To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

APPLICATION NOTE

R8C/10 Group

A Software control of I^{2}C-BUS using General-purpose Ports

1. Abstract

This application note describes a software control program of I^{2}C-BUS and its application example. This program can be also used for a control of EEPROM.

2. Introduction

A single master I^{2}C-BUS can be controlled by software using general-purpose ports.
The external pull-up resistances should be attached toP1$_2$(SDA) and P1$_3$(SCL).
Table 1 shows the functional performance of I^{2}C-BUS interface.

Table 1 Functional performance of single master I^{2}C-BUS interface

Item	Functional Performance
Communication mode	Master transmission (single master)
SCL Clock Frequency	100kHz approx.

Note 1 This is a value for a CPU clock operated at 16MHz when no interrupt is used.
When a CPU clock operates at other than 16MHz, some adjustment is necessary to set this value.

This program can also be used when operating other microcomputers within the M16C family, provided they have the same SFR (Special Function Registers) as the R8C/10 microcomputers. However, some functions may have been modified. Refer to the User's Manual for details. Use functions covered in this Application Note only after careful evaluation.

3. I^{2}C-BUS

3.1 START Condition / STOP Condition

(1) START Condition
Change SDA from high to low when SCL is high.
Later, change SCL to low.
(2) STOP Condition
Change SDA from low to high when SCL is high.
Later, change SCL to low.

Figure 1 shows a configuration of START condition generation timing, and Figure 2 shows a configuration of STOP condition generation timing.　A list of START condition / STOP condition generation timing is shown in Table 2 below.

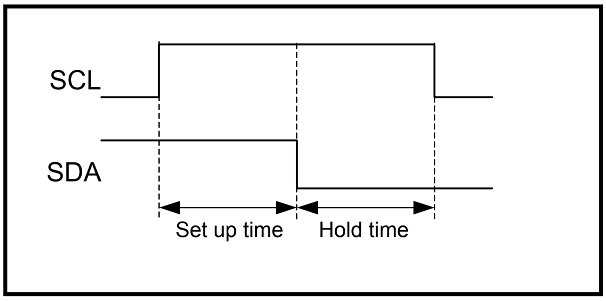


Figure 1　START condition generation timing

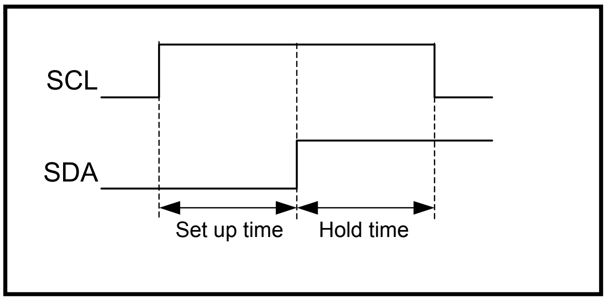


Figure 2　STOP condition generation timing

Table2　a list of START condition / STOP condition generation timing

Timing	START condition	STOP condition
Set up time	2.0μs approx.	1.6μs approx.
Hold time	3.0μs approx.	3.0μs approx.

Note 1　This is a value for a CPU clock operated at 16MHz when interrupt is not used
When a CPU clock operates at other than 16MHz, some adjustment is necessary to set this value.

3.2 Data Input / Output

(1) Data output
Data is output to SDA pin. After data setup time passes, a clock is output from SCL pin. ("L"→"H"→"L")
(2) Data input
Input data after driving SCL high, and then drive SCL low.

Figure 3 shows a configuration of data input/output timing , and Table 3 shows a list of data input/output timing.

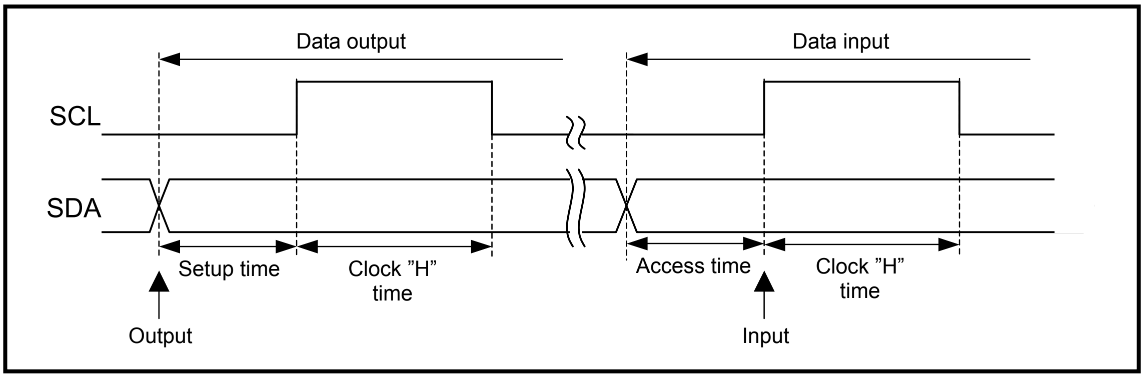


Figure 3 A configuration of data input /output timing

Table 3 A list of data input / output timing

Timing	Data output	Data input
Setup time	3.3µs approx.	-
Access time	-	Over 1µs approx.
Clock "H" time	3.0µs approx.	4.7µs approx.

Note 1 This is a value for a CPU clock operated at 16MHz when interrupt is not used
When a CPU clock operates at other than 16MHz, some adjustment is necessary to set this value.

3.3 Byte Format

1 byte consists of 8-bit-length data and 1-bit-length Acknowledge.

Acknowledge is a signal to indicate whether data is normally transferred or not. When Acknowledge indicates "L", data is normally transferred. When it is "H", data is not normally transferred.

When the master device transfers the data to the slave device, the master device releases SDA line (high-impedance) at the 9th transmit clock pulse and the slave device returns an acknowledge signal. When the master device receives the data from the slave device, the slave device releases SDA line (high-impedance) at the 9th transmit clock pulse and the master device returns an acknowledge signal.

Figure 4 shows a configuration of byte format.

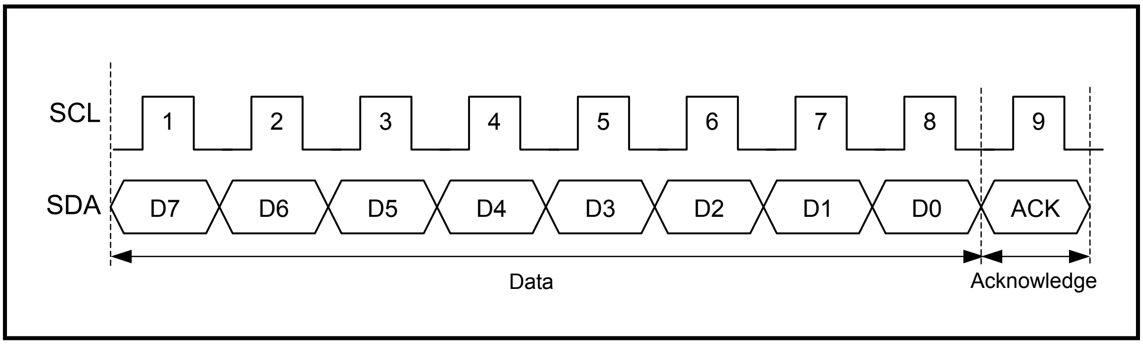


Figure 4 Byte Format

4. Application Example(a control of EEPROM)

Write / read the data to 2k-bit EEPROM(HN58X2402SI).

