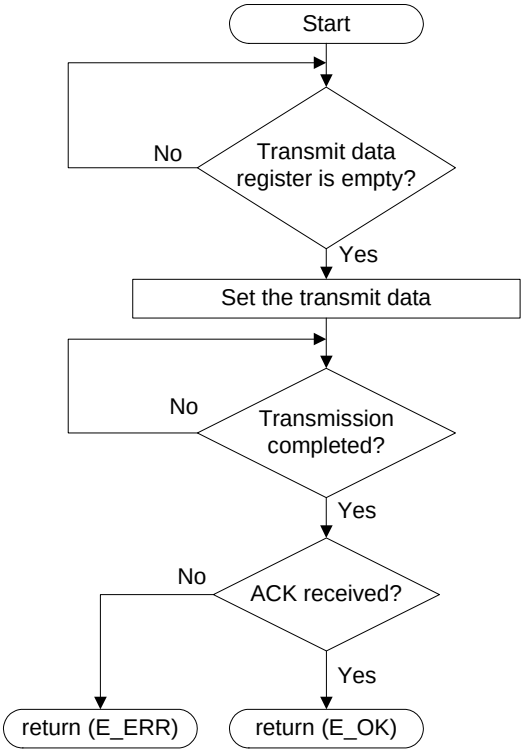


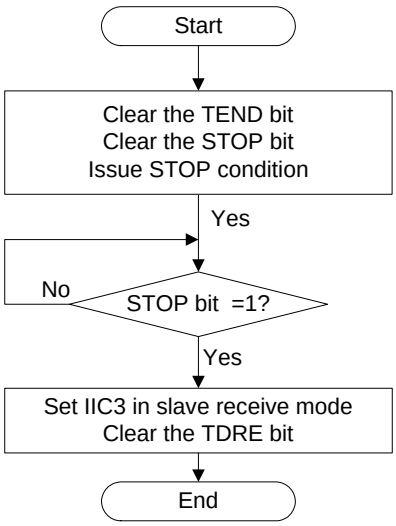
Figure 8 Sample Program Flow Chart (3/8)

Transmit data
(io_iic3_data_send)



- Make sure that the transmit data register is empty, and set the transmit data
- After the transmission is completed, check if an ACK is received
- Return an error when not receiving an ACK

Issue STOP condition, Slave Receive Mode
(io_iic3_mst_send_end)



- I²C bus status register (ICSR)
  TEND: Transmit end
  STOP: STOP condition detect flag
- After issuing the STOP condition, continue to poll until the STOP bit in the I²C bus status register (ICSR) becomes 1
- I²C bus status register (ICSR)
  TDRE: Transmit data empty

Figure 9 Sample Program Flow Chart (4/8)

