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.

SH7764 Group

Example of Single-Master Transmit/Receive Using I²C Bus Interface

Introduction

This application note presents a sample program that uses the I²C bus interface (IIC) of the SH7764 to perform single-master transmit and receive operation. Data write and read access to the EEPROM is described.

Target Device

SH7764 (Renesas Technology R0K507764E001BR)

Contents

1. Introduction

1.1 Specifications

Data transmit and receive are performed using the SH7764 as the master device and the EEPROM as the slave device. The connection is in fast mode (SCL frequency: max. 400 kHz). (The sample program uses a frequency setting of 346.15 kHz.)

Note: Select a setting that matches the specifications of the EEPROM.

1.2 Functions Used

I²C Bus Interface (IIC)

1.3 Applicable Conditions

• MCU	SH7764
• Operating frequency	CPU clock: 324 MHz SuperHyway clock: 108 MHz Peripheral clock: 54 MHz Bus clock: 108 MHz
• Integrated development environment	Renesas Technology
• C compiler	Renesas Technology SuperH RISC Engine Family C/C++ Compiler Package, Ver. 9.03, Release 00
• Compile options	High-performance Embedded Workshop default settings -cpu=sh4a -endian=little -include="$(WORKSPDIR)\inc" -object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -optimize=0 -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo

1.4 Related Application Notes

Application notes related to this application note are listed below. Please refer to them in conjunction with this application note.

SH7764 Group: *SH7764 Example of Initial Setting (RJJ05B1508)*

2. Description of Sample Program

The sample program uses the I²C bus interface (IIC) to write data from the SH7764 master device to the EEPROM slave device, then reads the data that has been written.

For a detailed description of the IIC, see *SH7764 Group Hardware Manual*.

2.1 IIC Operation

2.1.1 Features

The I²C bus interface (IIC) has the following features.

- Support for I²C bus interface
- Multi-master support
- 7- or 10-bit address compatible master
- 7-bit slave address
- Fast mode support
- Variable SCL clock frequency

Figure 1 is a block diagram of the IIC.

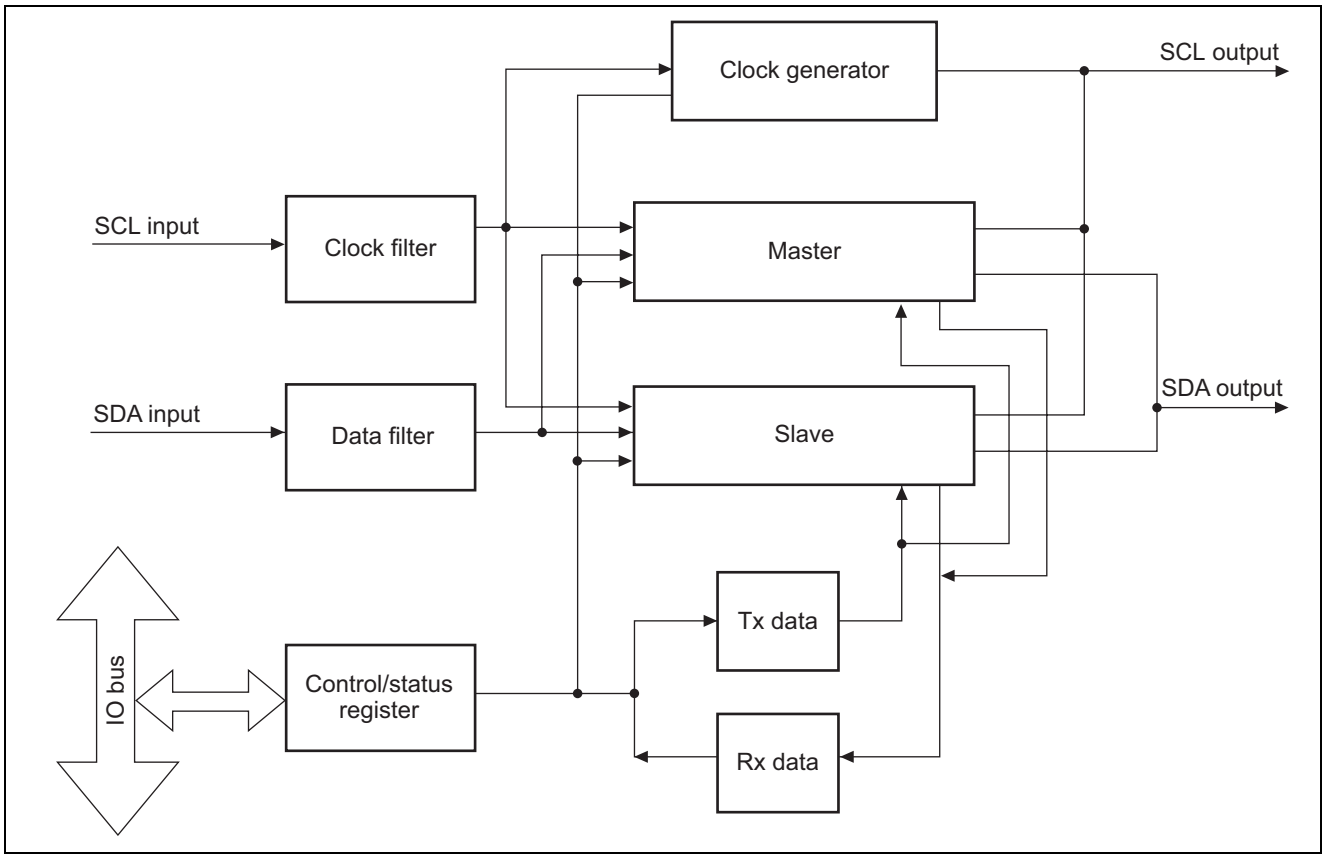

Figure 1 IIC Block Diagram

2.1.2 7-Bit Address Communication Format

Figure 2 shows the data transfer format (master data transmit format) used to transmit data from the master device to the slave device. Figure 3 shows the combined transfer format in which the data transfer direction changes during a single transfer.

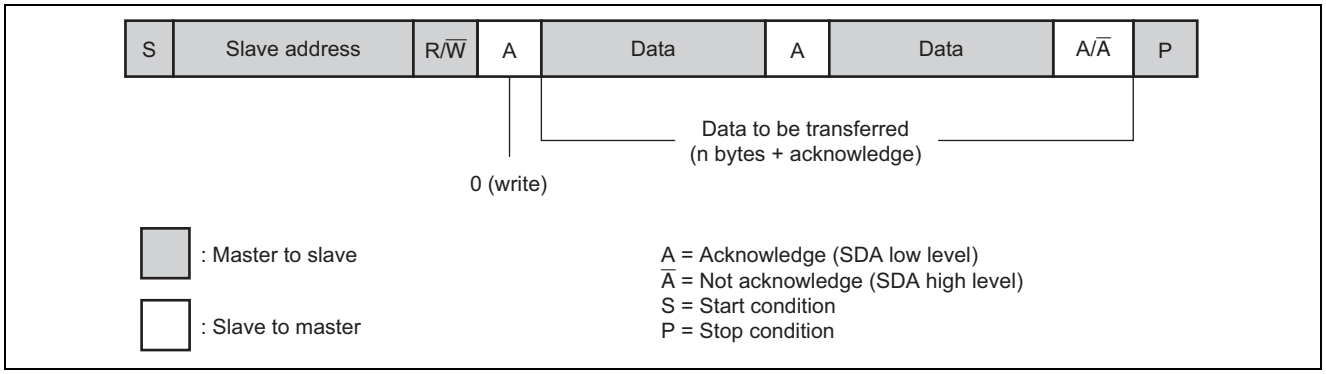

Figure 2 Master Data Transmit Format

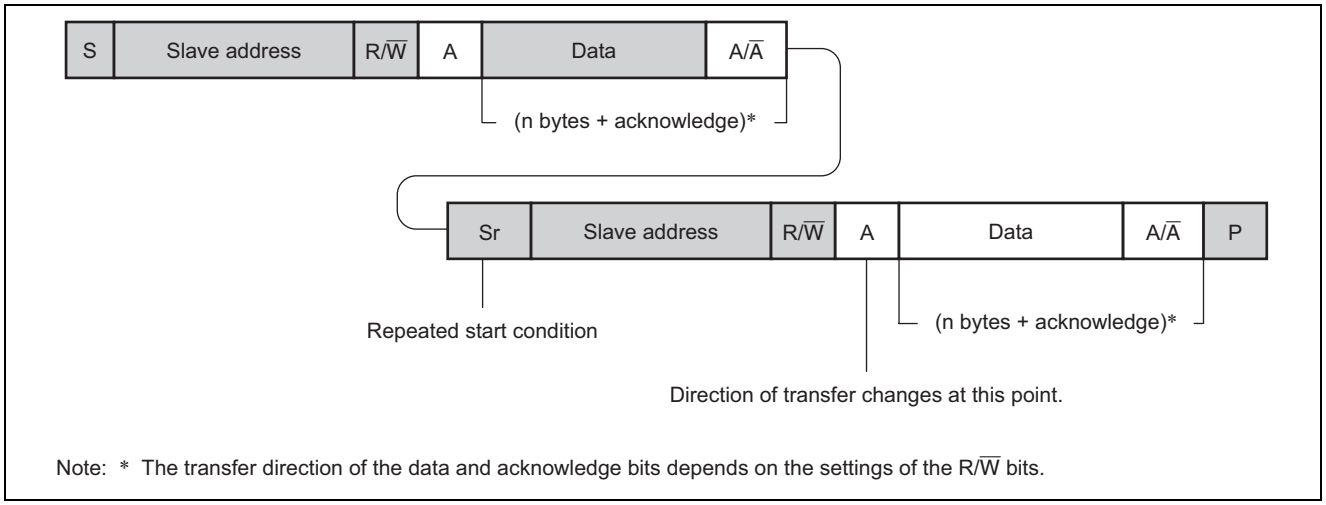

Figure 3 Combined Transfer Format of Master Transfer

2.2 Pin Connection Example

Figure 4 shows a pin connection example for the sample program.

