R8C/35C Group

I²C bus Single Master Control Program (Master Transmit/Receive)

R01AN0074EJ0100
Rev.1.00
Aug. 31, 2010

1. Abstract

This document describes the master transmit/receive processes in the I²C bus single master control program using the R8C/35C Group I²C bus interface.

2. Introduction

The application example described in this document applies to the following microcomputer (MCU) and parameter:

- MCU: R8C/35C Group
- XIN Clock: 20 MHz

This application note can be used with other R8C Family MCUs which have the same special function registers (SFRs) as the above group. Check the manual for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.

3. Application Example

3.1 Program Outline

Transmission is performed in 3-byte data both in master transmission and reception. Master transmission and reception are repeated alternately. This transmission procedure conforms to the I²C bus communication protocol when used under the following conditions:

- Slave address: 7 bits
- Transfer rate: Approximately 357 kHz (Standard-mode and Fast-mode supported)
- Transfer data length: 1 to 255 bytes (not including the slave address)
- Single master communication (multimaster is not supported)
- Restart condition generation is not supported.

Figure 3.1 shows the Communication Format, Figure 3.2 shows the Block Diagram, Figure 3.3 shows the Outline Flowchart, and Figure 3.4 to Figure 3.6 show Timing Diagrams.

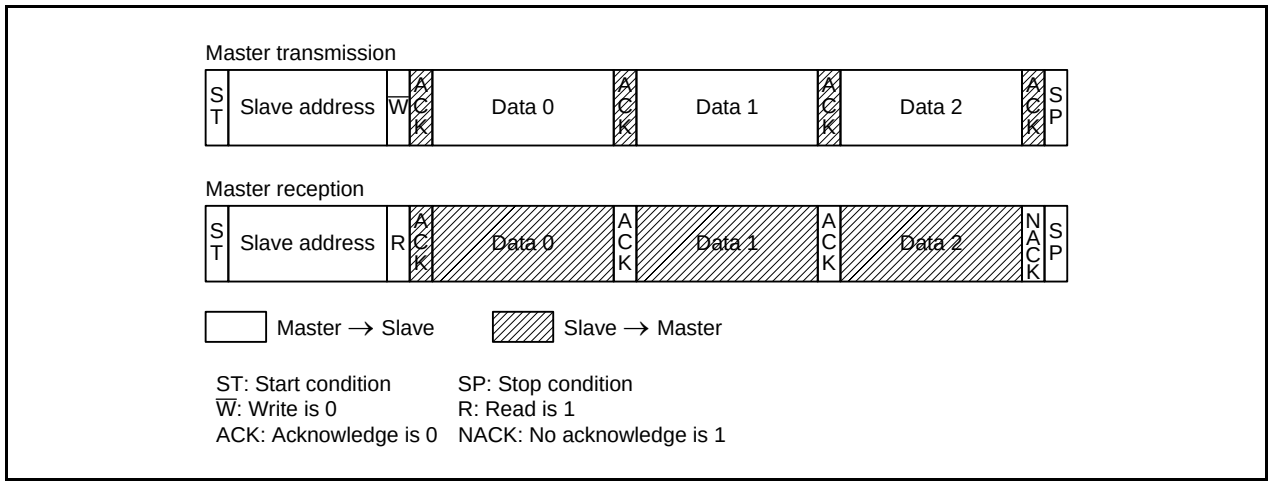


Figure 3.1 Communication Format

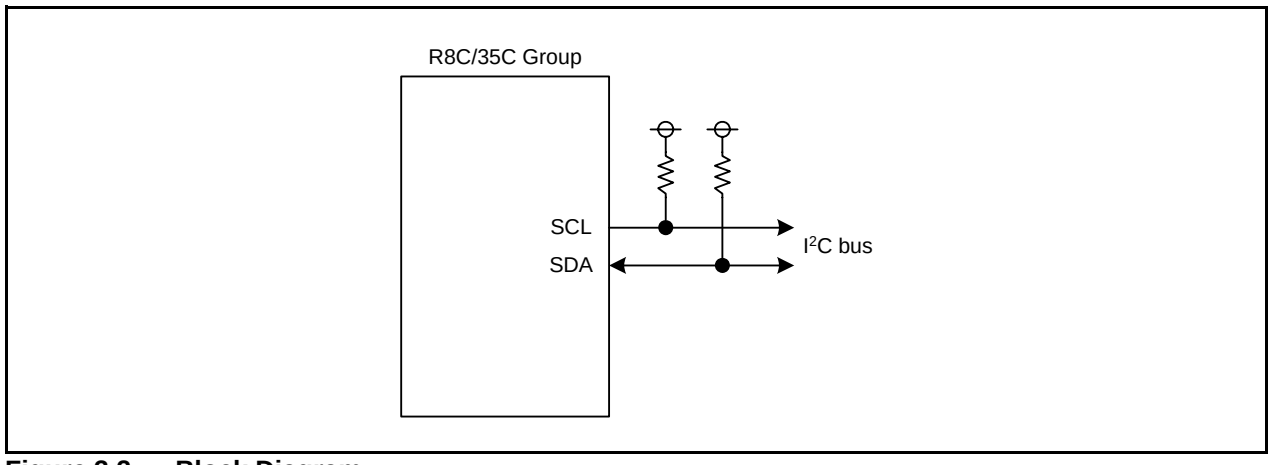


Figure 3.2 Block Diagram

The numbers in Figure 3.3 correspond to the numbers indicated in the program processing in the operating timing charts in Figure 3.4 to Figure 3.6.