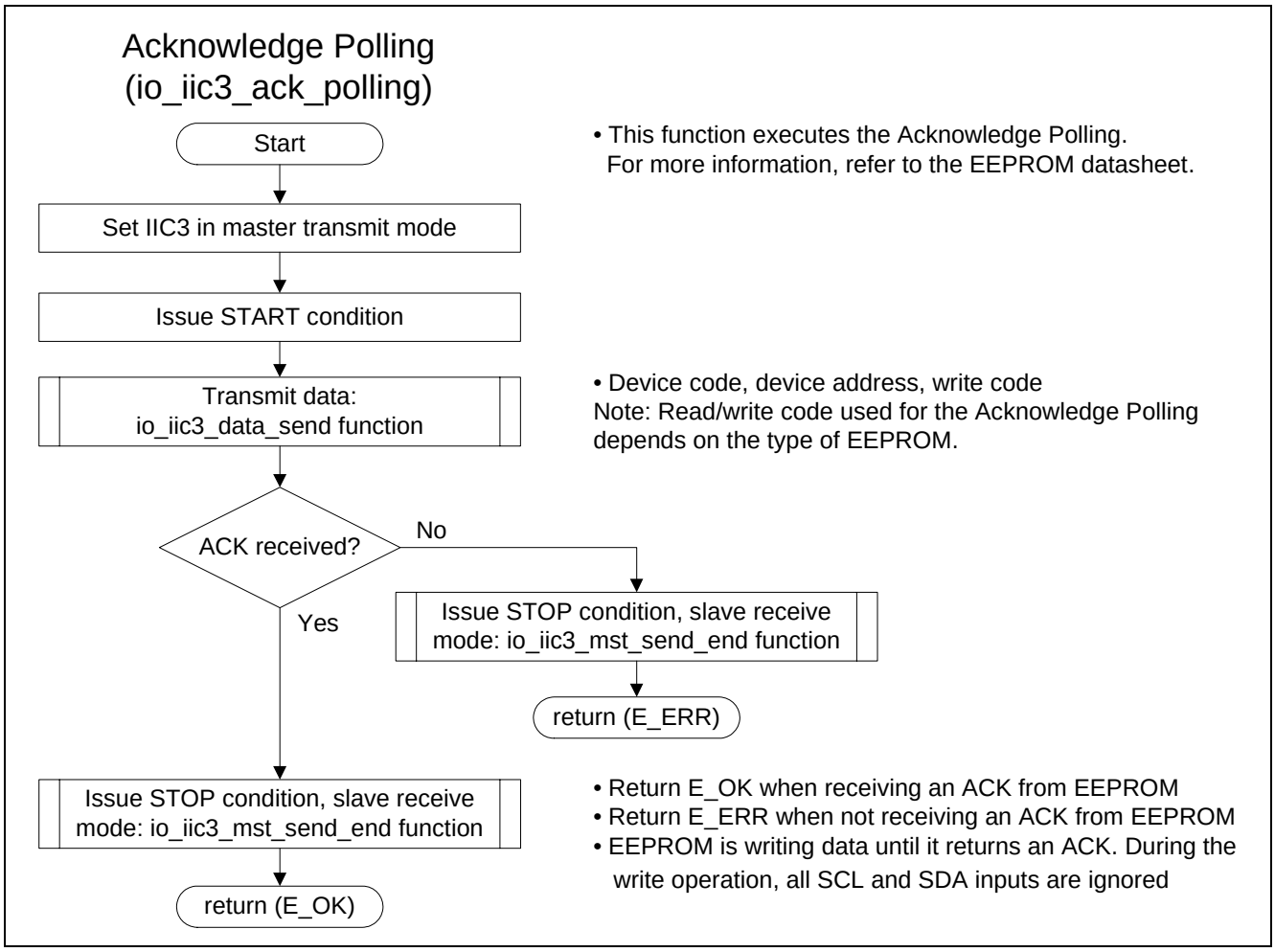


Figure 10 Sample Program Flow Chart (5/8)