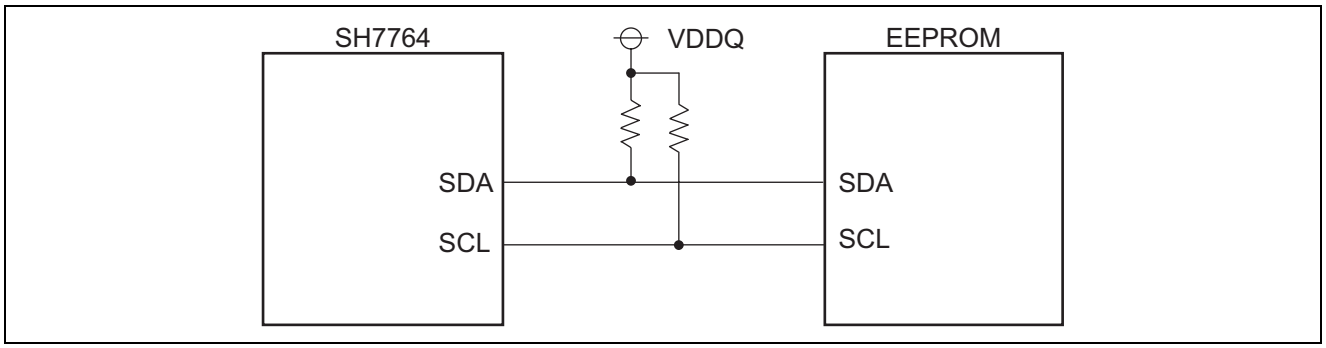

Figure 4 I²C Bus Interface Pin Connection Example

2.3 Specifications of Sample Program

The specifications of the sample program and flowcharts of its operations are presented below.

2.3.1 Specifications

- Write and read access are performed to transfer 10 bytes of data from the master, the IIC of the SH7764, to the slave, the EEPROM, and back.
- The bus is forcibly released when the IIC is initialized after the MCU is reset. (This is intended as a countermeasure against hardware noise.)
- When a not acknowledge (NACK) is detected during transmit or receive operation by the IIC, communication is suspended and the transfer operation is retried (up to five times).

The sample program uses the device code B'1010 and the device address B'000. The memory address in EEPROM is specified with one byte.

Figure 5 shows the communication format used when writing 10 bytes of data to the EEPROM, and figure 6 shows the communication format used when reading 10 bytes of data from the EEPROM.

Note: Control operation to match the specifications of the EEPROM.

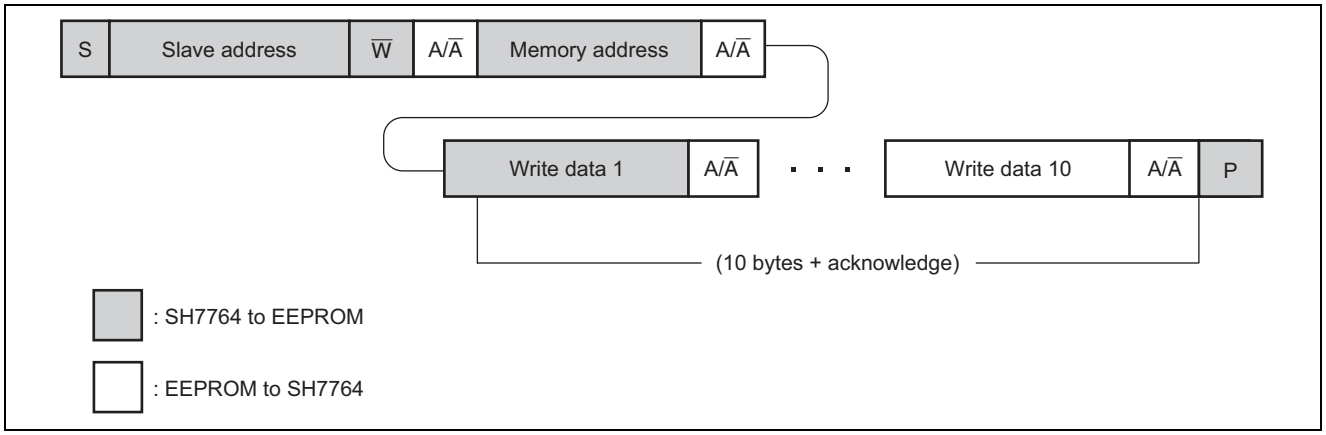

Figure 5 Communication Format When Writing 10 Bytes of Data to EEPROM

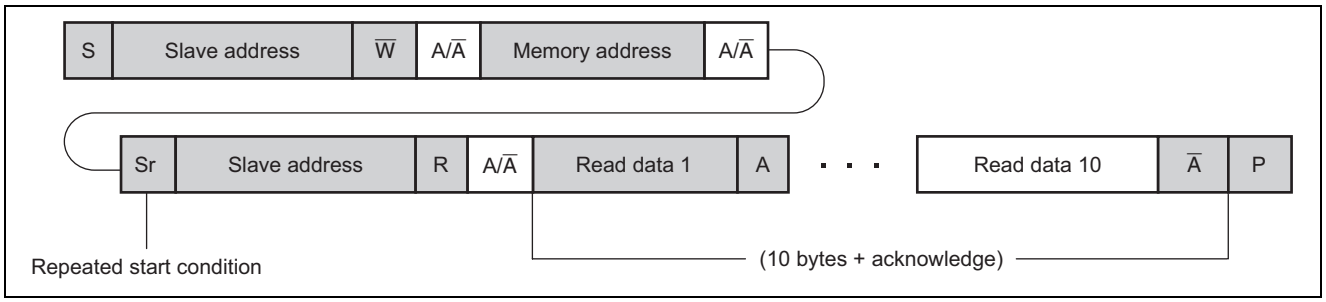

Figure 6 Communication Format When Reading 10 Bytes of Data from EEPROM

2.3.2 Main Sequence of Sample Program

Figure 7 shows the main sequence of the sample program.

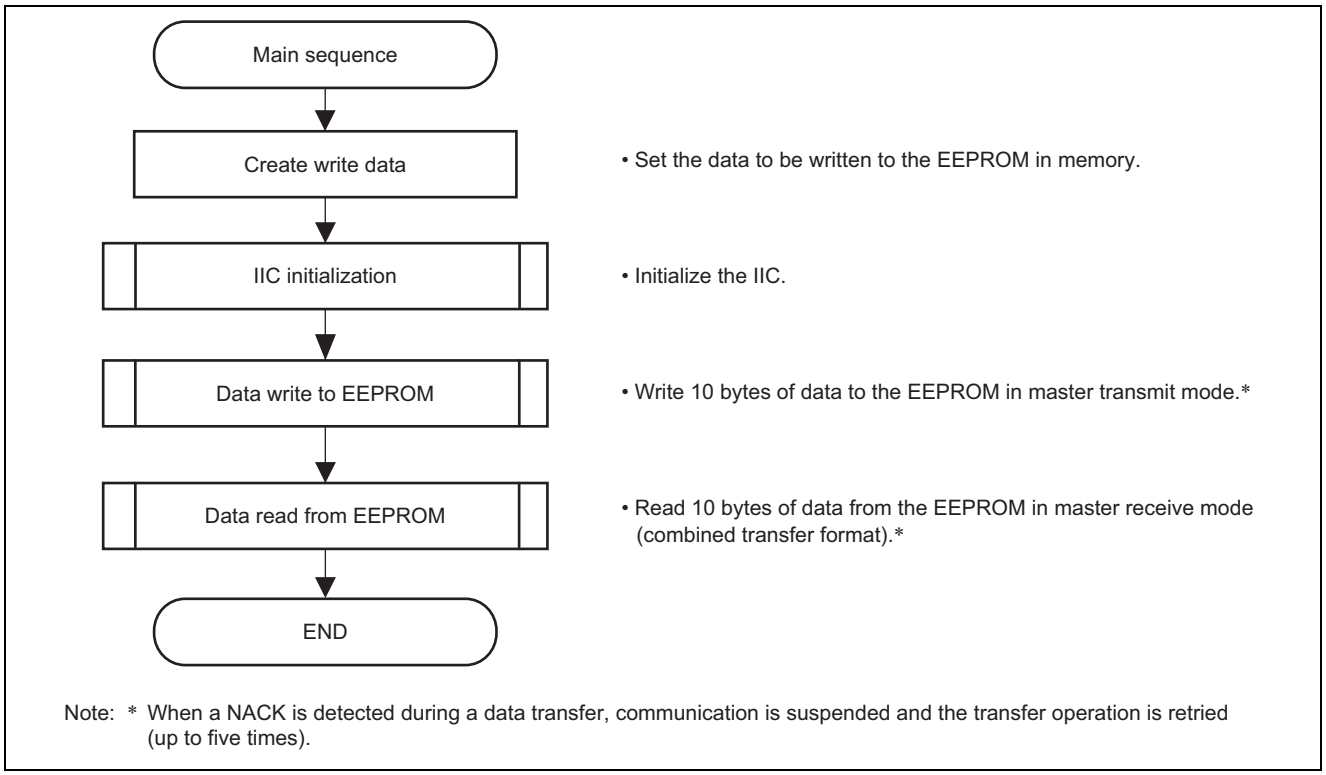

Figure 7 Main Sequence of Sample Program

2.3.3 IIC Initialization

Figure 8 shows the IIC initialization sequence.