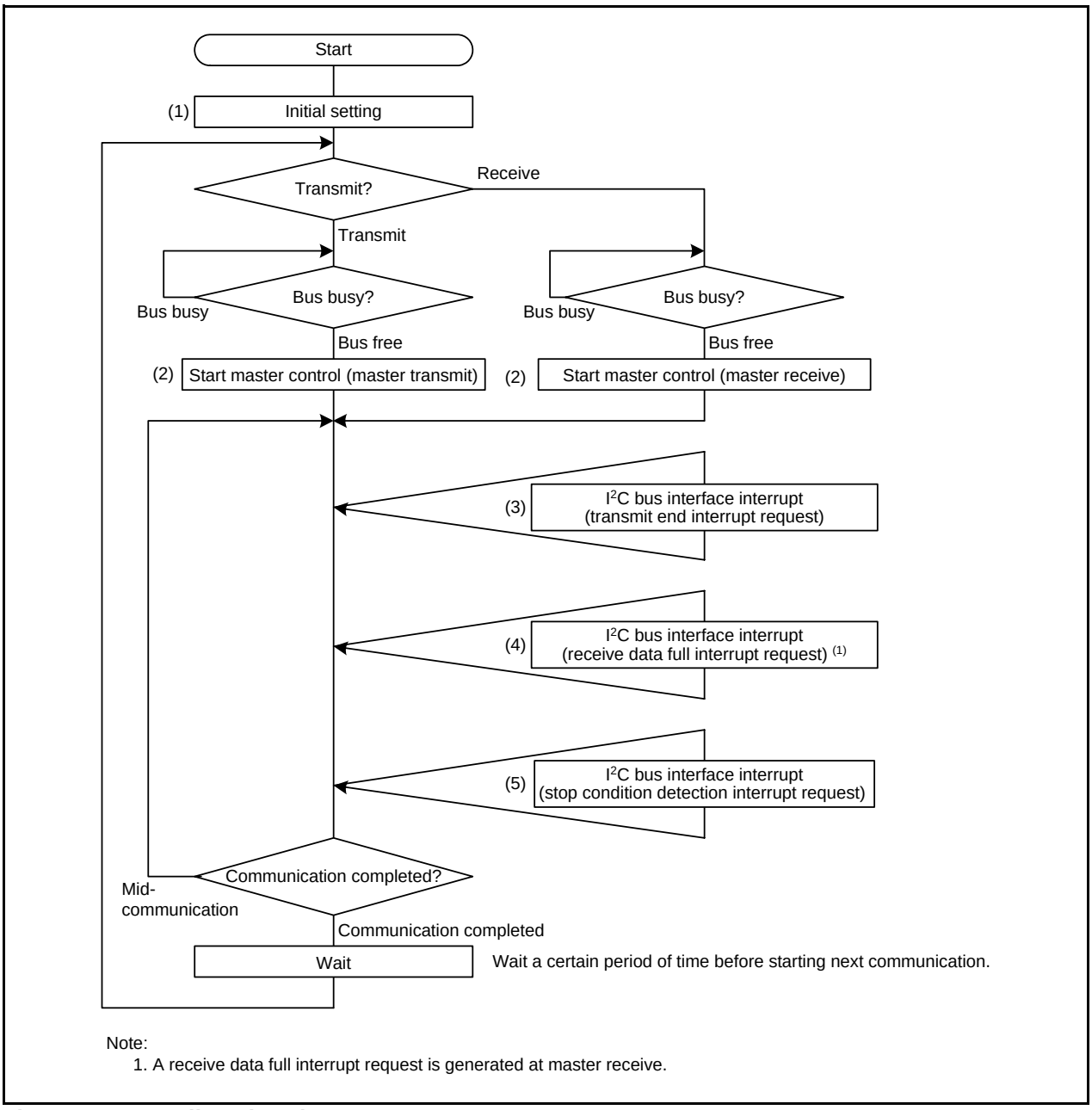


Figure 3.3 Outline Flowchart

A process outline is described as follows:

(1) Initial setting
Initialize the system clock, I²C bus interface associated SFRs, and variables used.

(2) Start master control
Generate a start condition. Enable the I²C bus interface interrupt (transmit end interrupt request) and transmit the slave address.

(3) I²C bus interface interrupt (transmit end interrupt request)
An interrupt is generated at the rising edge of the ninth bit of the SCL clock.

At master transmit
• Determine ACK/NACK and set the next byte transmit data.
At master receive
• Disable the transmit end interrupt request and enable the receive data full interrupt request. When communication is completed, disable the transmit end interrupt request and receive data full interrupt request. Generate a stop condition and enable the stop condition detection interrupt request.

(4) I²C bus interface interrupt (receive data full interrupt request)
An interrupt is generated at the rising edge of the ninth bit of the SCL clock at master receive. Set the next byte ACK/NACK and read the receive data. Disable the transmit end interrupt request and receive data full interrupt request when communication is completed. Then generate a stop condition and enable the stop condition detection interrupt request.

(5) I²C bus interface interrupt (stop condition detection interrupt request)
An interrupt is generated when the stop condition is detected. Disable the stop condition detection interrupt request. Read the last receive data at master receive. Set to slave receive mode and disable the I²C bus interface interrupt.

