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.

M16C/64A, M16C/65 Group

Multi-Master I^{2}C-bus Interface

1. Abstract

The multi-master I^{2}C-bus interface (I^{2}C interface) is a serial communication circuit based on the I^{2}C-bus data transmit/receive format, and is equipped with arbitration lost detection that makes multi-master communication possible.

This document describes how to use the I^{2}C interface function.

Note: I^{2}C-bus is a trademark of Phillips, Netherlands.

2. Introduction

The application example described in this document applies to the following microcomputers (MCUs):

- MCUs: M16C/64A Group
  M16C/65 Group

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

3. Overview

The I^2C interface is a serial communication circuit based on the I^2C-bus data transmit/receive format, and is equipped with arbitration lost detection and clock synchronous functions.

3.1 General Call

A general call can be detected when the address data is all 0's (1).

Note:

1. The master transmits general call address 00h to all slaves.

3.2 Addressing Format

7-bit addressing format is supported.

Only the 7 high-order bits of the I^2C address register (slave address) are compared with the address data.

3.3 I^2C Interface Related Pins

- SCLMM pins: Clock I/O pins of the I^2C interface.
- SDAMM pins: Data I/O pins of the I^2C interface.

3.4 Selectable Functions

The functions below can be selected when using the I^{2}C interface.

(1) Communication mode
There are four communication modes available when performing data communication:

- Master transmission: Start and stop conditions are generated (master mode). Address and control data are output to the SDA in synchronization with the SCLMM clock generated by the master device.
- Master reception: Data from the transmitting device is received in synchronization with the SCLMM clock generated by the master device.
- Slave transmission: Start and stop conditions generated by the master device are received (slave mode). Control data is output in synchronization with the clock generated by the master device.
- Slave reception: Data from the transmitting device is received in synchronization with the clock generated by the master device.

(2) SCL mode
SCL mode can be selected from the following:

- Standard clock mode: The bit rate can be selected in the range 16.1 to 100 kHz.
- High-speed clock mode: The bit rate can be selected in the range 32.3 to 400 kHz.

(3) ACK clock
ACK clock can be selected from the following:

- ACK clock not available: No ACK clocks are generated after a data transfer.
- ACK clock available: The master generates an ACK clock each time 1 byte of data is transferred.

(4) Data format
Data format can be selected from the following:

- Addressing format: The received slave address and bits SAD6 to SAD0 in the S0Di register are compared. (i = 0 to 2). When an address match is found, or when a general call is received, an interrupt request is generated and additional data is transmitted and received.
- Free data format: An interrupt request is generated and additional data is transmitted and received regardless of the received slave address.

4. Data Transmit/Receive Example

The data transmit/receive examples are described in this section. The conditions for the examples are below.

- Slave address: 7 bits
- Data: 8 bits
- ACK clock available
- Standard clock mode, bit rate: 100 kbps (fIIC: 20 MHz; fVIIC: 4 MHz)
  20 MHz (fIIC) divided-by-5 = 4 MHz (fVIIC),
  4 MHz (fVIIC) divided-by-8 and further divided-by-5 = 100 kbps (bit rate).
- In receive mode, ACK is returned for data other than the last data. NACK is returned after the last data is received.
- When receiving data, I^{2}C-bus interrupt at the eighth clock (before the ACK clock): Disabled
- Stop condition detection interrupt: Enabled
- Timeout detection interrupt: Disabled
- Set own slave address to the S0D0 register (do not use registers S0D1 or S0D2).

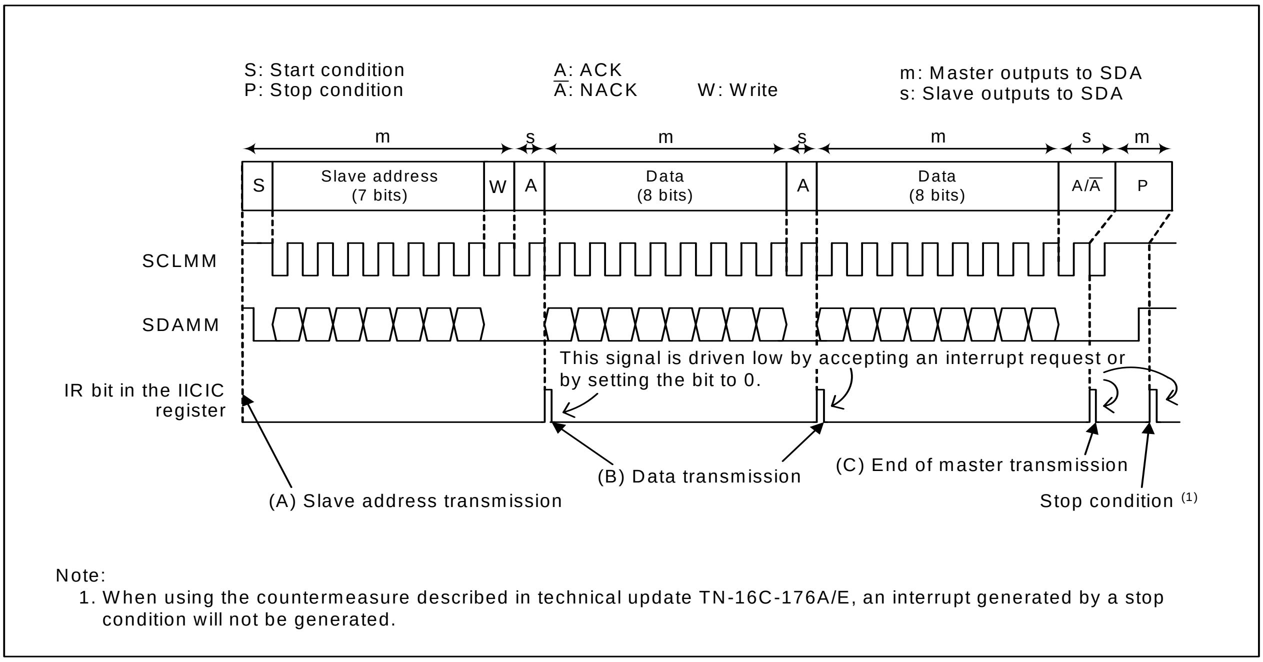
While receiving data, if an interrupt is enabled at the eighth clock, ACK or NACK can be set after each byte of received data is checked.

4.1 Initial Settings

Follow the initial setting procedures below for 4.2 Master Transmission to 4.5 Slave Transmission.
(1) Write own slave address to bits SAD6 to SAD0 in the S0D0 register.
(2) Write 85h to the S20 register (CCR value: 5, standard clock mode selected, ACK clock available).
(3) Write 18h to the S4D0 register (fVIIC: fIIC divided-by-5, timeout detection, interrupt disabled).
(4) Write 01h to the S3D0 register (while receiving data, an I^{2}C-bus interrupt is disabled at the eighth clock (before the ACK clock) and a stop condition detection interrupt is enabled).
(5) Write 0Fh to the S10 register (slave receive mode).
(6) Write 98h to the S2D0 register (SSC value: 18h; start/stop condition generation timing: long mode).
(7) Write 08h to the S1D0 register (bit counter: 8, I^{2}C interface enabled, addressing format; input level: I^{2}C-bus input).
If the MCU uses a single-master system and the MCU itself is the master, start the initial setting procedures from step (2).

4.2 Master Transmission

Master transmission is described in this section. The initial settings are described in 4.1 Initial Settings.
Initial settings are assumed to be completed. Programs (A) to (C) below refer to (A) to (C) in the following figure.



Note:
1. When using the countermeasure described in technical update TN-16C-176A/E, an interrupt generated by a stop condition will not be generated.

Figure 4.1 Example of Master Transmission

(A) Slave address transmission
- (1) The BB bit in the S10 register must be 0 (bus free).
- (2) Write E0h to the S10 register (start condition standby).
- (3) Write a slave address to the upper 7 bits and set the least significant bit to 0 (start condition generated, then slave address transmitted).

After a stop condition is generated and the BB bit becomes 0, the S10 register is write disabled for 1.5 cycles of fVIIC. Therefore, when writing E0h to the S10 register and a slave address to the S00 register during the 1.5 fVIIC cycles, a start condition is not generated.
When generating a start condition immediately after the BB bit changes from 1 to 0, confirm that both the TRX and MST bits are 1 (transmission mode and master mode) after step (1), and then execute step (2).

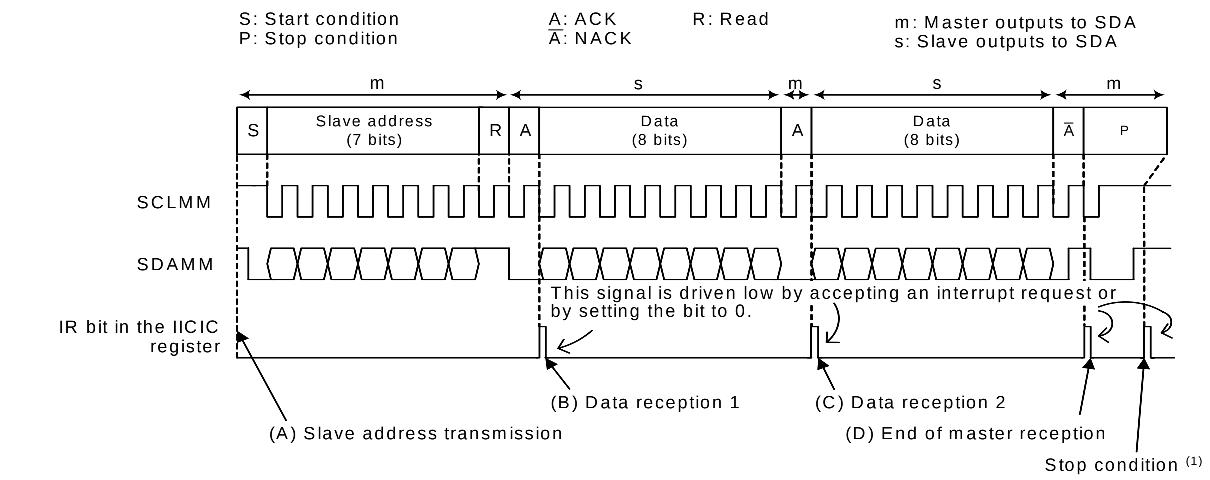
(B) Data transmission (in the I^{2}C-bus interrupt routine)
- (1) Write transmit data to the S00 register (data transmission).

(C) Completion of master transmission (in the I^{2}C-bus interrupt routine)
- (1) Write C0h to the S10 register (stop condition standby).
- (2) Write dummy data to the S00 register (stop condition generated).

When the transmission is completed or ACK is not returned from the slave device (NACK returned), master transmission should be completed as shown in the example above.
If the slave device or other master device drives the SCLMM line low, use the countermeasure described in technical update (TN-16C-176A/E). An I^{2}C-bus interface interrupt generated by a stop condition will not be generated.

4.3 Master Reception

Master reception is described in this section. The initial settings are described in 4.1 Initial Settings.
Initial settings are assumed to be completed. Programs (A) to (D) below refer to (A) to (D) in the following figure.



Note:
1. When using the countermeasure described in technical update TN-16C-176A/E, an interrupt generated by a stop condition will not be generated.