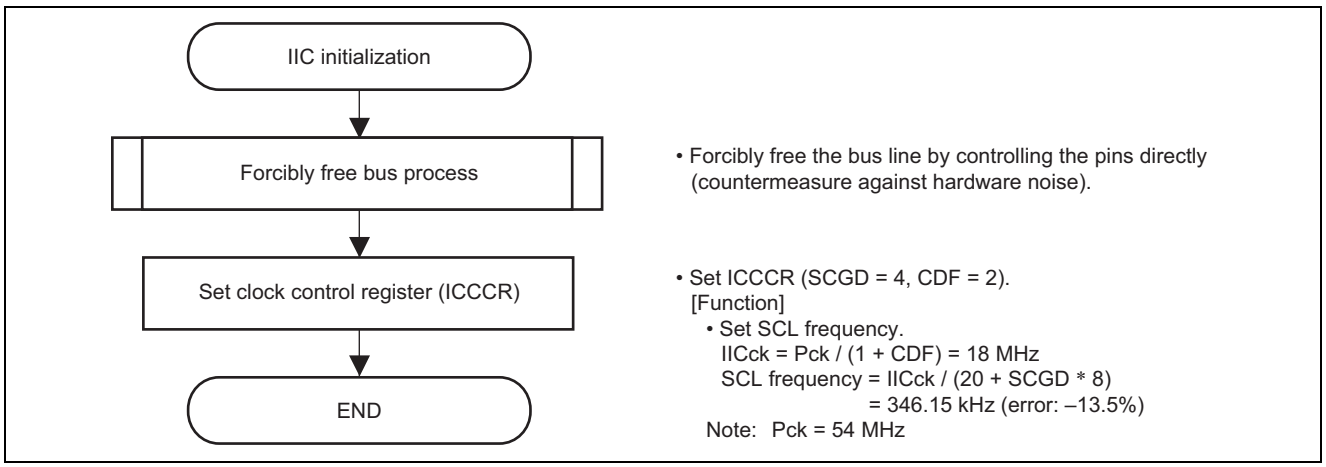

Figure 8 IIC Initialization Sequence

2.3.4 Data Write to EEPROM

Figure 9 shows an example of writing data to the EEPROM using the IIC. Continuous data writes of two or more bytes of data are supported.

Note: Control operation to match the specifications of the EEPROM.

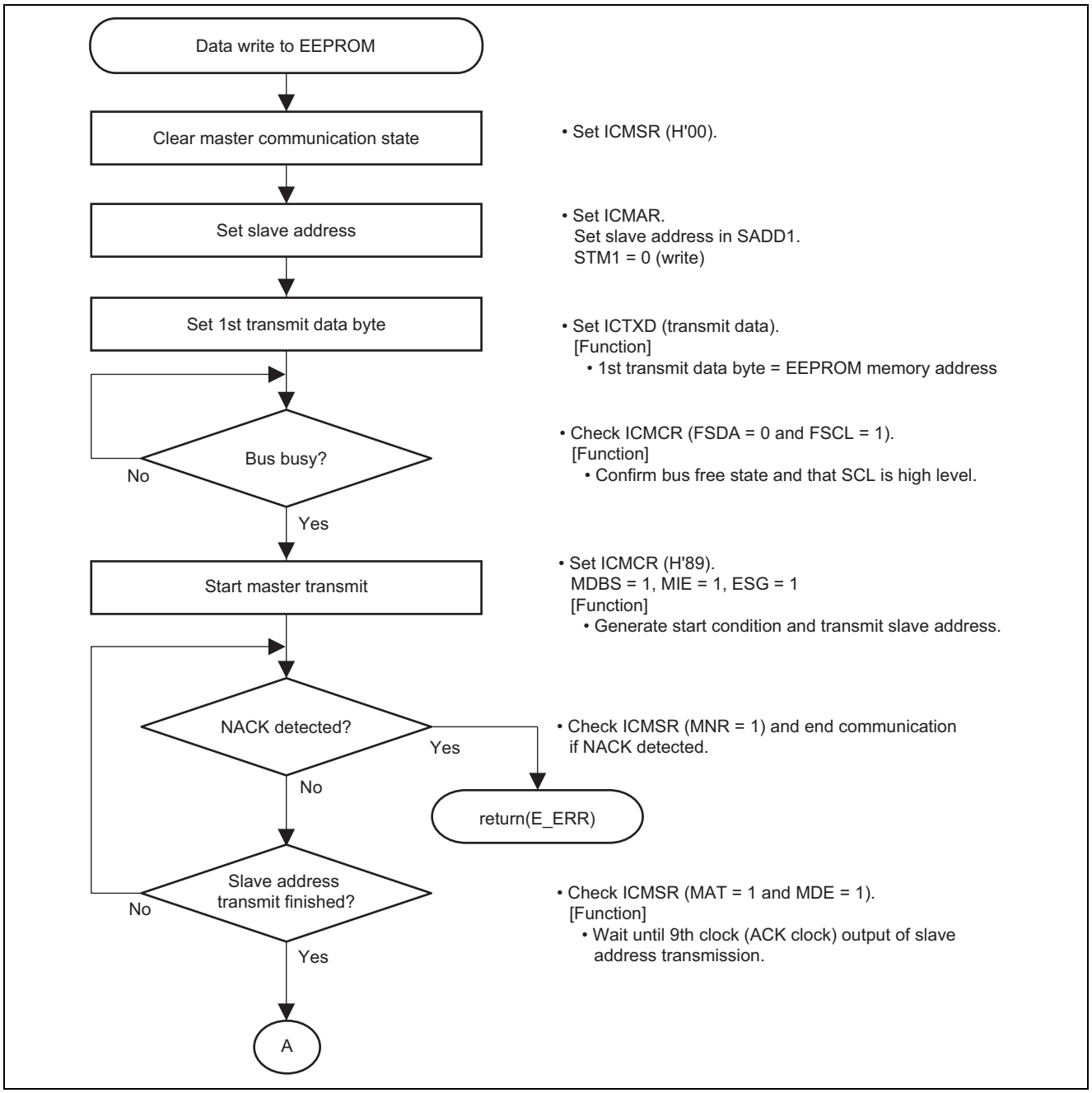

Figure 9 Data Write to EEPROM (1)