In 7 bit addressing mode, Device Address Code (A2,A1,A0)can be assigned by the lower 3 bit of Device Address Word.

Figure 5 shows an example of connection between a microcomputer and EEPROM(HN58X2402SI).

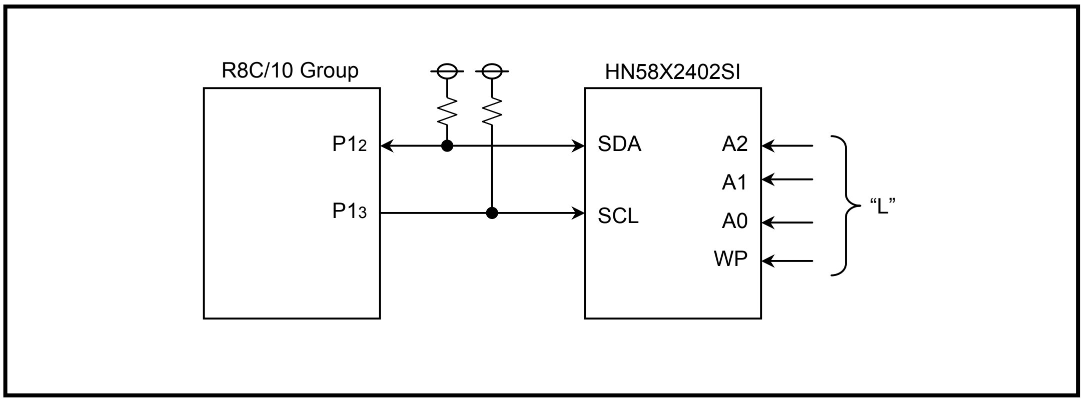
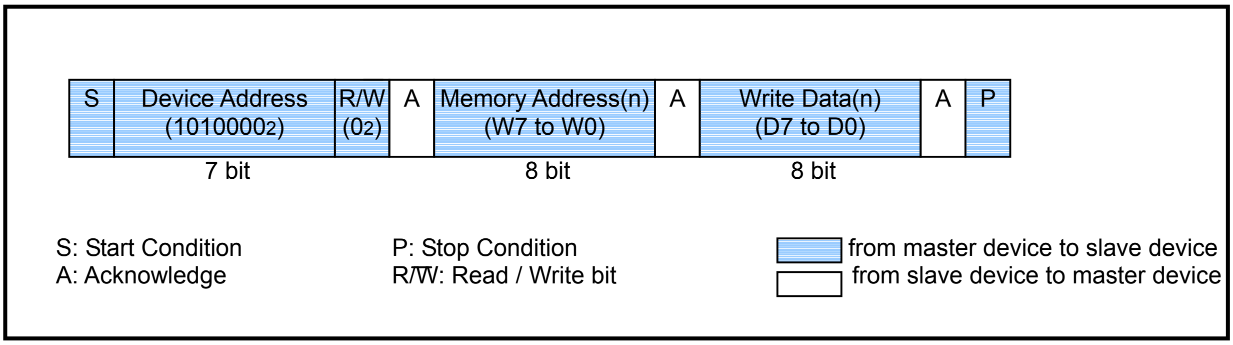


Figure 5 An example of connection

4.1 Byte Write

Write "Write Data" to an address (n) assigned to Memory Address(W7 to W0).
Confirm Acknowledge and generate Stop Condition after 8-bit Write Data is output.



S	Device Address (1010000_2)	R/W (0_2)	A	Memory Address(n) (W7 to W0)	A	Write Data(n) (D7 to D0)	A	P
	7 bit			8 bit		8 bit		

S: Start Condition
A: Acknowledge

P: Stop Condition
R/$\overline{W}$: Read / Write bit

▨ from master device to slave device
□ from slave device to master device

Figure 6 Byte Write

4.2 Page Write

Write multi-bytes (m+1) of “Write Data” to address assigned to Memory Address(W7 to W0).*
Confirm Acknowledge and generate Stop Condition after the assigned byte of “Write Data” is output.

*Page Write provides a sequential write of up to 8 byte-data.
Refer to EEPROM(HN58X2402SI) datasheet for details.

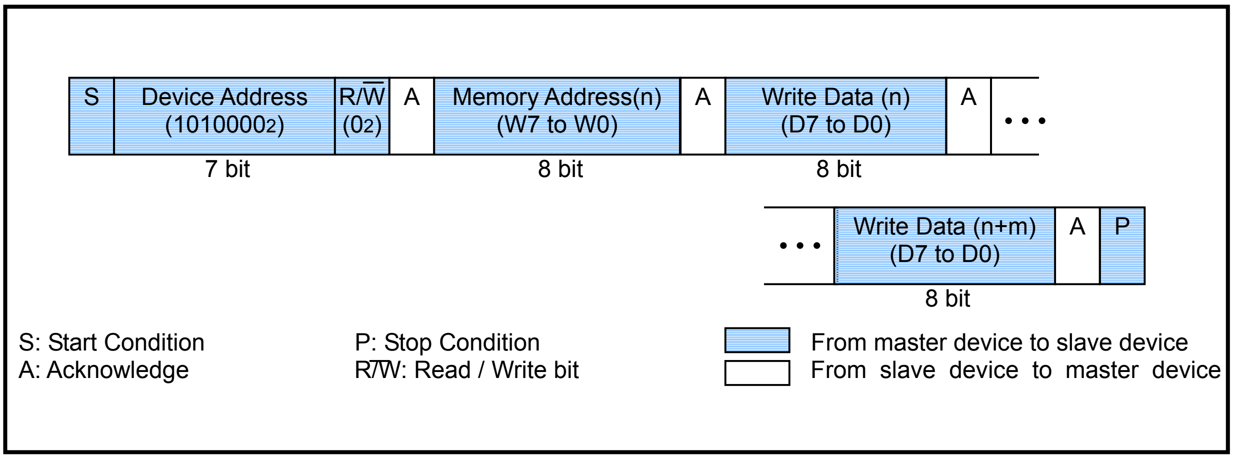

S
Device Address (10100002)
R/W̄ (02)
A
Memory Address(n) (W7 to W0)
A
Write Data (n) (D7 to D0)
A
• • •
7 bit
8 bit
8 bit
• • •
Write Data (n+m) (D7 to D0)
A
P
8 bit
S: Start Condition
A: Acknowledge
P: Stop Condition
R/W̄: Read / Write bit
From master device to slave device
From slave device to master device

Figure 7 Page Write

4.3 Sequential Read

Read “Read Data” from an address (n) assigned to Memory Address(W7 to W0).
Output Acknowledge “0” to read multi-byte (m+1) of Read Data after Read Data is input.
Output Acknowledge “1” and generate Stop Condition after the assigned byte of Read Data is input.