Figure 4.2 Example of Master Reception

(A) Slave address transmission
(1) The BB bit in the S10 register must be 0 (bus free).
(2) Write E0h to the S10 register (start condition standby).
(3) Write a slave address to the upper 7 bits (MSB) and a 1 to the LSB (start condition generated, then slave address transmitted).

(B) Data reception 1 (after slave address transmission) (in the I^{2}C-bus interrupt routine)
(1) Write AFh to the S10 register (master receive mode).
(2) Set the ACKBIT bit in the S20 register to 0 (ACK is available) because the data is not the last one.
(3) Write dummy data to the S00 register.

(C) Data reception 2 (data reception) (in the I^{2}C-bus interrupt routine)
(1) Read the received data from the S00 register.
(2) Set the ACKBIT bit in the S20 register to 1 (no ACK) because the data is the last one.
(3) Write dummy data to the S00 register.

(D) End of master reception (in the I^{2}C-bus interrupt routine).
(1) Read the received data from the S00 register.
(2) Write C0h to the S10 register (stop condition standby state).
(3) Write dummy data to the S00 register (stop condition generated).

If the slave device or other master device drives the SCLMM line low, use the countermeasure described in technical update TN-16C-176A/E. An I^{2}C-bus interface interrupt generated by a stop condition will not be generated.

4.4 Slave Reception

Slave reception is described in this section. The initial settings are described in 4.1 Initial Settings.
Initial settings are assumed to be completed. Programs (A) to (C) below refer to (A) to (C) in the following diagram.

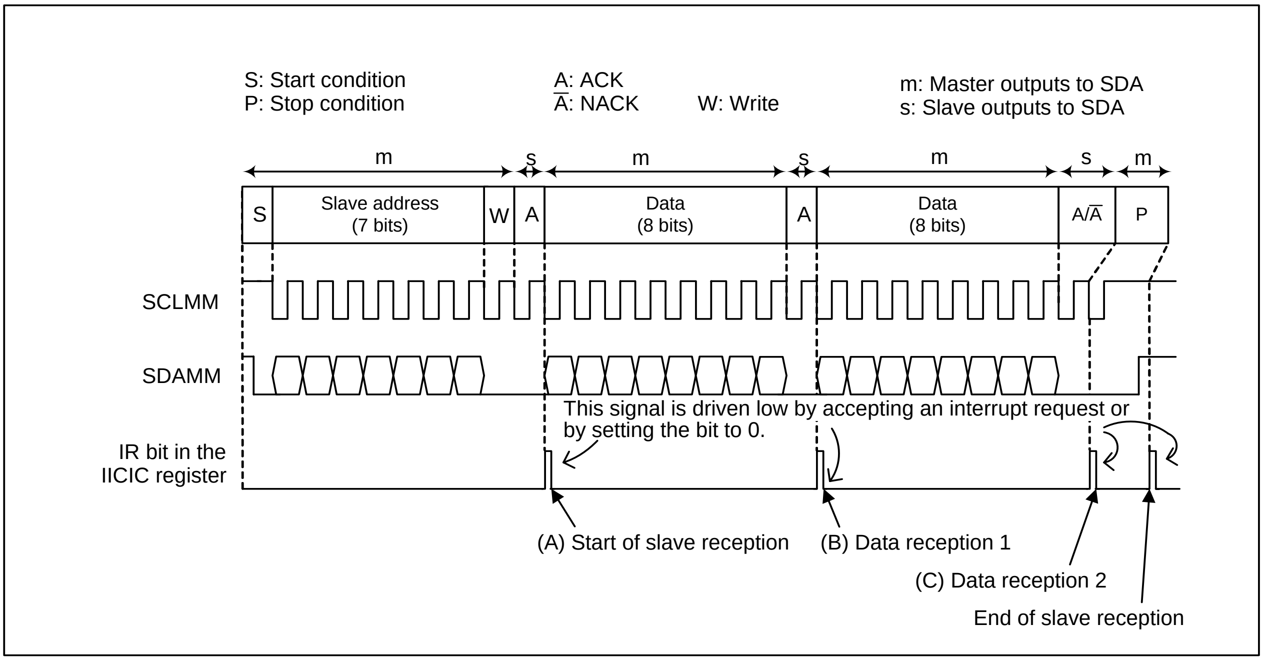


Figure 4.3 Example of Slave Reception

(A) Start of slave reception (in the I^{2}C-bus interrupt routine)
(1) Check the content of S10 register. When the TRX bit is 0, the I^{2}C interface is in slave receive mode.
(2) Write dummy data to the S00 register.

(B) Data reception 1 (in the I^{2}C-bus interrupt routine)
(1) Read the received data from the S00 register.
(2) Set the ACKBIT bit in the S20 register to 0 (ACK is available) because the data is not the last one.
(3) Write dummy data to the S00 register.

(C) Data reception 2 (in the I^{2}C-bus interrupt routine)
(1) Read the received data from the S00 register
(2) Set the ACKBIT bit in the S20 register to 1 (no ACK) because the data is the last one.
(3) Write dummy data to the S00 register.

4.5 Slave Transmission

Slave transmission is described in this section. The initial settings are described in 4.1 Initial Settings.
Initial settings are assumed to be completed. Programs (A) and (B) below refer to (A) and (B) in the following diagram.
When arbitration lost is detected, the TRX bit becomes 0 (receive mode) even when the bit after the slave address is 1 (read). Therefore, after arbitration lost is detected, read the S00 register. When bit 0 in the S00 register is 1, write 4Fh (slave transmit mode) to the S10 register and execute slave transmission.

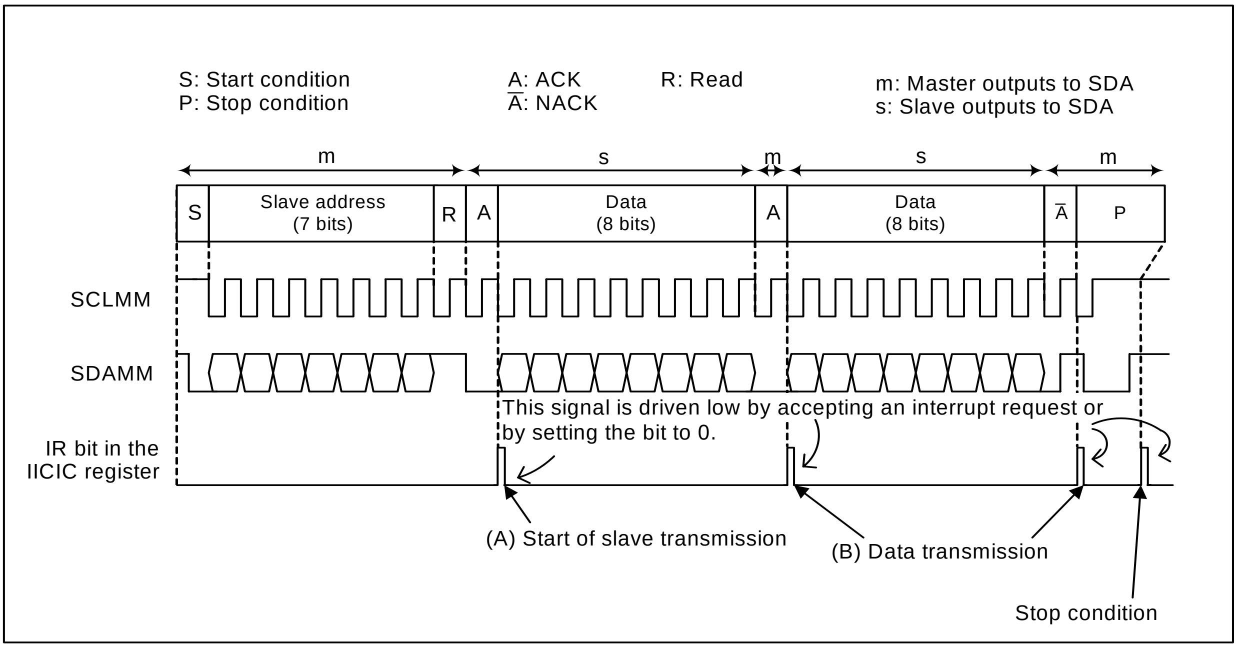


Figure 4.4 Example of Slave Transmission

(A) Start of slave transmission (in the I^{2}C-bus interrupt routine)

(1) Check the content of the S10 register. When the TRX bit is set to 1, the I^{2}C interface is in slave transmit mode.

(2) Write transmit data to the S00 register.

(B) Data transmission (in the I^{2}C-bus interrupt routine)

(1) Write transmit data to the S00 register

Write dummy data to the S00 register even if an interrupt occurs at an ACK clock of the last transmit data. When the S00 register is written, the SCLMM pin becomes high-impedance.

5. Arbitration Lost

The following describes the operation of the I^2C-bus interface when arbitration lost occurs. Figure 5.1 shows the Operation Timing of the Arbitration Lost Detect Flag.

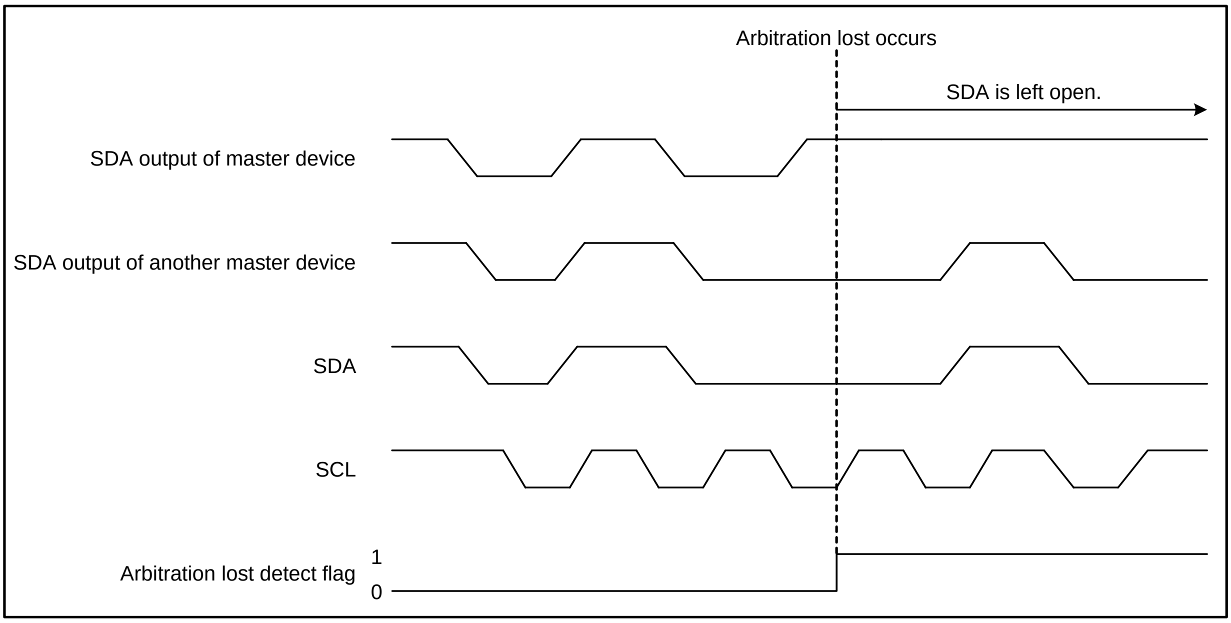

Arbitration lost occurs
SDA is left open.
SDA output of master device
SDA output of another master device
SDA
SCL
Arbitration lost detect flag
1
0

Figure 5.1 Operation Timing of the Arbitration Lost Detect Flag

When arbitration lost occurs, the arbitration lost detect flag becomes 1.