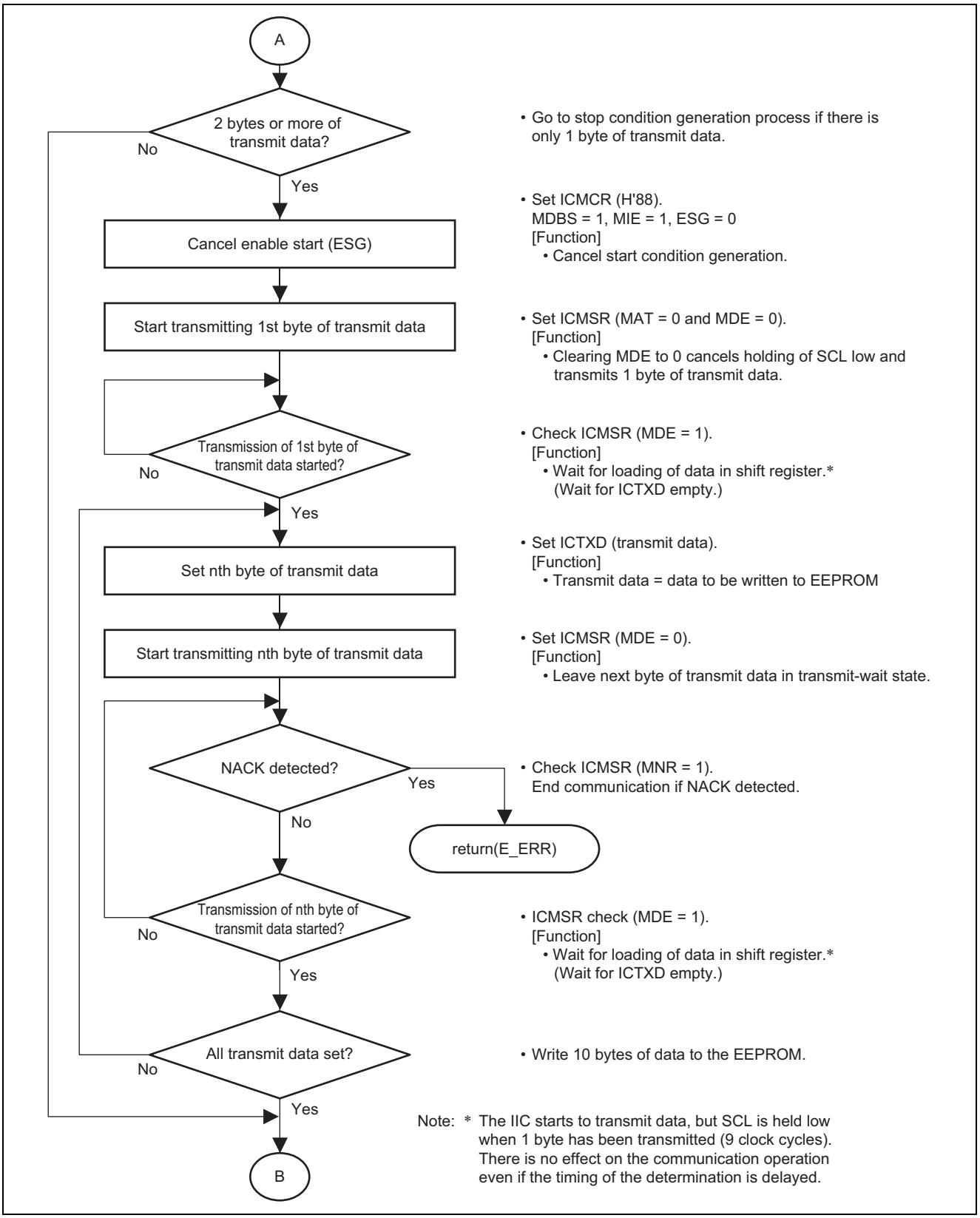

Figure 9 Data Write to EEPROM (2)

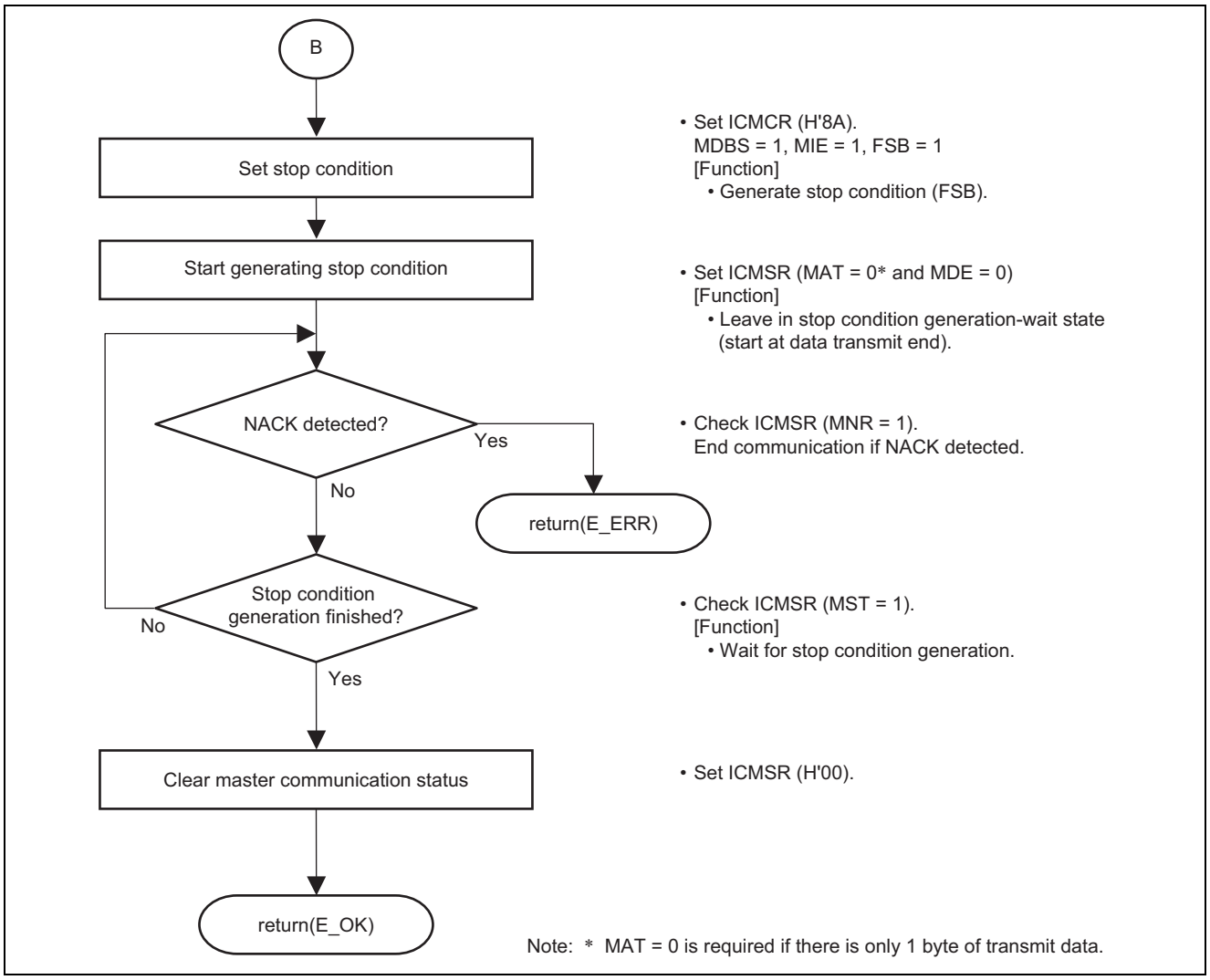

Figure 9 Data Write to EEPROM (3)

2.3.5 Data Read from EEPROM

Figure 10 shows an example of reading data from the EEPROM using the IIC. Continuous data reads of two or more bytes of data are supported.

Note: Control operation to match the specifications of the EEPROM.

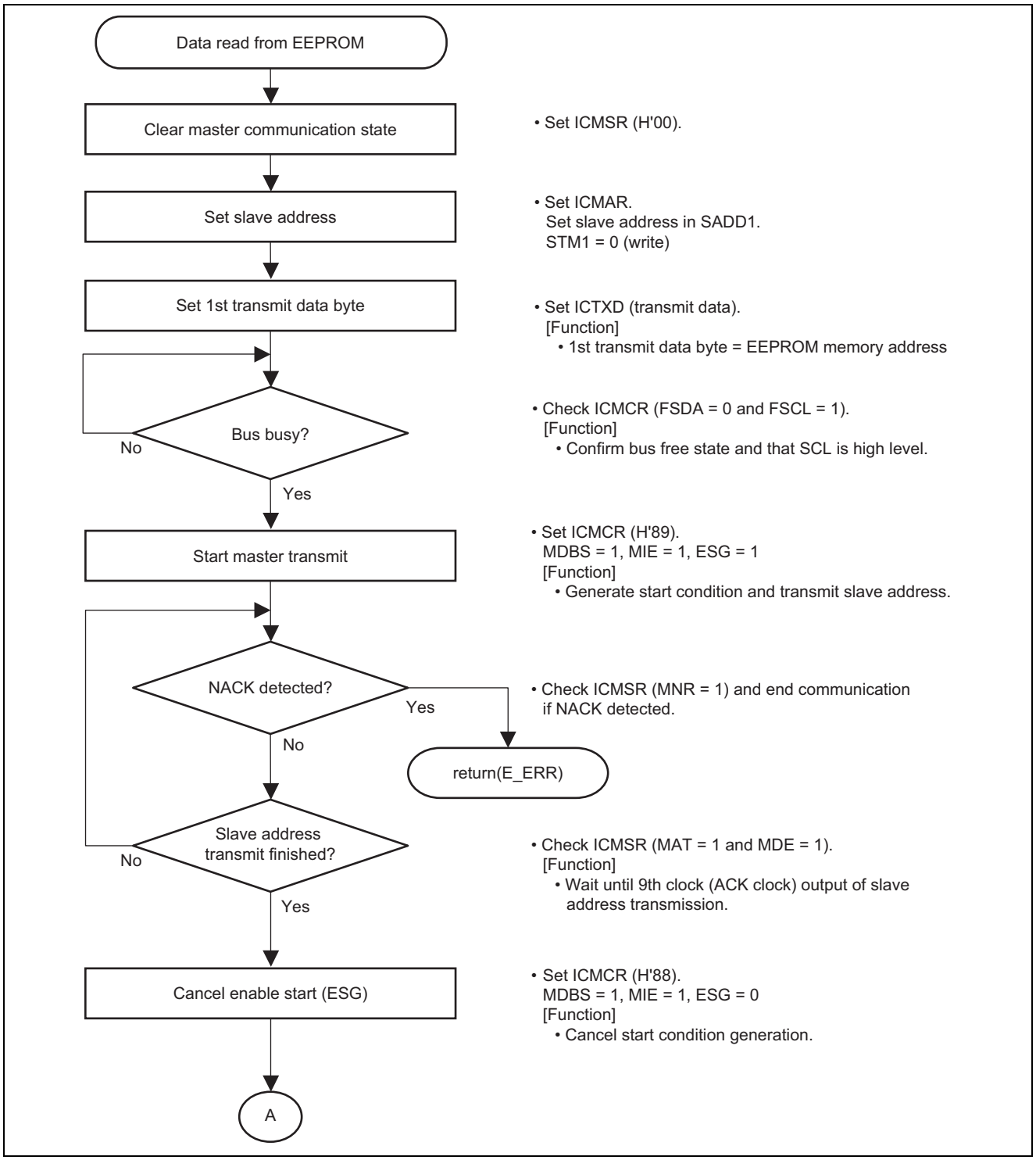

Figure 10 Data Read from EEPROM (1)