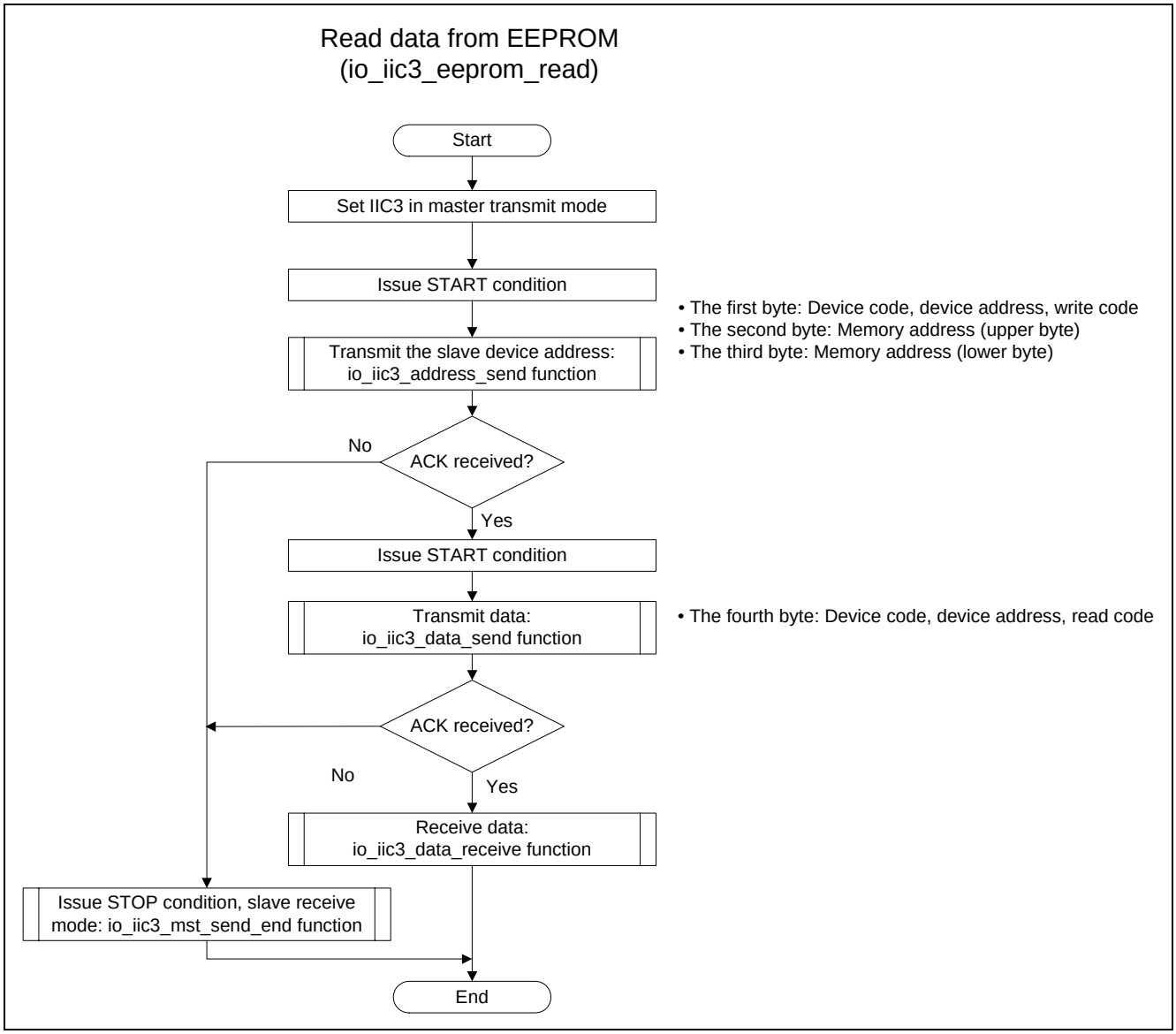


Figure 11 Sample Program Flow Chart (6/8)

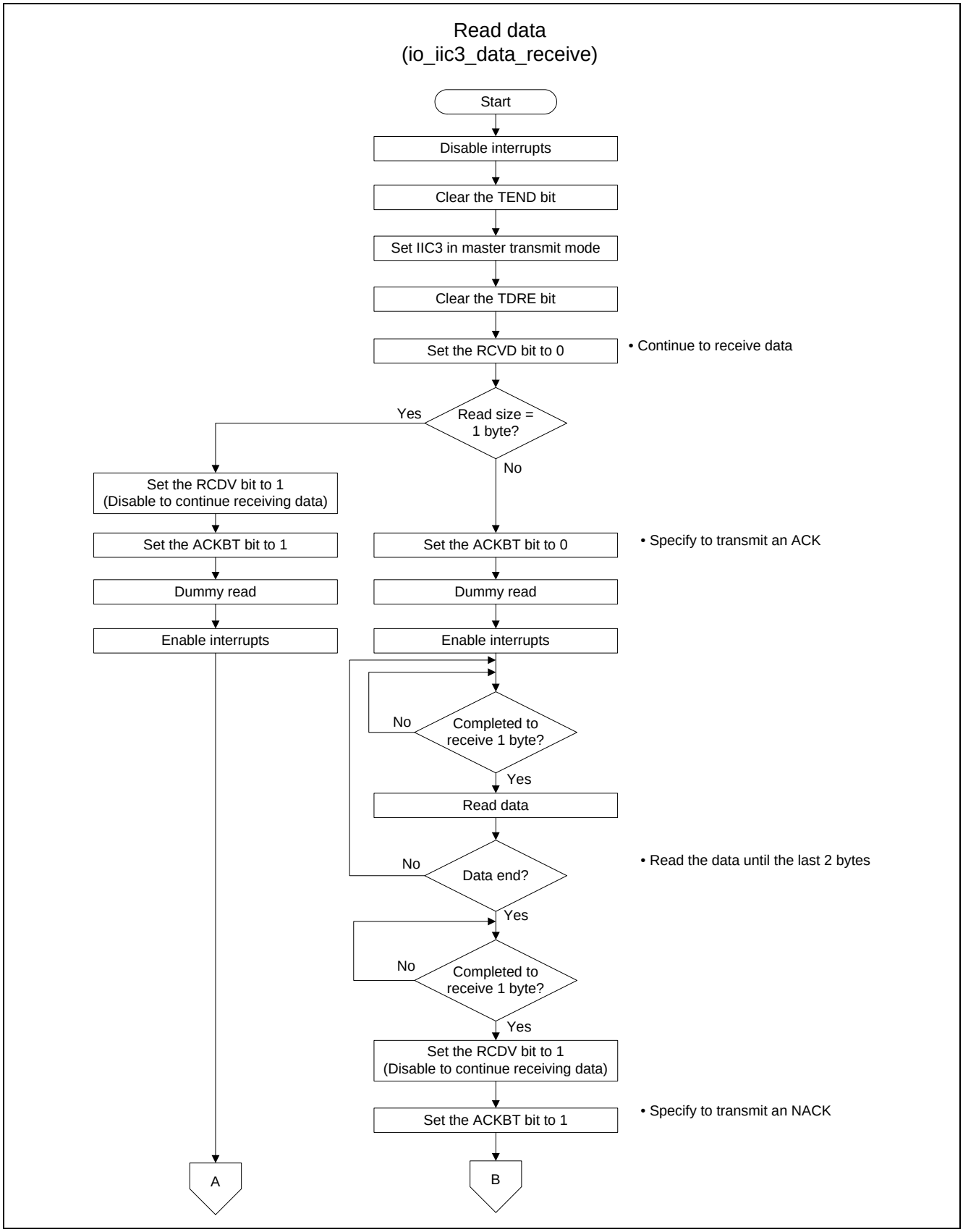


Figure 12 Sample Program Flow Chart (7/8)