(1) Arbitration lost occurs while transmitting a slave address.
When arbitration lost is detected, the communication mode automatically changes to slave reception enabling the slave address to be received.
If the selected data format is the addressing format, the slave address can be resolved by reading the AAS bit in the S10 register.

(2) Arbitration lost occurs while transmitting data following the slave address.
When arbitration lost is detected, the communication mode automatically changes to slave reception, enabling the data to be received.

6. Interrupt

The I^{2}C-bus interface has the following interrupt sources:

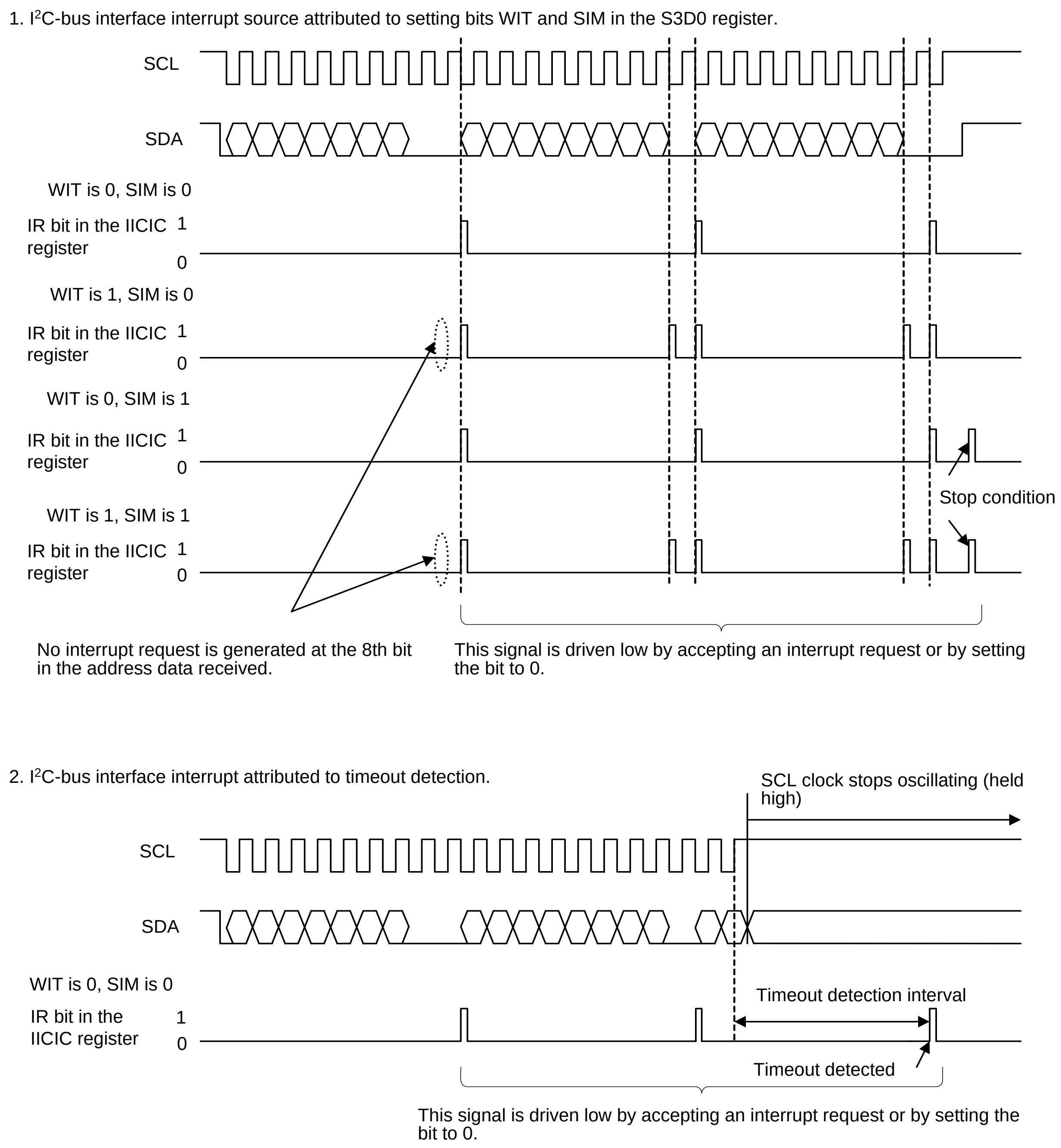
(1) Interrupt when 9-bit transmission/reception is completed (including ACK/NACK)
The interrupt source can be determined by reading the WIT bit in the S3D0 register. When the WIT bit is 0, it is determined that the generated interrupt is attributable to this interrupt source.

(2) Interrupt when 8 bits are received
Setting the WIT bit to 1 enables this interrupt source.
The interrupt source can be determined by reading the WIT bit. When the WIT bit is 1, it is determined that the generated interrupt is attributable to this interrupt source.
If no determination is made of ACK/NACK transmissions, there is no need to use this interrupt.

(3) Interrupt when a stop condition is detected
Setting the SIM bit in the S3D0 register to 1 enables this interrupt source.
The interrupt source can be determined by reading the SCPIN bit in the S4D0 register. When a stop condition is detected, the SCPIN bit becomes 1.

(4) Interrupt when the SCL clock remains high for more than a predetermined time during communication
Setting the TOE bit in the S4D0 register to 1 enables this interrupt source.
The interrupt source can be determined by reading the TOF bit in the S4D0 register. When the SCL clock remains high for more than a predetermined time during communication, the TOF bit becomes 1.

Figure 6.1 shows the I^{2}C-bus Interface Interrupt Request Generation Timing.


1. I2C-bus interface interrupt source attributed to setting bits WIT and SIM in the S3D0 register.
SCL
SDA
WIT is 0, SIM is 0
IR bit in the IICIC register
1
0
WIT is 1, SIM is 0
IR bit in the IICIC register
1
0
WIT is 0, SIM is 1
IR bit in the IICIC register
1
0
Stop condition
WIT is 1, SIM is 1
IR bit in the IICIC register
1
0
No interrupt request is generated at the 8th bit in the address data received.
This signal is driven low by accepting an interrupt request or by setting the bit to 0.
2. I2C-bus interface interrupt attributed to timeout detection.
SCL clock stops oscillating (held high)
SCL
SDA
WIT is 0, SIM is 0
IR bit in the IICIC register
1
0
Timeout detection interval
Timeout detected
This signal is driven low by accepting an interrupt request or by setting the bit to 0.

The above figure assumes the following:
1. The TRX bit in the S10 register is 0 (receive mode).
2. The ACKCLK bit in the S20 register is 1 (ACK is available).

Figure 6.1 I^{2}C-bus Interface Interrupt Request Generation Timing

7. Notes on I^{2}C Interface

7.1 Generating Start Condition

After a stop condition is generated and the BB bit becomes 0 (bus free), the S10 register is write disabled for 1.5 cycles of fVIIC. Therefore, when writing E0h to the S10 register and a slave address to the S00 register during the 1.5 fVIIC cycles, and a start condition is not generated.
When generating a start condition immediately after the BB bit changes from 1 to 0, confirm that both the TRX and MST bits are 1 after step (1), and then execute step (2).

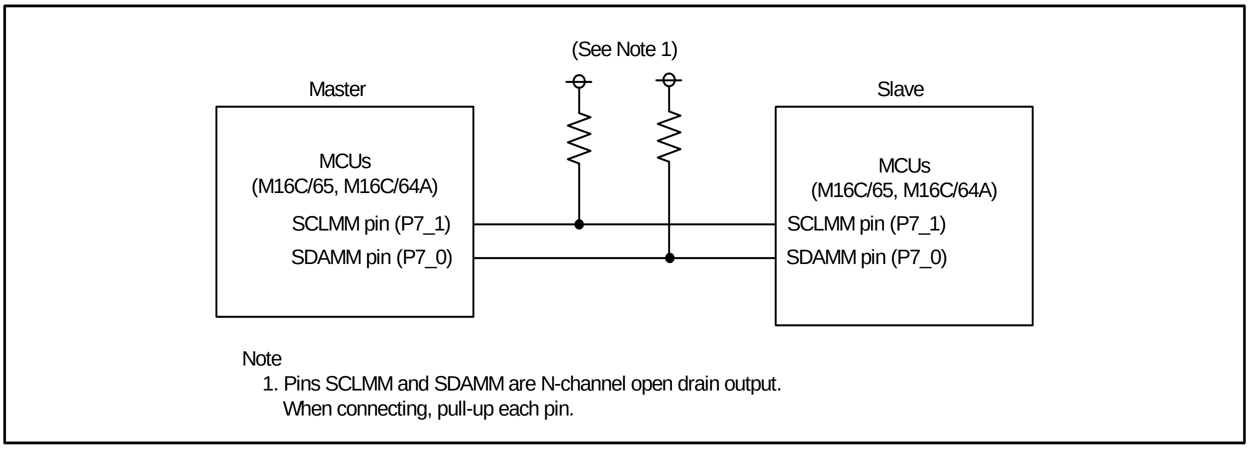
Step:
(1) Write E0h to the S10 register.
The I^{2}C interface enters the start condition standby state and the SDAMM pin is left open.
(2) Write a slave address to the S00 register.
A start condition is generated. Then, the bit counter becomes 000b, the SCL clock signal is output
for 1 byte, and the slave address is transmitted.

8. Sample Program

This sample program is provided for reference purposes only, and is not guaranteed to operate properly in all systems.
When incorporating it into a system, careful examination is recommended before using this sample program.
Furthermore, since its functionality as integral part of a system cannot be evaluated with this program alone, evaluation with the final system is indispensable.

8.1 Connection Example

Figure 8.1 shows the Sample Program Operating Environment.



Note
1. Pins SCLMM and SDAMM are N-channel open drain output.
When connecting, pull-up each pin.