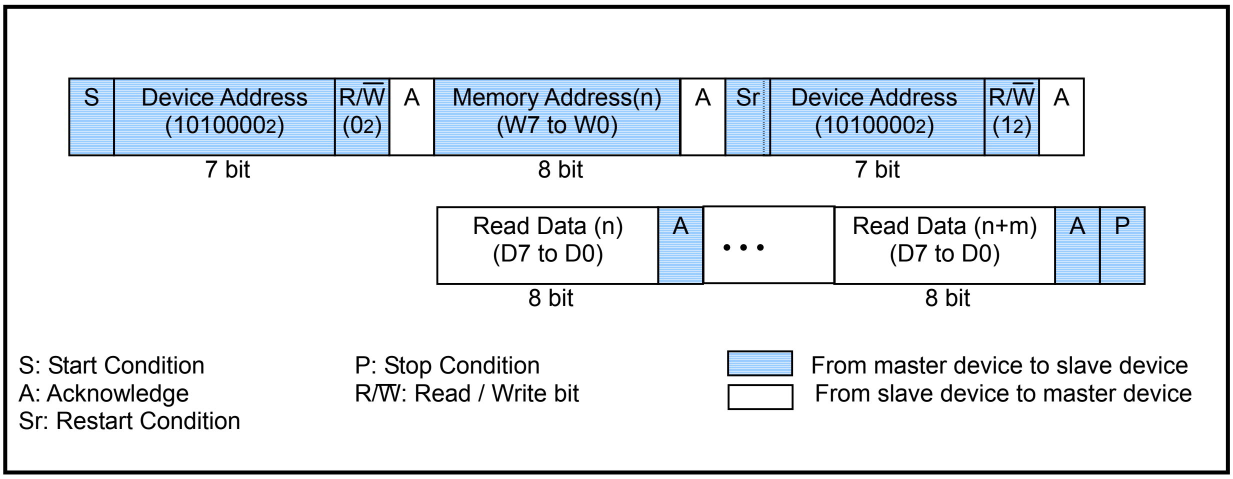
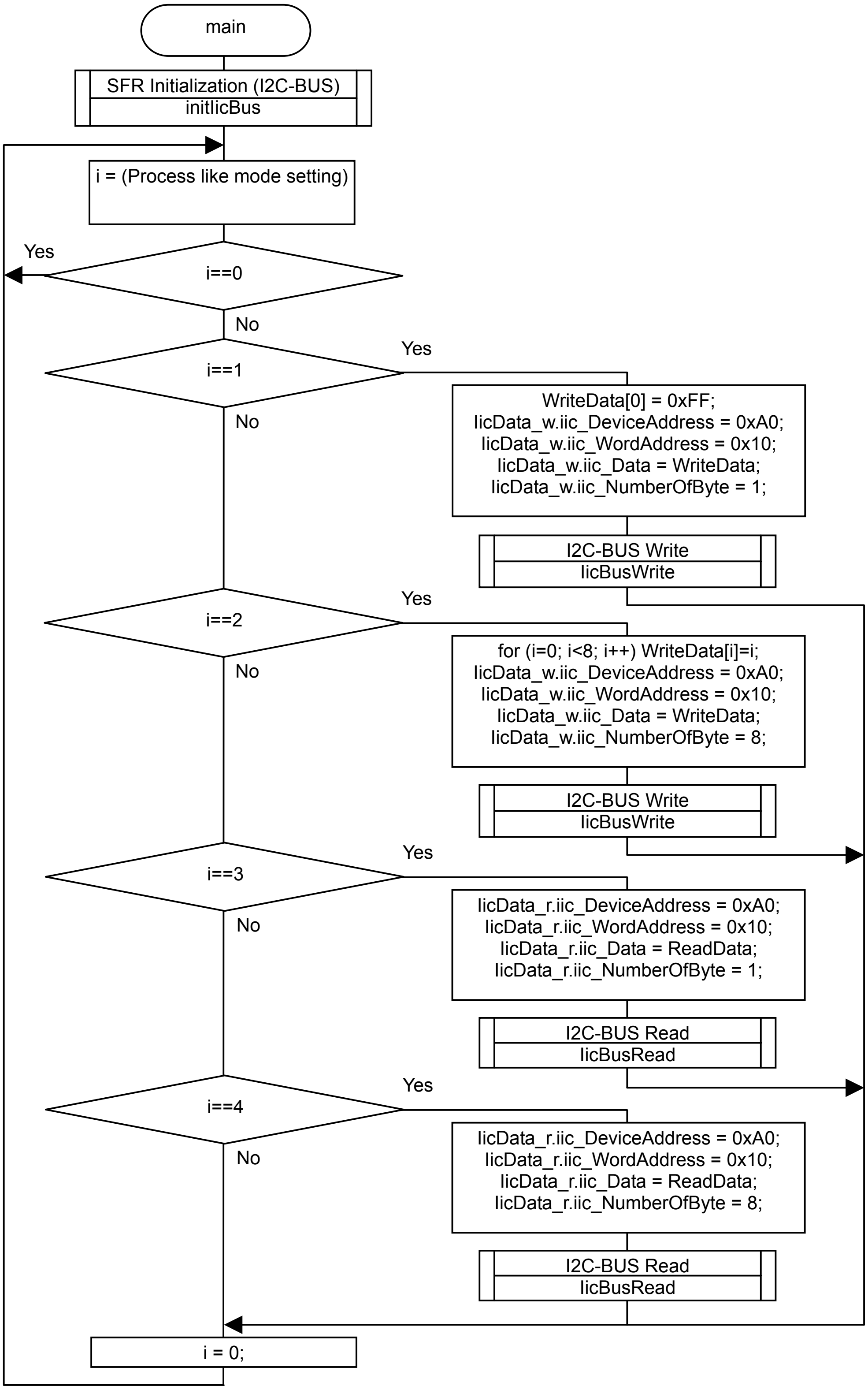