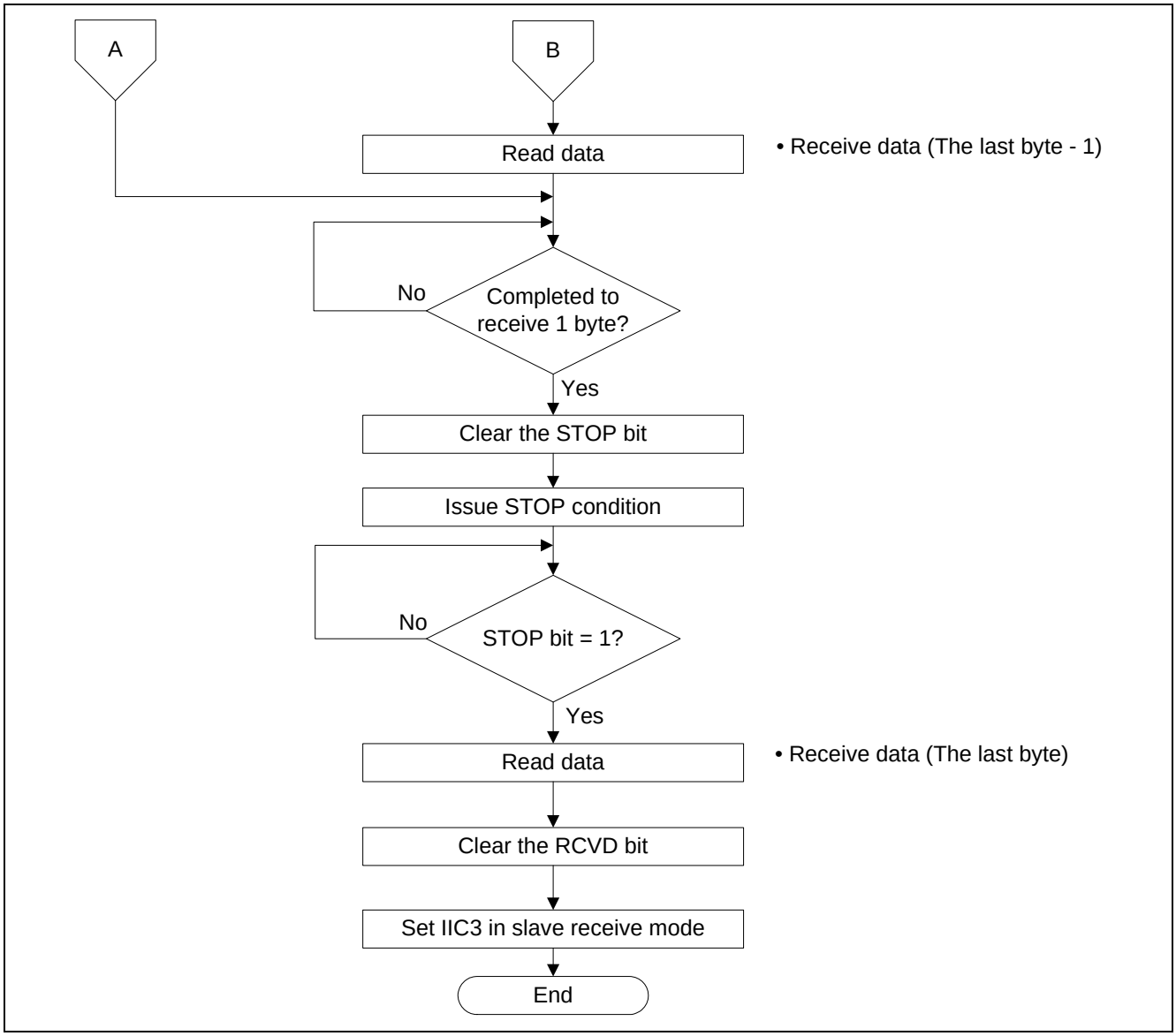


Figure 13 Sample Program Flow Chart (8/8)

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 ICDRR at the falling edge of around the 8th clock, STOP condition may not be issued.

Read the ICDRR in master receive mode before the rising edge of the 8th clock.

2.6 Notes for Setting the ACKBT Bit in Master Receive Mode

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

2.7 Notes for Using the IICRST Bit

When writing 0 to the ICE bit in ICCR1 register or writing 1 to the IICRST 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 Sample Program Listing "main.c" (1/13)

```
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) 2010 Renesas Electronics Corporation. All rights reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7216 Sample Program
31  *   File Name   : main.c
32  *   Abstract    : Reading/Writing EEPROM Using IIC3
33  *   Version     : 1.00.00
34  *   Device      : SH7216
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: R0K572167 (CPU board)
40  *   Description :
41  ***********************************************************************
42  *   History     : Jun.30,2010 Ver.1.00.00
43  *""FILE COMMENT END""*****************************************************/
44  #include <machine.h>
45  #include <stdio.h>
46  #include "iodefine.h"
47
```