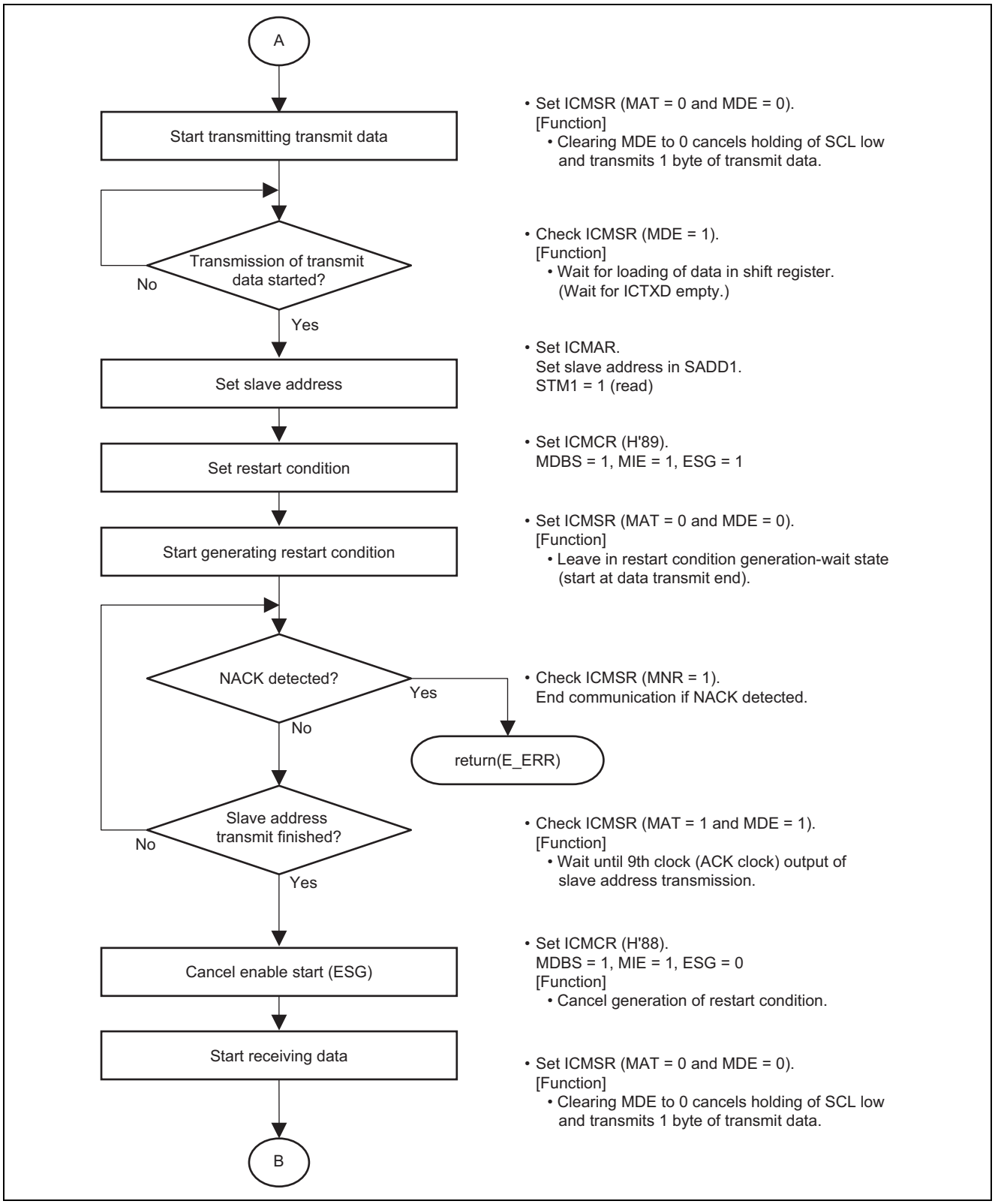

Figure 10 Data Read from EEPROM (2)

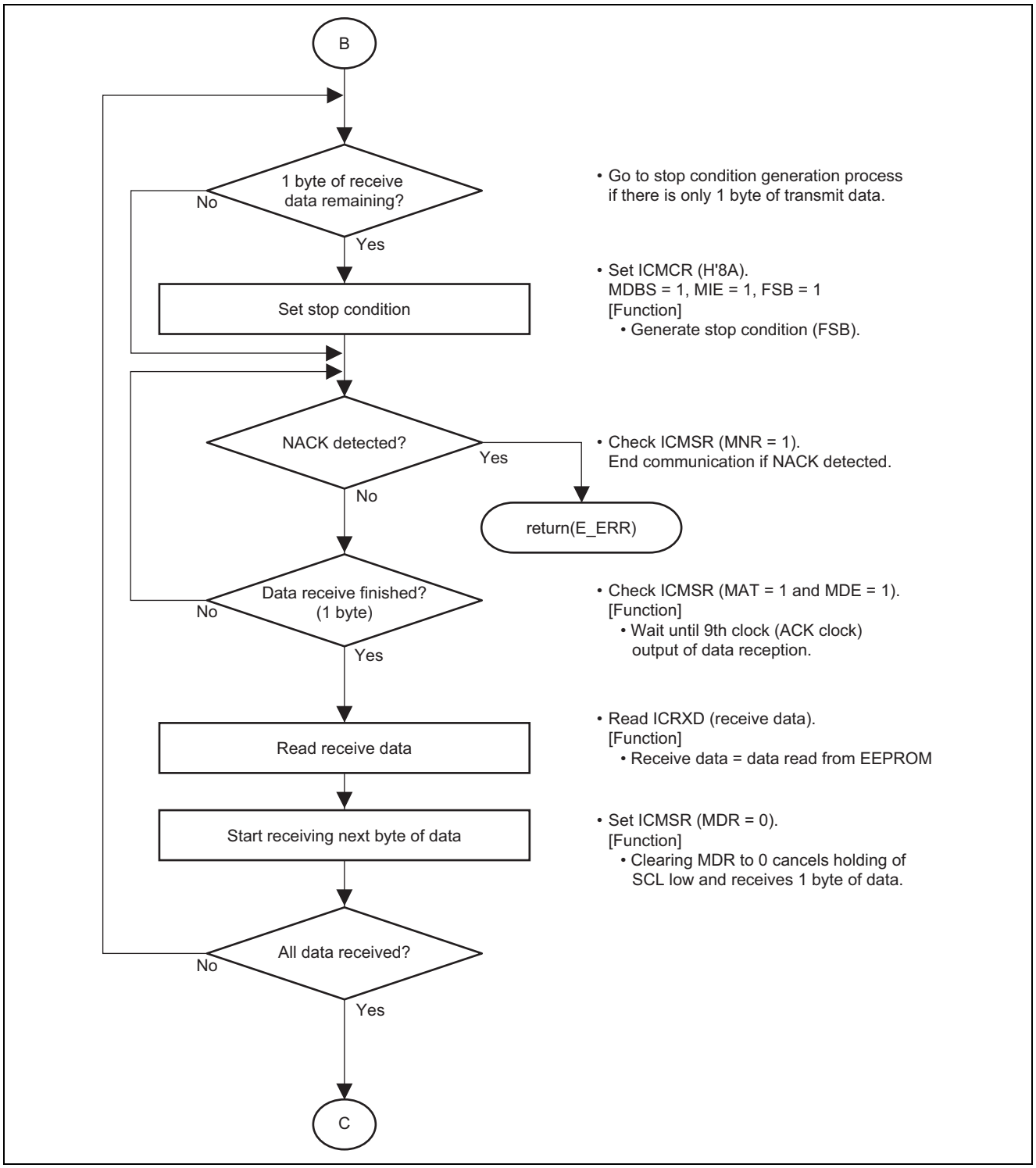

Figure 10 Data Read from EEPROM (3)