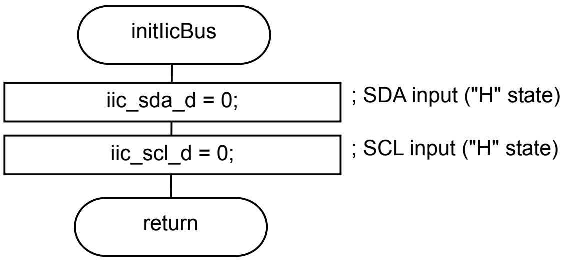
Figure 8 Sequential Read Cycle

5. Flowchart

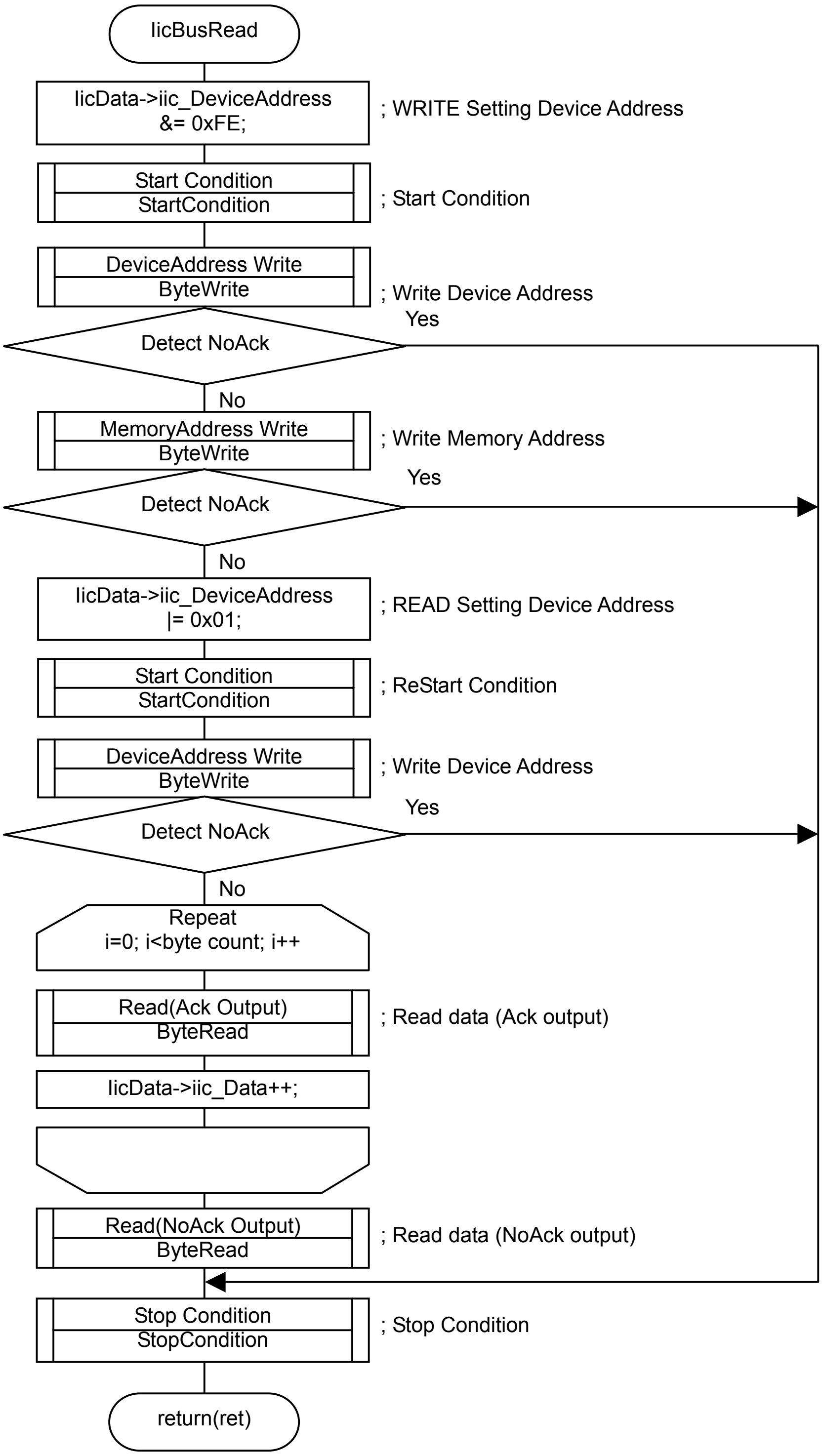
5.1 Initial Operation and Main loop


main
SFR Initialization (I2C-BUS)
initIicBus
i = (Process like mode setting)
i==0
Yes
No
i==1
Yes
No
WriteData[0] = 0xFF;
IicData_w.iic_DeviceAddress = 0xA0;
IicData_w.iic_WordAddress = 0x10;
IicData_w.iic_Data = WriteData;
IicData_w.iic_NumberOfByte = 1;
I2C-BUS Write
IicBusWrite
i==2
Yes
No
for (i=0; i<8; i++) WriteData[i]=i;
IicData_w.iic_DeviceAddress = 0xA0;
IicData_w.iic_WordAddress = 0x10;
IicData_w.iic_Data = WriteData;
IicData_w.iic_NumberOfByte = 8;
I2C-BUS Write
IicBusWrite
i==3
Yes
No
IicData_r.iic_DeviceAddress = 0xA0;
IicData_r.iic_WordAddress = 0x10;
IicData_r.iic_Data = ReadData;
IicData_r.iic_NumberOfByte = 1;
I2C-BUS Read
IicBusRead
i==4
Yes
No
IicData_r.iic_DeviceAddress = 0xA0;
IicData_r.iic_WordAddress = 0x10;
IicData_r.iic_Data = ReadData;
IicData_r.iic_NumberOfByte = 8;
I2C-BUS Read
IicBusRead
i = 0;

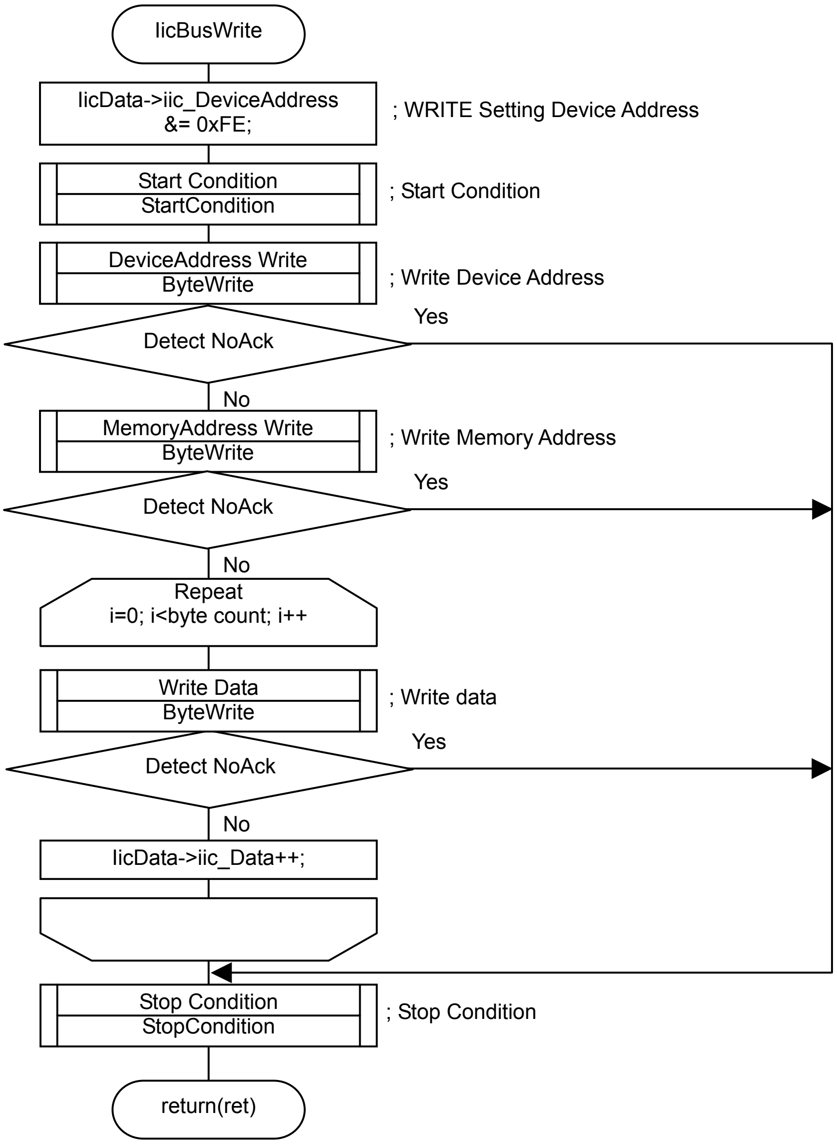
5.2 SFR Initial Setting(I^{2}C-BUS)