3.2 Sample Program Listing "main.c" (2/13)

```
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  unsigned char WriteData[IIC3_DATA];
62
63  /* ==== prototype declaration ==== */
64  void main(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_eeprom_write(unsigned char d_code,unsigned char d_adr,
69                           unsigned short w_adr,unsigned int w_size,unsigned char* w_buf);
70  int io_iic3_address_send(unsigned char* data);
71  int io_iic3_data_receive(unsigned char* r_buf,unsigned int r_size);
72  int io_iic3_data_send(unsigned char data);
73  void io_iic3_mst_send_end(void);
74  int io_iic3_ack_polling(void);
75
76  /*""FUNC COMMENT""*****************************************************************
77   * ID           :
78   * Outline      : Sample program main
79   *-------------------------------------------------------------------------------
80   * Include      :
81   *-------------------------------------------------------------------------------
82   * Declaration  : void main(void);
83   *-------------------------------------------------------------------------------
84   * Description  : Writes data to EEPROM using IIC3 master transmit mode.
85   *              : Reads data from EEPROM using IIC3 master receive mode.
86   *-------------------------------------------------------------------------------
87   * Argument     : void
88   *-------------------------------------------------------------------------------
89   * Return Value : void
90   *-------------------------------------------------------------------------------
91   * Note         : None
92   *""FUNC COMMENT END""*************************************************************/
93  void main(void)
94  {
95    int i,ack;
96
```

3.3 Sample Program Listing "main.c" (3/13)

```
97      /* ==== Clears the buffer storing the data ==== */
98      for(i=0;i<IIC3_DATA;i++){
99        ReadData[i] = 0x00;
100     }
101     /* ==== Creates the write data ==== */
102     for(i=0;i<IIC3_DATA;i++){
103       WriteData[i] = IIC3_DATA+i;
104     }
105
106     /* ==== Configures IIC3 ==== */
107     io_iic3_init();
108
109     /* ==== Transmits data in IIC3 master transmit mode ==== */
110     ack = io_iic3_eeprom_write(DEVICE_CODE,    /* Device code       */
111                     DEVICE_ADDR,          /* Device address     */
112                     0x0000,               /* Write start address */
113                     sizeof(WriteData),        /* Write data size */
114                     WriteData);           /* Buffer storing data       */
115
116     /* ==== Write error check ==== */
117     if( ack != E_OK ){
118       while(1){
119           /* error */
120       }
121     }
122     /* ==== Acknowledge Polling ==== */
123     while(io_iic3_ack_polling() != E_OK){
124       /* Waits until reprogramming EEPROM internally is completed */
125     }
126
127     /* ==== Receives data in IIC3 master receive mode ==== */
128     io_iic3_eeprom_read(DEVICE_CODE,        /* Device code       */
129                     DEVICE_ADDR,       /* Device address     */
130                     0x0000,            /* Read start address */
131                     sizeof(ReadData),  /* Read data size */
132                     ReadData);         /* Buffer storing data       */
133
134     /* ==== Compares the result ==== */
135     for(i=0; i<IIC3_DATA; i++){
136       if( WriteData[i] != ReadData[i] ){
137           while(1){
138               /* error */
139           }
140       }
141     }
142     while(1){
143       /* Loop */
144     }
145   }
146
```

3.4 Sample Program Listing "main.c" (4/13)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : IIC3 module configuration
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : int io_iic3_init(void);
 *-----------------------------------------------------------------------------
 * Description  : Configures IIC3
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : E_OK
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
int io_iic3_init(void)
{

  /* ---- STBCR3 ---- */
  STB.CR3.BIT._IIC3    = 0;    /* IIC3 is operating   */

  /* ---- PORT ---- */
  PFC.PBCRL4.BIT.PB13MD = 6;  /* SDA select */
  PFC.PBCRL4.BIT.PB12MD = 6;  /* SCL select */


  /* ----IIC31 module operation enabled ---- */
  IIC3.ICCR1.BIT.ICE = 1u;    /* IIC3 module is enabled to operate */
  IIC3.ICCR1.BIT.RCVD = 0u;       /* Continues the next reception */
  IIC3.ICCR1.BIT.MST = 1u;    /* Specifies master mode */
  IIC3.ICCR1.BIT.TRS = 1u;    /* Specifies transmit mode */
  IIC3.ICCR1.BIT.CKS = 8u;    /* Transfer clock rate: P-clock/256 (195 kHz) */

  /*  ---IIC bus mode register (ICMR) setting --- */
  IIC3.ICMR.BYTE = 0x30u;
          /*
                bit 7     : MLS:0 ---------------------- MSB first
                bit 6     : Reserve:0 ----------------- Reserve bit
                bits 5 to 4   : Reserve:1 ----------------- Reserve bit
                bit 3     : BCWP:0---------------------- Not set
                bits 2 to 0   : BC0:0, BC1:0,BC0:0--------- IIC format 9-bit
          */