Figure 8.1 Sample Program Operating Environment

8.2 Operation Conditions

Table 8.1 lists the Sample Program Operation Conditions

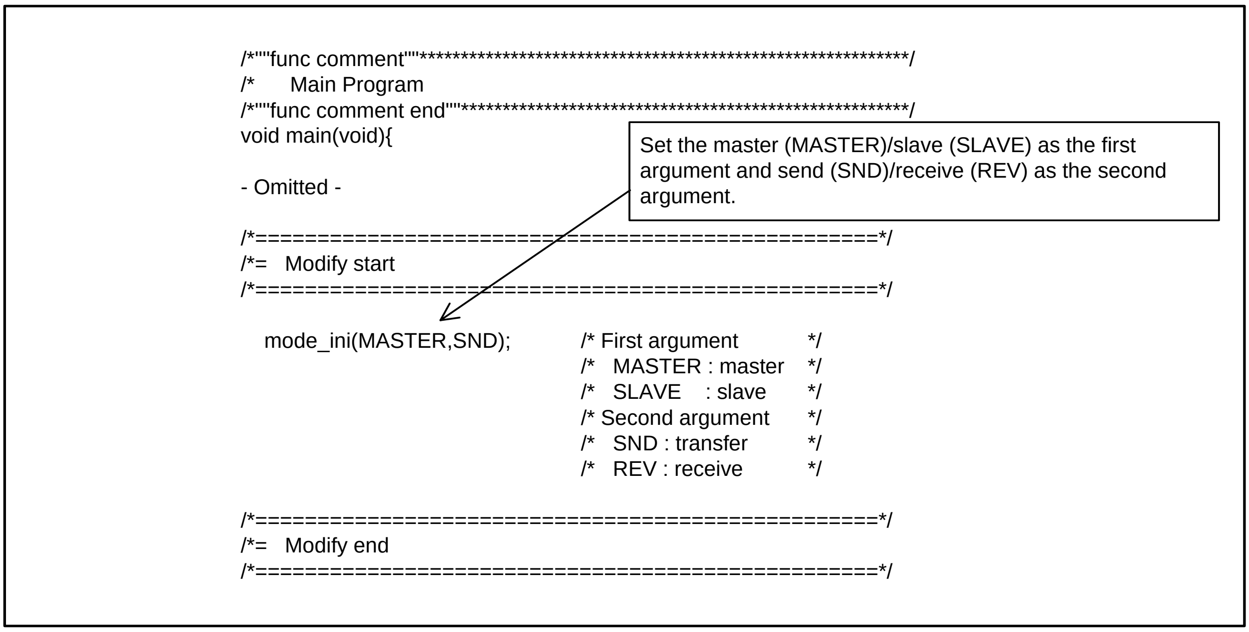
Table 8.1 Sample Program Operation Conditions

Item	Content
Peripheral function clock (fIIC)	24 MHz (XIN: 6 MHz; PLL clock: Divided-by-2, and then multiplied-by-8)
I^{2}C-bus system clock (fVIIC)	4 MHz (fIIC divided-by-6)
Bit rate	100 kbps (fVIIC divided-by-8 and further divided-by-5)
SCL mode	Standard clock mode
Data format	Addressing mode
Slave address compare	S0D0 register only
Stop condition detect interrupt	Enabled
Data receive interrupt	Enabled
Timeout detection function	Enabled

8.3 Sample Program Setting

Four communication modes can be used in the sample program: master transmission, master reception, slave reception, and slave transmission. When calling the “mode_ini” function, the communication modes can be selected by setting arguments.
Set the other slave address and own slave address in define declaration area in the sample program.

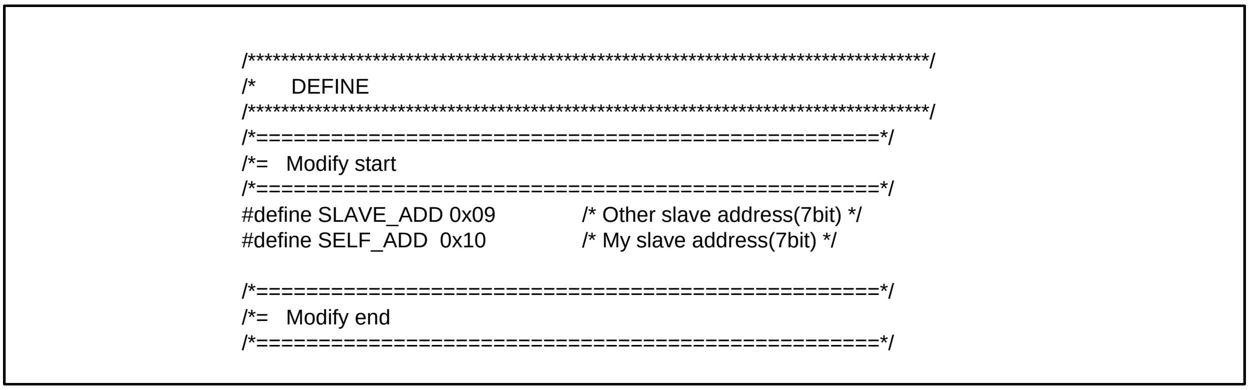
Figure 8.2 shows the Setting Example of Master Transmission. Figure 8.3 shows the Setting Example of Slave Address (0x09) and Own Slave Address (0x10).



```
/*""func comment""**********************************************************/
/*      Main Program
/*""func comment end""******************************************************/
void main(void){

- Omitted -

/*==================================================*/
/*=   Modify start
/*==================================================*/

    mode_ini(MASTER,SND);          /* First argument        */
                                   /*   MASTER : master     */
                                   /*   SLAVE    : slave    */
                                   /* Second argument       */
                                   /*   SND : transfer      */
                                   /*   REV : receive       */

/*==================================================*/
/*=   Modify end
/*==================================================*/
```

Set the master (MASTER)/slave (SLAVE) as the first argument and send (SND)/receive (REV) as the second argument.

Figure 8.2 Communication Mode Setting Example



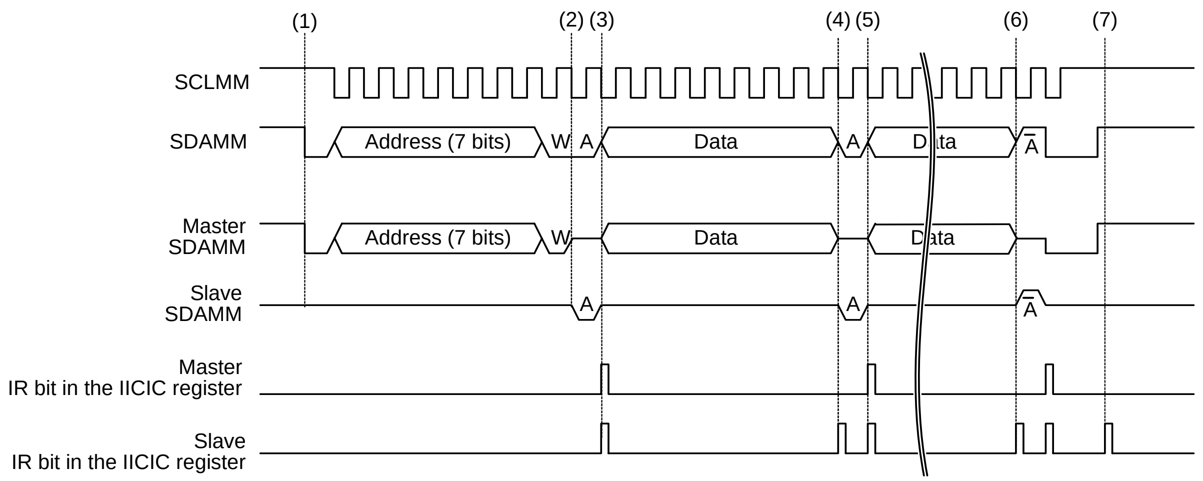
```
/**********************************************************************************/
/*      DEFINE
/**********************************************************************************/
/*==================================================*/
/*=   Modify start
/*==================================================*/
#define SLAVE_ADD 0x09             /* Other slave address(7bit) */
#define SELF_ADD  0x10             /* My slave address(7bit) */

/*==================================================*/
/*=   Modify end
/*==================================================*/
```

Figure 8.3 Slave Address Setting Example

8.4 Operation Example

8.4.1 Master Transmission and Slave Reception

Figure 8.4 shows the Master Transmission and Slave Reception Operation Example

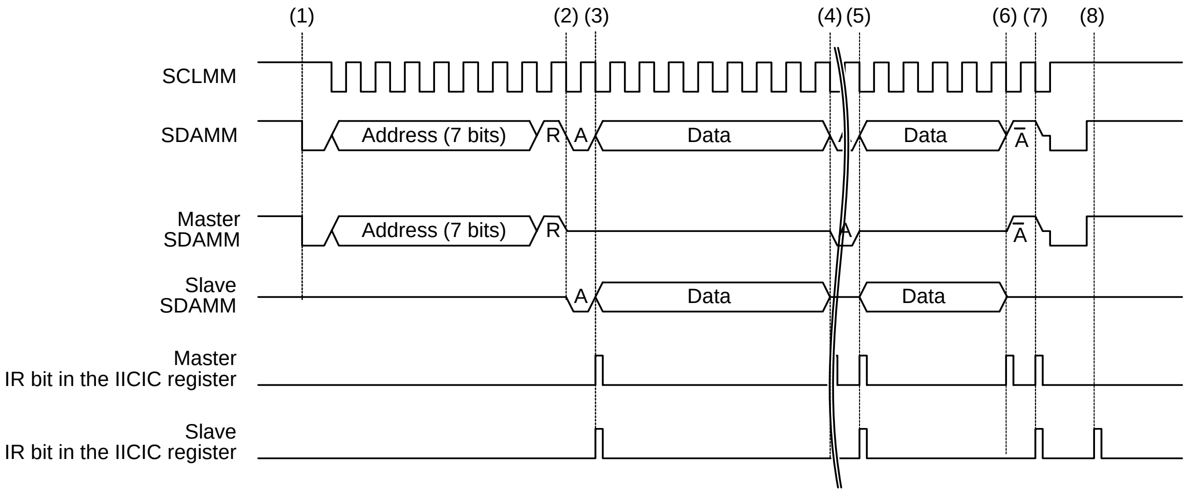


(1) Master: A start condition is generated after writing E0h to the S10 register and transmit data to the S00 register.
(2) Master: The slave address set to bits b7 to b1 in the S00 register and Write("0") set to b0 are output.
Slave: ACK is output when a match is found between the received slave address and the value in the S0Di register.
(3) Master: After ACK reception, the IR bit in the IICIC register becomes 1.
Slave: After ACK transmission, the IR bit becomes 1.
(4) Slave: After data reception, the IR bit becomes 1. During interrupt handling, set the ACKBIT bit to 0 and ACK is output.
(5) Master: After ACK reception, the IR bit becomes 1.
Slave: After ACK transmission, the IR bit becomes 1.
(6) Slave: After receiving 5 bytes, set the ACKBIT bit to 1 and NACK is output.
(7) Master: A stop condition is generated.
The IR bit does not become 1 when a stop condition is generated, by using the countermeasure described in technical update TN-16C-A176A/E.
Slave: When detecting the stop condition, the IR bit becomes 1.

Figure 8.4 Master Transmission and Slave Reception Operation Example

8.4.2 Master Reception and Slave Transmission

Figure 8.5 shows the Master Reception and Slave Transmission Operation



(1) Master: A start condition is generated after writing E0h to the S10 register and transmit data to the S00 register.

(2) Master: The slave address set to bits b7 to b1 in the S00 register and Read("1") set to b0 are output.
Slave: ACK is output when a match is found between the received slave address and the value in the S0Di register.
The TRX bit in the S10 register becomes 1 (transmit mode) (only when the ALS bit in the S1D0 register is 1 (addressing mode)).