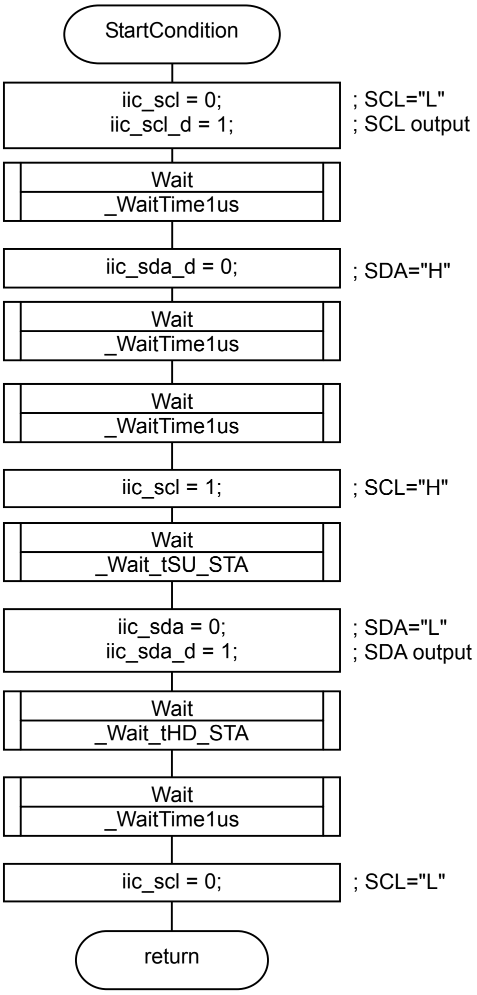
5.3 I^{2}C-BUS Read



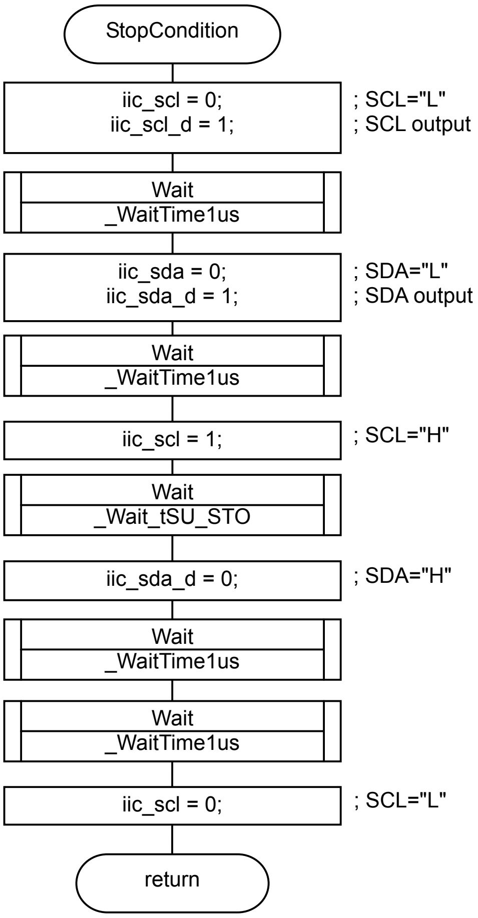
5.4 I^{2}C-BUS Write



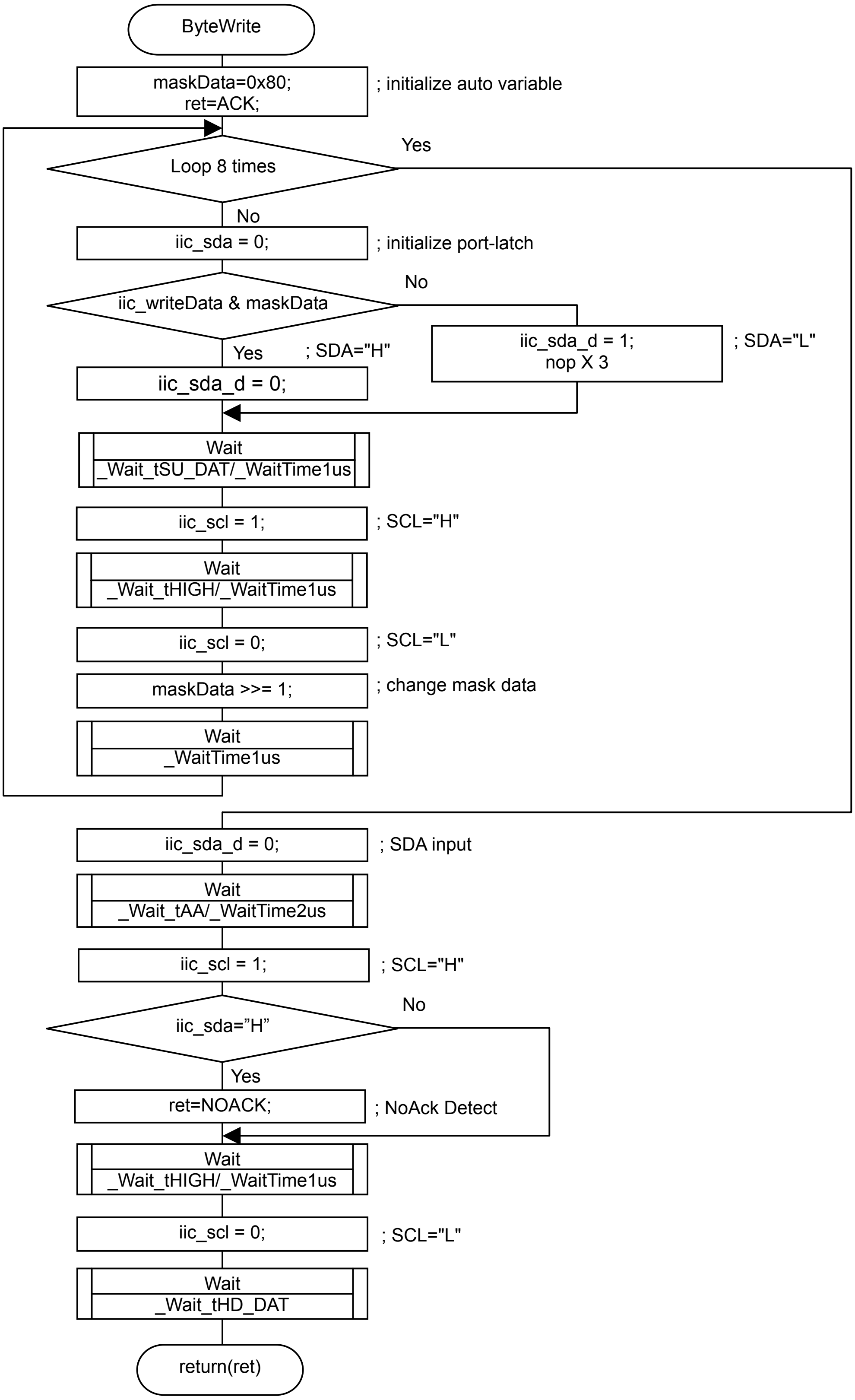
5.5 I^{2}C-BUS Start Condition



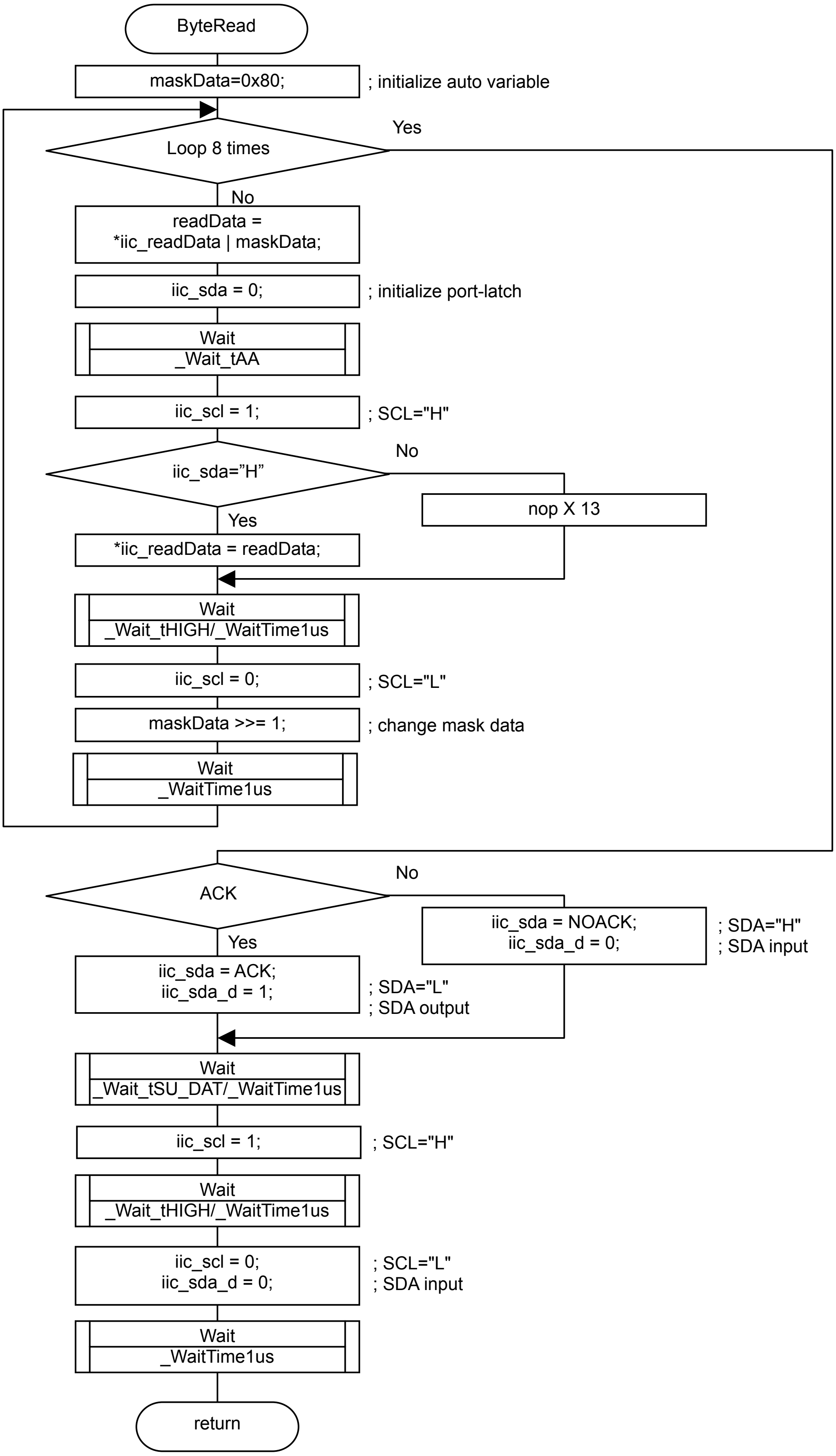
5.6 I^{2}C-BUS Stop Condition



5.7 I²C-BUS Byte Write



5.8 I^{2}C-BUS Byte Read



6. Program

```
/***************************************************************
 *                                                                          *
 *         File Name : main.c                                     *
 *         Contents  : main file                                  *
 *         Copyright : RENESAS TECHNOLOGY CORPORATION           *
 *                          AND RENESAS SOLUTIONS CORPORATION       *
 *         Version            : 1.0                                     *
 *         note               :                                         *
 *                                                                          *
***************************************************************/

#include "sfrr8c10.h"
#include "Iic_Bus.h"

void main (void)
{
    static unsigned char i=0;
    static unsigned char WriteData[8];
    static unsigned char ReadData[8];
    IicPack IicData_w;
    IicPack IicData_r;

    p1_4 = 1;                          /* test port */
    pd1_4 = 1;                         /* test port */
    p1_1 = 1;                          /* test port */
    pd1_1 = 1;                         /* test port */

    while(1){
        while(i==0) {
            i = mode();                                     /* Setting Access Mode */
        }
        p1_4 = 1;
        switch (i) {
        case 1:                                                      /* Write data 1Byte */
            WriteData[0] = 0xAA;                            /* Setting write data */
            IicData_w.iic_DeviceAddress = 0xA0;
            IicData_w.iic_MemoryAddress = 0x10;
            IicData_w.iic_Data = WriteData;
            IicData_w.iic_NumberOfByte = 1;
            p1_4 = 0;
            if (IicBusWrite(&IicData_w) == ACK) {
                p1_4 = 1;
            };
            break;
        case 3:                                                      /* Write data 8Bytes */
            for (i=0; i<8; i++) WriteData[i]=I*5;       /* Setting write data */
            IicData_w.iic_DeviceAddress = 0xA0;
            IicData_w.iic_MemoryAddress = 0x10;
            IicData_w.iic_Data = WriteData;
            IicData_w.iic_NumberOfByte = 8;
            p1_4 = 0;
}           if (IicBusWrite(&IicData_w) == ACK) {
                p1_4 = 1;
            };
            break;
        case 2:                                                     /* Read data 1Byte */
            IicData_r.iic_DeviceAddress = 0xA0;
            IicData_r.iic_MemoryAddress = 0x10;
            IicData_r.iic_Data = ReadData;
            IicData_r.iic_NumberOfByte = 1;
            p1_1 = 0;
            if(IicBusRead(&IicData_r) == ACK) {   /*   */
                p1_1 = 1;
            }
            break;
        case 4:                                                     /* Read data 8Bytes */
            IicData_r.iic_DeviceAddress = 0xA0;
            IicData_r.iic_MemoryAddress = 0x10;
            IicData_r.iic_Data = ReadData;
            IicData_r.iic_NumberOfByte = 8;
            p1_1 = 0;
            if(IicBusRead(&IicData_r) == ACK) {   /*   */
                p1_1 = 1;
            }
            break;
        default:
            asm("nop");
            break;
        }
        p1_4 = 0;
        p1_1 = 0;
        i = 0;
    }
}

void init(void)
{
    asm("fclr       i");
    prcr = 0x01;
    cm0 = 0x08;
    cm1 = 0x28;
    ocrd = 0x00;
    prcr = 0x00;
}

unsigned char mode(void)
{
    unsigned int loop;
    static unsigned char mode=0;

    for (loop=1; 0!=loop; loop++) {}                    /* about 82ms at 16MHz/1 */
    if (++mode > 4) mode=0;                             /* change mode */
    return(mode);
}
```

```
/****************************************************************
 *                                                                          *
 *          File Name : Iic_bus.h                            *
 *          Contents  : IIC Bus Definition file              *
 *          Copyright : RENESAS TECHNOLOGY CORPORATION          *
 *                               AND RENESAS SOLUTIONS CORPORATION     *
 *          Version              : 1.0                                     *
 *          note                 :                                         *
 *                                                                          *
****************************************************************/

#define ACK           0
#define NOACK         1

#define WRITE_MODE          0
#define READ_MODE           1

typedef unsigned char uchar;