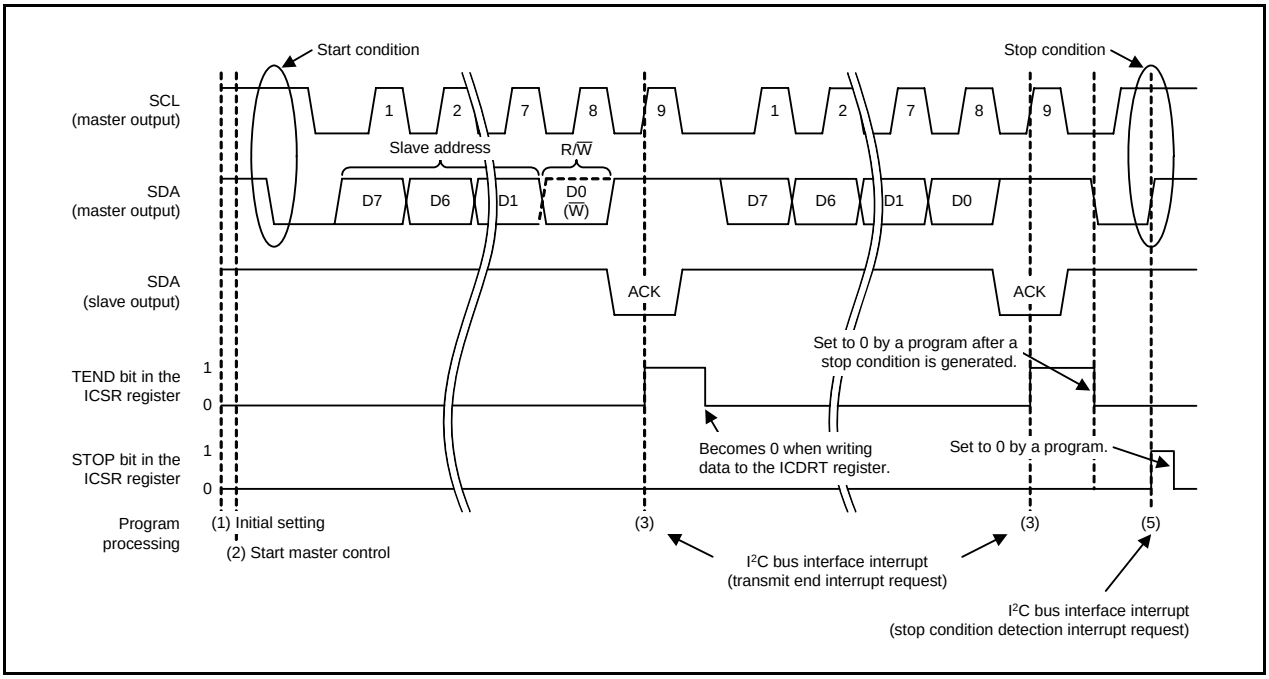


Figure 3.4 Master Transmit Timing

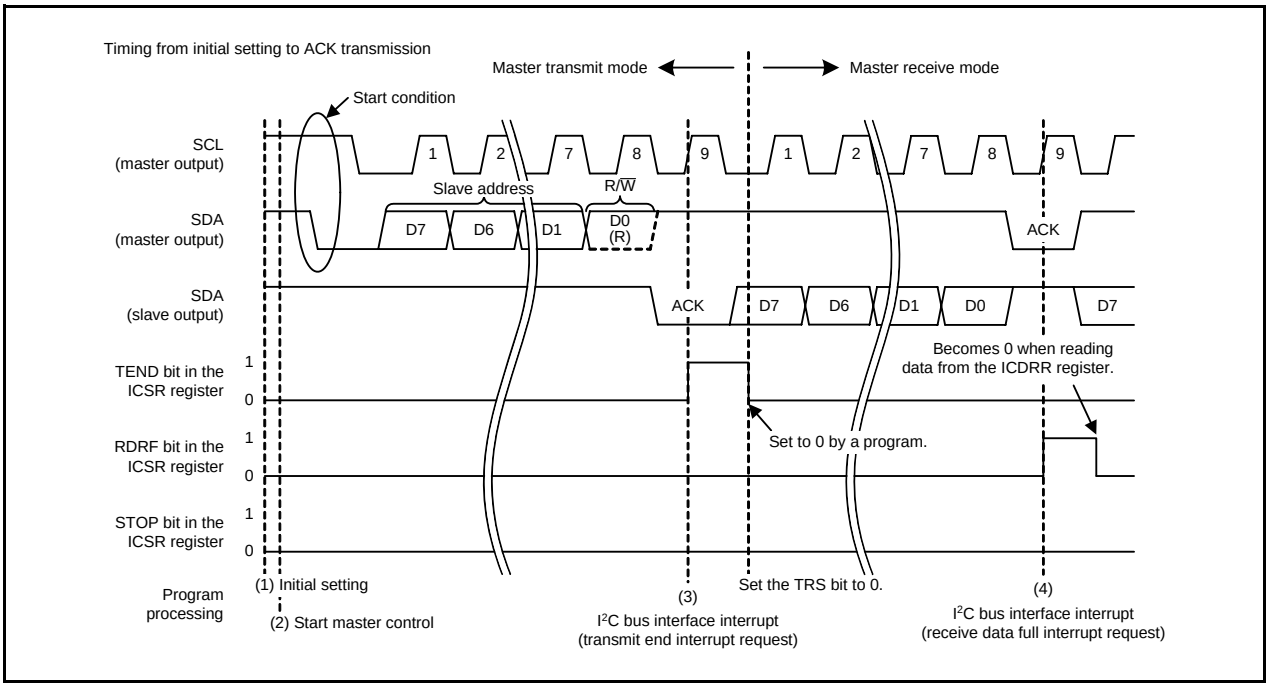


Figure 3.5 Master Receive Timing (1)

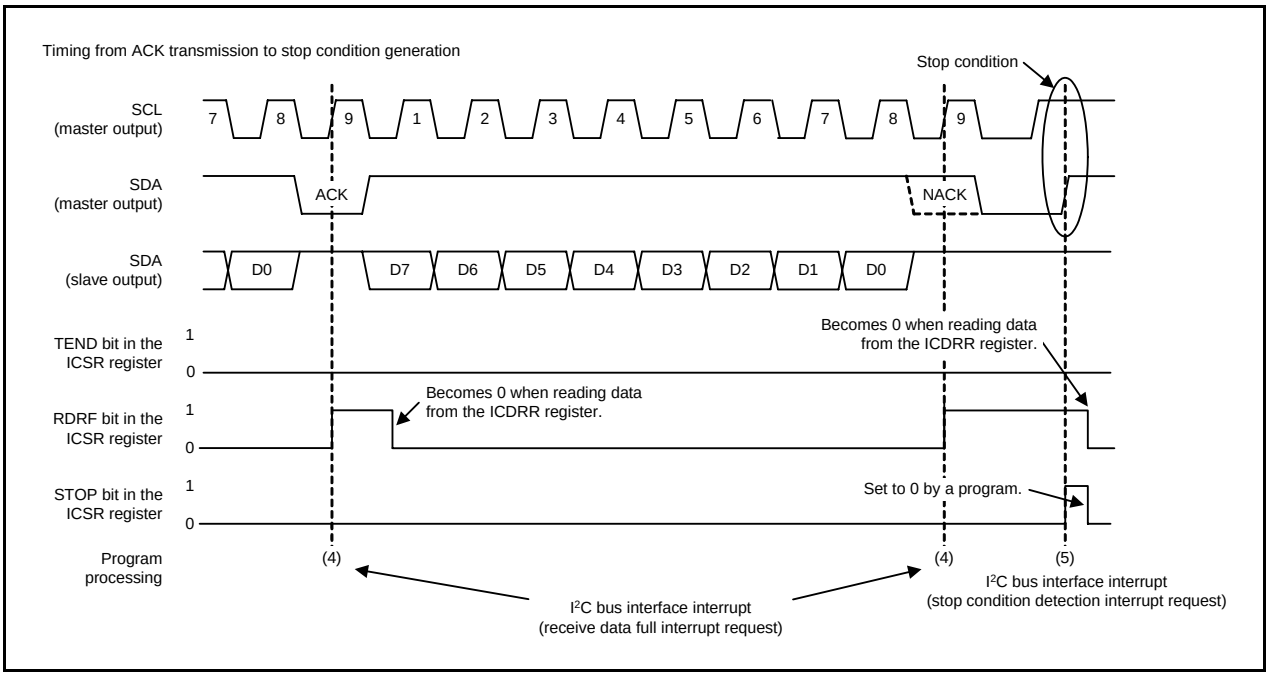


Figure 3.6 Master Receive Timing (2)

3.1.1 Peripheral Functions

The I²C bus interface mode of the I²C bus interface is used under the following setting conditions:

- I²C bus format is used.
- f1/56 is used for the transfer clock (approximately 357 kHz is set as the transfer rate).
- No wait states are set (data and the acknowledge bit are transferred consecutively).
- MSB first is used for the transfer format.
- 3 × f1 cycles are used for the SDA digital delay value.
- The receive acknowledge bit (ACKBR bit) is used to determine an acknowledge signal.
- The receive data full interrupt request is used.
- The transmit end interrupt request is used.
- The stop condition detection interrupt request is used.
- The transmit data empty interrupt request is not used.
- The NACK receive interrupt request and arbitration lost/overrun error interrupt request are not used.

Calculating the transfer rate

Transfer rate = Bits CKS3 to CKS0 in the ICCR1 register setting

= 20 MHz (f1) ÷ 56

≈ 357.142 kHz

Table 3.1 Pins Used and Their Functions

Pin	I/O	Function
P3_5/SCL	I/O	I²C bus clock I/O pin
P3_7/SDA	I/O	I²C bus data I/O pin

3.1.2 Notes on Using the Attached Sample Program

Note the following when using the program included with this application note:

- Do not use multiple interrupts.
- When setting the system clock to anything other than the 20 MHz XIN clock, change the setting value of bits CKS3 to CKS0 according to the transfer rate calculation shown in **3.1.1 Peripheral Functions**.

3.2 Memory

Table 3.1 Memory

Memory	Size	Remarks
ROM	568 bytes	In the iic.c module
RAM	6 bytes	In the iic.c module
Maximum user stack	18 bytes	
Maximum interrupt stack	25 bytes	

Usage memory size varies depending on C compiler version and compile options. The above applies under the following conditions:

C compiler: M16C Series, R8C Family C Compiler V.5.45 Release 01

Compile options: -c -finfo -dir “$(CONFIGDIR)” -R8C

4. Software

This section shows the program example to set the example described in section **3. Application Example**. Refer to the latest **R8C/35C Group** hardware user's manual for details on individual registers.