  return(E_OK);
}

```

3.5 Sample Program Listing "main.c" (5/13)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Read data from EEPROM
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : int io_iic3_eeprom_read(unsigned char d_code,
 *              :                         unsigned char d_adr,
 *              :                         unsigned short r_adr,
 *              :                         unsigned int r_size,
 *              :                         unsigned char* r_buf);
 *-----------------------------------------------------------------------------
 * Description  : Reads the r_size bytes of data from EEPROM specified by the
 *              : device code (d_code), device address (d_adr), and stores the
 *              : read data in the buffer specified by the r_buf. Specify the EEPROM
 *              : memory address by the r_adr.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char d_code : Device code
 *              : unsigned char d_adr  : Device address
 *              : unsigned short r_adr : Read start address
 *              : unsigned int r_size  : Read data size
 *              : unsigned char* r_buf : Buffer storing the read data
 *-----------------------------------------------------------------------------
 * Return Value : ACK received: E_OK
 *              : ACK not received: E_ERR
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
int io_iic3_eeprom_read(unsigned char d_code,unsigned char d_adr,unsigned short r_adr,
                        unsigned int r_size,unsigned char* r_buf)
{
  int ack = E_OK;
  unsigned char send[4];

  send[0] = (unsigned char)(d_code|((d_adr & 0x7)<<1)|IIC_DATA_WR);
  send[1] = (unsigned char)((r_adr>>8) & 0x00ff);
  send[2] = (unsigned char)(r_adr & 0x00ff);
  send[3] = (unsigned char)(d_code|((d_adr & 0x7)<<1)|IIC_DATA_RD);

  while(IIC3.ICCR2.BIT.BBSY == 1u){
    /* Waits until the bus is released */
  }
  IIC3.ICCR1.BYTE |= 0x30u;                                    /* Sets IIC3 in */
                                                          /* master transmit mode */
  IIC3.ICCR2.BYTE = ((IIC3.ICCR2.BYTE & 0xbfu)|0x80u);      /* Issues START condition */

  ack = io_iic3_address_send(send);                          /* Transmits the 1st, 2nd, */
                                                                /* and 3rd bytes */

```

3.6 Sample Program Listing "main.c" (6/13)

```
243     if(ack == E_OK){
244       /* ACK received from the specified device */
245       IIC3.ICCR2.BYTE=((IIC3.ICCR2.BYTE & 0xbfu) | 0x80u); /* Issues START condition */
246       ack = io_iic3_data_send(send[3]);                    /* Transmits the 4th byte */
247       if(ack == E_OK){
248           io_iic3_data_receive(r_buf,r_size);                 /* Receives data */
249       }
250       else{
251           io_iic3_mst_send_end();
252       }
253     }
254     else{
255       /* ACK not received from the specified device */
256       io_iic3_mst_send_end();
257     }
258     return(ack);
259   }
260
261   /*""FUNC COMMENT""*****************************************************************
262    * ID           :
263    * Outline      : Master receive mode
264    *-------------------------------------------------------------------------------
265    * Include      : #include "iodefine.h"
266    *-------------------------------------------------------------------------------
267    * Declaration  : int io_iic3_data_receive(unsigned char* r_buf,
268    *              :                          unsigned int r_size);
269    *-------------------------------------------------------------------------------
270    * Description  : Sets IIC3 in master receive mode to receive the r_size bytes
271    *              : of data, and stores the receive data in the r_buf.
272    *              : After receiving the specified number of bytes of data is completed,
273    *              : it switches IIC3 in slave receive mode.
274    *-------------------------------------------------------------------------------
275    * Argument     : unsigned char* r_buf : Buffer storing the read data
276    *              : unsigned int r_size  : Read data size
277    *-------------------------------------------------------------------------------
278    * Return Value : Always returns E_OK
279    *-------------------------------------------------------------------------------
280    * Note         : None
281    *""FUNC COMMENT END""*************************************************************/
282   int io_iic3_data_receive(unsigned char* r_buf,unsigned int r_size)
283   {
284     int cnt, mask;
285     unsigned char dummy;
286
287     mask = get_imask();
288     set_imask(15);                    /* Interrupts are disabled */
289
```