typedef struct {
        unsigned char iic_DeviceAddress;
        unsigned char iic_MemoryAddress;
        unsigned char *iic_Data;
        unsigned char iic_NumberOfByte;
}IicPack;

void initIicBus(void);
unsigned char IicBusRead(IicPack *);
unsigned char IicBusWrite(IicPack *);
void StartCondition(void);
void StopCondition(void);
unsigned char ByteWrite(unsigned char);
void ByteRead (unsigned char *, unsigned char);
```

```
/**************************************************************
 *                                                                                  *
 *          File Name : Iic_bus.c                              *
 *          Contents  : IIC Bus file                                   *
 *          Copyright : RENESAS TECHNOLOGY CORPORATION          *
 *                                   AND RENESAS SOLUTIONS CORPORATION     *
 *          Version                 : 1.0                                            *
 *          note                    :                                                *
 *                                                                                  *
 **************************************************************/

#include "sfrr8c10.h"
#include "Iic_Bus.h"

#define iic_sda_d     pd1_2
#define iic_sda                  p1_2
#define iic_scl_d     pd1_3
#define iic_scl                  p1_3

void _WaitTime0us(void);
void _WaitTime1us(void);
void _WaitTime2us(void);

#define _Wait_tHIGH              _WaitTime1us()       /* Clock pulse width high */
#define _Wait_tLOW _WaitTime2us()       /* Clock pulse width low */
#define _Wait_tHD_STA          _WaitTime1us()       /* Start hold time */
#define _Wait_tSU_STA          _WaitTime1us()       /* Start setup time */
#define _Wait_tHD_DAT          _WaitTime0us()       /* Data in hold time */
#define _Wait_tSU_DAT          _WaitTime1us()       /* Data in setup time */
#define _Wait_tAA    _WaitTime1us()       /* Access time */
#define _Wait_tSU_STO          _WaitTime1us()       /* Stop setup time */
#define _Wait_tBUF  _WaitTime2us()       /* Bus free time for next mode */


/********************************************************
 Name                  : initIicBus
 Parameters          : None
 Returns  : None
 Description          : initialize I2C-BUS port
********************************************************/
void initIicBus(void)
{
      iic_sda_d = 0;                 /* SDA input ("H" state) */
      iic_scl_d = 0;                 /* SCL input ("H" state) */
}
```

```
/******************************************************
 Name                   : IicBusRead
 Parameters             : structure IicPack pointer
 Returns  : Acknowledge
 Description            : Sequential Ramdom Read Cycle (I2C-BUS)
******************************************************/
unsigned char IicBusRead(IicPack *IicData)
{
    unsigned char i,ret;

    /* Ramdom Read Cycle / Sequential Ramdom Read Cycle */
    IicData->iic_DeviceAddress &= 0xFE;            /* WRITE Setting DeviceAddress */
    StartCondition();                              /* Start Condition */
    while (1) {
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                           /* WRITE DeviceAddress */
            break;                                 /* NoAck Detect */
        if ((ret=ByteWrite(IicData->iic_MemoryAddress)) == NOACK)
                                                           /* WRITE MemoryAddress */
            break;                                 /* NoAck Detect */
        IicData->iic_DeviceAddress |= 0x01;        /* READ Setting DeviceAddress */
        StartCondition();                          /* ReStart Condition */
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                           /* DeviceAddress WRITE */
            break;                                 /* NoAck Detect */
        for (i=1; i<IicData->iic_NumberOfByte; i++) {   /* specified bytes as loop */
            ByteRead (IicData->iic_Data, ACK);     /* Read data (Ack output) */
            IicData->iic_Data++;                   /*    */
        }
        ByteRead (IicData->iic_Data, NOACK);       /* Read data (NoAck output) */
        break;
    }
    StopCondition();                               /* Stop Condition */
    return(ret);
}
```

```
/********************************************************
 Name                : IicBusWrite
 Parameters          : structure IicPack pointer
 Returns  : Acknowledge
 Description         : Byte Write or Page Write Cycle (I2C-BUS)
********************************************************/
unsigned char IicBusWrite(IicPack *IicData)
{
    unsigned char i,ret;

    /* Byte Write / Page Write */
    IicData->iic_DeviceAddress &= 0xFE;             /* WRITE Setting DeviceAddress */
    StartCondition();                                         /* Start Condition */