4.1 Usage Variables

Table 4.1 Definition File Name: r01an0074_src.c

Variable Name	Size	Description
unsigned char iic_tx[BUFSIZE]	255 bytes	Transmit buffer
unsigned char iic_rx[BUFSIZE]	255 bytes	Receive buffer

Table 4.2 Definition File Name: iic.c

Variable Name		Size /Bit-number	Description
static byte_dt iic_str1		—	Structure to store slave address
Structure member	iic_slave_addr	1 byte	Slave address
	iic_rw	b0	R/$\overline{W}$ flag 0: Write ($\overline{W}$) 1: Read (R)
	—	b7 to b1	7-bit address
static byte_dt iic_str2		—	Structure to store status
Structure member	iic_status	1 byte	All statuses
	iic_start	b0	Mid-communication flag 0: Communication completed 1: Mid-communication
	iic_err_par	b1	Parameter error flag 0: No error 1: Parameter error
	iic_err_nack	b2	NACK detection error flag 0: No error 1: NACK detection error
	iic_err_addr	b3	No address match error flag 0: No error 1: No address match error
	—	b7 to b4	Not used (undefined)
unsigned char iic_length		1 byte	Transfer data length
unsigned char iic_index		1 byte	Number of transmit/receive bytes
unsigned char far *iic_pointer		2 bytes	Transmit/receive buffer pointer

4.2 Function Tables

Declaration	void main (void)		
Outline	Main processing		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	unsigned char iic_tx[BUFSIZE]		Transmit buffer
	unsigned char iic_rx[BUFSIZE]		Receive buffer
Returned value	Type	Value	Meaning
	None	—	—
Function	After initializing the system clock and I²C bus interface, master transmission and reception are repeated alternately. Call the iic_master_start function to start master control and call the iic_master_end function to wait for completion of master control. After master control is completed, insert a given length of wait time before starting the next communication.		

Declaration	void mcu_init (void)		
Outline	System clock setting		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	None		—
Returned value	Type	Value	Meaning
	None	—	—
Function	Call this function from the main processing. Set the system clock (XIN clock).		

Declaration	void iic_init (unsigned char ini)		
Outline	Initial setting of I²C bus interface		
Argument	Argument name	Meaning	
	unsigned char ini	0: I²C module disabled 1: I²C module enabled	
Variable (global)	Variable name		Contents
	None		—
Returned value	Type	Value	Meaning
	None	—	—
Function	This function is called from the main processing. Initialize SFRs to use I²C bus interface.		

<table>
<tr><td>Declaration</td><td colspan="3">unsigned char iic_master_start (
unsigned char addr,
unsigned char rw,
unsigned char far *buf,
unsigned char len)</td></tr>
<tr><td>Outline</td><td colspan="3">Master control start processing</td></tr>
<tr><td rowspan="5">Argument</td><td>Argument name</td><td colspan="2">Meaning</td></tr>
<tr><td>unsigned char addr</td><td colspan="2">0x00 to 0x7F: Specify slave address</td></tr>
<tr><td>unsigned char rw</td><td colspan="2">0x00: Master transmit
0x01: Master receive</td></tr>
<tr><td>unsigned char far *buf</td><td colspan="2">Transmit or receive buffer pointer</td></tr>
<tr><td>unsigned char len</td><td colspan="2">0x01 to 0xFF: Transfer data length</td></tr>
<tr><td rowspan="8">Variable (global)</td><td>Variable name</td><td colspan="2">Contents</td></tr>
<tr><td>(structure member) iic_status</td><td colspan="2">All statuses</td></tr>
<tr><td>(structure member) iic_start</td><td colspan="2">Mid-communication flag</td></tr>
<tr><td>(structure member) iic_err_par</td><td colspan="2">Parameter error flag</td></tr>
<tr><td>(structure member) iic_slave_addr</td><td colspan="2">Slave address</td></tr>
<tr><td>unsigned char iic_length</td><td colspan="2">Transfer data length</td></tr>
<tr><td>unsigned char *iic_pointer</td><td colspan="2">Transmit/receive buffer pointer</td></tr>
<tr><td>unsigned char iic_index</td><td colspan="2">Number of transmit/receive bytes</td></tr>
<tr><td rowspan="4">Returned value</td><td>Type</td><td>Value</td><td>Meaning</td></tr>
<tr><td rowspan="3">unsigned char</td><td>0</td><td>Bus busy</td></tr>
<tr><td>1</td><td>Bus free</td></tr>
<tr><td>0xFF</td><td>Parameter error</td></tr>
<tr><td>Function</td><td colspan="3">This function is called from the main function to perform master control start processing. Before executing this function, execute the iic_init function to enable the I2C module.
In the function header, all statuses are initialized and argument parameters are checked. If any parameter value is invalid, the parameter error flag is set to 1 and 0xFF is returned. Master control start processing is not performed when a parameter error is detected. Next, the bus status is checked.
• When the bus is busy, the returned value is 0 and master control start processing is not performed.
• When the bus is free, the returned value is 1 and master control start processing is performed. After setting the mid-communication flag to 1, a start condition is generated and a slave address is transmitted.</td></tr>
</table>

Declaration	void _ssuic (void)		
Outline	I²C bus interface interrupt handling		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	unsigned char iic_index		Number of transmit/receive bytes
	(structure member) iic_err_addr		No address match error flag
	(structure member) iic_rw		R/$\overline{W}$ flag
Returned value	Type	Value	Meaning
	None	—	—
Function	An interrupt is generated at the rising edge of the ninth bit of the SCL clock or when a stop condition is detected. When a stop condition is detected, call the stp_int function. When a stop condition is not detected, call the master_trn_int function at master transmit and the master_rcv_int function at master receive. When communication is completed, generate a stop condition and enable the stop condition detection interrupt request.		