(3) Master: After ACK reception, the IR bit in the IICIC register becomes 1.
Slave: After ACK transmission, the IR bit becomes 1.

(4) Slave: After data reception, the IR bit becomes 1. During interrupt handling, set the ACKBIT bit to 0 and ACK is output.

(5) Master: After ACK transmission, the IR bit becomes 1.
Slave: After ACK reception, the IR bit becomes 1.

(6) Slave: After receiving 5 bytes, set the ACKBIT bit to 1 and NACK is output.

(7) Slave: After receiving NACK, the TRX bit becomes 0 (receive mode) only when the ALS bit is 1.

(8) Master: A stop condition is generated.
The IR bit does not become 1 when a stop condition is generated, by using the countermeasure described in technical update TN-16C-A176A/E.
Slave: When detecting the stop condition, the IR bit becomes 1.

Figure 8.5 Master Reception and Slave Transmission Operation

8.5 Function Tables

Declaration	void iic_ini(unsigned char ini, unsigned char sub_address)	
Outline	I^{2}C-bus initialization function	
Argument	Argument name	Meaning
	ini	I^{2}C-bus function enabled/disabled
		ENABLED: I^{2}C-bus function enabled
		DISABLED: I^{2}C-bus function disabled
	sub_address	Slave address setting
Variable (global)	Variable name	Content
	iic_mode	For selecting communication mode
	iic_index	For the number of transfers
Returned value	None	
Function	Argument ini = When ENABLED (I^{2}C-bus function enabled), initialize the I^{2}C-bus before enabling interrupts. Argument ini = When DISABLED (I^{2}C-bus function disabled), disable the I^{2}C-bus interface and the I^{2}C-bus interrupt.	

Declaration	void mode_ini(unsigned char ms, unsigned char sr)	
Outline	Function for setting respective communication modes	
Argument	Argument name	Meaning
	ms	Select master or slave
		MASTER: Master SLAVE: Slave
	sr	Select transmission or reception
		SND: Transmission mode REV: Reception mode
Variable (global)	Variable name	Content
	iic_ram[]	Data storage alignment for master transmit
	iic_length	For transmit and receive size
Returned value	None	
Function	Set the respective communication modes.	

Declaration	unsigned char iic_master_start (unsigned char slave, unsigned char sr, unsigned char *buf, unsigned char len)	
Outline	Master start function	
Argument	Argument name	Meaning
	slave	Specified slave address (0x00 to 0x7f)
	sr	Select transmission or reception
		SND: Transmission mode REV: Reception mode
	*buf	Pointer for transmit buffer
	len	Transmit/receive data size (0x00 to 0xff)
Variable (global)	Variable name	Content
	iic_slave	Variable for storing slave address
	iic_length	For transmit and receive size
	iic_pointer	Pointer for transmission buffer
	iic_mode	For selecting communication mode
	iic_rw	READ/WRITE
Returned value	Type	Meaning
	unsigned char	Master start failure/start successful
		FALSE: Master start failure
		TRUE: Master start successful
Function	Transmit the start condition and slave address after master setting.	

Declaration	void master_transfer(void)	
Outline	Master transmit function	
Argument	None	
Variable (global)	Variable name	Content
	iic_mode	For selecting communication mode
	iic_length	For transmit and receive size
	iic_pointer	Transmit buffer pointer
Returned value	None	
Function	Detect arbitration lost, confirming ACK/NACK reception, and transmitting data.	

Declaration	void master_receive(void)	
Outline	Master receive function	
Argument	None	
Variable (global)	Variable name	Content
	iic_mode	For selecting communication mode
	iic_length	For transmit and receive size
	iic_pointer	Receive buffer pointer
Returned value	None	
Function	Detecting arbitration lost, transmitting ACK/NACK, and receiving data.	

Declaration	void slave_receive(void)	
Outline	Slave receive function	
Argument	None	
Variable (global)	Variable name	Content
	iic_length	For transmit and receive size
	iic_index	Number of transfers
	iic_pointer	Receive buffer pointer
Returned value	None	
Function	Receive data and transmit ACK/NACK.	

Declaration	void slave_transfer(void)	
Outline	Slave transmit function	
Argument	None	
Variable (global)	Variable name	Content
	iic_length	For transmit and receive size
	iic_index	Number of transfers
	iic_pointer	Transmit buffer pointer
Returned value	None	
Function	After receiving ACK/NACK, transmit data.	

Declaration	void idle_mode(void)	
Outline	Transmit and receive mode select function	
Argument	None	
Variable (global)	Variable name	Content
	iic_mode	For selecting communication mode
Returned value	None	
Function	Select transmit mode or receive mode when receiving data.	

Declaration	unsigned char* select_buffer(unsigned char RW)	
Outline	Function for obtaining transmit and receive buffer addresses	
Argument	Variable name	Meaning
	RW	Select transmit and receive buffer
		0: Slave receive buffer 1: Slave transmit buffer
Variable (global)	None	
Returned value	Type	Meaning
	unsigned char*	Transmit and receive buffer address
Function	Obtain transmit and receive buffer addresses.	

Declaration	void receive_stop_condition(void)	
Outline	Stop condition reception state processing function	
Argument	None	
Variable (global)	Variable name	Content
	iic_mode	For selecting communication mode
	iic_index	Number of transfers
Returned value	None	
Function	Clear the stop condition detection interrupt request bit and initialize the communication mode.	

Declaration	void iic_master_end(unsigned char status)	
Outline	Master control completion function	
Argument	Argument name	Meaning
	status	Status after master control
		0x10: Master transmission completed
		0x11: Arbitration lost is detected during master transmission.
		0x12: NACK is received during master transmission
		0x20: Master reception completed
		0x21: Arbitration lost is detected during master reception.
		0x22: NACK is received during master reception
Variable (global)	None	
Returned value	None	
Function	Carry out the processing after master control is completed. This application note does not include any processing. Add if the need arises.	

Declaration	void iic_slave_end(unsigned char status)	
Outline	Slave control completion function	
Argument	Argument name	Meaning
	status	Status after slave control completed
		0x10: Slave transmission completed
Variable (global)	None	
Returned value	None	
Function	Carry out the processing after slave control is completed. This application note does not include any processing. Add if the need arises.	

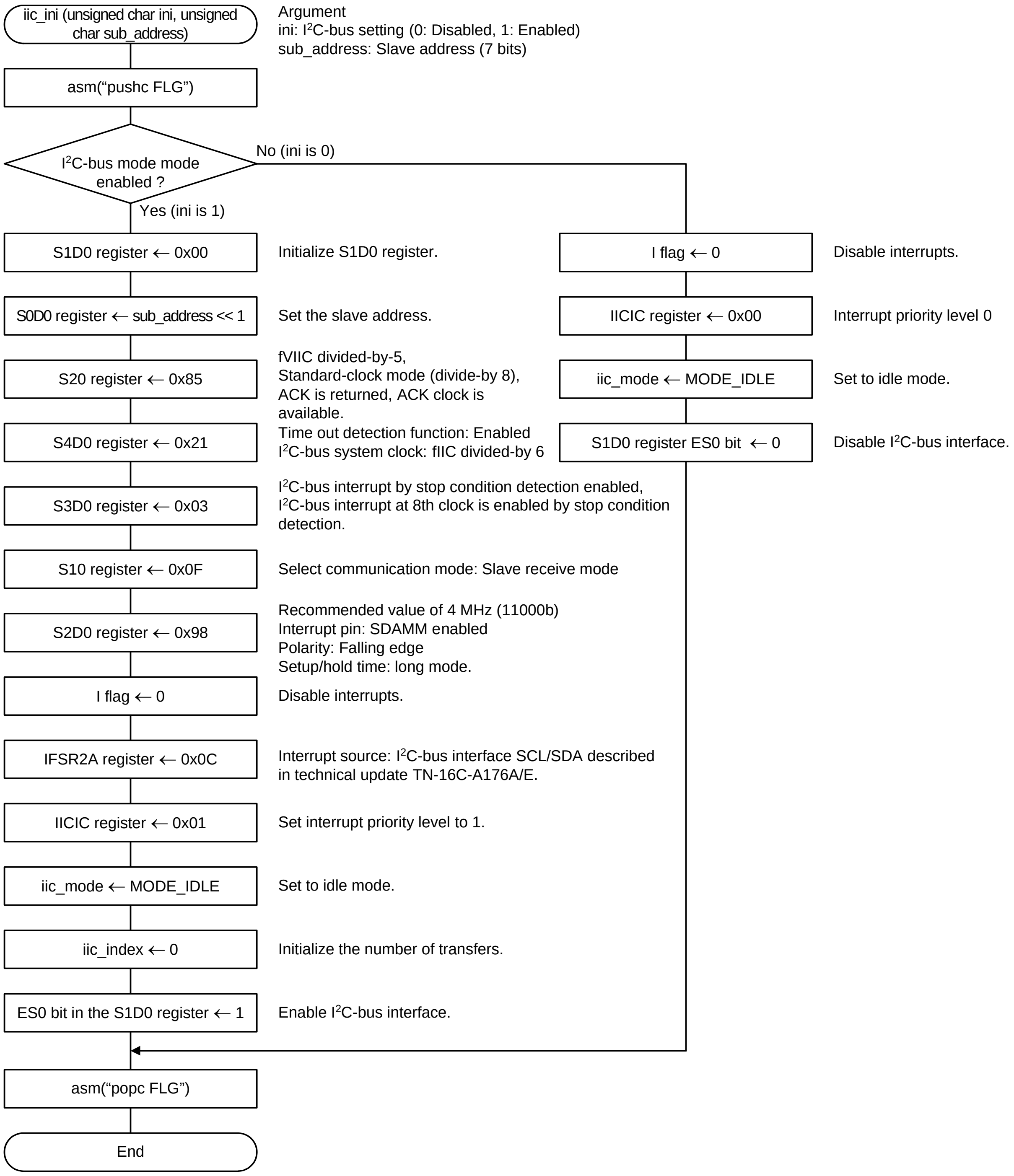
Declaration	void stop_condition(void)
Outline	Stop condition generation function
Argument	None
Variable (global)	None
Returned value	None
Function	A stop condition is generated using the countermeasure described in technical update TN-16C-176A/E.

Declaration	void soft_wait(unsigned int time)	
Outline	Software wait function	
Argument	Argument name	Meaning
	time	Wait time
		SETUP_TIME: Setup time for generating a stop condition WAIT_TIME: Wait time (approx. 5 µs)
Variable (global)	None	
Returned value	None	
Function	Wait time is generated.	