    while (1) {
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                                        /* WRITE DeviceAddress */
            break;                                            /* NoAck Detect */
        if ((ret=ByteWrite(IicData->iic_MemoryAddress)) == NOACK)
                                                                        /* WRITE MemoryAddress */
            break;                                            /* NoAck Detect */
        for (i=0; i<IicData->iic_NumberOfByte; i++) {         /* specified bytes as loop */
            if ((ret=ByteWrite(*(IicData->iic_Data))) == NOACK)         /* Write Data */
                break;                                                  /* NoAck Detect */
            IicData->iic_Data++;                              /*    */
        }
        break;
    }
    StopCondition();                                          /* Stop Condition */
    return(ret);
}
```

```
/*********************************************************
 Name                 : StartCondition
 Parameters           : None
 Returns  : None
 Description          : Output Start Condition (I2C-BUS)
 Note                 : *1 adjust a wait time
*********************************************************/
void StartCondition(void)
{
     iic_scl = 0;                              /* SCL="L" */
     iic_scl_d = 1;                            /* SCL output */
     _WaitTime1us();                           /* wait *1 */
     iic_sda_d = 0;                            /* SDA="H" */
     _WaitTime1us();                           /* wait */
     _WaitTime1us();                           /* wait *! */
     iic_scl = 1;                              /* SCL="H" */
     _Wait_tSU_STA;                                       /* wait */
     iic_sda = 0;                              /* SDA="L" */
     iic_sda_d = 1;                            /* SDA output */
     _Wait_tHD_STA;                                       /* wait */
     _WaitTime1us();                           /* wait *1 */
     iic_scl = 0;                              /* SCL="L" */
}

/*********************************************************
 Name                 : StopCondition
 Parameters           : None
 Returns  : None
 Description          : Output Stop Condition (I2C-BUS)
 Note                 : *1 adjust a wait time
*********************************************************/
void StopCondition(void)
{
     iic_scl = 0;                              /* SCL="L" */
     iic_scl_d = 1;                            /* SCL output */
     _WaitTime1us();                           /* wait *1 */
     iic_sda = 0;                              /* SDA="L" */
     iic_sda_d = 1;                            /* SDA output */
     _WaitTime1us();                           /* wait *1 */
     iic_scl = 1;                              /* SCL="H" */
     _Wait_tSU_STO;                                       /* wait */
     iic_sda_d = 0;                            /* SDA="H" */
     _WaitTime1us();                           /* wait */
     _WaitTime1us();                           /* wait *1 */
     iic_scl = 0;                              /* SCL="L" */
}
```

```
/*******************************************************
 Name               : ByteWrite
 Parameters         : Write data
 Returns  : Acknowledge
 Description        : byte data Output (I2C-BUS)
 Note               : *1 adjust a wait time
*******************************************************/
unsigned char ByteWrite(unsigned char iic_writeData)
{
    unsigned char maskData=0x80;                  /* MSB first */
    unsigned char ret=ACK;                        /* Ack/NoAck */

    while (maskData) {                                      /* 8times as loop */
        iic_sda = 0;                              /* initialize port-latch */
        if (iic_writeData & maskData) {  /* "H" output ? */
            iic_sda_d = 0;                        /* Yes SDA="H" */
        }else{
            iic_sda_d = 1;                        /* No   SDA="L" */
            asm("nop");                                     /* wait *1 */
            asm("nop");                                     /* wait *1 */
            asm("nop");                                     /* wait *1 */
        }
        _Wait_tSU_DAT;                                      /* wait */
        _WaitTime1us();                                     /* wait *1 */
        iic_scl = 1;                              /* SCL="H" */
        _Wait_tHIGH;                                        /* wait */
        _WaitTime1us();                                     /* wait *1 */
        iic_scl = 0;                              /* SCL="L" */
        maskData >>= 1;                                     /* change mask data */
        _WaitTime1us();                                     /* wait *1 */
    }