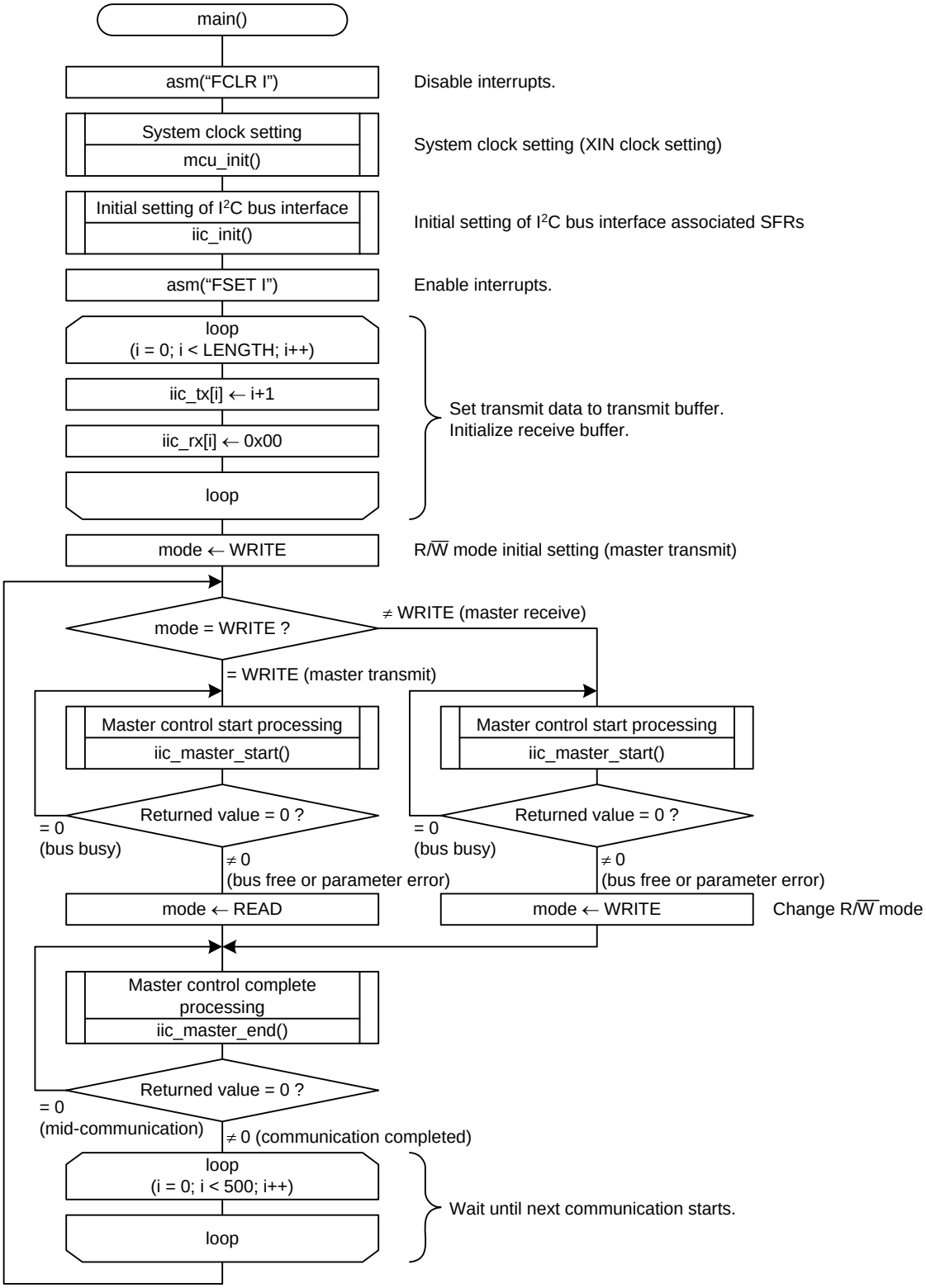
Declaration	static void stp_int (void)		
Outline	Stop condition detection processing		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	unsigned char far *iic_pointer		Transmit or receive buffer pointer
	unsigned char iic_index		Number of transmit/receive bytes
	(structure member) iic_start		Mid-communication flag
Returned value	Type	Value	Meaning
	None	—	—
Function	This function is called from the I²C bus interface interrupt handling. The I²C bus interface associated SFRs changed during communication are reset, and the mid-communication flag is set to 0.		

Declaration	static unsigned char master_trn_int (void)		
Outline	Master transmit processing		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	(structure member) iic_err_nack		NACK detection error flag
	unsigned char iic_index		Number of transmit/receive bytes
	unsigned char iic_length		Transfer data length
	unsigned char far *iic_pointer		Transmit/receive buffer pointer
Returned value	Type	Value	Meaning
	unsigned char	IIC_SP_ON	0: Stop condition generated
		IIC_SP_OFF	1: Stop condition not generated
Function	This function is called from the I2C bus interface interrupt handling. IIC_SP_OFF is returned in the following case: ACK is detected and not the last byte (starts the next transmission). IIC_SP_ON is returned in the following cases: • NACK is detected (NACK detection error flag is set to 1). • The last byte transmission is completed.		

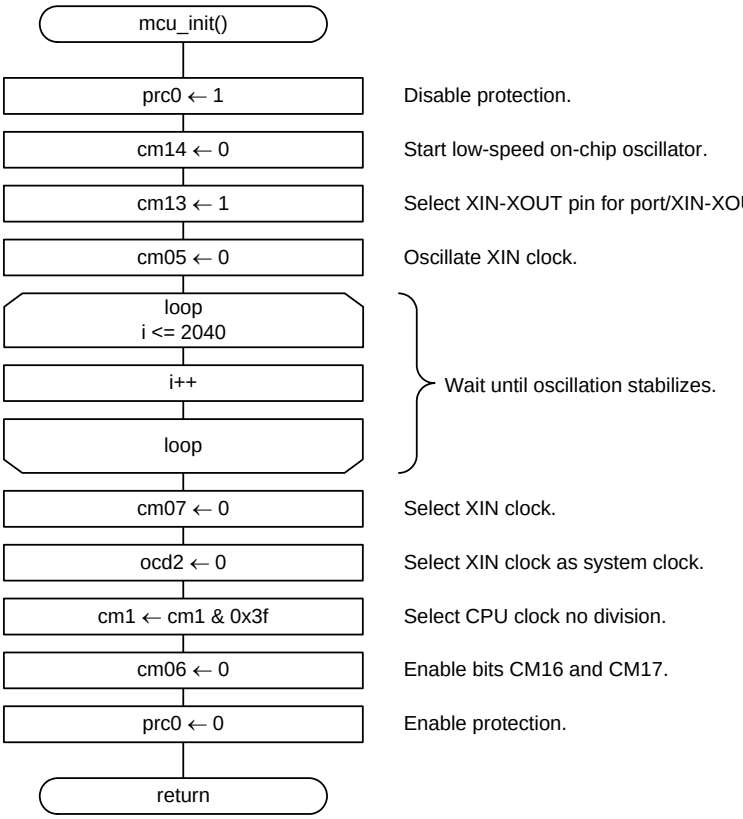
Declaration	static unsigned char master_rcv_int (void)		
Outline	Master receive processing		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	unsigned char iic_index		Number of transmit/receive bytes
	unsigned char iic_length		Transfer data length
	unsigned char far *iic_pointer		Transmit/receive buffer pointer
Returned value	Type	Value	Meaning
	unsigned char	IIC_SP_ON	0: Stop condition generated
		IIC_SP_OFF	1: Stop condition not generated
Function	This function is called from the I2C bus interface interrupt handling. After transmitting the first byte (slave address), set to master receive mode and enable the receive data full interrupt request. IIC_SP_OFF is returned in the following case: • The following data is not the last byte data. IIC_SP_ON is returned in the following case: • The last byte reception is completed.		

Declaration	unsigned char iic_master_end (void)		
Outline	Master control complete processing		
Argument	Argument name		Meaning
	None		—
Variable (global)	Variable name		Contents
	(structure member) iic_status		All statuses
	(structure member) iic_start		Mid-communication flag
	(structure member) iic_err_par		Parameter error flag
	(structure member) iic_err_nack		NACK detection error flag
	(structure member) iic_err_addr		No address match error flag
Returned value	Type	Value	Meaning
	unsigned char	0	Mid-communication
		1	Communication completed
Function	This function is called from the main function. It informs the user of the master control state. During communication, this function returns 0. When communication is completed, this function returns 1. Additional processing after communication is completed can be added as needed.		