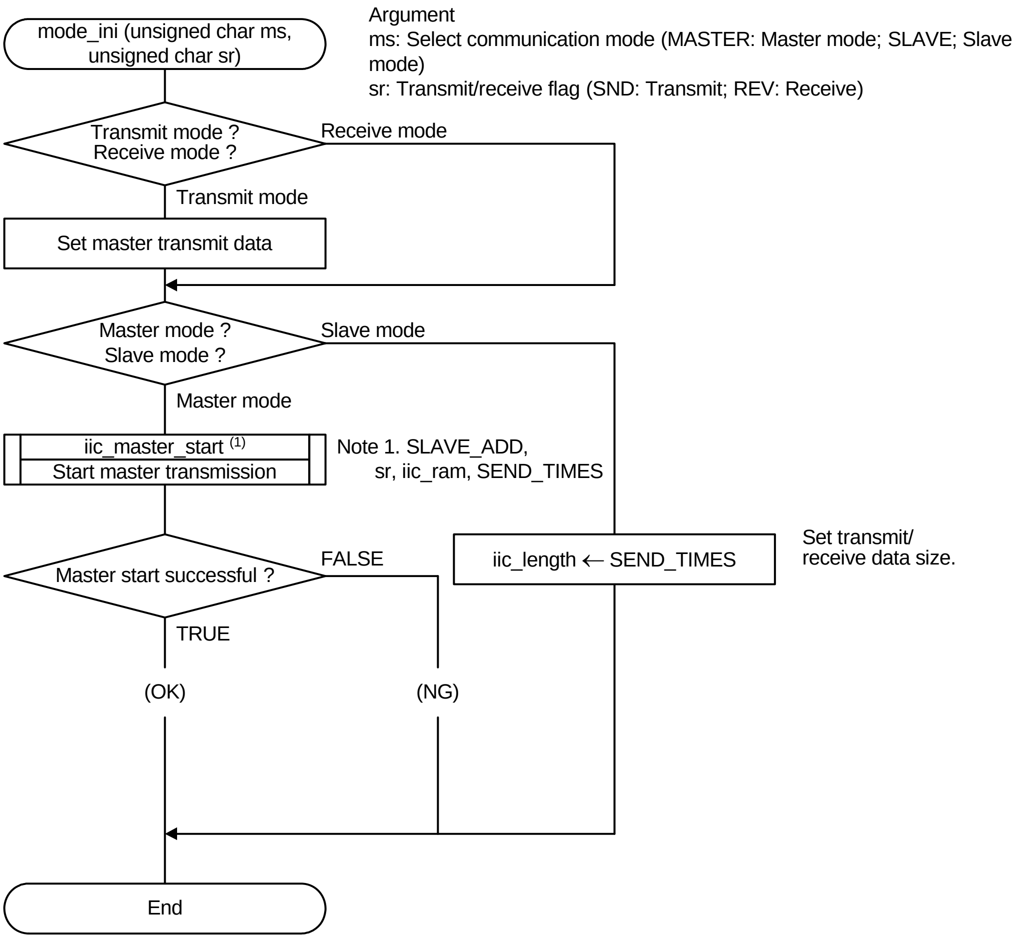
Declaration	void reset_mmi2c(void)
Outline	I^{2}C-bus interface reset function
Argument	None
Variable (global)	None
Returned value	None
Function	Set the SDAMM pin to high and reset the I^{2}C-bus interface.

8.6 Flowcharts

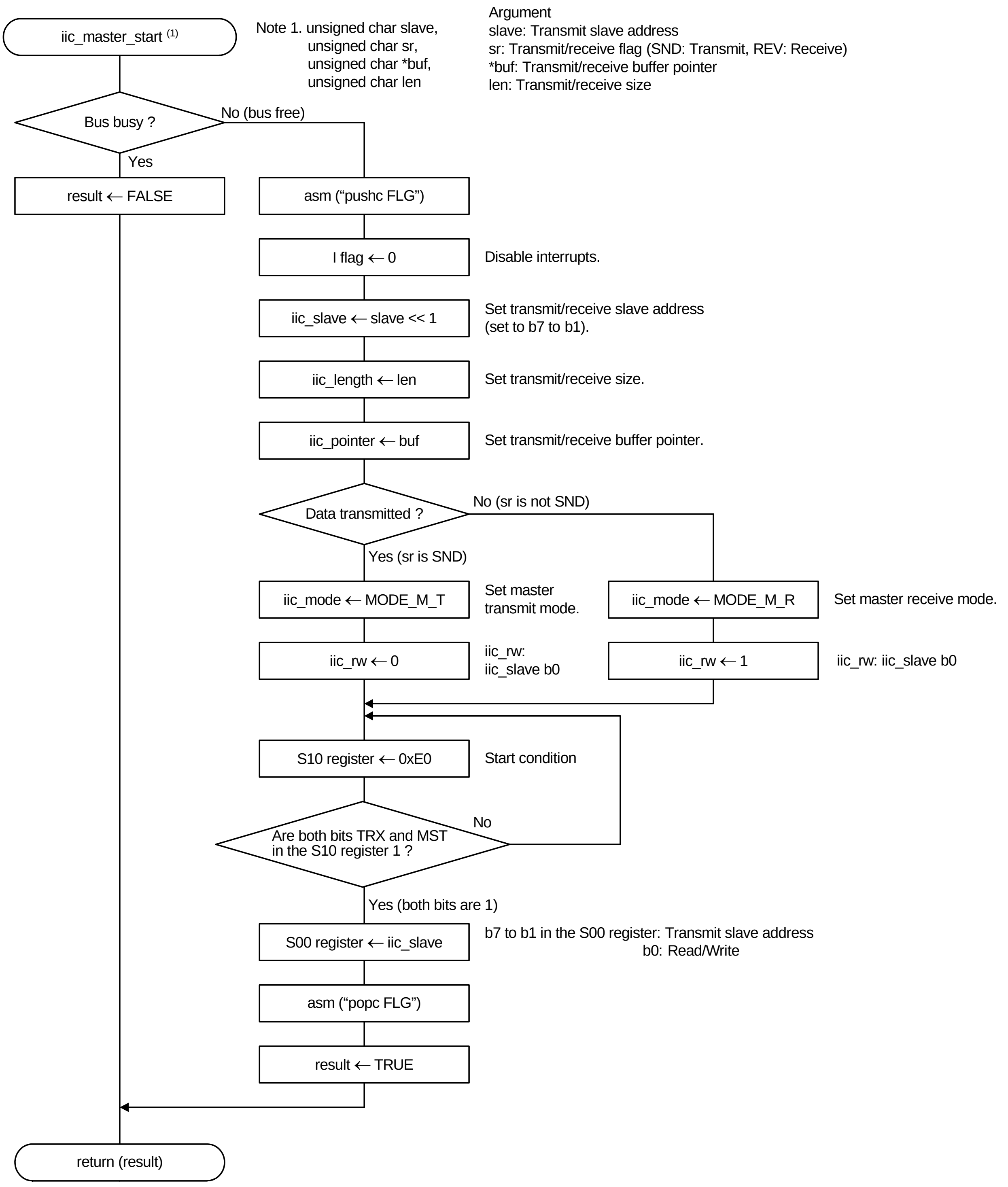
8.6.1 I^{2}C-bus Initialization Function


iic_ini (unsigned char ini, unsigned char sub_address)
Argument
ini: I2C-bus setting (0: Disabled, 1: Enabled)
sub_address: Slave address (7 bits)
asm("pushc FLG")
I2C-bus mode mode enabled ?
No (ini is 0)
Yes (ini is 1)
S1D0 register ← 0x00
Initialize S1D0 register.
S0D0 register ← sub_address << 1
Set the slave address.
S20 register ← 0x85
fVIIC divided-by-5,
Standard-clock mode (divide-by 8),
ACK is returned, ACK clock is available.
S4D0 register ← 0x21
Time out detection function: Enabled
I2C-bus system clock: fIIC divided-by 6
S3D0 register ← 0x03
I2C-bus interrupt by stop condition detection enabled,
I2C-bus interrupt at 8th clock is enabled by stop condition detection.
S10 register ← 0x0F
Select communication mode: Slave receive mode
S2D0 register ← 0x98
Recommended value of 4 MHz (11000b)
Interrupt pin: SDAMM enabled
Polarity: Falling edge
Setup/hold time: long mode.
I flag ← 0
Disable interrupts.
IFSR2A register ← 0x0C
Interrupt source: I2C-bus interface SCL/SDA described in technical update TN-16C-A176A/E.
IICIC register ← 0x01
Set interrupt priority level to 1.
iic_mode ← MODE_IDLE
Set to idle mode.
iic_index ← 0
Initialize the number of transfers.
ES0 bit in the S1D0 register ← 1
Enable I2C-bus interface.
I flag ← 0
Disable interrupts.
IICIC register ← 0x00
Interrupt priority level 0
iic_mode ← MODE_IDLE
Set to idle mode.
S1D0 register ES0 bit ← 0
Disable I2C-bus interface.
asm("popc FLG")
End

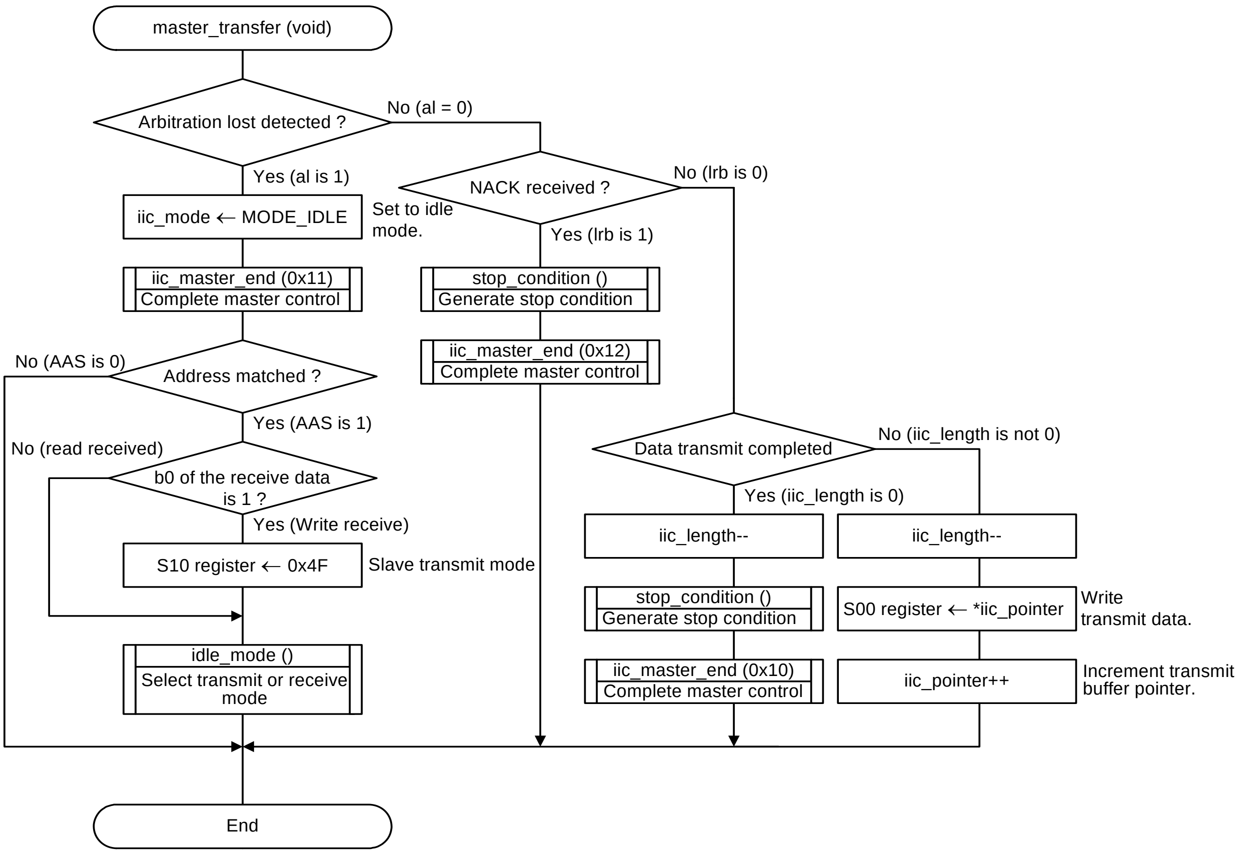
8.6.2 Function for Setting Respective Communication Modes


mode_ini (unsigned char ms, unsigned char sr)
Argument
ms: Select communication mode (MASTER: Master mode; SLAVE; Slave mode)
sr: Transmit/receive flag (SND: Transmit; REV: Receive)
Transmit mode ? Receive mode ?
Receive mode
Transmit mode
Set master transmit data
Master mode ? Slave mode ?
Slave mode
Master mode
iic_master_start (1)
Start master transmission
Note 1. SLAVE_ADD, sr, iic_ram, SEND_TIMES
iic_length ← SEND_TIMES
Set transmit/ receive data size.
Master start successful ?
FALSE
TRUE
(OK)
(NG)
End