    iic_sda_d = 0;                                /* SDA input */
    _Wait_tAA;                                              /* wait */
    _WaitTime2us();                               /* wait *1 */
    iic_scl = 1;                                  /* SCL="H" */
    if (iic_sda) ret=NOACK;                       /* NoAck Detect */
    _Wait_tHIGH;                                  /* wait */
    _WaitTime1us();                               /* wait *1 */
    iic_scl = 0;                                  /* SCL="L" */
    _Wait_tHD_DAT;                                          /* wait */
    return(ret);
}
```

```
/*******************************************************
 Name                : ByteRead
 Parameters          : Read data strage location pointer, Select Ack/NoAck
 Returns   : None
 Description         : byte data input with Ack output (I2C-BUS)
 Note                : *1 adjust a wait time
*******************************************************/
void ByteRead(unsigned char *iic_readData, unsigned char ackData)
{
    unsigned char maskData=0x80;              /* MSB first */
    unsigned char readData;

    *iic_readData = 0;                                  /*    */
    while (maskData) {                                  /* 8times as loop */
        readData = *iic_readData | maskData;  /*    */
        iic_sda_d = 0;                                  /* initialize port-latch */
        _Wait_tAA;                            /* wait */
        iic_scl = 1;                          /* SCL="H" */
        if (iic_sda) {                        /* SDA="H" ? */
            *iic_readData = readData;         /* Yes    */
        }else{
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
            asm("nop");                                 /* wait *1 */
        }
        _Wait_tHIGH;                                    /* wait */
        _WaitTime1us();                                 /* wait *1 */
        iic_scl = 0;                          /* SCL="L" */
        maskData >>= 1;                                 /* Change mask data */
        _WaitTime1us();                                 /* wait *1 */
    }
    if (!ackData) {                           /* Ack output ? */
    /* Ack output */
        iic_sda = ACK;                                  /* Yes SDA="L" */
        iic_sda_d = 1;                                  /* SDA output */
    }else{
    /* NoAck output */
        iic_sda = NOACK;                      /* No    SDA="H" */
        iic_sda_d = 0;                                  /* SDA input */
    }
    _Wait_tSU_DAT;                                      /* wait */
    _WaitTime1us();                           /* wait *1 */
    iic_scl = 1;                              /* SCL="H" */
	_Wait_tHIGH;                                    /* wait */
	_WaitTime1us();                                 /* wait *1 */
	iic_scl = 0;                                    /* SCL="L" */
	iic_sda_d = 0;                                  /* SDA input */
	_WaitTime1us();                                 /* wait *1 */
}


/*******************************************************
 Name                 : _WaitTime0us
 Parameters           : None
 Returns  : None
 Description          : a 0us wait
*******************************************************/
void _WaitTime0us(void)
{
}


/*******************************************************
 Name                 : _WaitTime1us
 Parameters           : None
 Returns  : None
 Description          : a 1us wait
*******************************************************/
void _WaitTime1us(void)
{
	/* +14cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle = 16cycle */
}


/*******************************************************
 Name                 : _WaitTime2us
 Parameters           : None
 Returns  : None
 Description          : a 2us wait
*******************************************************/
void _WaitTime2us(void)
{
	/* +14cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
	asm("nop");                     /* +1cycle */
```

```
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle */
    asm("nop");                 /* +1cycle = 32cycle */
}
```

7. Reference

Hardware Manual

R8C/10 Group Hardware Manual
(Acquire the most current version from Renesas web-site)

8. Web-site and contact for support

Renesas Web-site

http://www.renesas.com/

Information on Renesas Products

Mail to : csc@renesas.com (Customer Support Center)

Contact for technical information on M16C family

Mail to:support_apl@renesas.com (M16C family MCU technical support)

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	DEC 10, 2003	-	First edition issued

Keep safety first in your circuit designs!

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

Notes regarding these materials

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