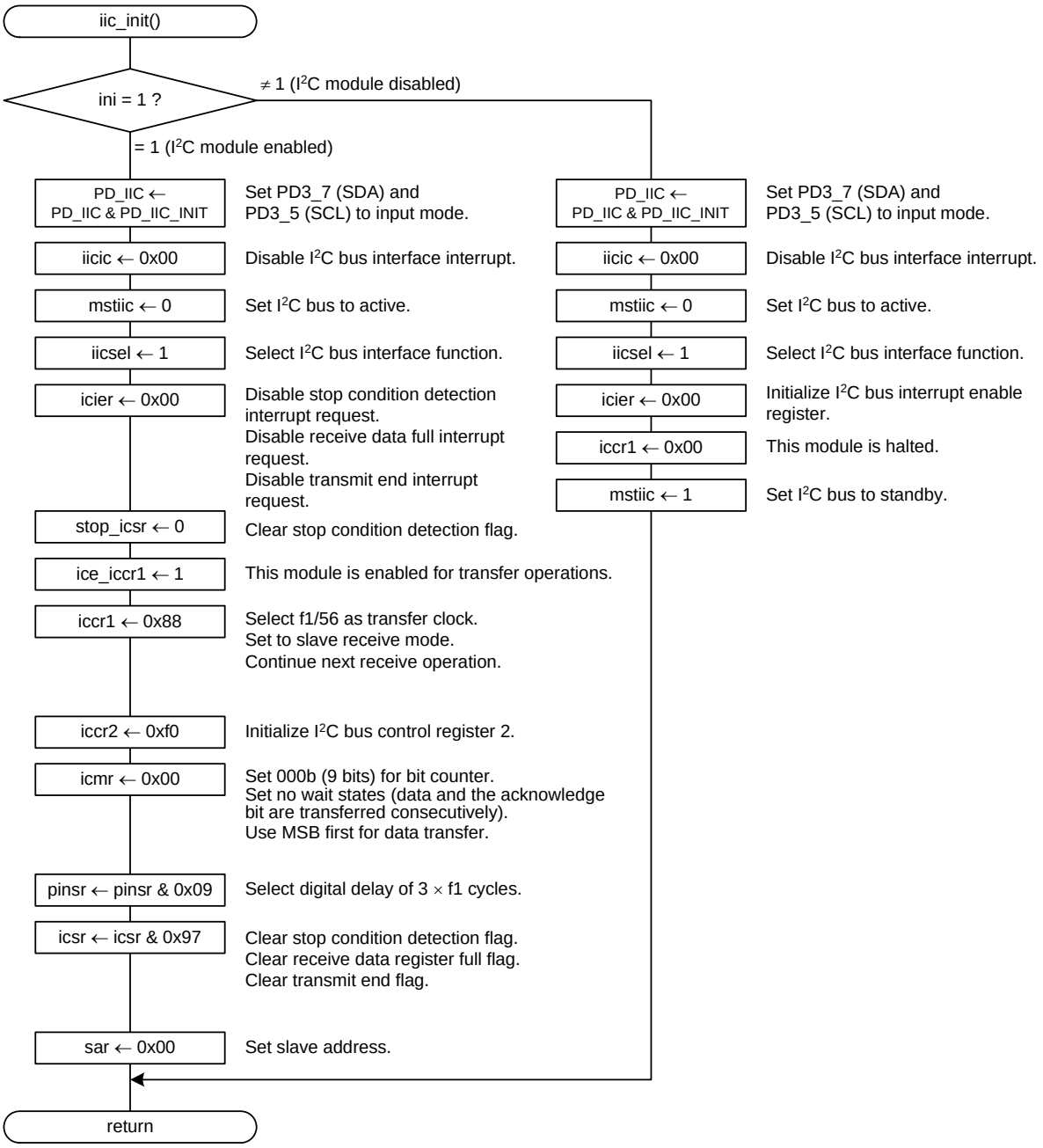
4.3 Main Processing



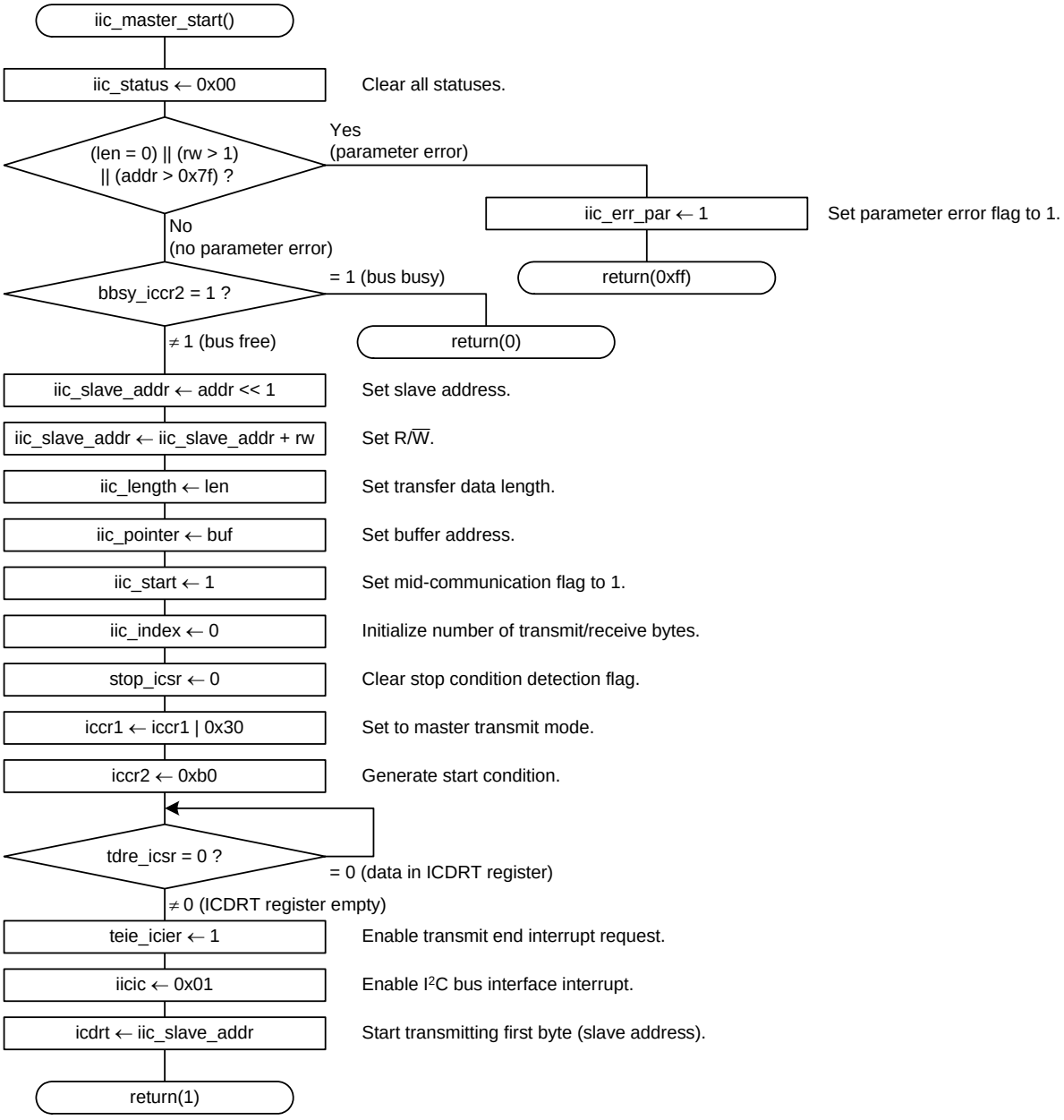