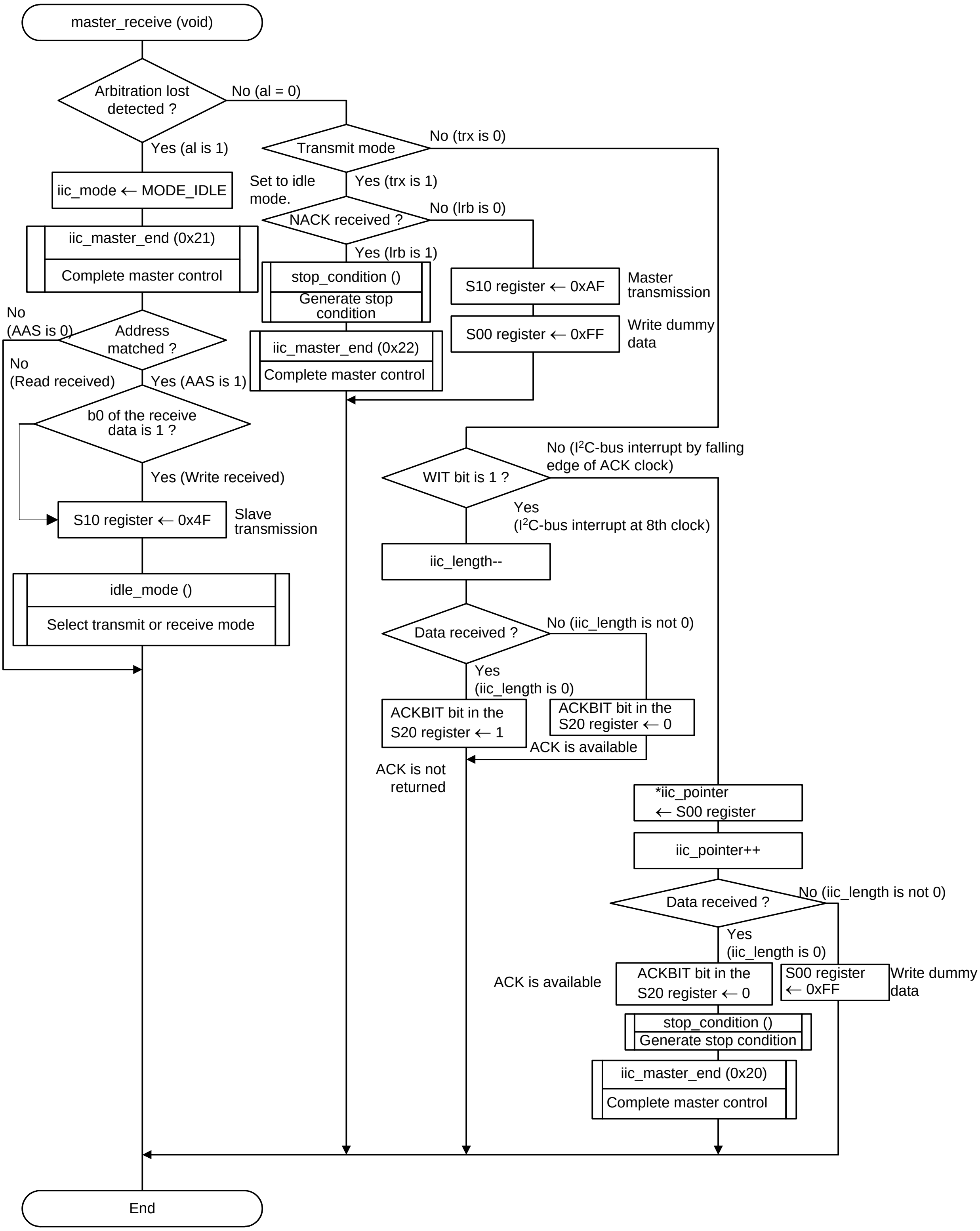
8.6.3 Master Start Function



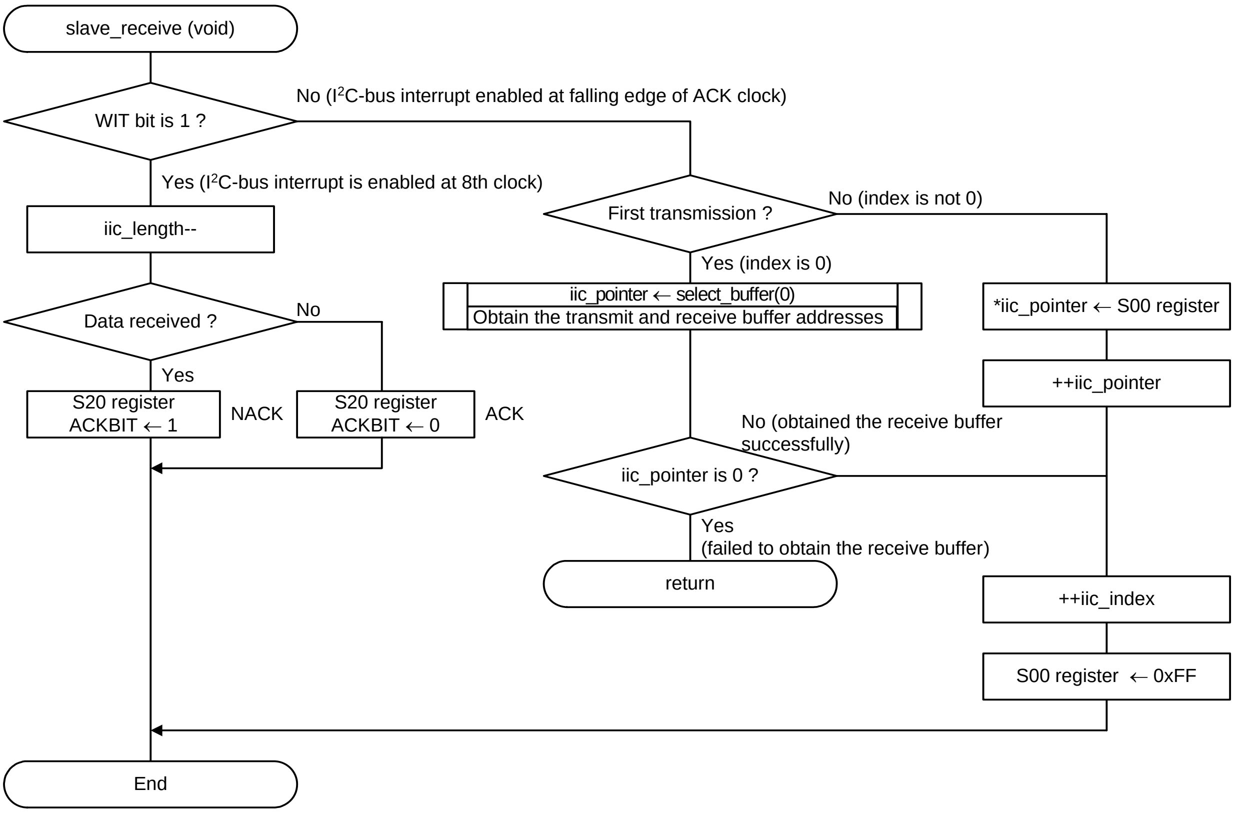
8.6.4 Master Transmit Function



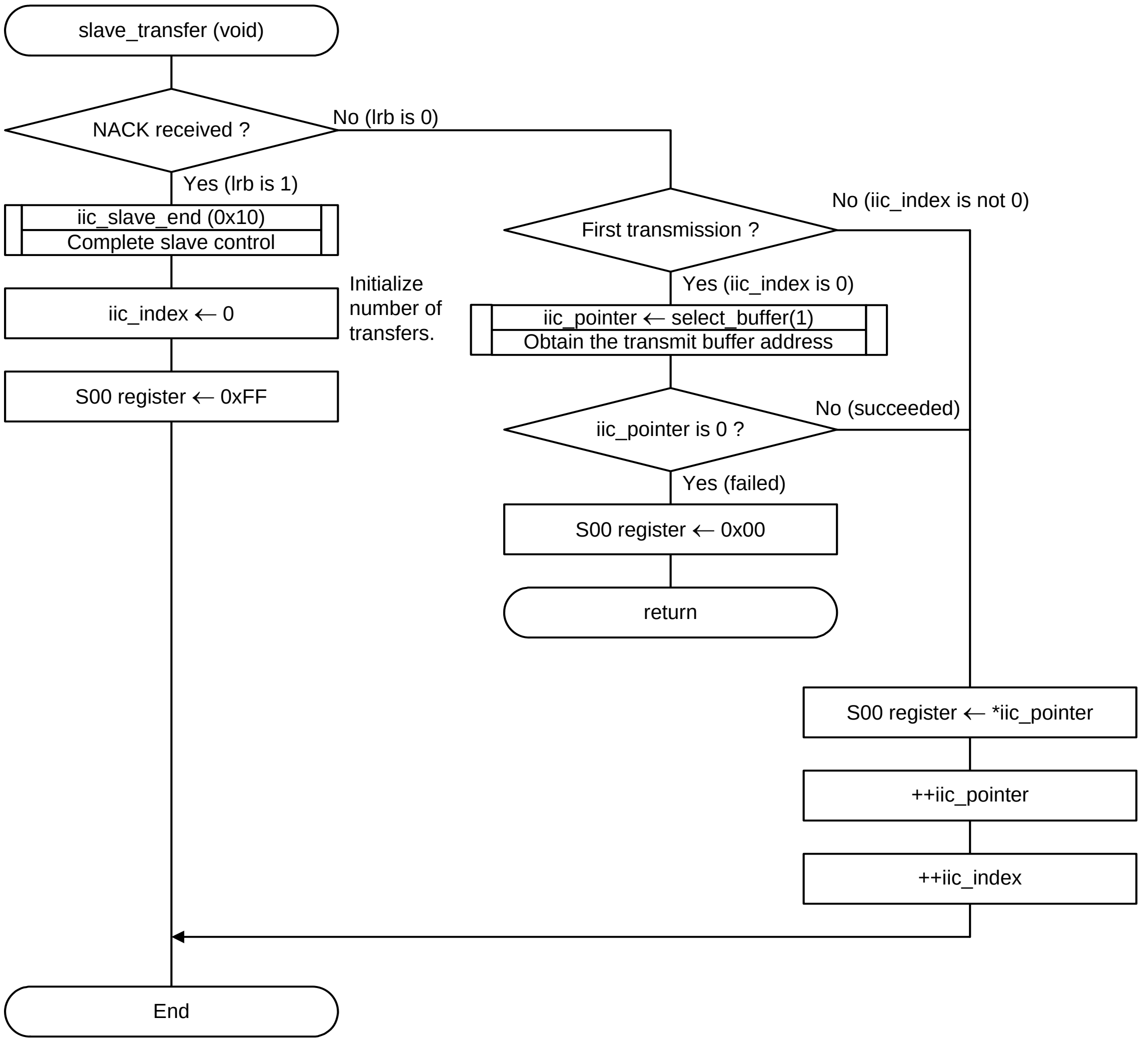
8.6.5 Master Receive Function



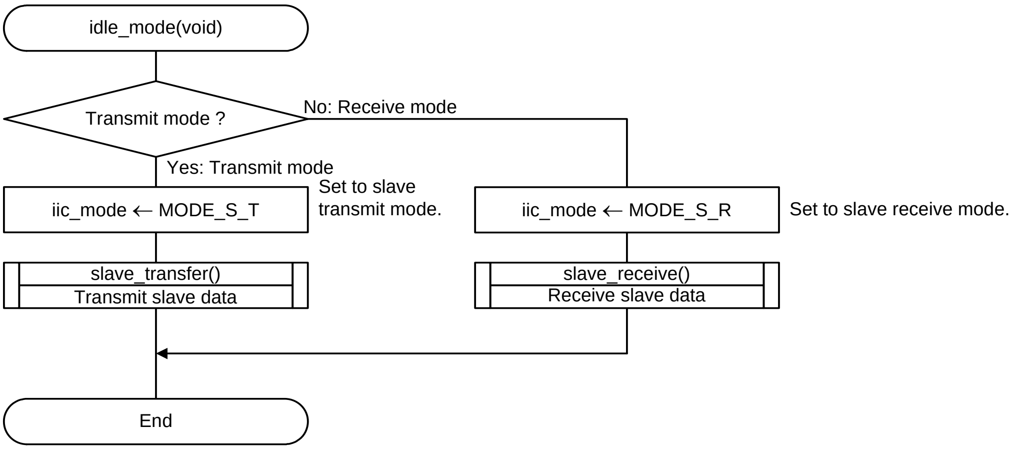
8.6.6 Slave Receive Function


slave_receive (void)
WIT bit is 1 ?
No (I²C-bus interrupt enabled at falling edge of ACK clock)
Yes (I²C-bus interrupt is enabled at 8th clock)
iic_length--
Data received ?
No
Yes
S20 register ACKBIT ← 1
NACK
S20 register ACKBIT ← 0
ACK
First transmission ?
No (index is not 0)
Yes (index is 0)
iic_pointer ← select_buffer(0)
Obtain the transmit and receive buffer addresses
*iic_pointer ← S00 register
++iic_pointer
No (obtained the receive buffer successfully)
iic_pointer is 0 ?
Yes
(failed to obtain the receive buffer)
return
++iic_index
S00 register ← 0xFF
End

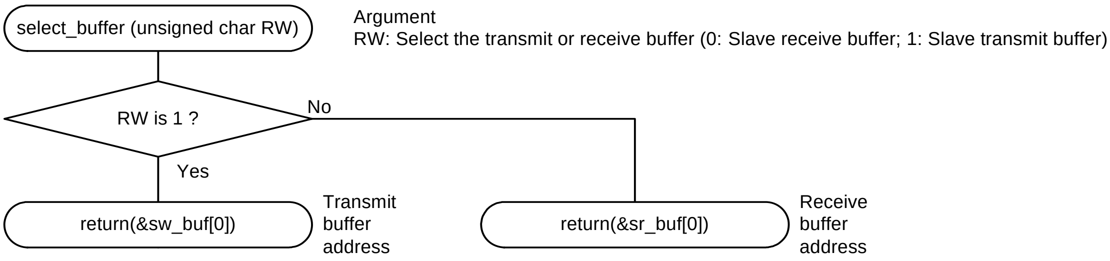
8.6.7 Slave Transmit Function



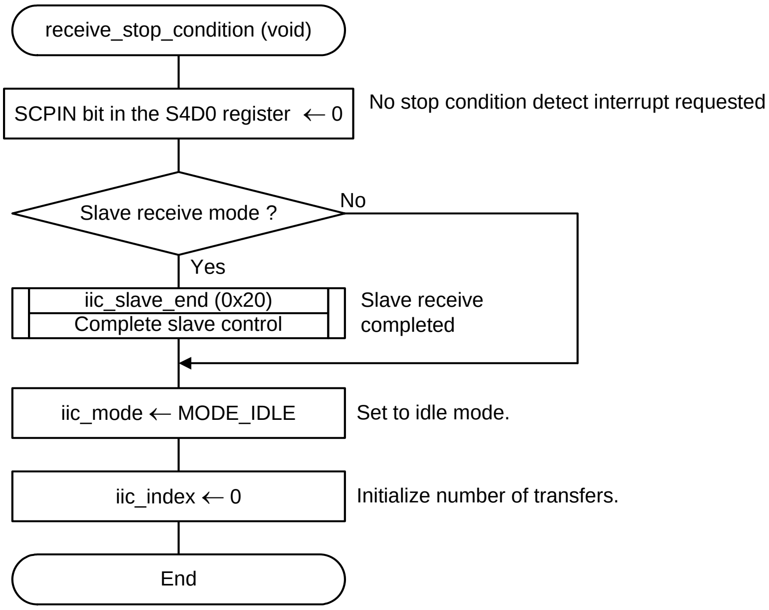
8.6.8 Transmit and Receive Mode Select Function



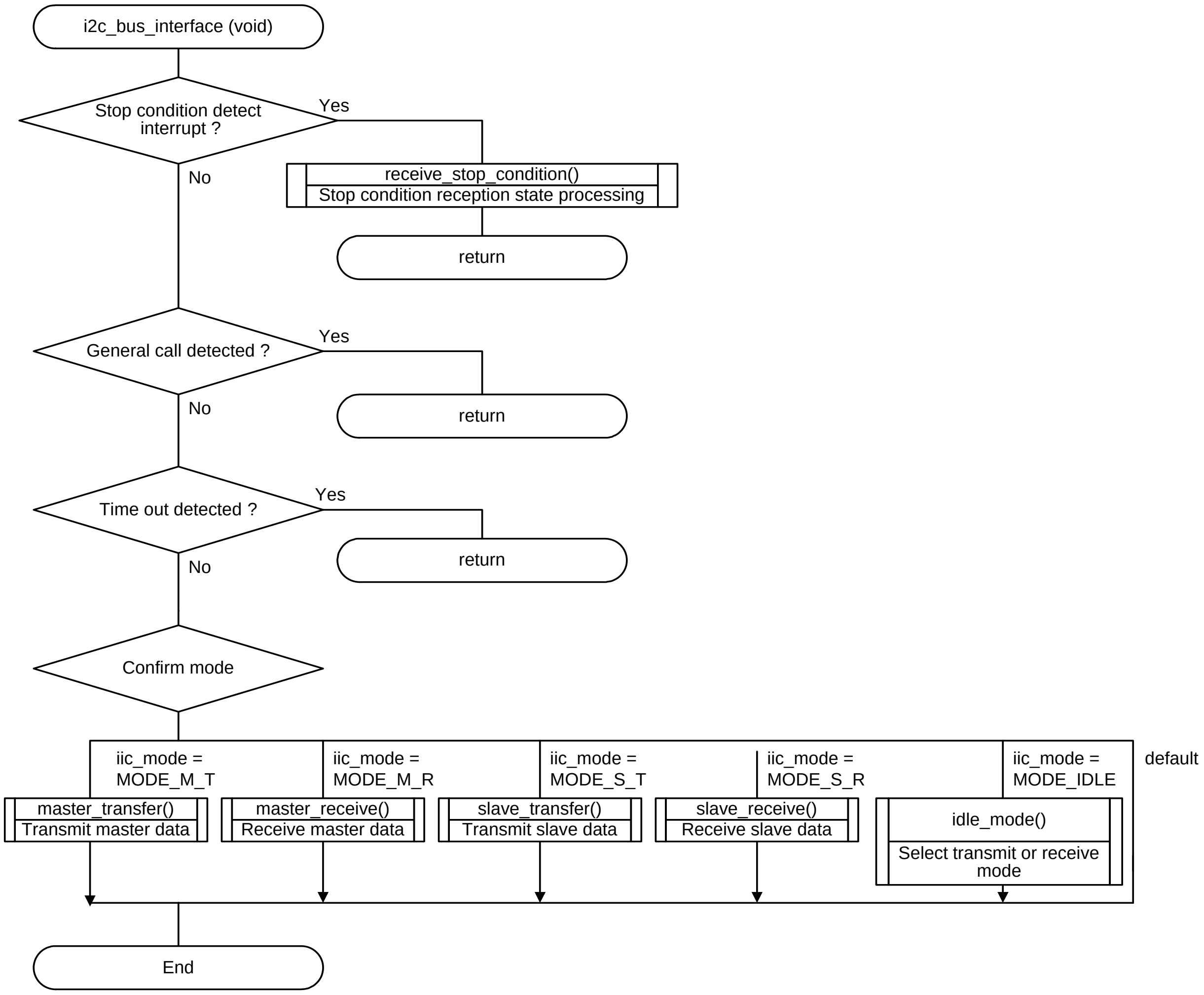
8.6.9 Function for Obtaining Transmit and Receive Buffer Addresses


select_buffer (unsigned char RW)
Argument
RW: Select the transmit or receive buffer (0: Slave receive buffer; 1: Slave transmit buffer)
RW is 1 ?
No
Yes
return(&sw_buf[0])
Transmit buffer address
return(&sr_buf[0])
Receive buffer address

8.6.10 Stop Condition Reception State Processing Function



8.6.11 I^{2}C-bus Interface Interrupt Handling



9. Sample Program

A sample program can be downloaded from the Renesas Technology website.
To download, click “Application Notes” in the left-hand side menu of the M16C Family page.

10. Reference Documents

Hardware Manuals
M16C/64A Group Hardware Manual
M16C/65 Group Hardware Manual
The latest version can be downloaded from the Renesas Technology website.

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

Website and Support

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

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

REVISION HISTORY	M16C/64A, M16C/65 Group Multi-Master I^{2}C-bus Interface

Rev.	Date	Description	
		Page	Summary
1.00	Nov 30, 2009	–	First Edition issued
1.01	Dec 01, 2009	4	“4.1 Initial Settings” (3),(4) enabled → disabled

All trademarks and registered trademarks are the property of their respective owners

.

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.