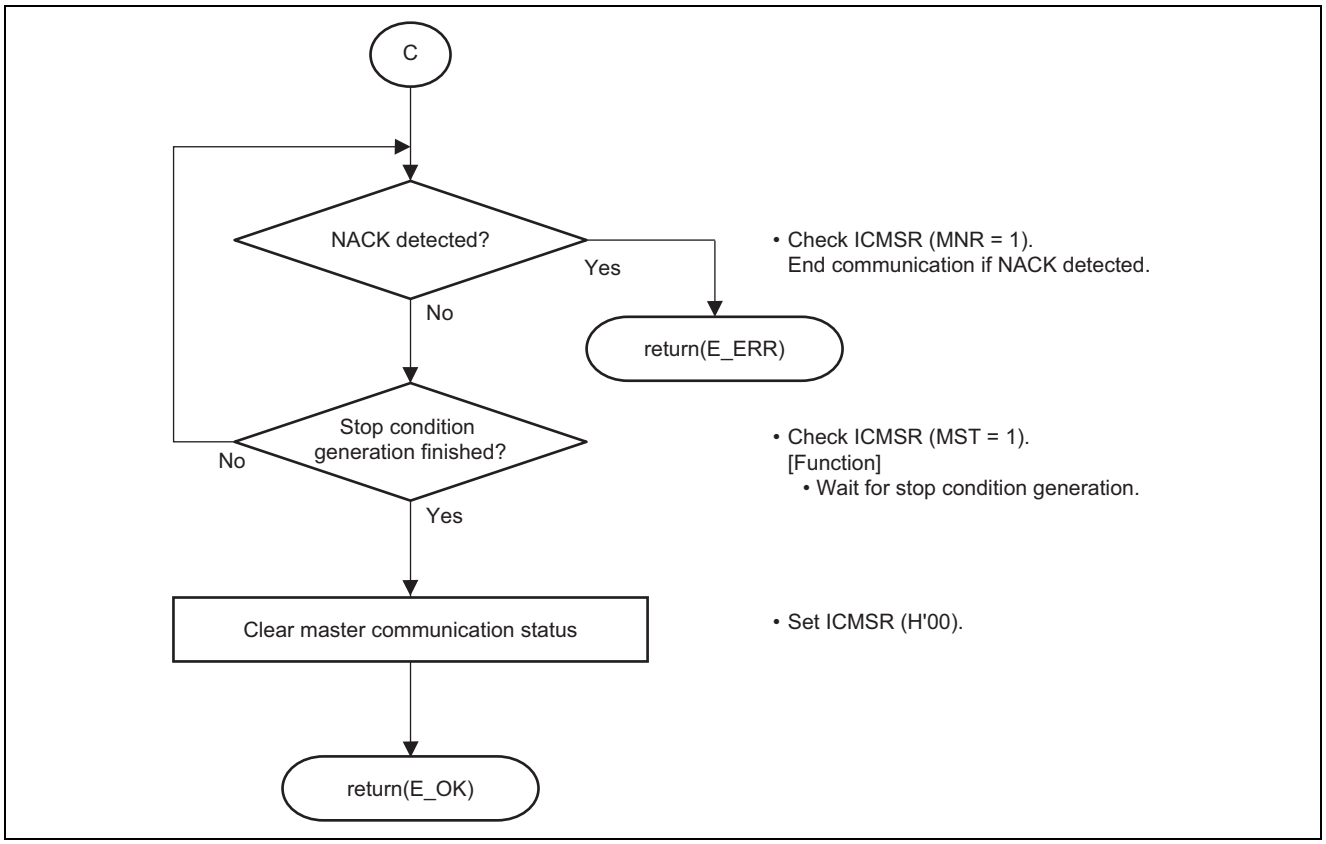

Figure 10 Data Read from EEPROM (4)

2.3.6 Communication Suspension Process

Figure 11 shows the processing sequence for suspending communication.

When a not acknowledge (NACK) is detected in response to transmission of the slave address during master transmit or receive operation by the I²C bus interface (IIC) of the SH7764, the slave address is retransmitted automatically. The retry process is repeated until an acknowledge (ACK) is returned for the slave device.

When communication is suspected, the operation of the IIC is stopped and then the bus is released. When stopping the operation of the IIC, stop automatic resending of the slave address in addition to normal operation.

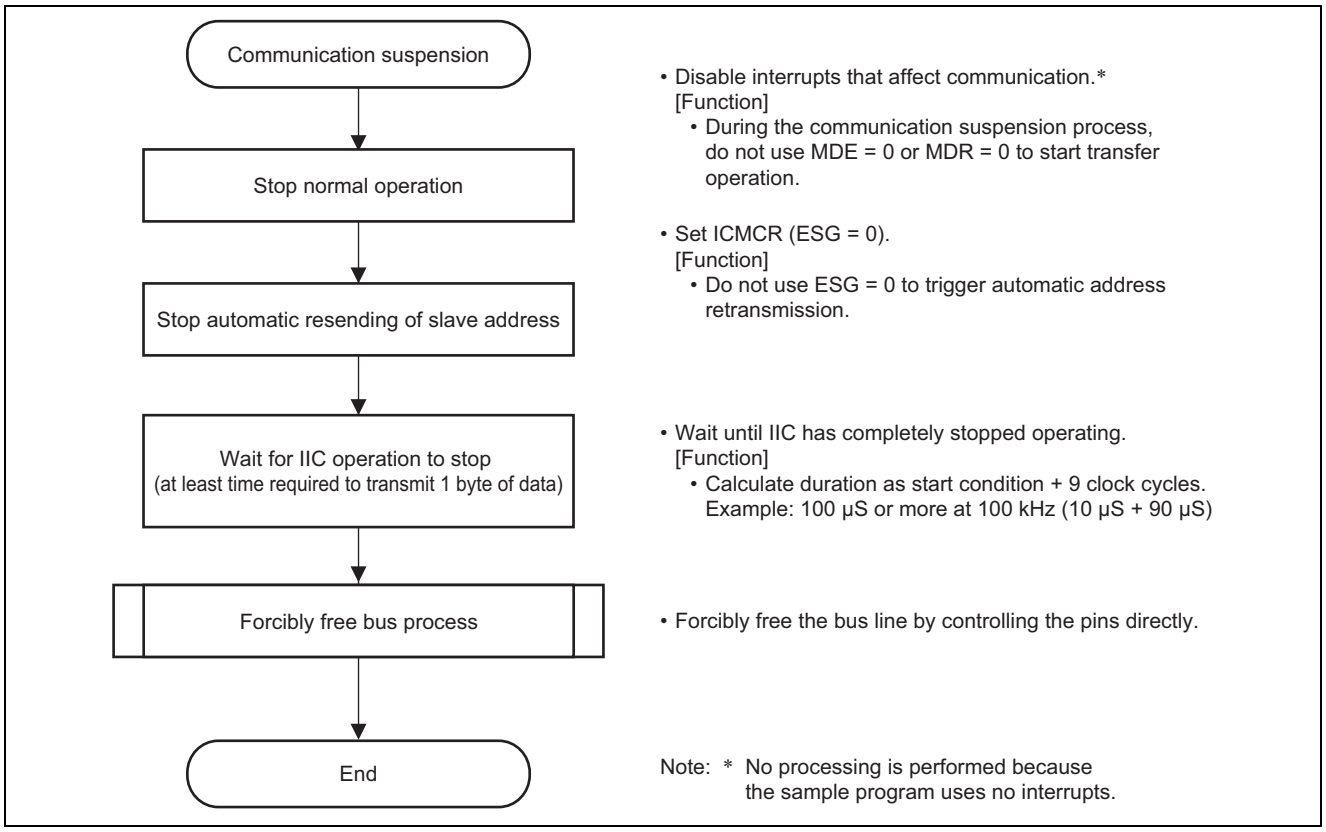

Figure 11 Communication Suspension Process Triggered by NACK

2.3.7 Forcibly Free Bus Process

Figure 12 shows the processing sequence for forcibly freeing the bus line.

With the IIC set as the master, a stop condition is forcibly generated to release the bus line. The sequence outlined here will free the bus line even if it has been put into a busy state by the effects of noise, etc. In addition, this sequence is also followed when a NACK is detected to release the bus line so the communication can be retried.

There may be cases were the IIC has released the bus but the other communication device continues to hold the bus low level. In such a case it is necessary to take measures such as a hardware reset of the other communication device.