4.4 System Clock Setting



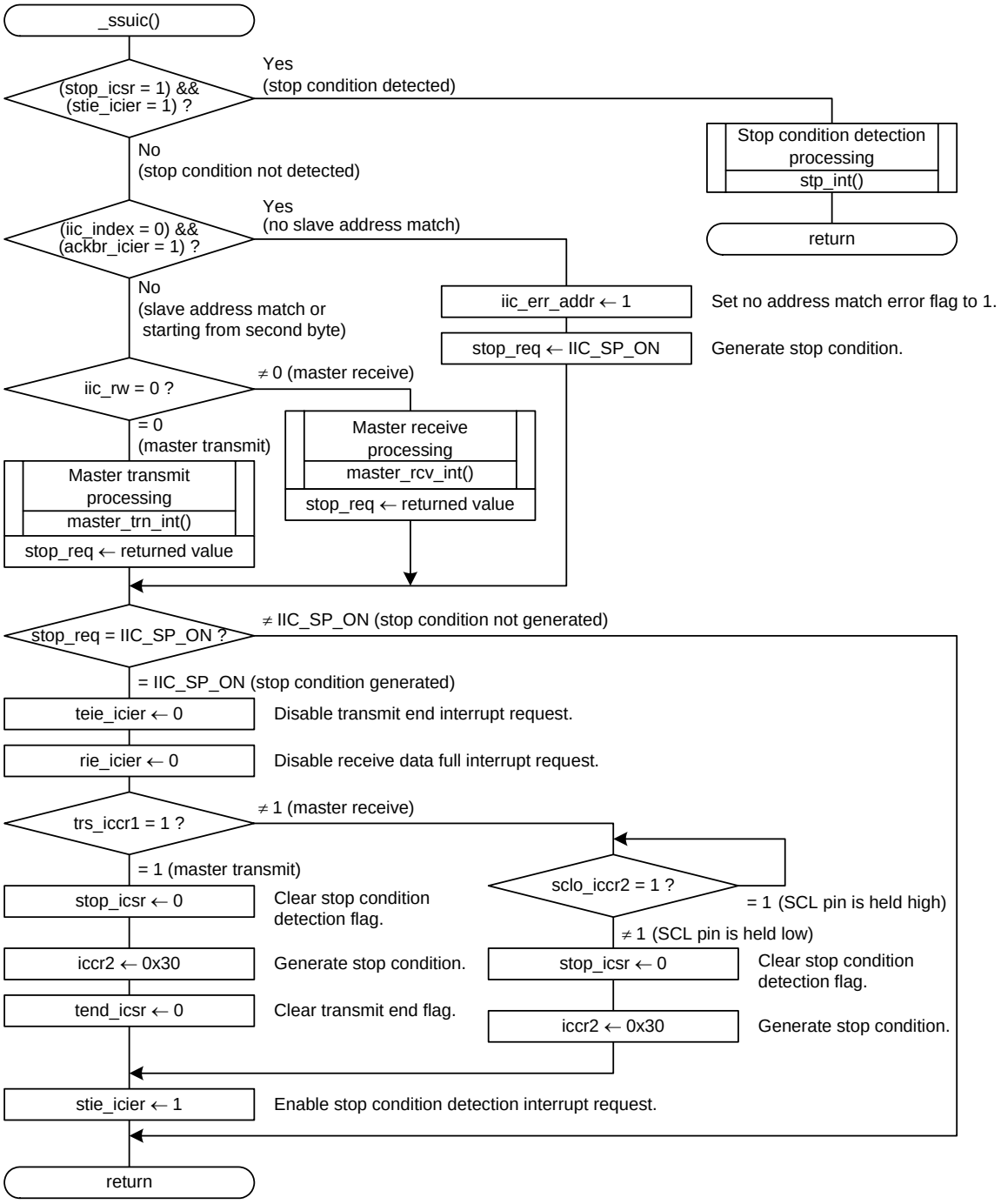
4.5 Initial Setting of I²C Bus Interface