3.7 Sample Program Listing "main.c" (7/13)

```
290     /* ==== Sets IIC3 in master receive mode (continuous reception) ==== */
291     IIC3.ICSR.BIT.TEND = 0u;        /* Clears the TEND bit */
292     IIC3.ICCR1.BIT.MST = 1u;        /* Master mode */
293     IIC3.ICCR1.BIT.TRS = 0u;        /* Receive mode */
294     IIC3.ICSR.BIT.TDRE = 0u;        /* Clears the TDRE bit */
295     IIC3.ICCR1.BIT.RCVD = 0u;           /* Receives data continuously */
296
297     /* ==== Starts receiving data (only one byte) ==== */
298     if(r_size == 1){                /* When receiving a single byte */
299       IIC3.ICCR1.BIT.RCVD = 1u;     /* Disables to receive data continuously */
300       IIC3.ICIER.BIT.ACKBT = 1u;    /* Sets the ACK bit to high */
301       dummy = IIC3.ICDRR;               /* Dummy read */
302       set_imask(mask);              /* Interrupts are enabled */
303     }
304     /* ==== Starts receiving data (more than two bytes) ==== */
305     else{
306       IIC3.ICIER.BIT.ACKBT = 0u;    /* Sets the ACK bit to low */
307       dummy = IIC3.ICDRR;               /* Dummy read */
308       set_imask(mask);              /* Interrupts are enabled */
309
310       /* ==== Reads data until the last 2 bytes ==== */
311       cnt = r_size;
312       while( cnt > 2 ){
313           /* ---- Waits until receiving one byte of data is completed ---- */
314           while(IIC3.ICSR.BIT.RDRF == 0u){
315               /* wait */
316           }
317           /* ---- Reads data ---- */
318           *r_buf++ = IIC3.ICDRR;
319           cnt--;
320       }
321
322       /* ==== Waits until receiving the last byte -1 data is completed ==== */
323       while(IIC3.ICSR.BIT.RDRF == 0u){
324           /* wait */
325       }
326       /* ==== Prepares for receiving the last byte ==== */
327       IIC3.ICCR1.BIT.RCVD = 1u;     /* Disables to receive data continuously */
328       IIC3.ICIER.BIT.ACKBT = 1u;    /* Sets the ACK bit to high */
329
330       /* ==== Reads the last byte -1 data ==== */
331       *r_buf++ = IIC3.ICDRR;
332     }
333
334     /* ==== Waits until receiving the last byte data is completed==== */
335     while(IIC3.ICSR.BIT.RDRF == 0u){
336       /* wait */
337     }
338
```

3.8 Sample Program Listing "main.c" (8/13)

```
339     /* ==== Issues STOP condition ==== */
340     /* ---- Starts issuing ---- */
341     IIC3.ICSR.BIT.STOP = 0u;        /* Clears the STOP flag */
342     IIC3.ICCR2.BYTE &= 0x3fu;           /* Issues STOP condition */
343
344     /* ---- Waits until issuing STOP condition is completed ---- */
345     while(IIC3.ICSR.BIT.STOP == 0u){
346       /* wait */
347     }
348
349     /* ==== Reads the last byte data ==== */
350     *r_buf = IIC3.ICDRR;            /* The last byte */
351
352     /* ==== Sets IIC3 in slave receive mode again ==== */
353     IIC3.ICCR1.BIT.RCVD = 0u;           /* Clears the RCVD bit */
354     IIC3.ICCR1.BYTE &= 0xcfu;           /* Slave receive mode */
355
356     return(E_OK);
357   }
358
359   /*""FUNC COMMENT""*****************************************************************
360    * ID           :
361    * Outline      : Write data to EEPROM
362    *-----------------------------------------------------------------------------
363    * Include      : iodefine.h
364    *-----------------------------------------------------------------------------
365    * Declaration  : int io_iic3_eeprom_write(unsigned char d_code,
366    *              :                          unsigned char d_adr,
367    *              :                          unsigned short w_adr,
368    *              :                          unsigned int w_size,
369    *              :                          unsigned char* w_buf);
370    *-----------------------------------------------------------------------------
371    * Description  : Writes the w_size bytes of data stored in the buffer specified
372    *              : by the w_buf to EEPROM specified by the device code d_code,
373    *              : device address d_adr. Specify the memory address of EEPROM by
374    *              : the w_adr.
375    *-----------------------------------------------------------------------------
376    * Argument     : unsigned char d_code ; I : Device code
377    *              : unsigned char d_adr  ; I : Device address
378    *              : unsigned short w_adr ; I : Write start address
379    *              : unsigned int w_size  ; I : Write data size
380    *              : unsigned char* w_buf ; O : Buffer storing the write data
381    *-----------------------------------------------------------------------------
382    * Return Value : ACK received: E_OK
383    *              : ACK not received: E_ERR
384    *-----------------------------------------------------------------------------
385    * Note         : None
386    *""FUNC COMMENT END""*************************************************************/
387   int io_iic3_eeprom_write(unsigned char d_code,unsigned char d_adr,unsigned short w_adr,
388                            unsigned int w_size,unsigned char* w_buf)
```

3.9 Sample Program Listing "main.c" (9/13)

```
389  {
390    int ack = E_OK;
391    int i;
392    unsigned char send[3];
393
394    send[0] = (unsigned char)(d_code|((d_adr & 0x7)<<1)|IIC_DATA_WR);
395    send[1] = (unsigned char)((w_adr>>8) & 0x00ff);
396    send[2] = (unsigned char)(w_adr & 0x00ff);
397
398    while(IIC3.ICCR2.BIT.BBSY == 1u){
399      /* Waits until the bus is released */
400    }
401    IIC3.ICCR1.BYTE |= 0x30u;                              /* Sets IIC3 in */
402                                                            /* master transmit mode */
403    IIC3.ICCR2.BYTE = ((IIC3.ICCR2.BYTE & 0xbfu)|0x80u);   /* Issues START condition */
404
405    ack = io_iic3_address_send(send);                      /* Transmits the first, second */
406                                                              /* and third bytes */
407
408    if(ack == E_OK){
409      /* Receives an ACK from the specified device */
410      for(i=0;i<w_size;i++){
411          ack = io_iic3_data_send(*w_buf++);           /* Transmits data */
412          if(ack == E_ERR){
413              break;
414          }
415      }
416      io_iic3_mst_send_end();
417    }
418    else{
419      /* ACK not received from the specified device */
420      io_iic3_mst_send_end();
421    }
422    return(ack);
423  }
424
```