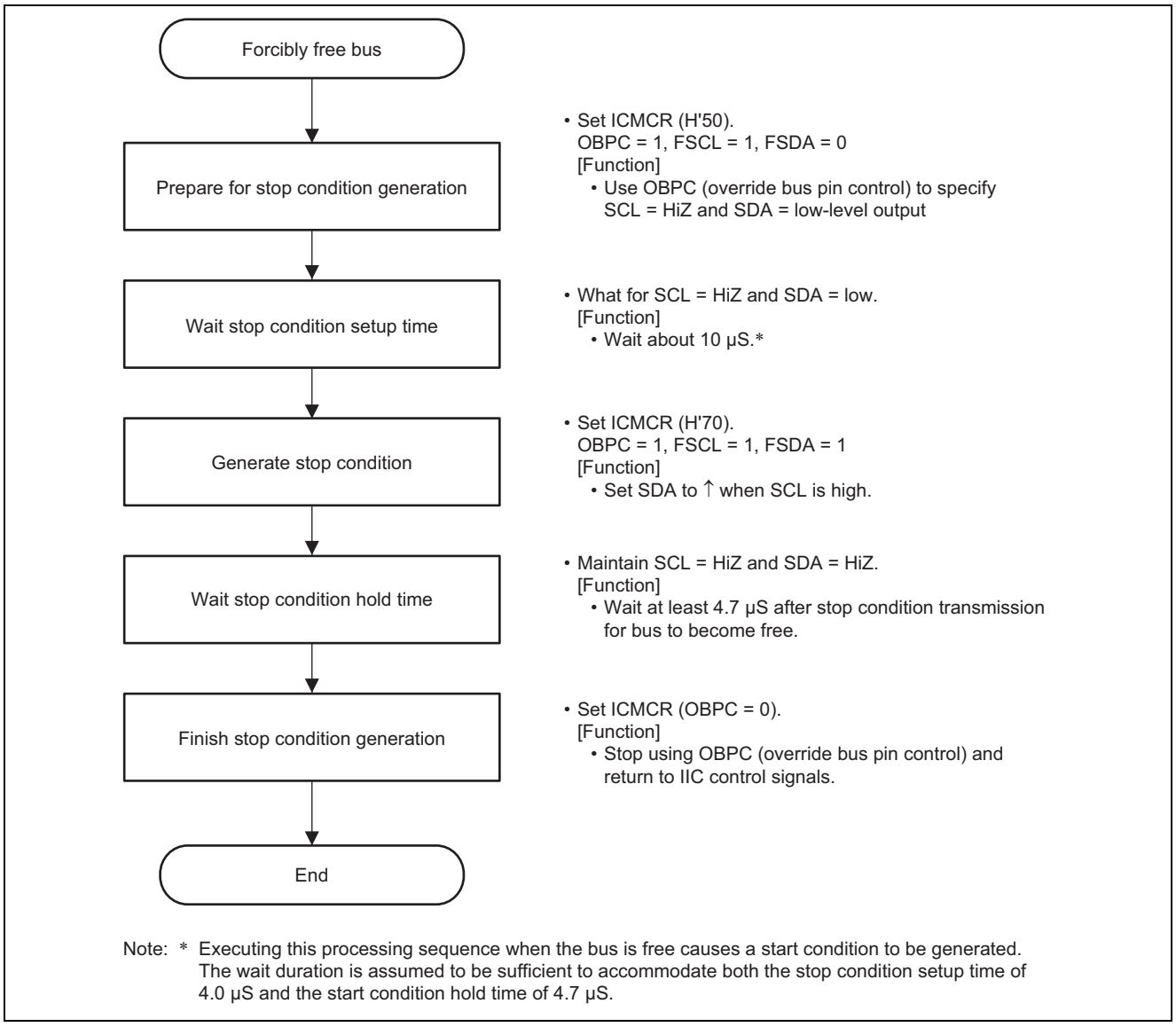

Figure 12 Forcibly Free Bus

3. Sample Program “iic.c”

3.1 Sample Program Listing: Macro Definition

```
1.  /*""FILE COMMENT""************ Technical reference data *************************
2.  *
3.  *       System Name : SH7764 Sample Program
4.  *       File Name   : iic.c
5.  *       Abstract    : I2C single-master transmit/receive
6.  *       Version     : 1.00.00
7.  *       Device      : SH7764
8.  *       Tool-Chain  : High-performance Embedded Workshop (Ver.4.05.01).
9.  *                   : C/C++ compiler package for the SuperH RISC engine family
10. *                   :                              (Ver.9.03 Release00).
11. *       OS          : none
12. *       H/W Platform: R0K507764E001BR
13. *       Disclaimer  :
14. *                     Note
15. *                     The entirety of this sample program is intended for reference,
16. *                     and its operation is not guaranteed. Customers should use this
17. *                     sample program as a technical reference when developing their
18. *                     own software.
19. *
20. *       The information described here may contain technical inaccuracies or
21. *       typographical errors. Renesas Technology Corporation and Renesas Solutions
22. *       assume no responsibility for any damage, liability, or other loss rising
23. *       from these inaccuracies or errors.
24. *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
25. *       AND Renesas Solutions Corp. All Rights Reserved
26. *
27. *       History     : Sep.01,2009 Ver.1.00.00
28. *""FILE COMMENT END""*************************************************************/
29. #include "iodefine.h"
30. /* ==== Macro definitions ==== */
31. #define      DEVICE_CODE    0xA0
32. #define      DEVICE_ADDR    0x00
33. #define      EEPROM_MEM_ADDR0x0000
34. #define RETRY_MAX           5
35. #define      E_OK           0
36. #define      E_ERR          -1
37. /* ==== Function prototype declarations ==== */
38. void iic_main(void);
39. void iic_initialize(void);
40. long iic_eeprom_write(unsigned char d_code,
41.                       unsigned char d_addr,
42.                       unsigned char w_addr,
43.                       unsigned short w_size,
44.                       unsigned char * w_buf);
45. long iic_eeprom_read(unsigned char d_code,
46.                      unsigned char d_addr,
47.                      unsigned char r_addr,
48.                      unsigned short r_size,
49.                      unsigned char * r_buf);
50. void iic_retry_func(void);
51. void iic_nack_end(void);
52. void iic_iic_bus_free(void);
53. void delay(unsigned long value);
```

3.2 Sample Program Listing: Main Process

```
/*""FUNC COMMENT""*****************************************************************
 * ID          :
 * Outline     : I2C main process
 *-------------------------------------------------------------------------------
 * Include     :
 *-------------------------------------------------------------------------------
 * Declaration : void iic_main(void);
 *-------------------------------------------------------------------------------
 * Function    : Uses the IIC's master transmit mode to perform data transmit and receive with the EEPROM.
 *-------------------------------------------------------------------------------
 * Argument    : void
 *-------------------------------------------------------------------------------
 * Return Value: void
 *""FUNC COMMENT END""*************************************************************/
void iic_main(void)
{
  unsigned char wrData[10];
  unsigned char rdData[10];
  int retry_cnt, i;

  for(i = 0; i < 10; i++)
      wrData[i] = i;

  /* ==== Initialize IIC module ==== */
  iic_initialize();

  /* ==== Write to EEPROM ==== */
  for(retry_cnt = 0; retry_cnt < RETRY_MAX; retry_cnt++){
      if(E_OK == iic_eeprom_write(DEVICE_CODE, DEVICE_ADDR, 0x0000, sizeof(wrData), wrData))
              break;

      iic_retry_func();
  }

  /* ==== Read from EEPROM ==== */
  for(retry_cnt = 0; retry_cnt < RETRY_MAX; retry_cnt++){
      if(E_OK == iic_eeprom_read(DEVICE_CODE, DEVICE_ADDR, 0x0000, sizeof(rdData), rdData))
              break;

      iic_retry_func();
  }
}
```

3.3 Sample Program Listing: IIC Initial Settings

```
/*""FUNC COMMENT""*****************************************************************
 * ID          :
 * Outline     : IIC initialization
 *-------------------------------------------------------------------------------
  * Include     :
  *-------------------------------------------------------------------------------
  * Declaration : void iic_initialize(void);
  *-------------------------------------------------------------------------------
  * Function    : Forcibly frees the bus, then sets the serial clock.
  *-------------------------------------------------------------------------------
  * Argument    : void
  *-------------------------------------------------------------------------------
  * Return Value: void
  *""FUNC COMMENT END""*************************************************************/
void iic_initialize(void)
{
    iic_iic_bus_free();

    /* ---- Set serial clock ---- */
    IIC.ICCCR.BIT._SCGD = 0x04;
    IIC.ICCCR.BIT._CDF = 0x02;
            /* I2C-clk 400kHz(346.15kHz) */
}
```