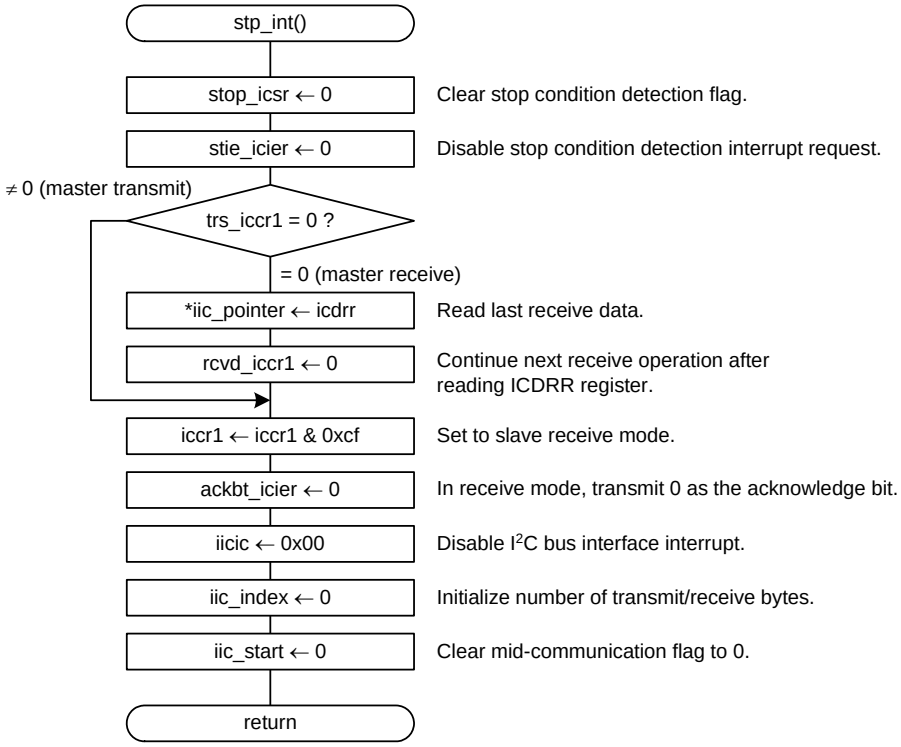
4.6 Master Control Start Processing



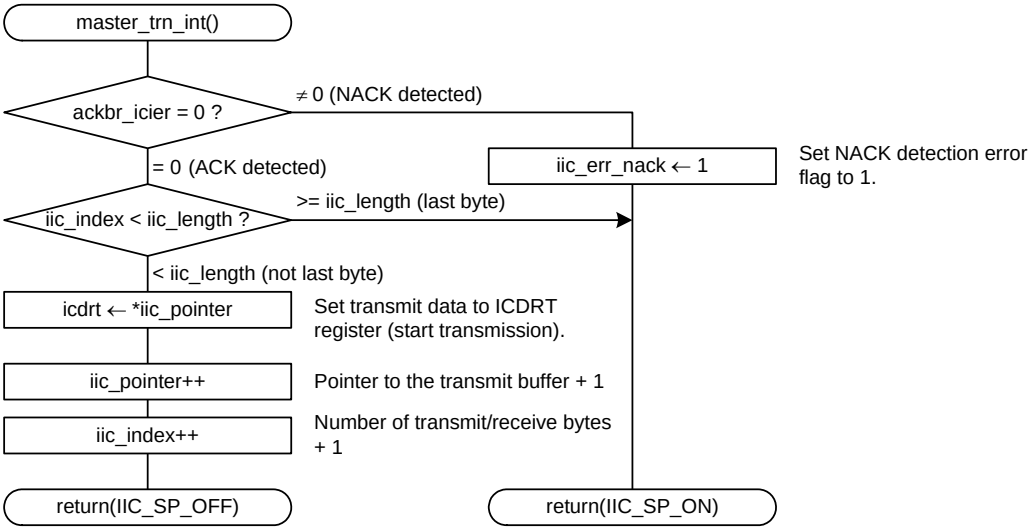
4.7 I²C bus Interface Interrupt Handling



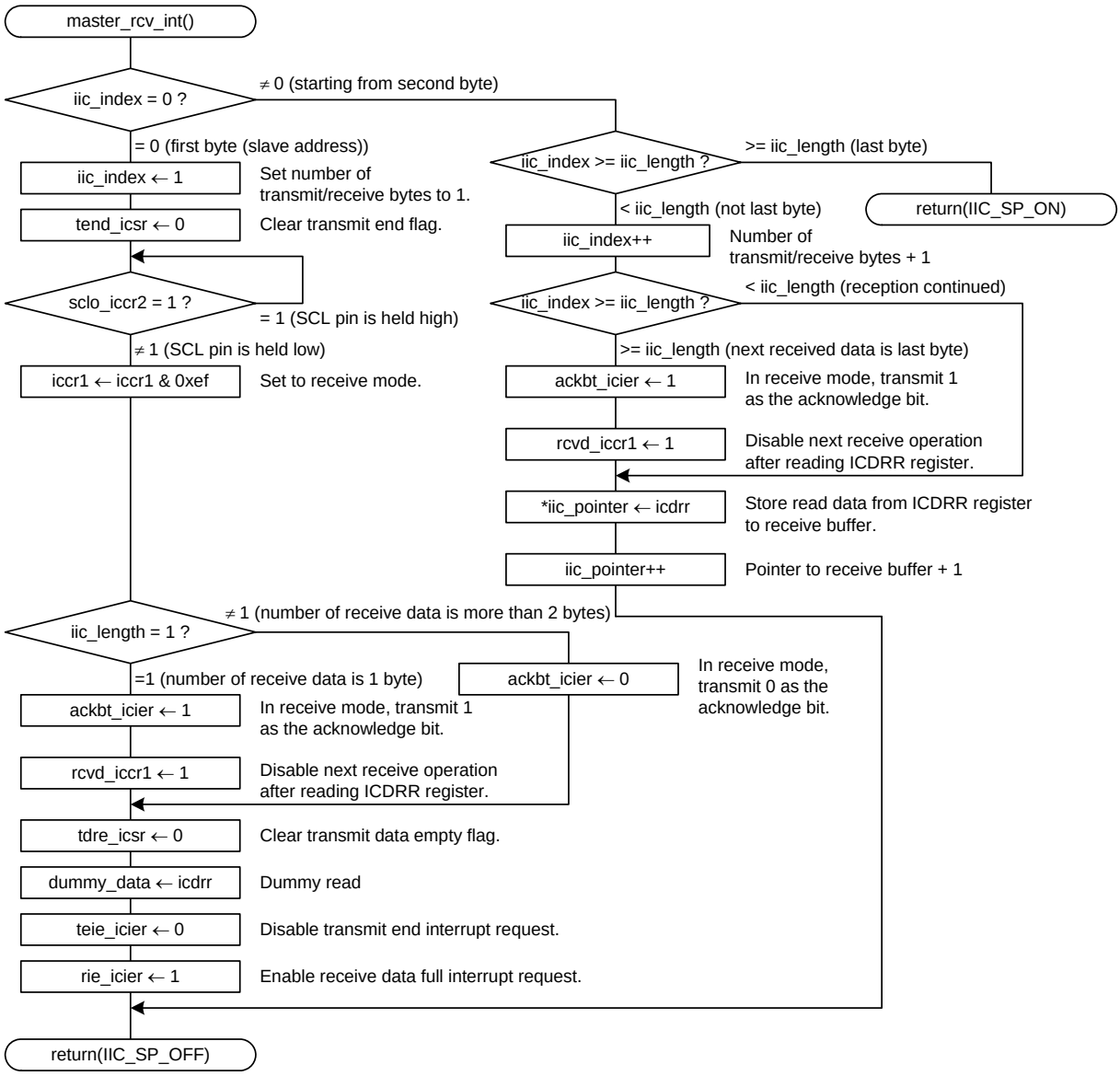
4.8 Stop Condition Detection Processing



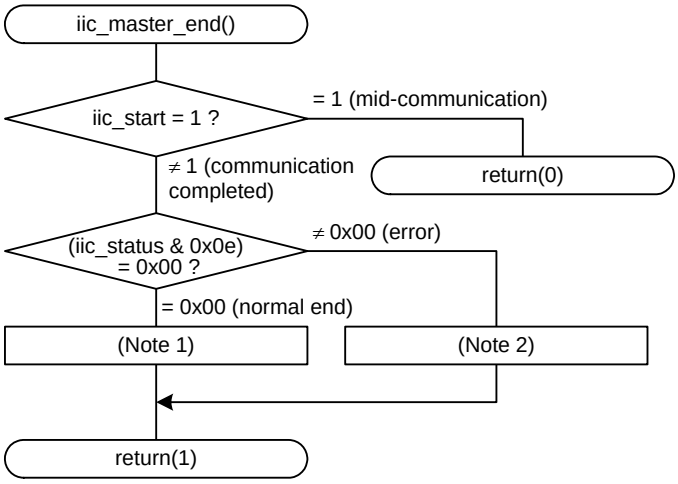
4.9 Master Transmit Processing



4.10 Master Receive Processing



4.11 Master Control Complete Processing



Notes:
1. Additional processing of communication completed normally can be added as needed.
2. Additional processing of communication completed with error can be added as needed.

5. Sample Program

A sample program can be downloaded from the Renesas Electronics website.
To download, click “Application Notes” in the left-hand side menu of the R8C Family page.

6. Reference Documents

R8C/35C Group User’s Manual: Hardware Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision History	R8C/35C Group I2C bus Single Master Control Program (Master Transmit/Receive)

Rev.	Date	Description	
		Page	Summary
1.00	July 15, 2010	—	First edition issued

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 part number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part 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.
7F, 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 1.0