3.10 Sample Program Listing "main.c" (10/13)

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Transmits the slave device address
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : int io_iic3_address_send(unsigned char* data);
 *-----------------------------------------------------------------------------
 * Description  : Transmits the slave device address (1 byte) and 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" (11/13)

```
462  /*""FUNC COMMENT""*****************************************************************
463   * ID           :
464   * Outline      : Transmit one byte of data
465   *-------------------------------------------------------------------------------
466   * Include      : iodefine.h
467   *-------------------------------------------------------------------------------
468   * Declaration  : int io_iic3_data_send(unsigned char data);
469   *-------------------------------------------------------------------------------
470   * Description  : Transmits the "data" as the following steps.
471   *              : 1. Waits until the ICDRT is empty
472   *              : 2. Sets the transmit data
473   *              : 3. Confirms the transmission is completed
474   *              : 4. Confirms an ACK is received
475   *-------------------------------------------------------------------------------
476   * Argument     : unsigned char data : Transmit data
477   *-------------------------------------------------------------------------------
478   * Return Value : ACK received: E_OK
479   *              : ACK not received: E_ERR
480   *-------------------------------------------------------------------------------
481   * Note         : None
482   *""FUNC COMMENT END""*************************************************************/
483  int io_iic3_data_send(unsigned char data)
484  {
485    int ack;
486
487    while(IIC3.ICSR.BIT.TDRE == 0u){
488      /* Waits until the ICDRT is empty */
489    }
490    IIC3.ICDRT = data;
491
492    while(IIC3.ICSR.BIT.TEND == 0u){
493      /* Waits until transmitting data is completed */
494    }
495    if(IIC3.ICIER.BIT.ACKBR == 0u){
496      ack = E_OK;
497    }
498    else{
499      ack = E_ERR;
500    }
501    return(ack);
502  }
503
```

3.12 Sample Program Listing "main.c" (12/13)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Issue STOP condition
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : void io_iic3_mst_send_end(void);
 *-----------------------------------------------------------------------------
 * Description  : Issues STOP condition, and switches IIC3 in slave receive mode.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void io_iic3_mst_send_end(void)
{
  IIC3.ICSR.BIT.TEND = 0u;    /* Clears the TEND flag */
  IIC3.ICSR.BIT.STOP = 0u;    /* Clears the STOP flag */
  IIC3.ICCR2.BYTE &= 0x3fu;       /* Issues STOP condition */

  while(IIC3.ICSR.BIT.STOP == 0u){
    /* Waits until the bus is released */
  }

  IIC3.ICCR1.BYTE &= 0xcfu;       /* Slave receive mode */
  IIC3.ICSR.BIT.TDRE = 0u;    /* Clears the TDRE bit */
}
```

3.13 Sample Program Listing "main.c" (13/13)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Acknowledge Polling
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : io_iic3_ack_polling
 *-----------------------------------------------------------------------------
 * Description  : This function checks if the write cycle of EEPROM is finished
 *              : or not. When the write cycle is not finished, EEPROM ignores
 *              : the input command and does not return an ACK. Make sure that
 *              : the write cycle of EEPROM is finished by this function before
 *              : accessing EEPROM. Read/Write codes to transmit upon the
 *              : Acknowledge Polling depends on the type of EEPROM. For more
 *              : information, refer to the EEPROM datasheet.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : E_OK : NOT_BUSY
 *              : E_ERR: BUSY (EEPROM is in the write cycle)
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
int io_iic3_ack_polling(void)
{
  int ack = E_OK;
  unsigned char send = (unsigned char)(DEVICE_CODE|((DEVICE_ADDR & 0x7)<<1)|IIC_DATA_WR);

  while(IIC3.ICCR2.BIT.BBSY == 1u){
    /* Waits until the bus is released */
  }
  IIC3.ICCR1.BYTE |= 0x30u;                              /* Sets IIC3 in */
                                                      /* master transmit mode */
  IIC3.ICCR2.BYTE = ((IIC3.ICCR2.BYTE & 0xbfu)|0x80u);   /* Issues START condition */

  ack = io_iic3_data_send(send);

  io_iic3_mst_send_end();     /* Issues STOP condition */

  return(ack);
}
/* 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
  SH7214 Group, SH7216 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 Record

Rev.	Date	Description	
		Page	Summary
1.00	Nov.19.10	—	First edition issued
1.01	Feb.10.12	17	Description amended 2.5 Notes for Master Receive Mode 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