3.4 Sample Program Listing: Data Write to EEPROM

```
/*""FUNC COMMENT""*****************************************************************
 * ID          :
 * Outline     : Data write to EEPROM
 *-------------------------------------------------------------------------------
 * Include     :
 *-------------------------------------------------------------------------------
 * Declaration : int iic_eeprom_write(unsigned char d_code,
 *             :                      unsigned char d_addr,
 *             :                      unsigned char w_addr,
 *             :                      unsigned short w_size,
 *             :                      unsigned char * w_buf);,
 *-------------------------------------------------------------------------------
 * Description : Writes the number of bytes specified by w_size of
 *             : the data in the area specified by w_buf to the EEPROM
 *             : designated by device code d_code and device address d_addr.
 *             : The EEPROM memory address is specified by w_adr.
 *-------------------------------------------------------------------------------
 * Argument    : unsigned char  d_code  : device code
 *             : unsigned char  d_addr  : device address
 *             : unsigned char  w_addr  : write start address
 *             : unsigned short w_size  : write data size
 *             : unsigned char* w_buf   : write data storage location
 *-------------------------------------------------------------------------------
 * Return Value: Normal        : E_OK
 *             : NACK detected : E_ERR
 *""FUNC COMMENT END""*************************************************************/
long iic_eeprom_write(unsigned char d_code,
                      unsigned char d_addr,
                      unsigned char w_addr,
                      unsigned short w_size,
                      unsigned char * w_buf)
{
    unsigned int i;

    /* ---- Clear master status ---- */
    IIC.ICMSR.BYTE = 0x00;

    /* ---- Set slave address ---- */
    IIC.ICMAR.BYTE = (d_code | d_addr) & 0xFE;
            /* bit0(STM) = 0 transmit */

    /* ---- Set 1st transmit data byte ---- */
    IIC.ICRXD_ICTXD = w_addr;

    /* ---- Check bus busy state ---- */
    while(1){
            if((IIC.ICMCR.BIT._FSDA == 0) && (IIC.ICMCR.BIT._FSCL == 1))
                break;
    }

    /* ---- Start master transmit ---- */
    IIC.ICMCR.BYTE = 0x89;

    while(1){
            /* ---- Check NACK ---- */
            if(IIC.ICMSR.BIT._MNR)
```

```
175.                 return E_ERR;
176.             /* Slave address transmit finished? */
177.             if(IIC.ICMSR.BIT._MDE && IIC.ICMSR.BIT._MAT)
178.                 break;
179.     }
180.
181.     /* 2 bytes or more of transmit data? */
182.     if(w_size){
183.             /* ---- Cancel enable start ---- */
184.             IIC.ICMCR.BYTE = 0x88;
185.             /* ---- Start transmitting 1st byte of transmit data ---- */
186.             IIC.ICMSR.BIT._MAT = 0;
187.             IIC.ICMSR.BIT._MDE = 0;
188.             /* Start transmitting 1st byte of transmit data */
189.             while(1){
190.                 if(IIC.ICMSR.BIT._MDE)
191.                     break;
192.             }
193.             for(i = 0; i < w_size; i++){
194.                 /* ---- Set nth byte of transmit data ---- */
195.                 IIC.ICRXD_ICTXD = *w_buf;
196.                 w_buf++;
197.                 /* Start transmitting nth byte of transmit data ---- */
198.                 IIC.ICMSR.BIT._MDE = 0;
199.
200.                 while(1){
201.                     /* ---- Check NACK ---- */
202.                     if(IIC.ICMSR.BIT._MNR)
203.                         return E_ERR;
204.                     /* Check start of transmission of nth byte of transmit data */
205.                     if(IIC.ICMSR.BIT._MDE)
206.                         break;
207.                 }
208.             }
209.     }
210.
211.     /* ---- Set stop condition ---- */
212.     IIC.ICMCR.BYTE = 0x8A;
213.     /* ---- Start generating stop condition ---- */
214.     IIC.ICMSR.BIT._MDE = 0;
215.     IIC.ICMSR.BIT._MAT = 0;
216.
217.     while(1){
218.             /* ---- Check NACK ---- */
219.             if(IIC.ICMSR.BIT._MNR)
220.                 return E_ERR;
221.             /* ---- Wait for generation of stop condition ---- */
222.             if(IIC.ICMSR.BIT._MST)
223.                 break;
224.     }
225.
226.     /* ---- Clear master status ---- */
227.     IIC.ICMSR.BYTE = 0x00;
228.
229.     return E_OK;
230. }
```

3.5 Data write to Data Read from EEPROM

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Data read from EEPROM
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration : int iic_eeprom_read(unsigned char d_code,
 *              :                    unsigned char d_addr,
 *              :                    unsigned char r_addr,
 *              :                    unsigned short r_size,
 *              :                    unsigned char * r_buf);,
 *-----------------------------------------------------------------------------
 * Description : Reads the number of bytes specified by w_size of
 *              : the data in the area specified by r_buf from the EEPROM
 *              : designated by device code d_code and device address d_addr.
 *              : The EEPROM memory address is specified by r_addr.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char  d_code  : device code
 *              : unsigned char  d_addr  : device address
 *              : unsigned char  r_addr  : read start address
 *              : unsigned short r_size  : read data size
 *              : unsigned char* r_buf   : read data storage location
 *-----------------------------------------------------------------------------
 * Return Value: Normal       : E_OK
 *              : NACK detected: E_ERR
 *""FUNC COMMENT END""*************************************************************/
long iic_eeprom_read(unsigned char d_code,
                     unsigned char d_addr,
                     unsigned char r_addr,
                     unsigned short r_size,
                     unsigned char * r_buf)
{
    unsigned int i;

    /* ---- Clear master status ---- */
    IIC.ICMSR.BYTE = 0x00;

    /* ---- Set slave address ---- */
    IIC.ICMAR.BYTE = (d_code | d_addr) & 0xFE;

    /* ---- Set transmit data ---- */
    IIC.ICRXD_ICTXD = r_addr;

    /* ---- Check bus busy state ---- */
    while(1){
            if((IIC.ICMCR.BIT._FSDA == 0) && (IIC.ICMCR.BIT._FSCL == 1))
                break;
    }

    /* ---- Start master transmit ---- */
    IIC.ICMCR.BIT._MDBS = IIC.ICMCR.BIT._MIE = IIC.ICMCR.BIT._ESG = 1;

    while(1){
            /* ---- Check NACK ---- */
            if(IIC.ICMSR.BIT._MNR)
                return E_ERR;
```

```
287.
288.             /* Wait for end of slave address transmit */
289.             if(IIC.ICMSR.BIT._MDE && IIC.ICMSR.BIT._MAT)
290.                 break;
291.     }
292.
293.     /* ---- Cancel enable start ---- */
294.     IIC.ICMCR.BYTE = 0x88;
295.
296.     /* ---- Start transmitting transmit data ---- */
297.     IIC.ICMSR.BYTE = 0x00;
298.
299.     /* ---- What for ICTXD empty ---- */
300.     while(1){
301.             if(IIC.ICMSR.BIT._MDE)
302.                 break;
303.     }
304.
305.     /* ---- Set slave address ---- */
306.     IIC.ICMAR.BIT._STM1 = 1;
307.             /* bit0(STM) = 1 receive */
308.
309.     /* ---- Set restart condition ---- */
310.     IIC.ICMCR.BYTE = 0x89;
311.
312.     /* Start generating restart condition ---- */
313.     IIC.ICMSR.BYTE = 0x00;
314.
315.     while(1){
316.             /* ---- Check NACK ---- */
317.             if(IIC.ICMSR.BIT._MNR)
318.                 return E_ERR;
319.
320.             /* Wait for slave address transmit */
321.             if(IIC.ICMSR.BIT._MDR && IIC.ICMSR.BIT._MAT)
322.                 break;
323.     }
324.
325.     /* ---- Cancel enable start ---- */
326.     IIC.ICMCR.BYTE = 0x88;
327.
328.     /* ---- Start receiving data ---- */
329.     IIC.ICMSR.BYTE = 0x00;
330.
331.     for(i = 0; i < r_size; i++){
332.             if(i == (r_size - 1))
333.                 /* ---- Set stop condition ---- */
334.                 IIC.ICMCR.BYTE = 0x8A;
335.             while(1){
336.                 /* ---- Check NACK ---- */
337.                 if(IIC.ICMSR.BIT._MNR)
338.                     return E_ERR;
339.
340.                 /* Wait for data receive (1 byte) */
341.                 if(IIC.ICMSR.BIT._MDR)
342.                     break;
343.             }
```

```
344.
345.             /* ---- Read receive data ---- */
346.             *r_buf = IIC.ICRXD_ICTXD;
347.             r_buf++;
348.
349.             /* ---- Start receiving next byte of data ---- */
350.             IIC.ICMSR.BYTE = 0x00;
351.     }
352.
353.     while(1){
354.             /* ---- Check NACK ---- */
355.             if(IIC.ICMSR.BIT._MNR)
356.                 return E_ERR;
357.
358.             /* Wait for generation of stop condition */
359.             if(IIC.ICMSR.BIT._MST)
360.                 break;
361.     }
362.
363.     /* ---- Clear master status ---- */
364.     IIC.ICMSR.BYTE = 0x00;
365.
366.     return E_OK;
367. }
```

3.6 Sample Program Listing: Communication Suspension Process

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Communication suspension process triggered by NACK
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration : void iic_nack_end(void);
 *-----------------------------------------------------------------------------
 * Description : Suspends communication and releases the bus when a NACK is received.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *""FUNC COMMENT END""*************************************************************/
void iic_nack_end(void)
{
    /* ---- Stop automatic resending of slave address ---- */
    IIC.ICMCR.BIT._ESG = 0;

    /* ---- Wait for IIC operation to stop (at least time required to transmit 1 byte of data) ---- */
    delay(10000);

    /* ---- Forcibly free bus process ---- */
    iic_iic_bus_free();
}
```

3.7 Sample Program Listing: Forcibly Free Bus

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Forcibly free bus process
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration : void iic_iic_bus_free(void);
 *-----------------------------------------------------------------------------
 * Description : Forces generation of a stop condition to free the bus.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *""FUNC COMMENT END""*************************************************************/
void iic_iic_bus_free(void)
{
    /* ---- Prepare to generate stop condition ---- */
    IIC.ICMCR.BYTE = 0x50;
    /* ---- Wait 10 µS (more than 4.7 µS + 4.0 µS) ---- */
    delay(1000);
    /* ---- Generate stop condition (set SDA to HiZ) ---- */
    IIC.ICMCR.BYTE = 0x70;
    /* ---- Wait 5 µS (more than 4.7 µS) ---- */
    delay(500);
    /* ---- Finish generating stop condition (stop SCL and SDA output) ---- */
    IIC.ICMCR.BIT._OBPC = 0;
}
```

4. Reference Documents

- The I²C-Bus Specification (Version 2.1), January 2000, Phillips Semiconductor.
- Hardware Manual
  SH7764 Hardware Manual
  (The latest version can be downloaded from the Renesas Technology Web site.)
- Software Manual
  SH-4A Software Manual
  (The latest version can be downloaded from the Renesas Technology Web site.)

Website and Support

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

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

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Jan.29.10	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.