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.

RENESAS

Application Note

Multimedia Processor for Mobile Applications

IIC Interface

EMMA Mobile1

Document No. S19905EJ1V0AN00
Date Published Aug.2009

PREFACE

Purpose The purpose of this document is to specify the usage of the IIC interface.

Organization This document includes the following:

- Introduction
- Usage of IIC Interface
- Example of IIC Operation
- IIC Driver Function

Notation Here explains the meaning of following words in text:

Note	Explanation of item indicated in the text
Caution	Information to which user should afford special attention
Remark	Supplementary information

Related document The following tables list related documents.

Reference Document

Document Name	Version/date	Author	Description
S19265EJ1V0UM00_ASMUGIO.pdf	1st edition	NECEL	ASMU/GIO User's Manual
S19268EJ1V0UM00_1chip.pdf	1st edition	NECEL	1 Chip User's Manual
S19256EJ1V0UM00_I2C.pdf	1st edition	NECEL	I2C User's Manual
ak4648_f01e.pdf	1st edition	AKM	AK4648 User's Manual
S19907EJ1V0AN00_GD.pdf	1st edition	NECEL	GD Spec

Disclaimers

- **The information contained in this document is subject to change without prior notice in the future. Refer to the latest applicable data sheet(s) and user's manual when designing a product for mass production.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC Electronics products listed in this documents or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customers' equipment shall be done under the full responsibility of the customer. NEC Electronics assume no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

Note)

1. "NEC Electronics" as used in this document means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
2. "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above)
3. All trademarks or registered trademarks are the property of their respective owners. Registered trademarks ® and trademarks™ are not noted in this document.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Outline

This document introduces how to use the IIC interface of EMMA Mobile1, including the following functions:

- How to write data as a master and slave.
- How to read data as a master and slave.

The IIC interface has the following functions: two channels, conformance to the I2C bus format (1995 updated version of Philips specification), automatic serial data identification, address-based chip selection, wake-up operation, acknowledge (ACK) transmission, wait (WAIT) notification. About the details of functions, please refer to "**CHAPTER 1 OVERVIEW**" of I2C user's manual.

1.2 Development Environment

- Hardware environment of this project is listed as below.

Table 1-1 Hardware Environment

Name	Version	Maker
EMMA Mobile1 evaluation board (PSKCH2Y-S-0016-01)	-	NEC Electronics
PARTNER-Jet ICE ARM	M20	Kyoto Microcomputer Co. Ltd

- Software used in this project is listed as below.

Table 1-2 Software Environment

Name	Version	Maker
GNUARM Toolchain	V4.3.2	GNU
WJETSET-ARM	V5.10a	Kyoto Microcomputer Co. Ltd

Chapter 2 Usage of IIC Interface

2.1 IIC Overview

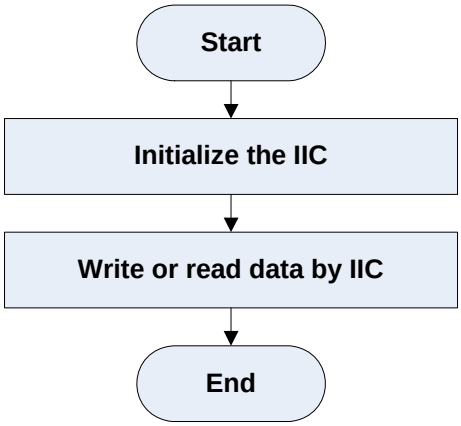

Figure 2-1 Overview of IIC

2.2 IIC Detail

(1). Initialize the IIC.

- Open the clock

The related registers are as follow:

ASMU_GCLKCTRL2ENA;

ASMU_GCLKCTRL2;

- Reset the IIC

The related registers are as follow:

ASMU_RESETREQ1ENA;

ASMU_RESETREQ1;

- Set the divisor

The related registers are as follow:

ASMU_DIVIICSCLK;

- Port setting

The related registers are as follow:

For port setting:

CHG_PINSEL_G80;

CHG_PINSEL_IIC2;

For input attribute setting:

CHG_PULL_G80;

CHG_PULL0;

For driver capability setting:

CHG_DRIVE0;

CHG_DRIVE1;

About the details, please refer to the "**Appendix A.2.1 Initialize**" of this document.

For port setting, the following table is the alternate port table:

Table 2-1 Alternate Port

Port Name	Function1	Function2
IIC_SCL	GIO_P83	IIC_SCL
IIC_SDA	GIO_P84	IIC_SDA
IIC2_SCL	IIC2_SCL	NAND_WE
IIC2_SDA	IIC2_SDA	NAND_RB0

- Wait control and read mode setting

The related registers are as follow:

For wait control register:

ASMU_AB1_IICWAITCTRL;

ASMU_AB1_IIC2WAITCTRL;

For read mode:

ASMU_AB1_IICREADCTRL;

ASMU_AB1_IIC2READCTRL;

- Mode selection and frequency setting

The related registers are as follow:

For wait control register:

IIC_IICCL0;

IIC_IICCL2;

(2). Write and read data by IIC

For write and read data, the related registers are as follow:

IIC_IIC0;

IIC_IICC0;

IIC_SVA0;

IIC_IICCL0;

IIC_IICSE0;

IIC_IICF0;

IIC_IIC2;

IIC_IICC2;

IIC_SVA2;

IIC_IICCL2;

IIC_IICSE2;

IIC_IICF2;

Caution:

EMMA Mobile1 IIC interface conformances the 1995 updated version IIC standard. So it only supports two modes:

Standard mode: maximum transfer rate 100 kbps.

High speed mode: maximum transfer rate 341 kbps.

Chapter 3 Example of IIC Operation

This following contents show 4 examples about IIC interface:

- How to write data as a master.
- How to read data as a master.
- How to write data as a slave.
- How to read data as a slave.

3.1 Example of Write as a Master

The hardware connection of write and read as a master is as follow figure.

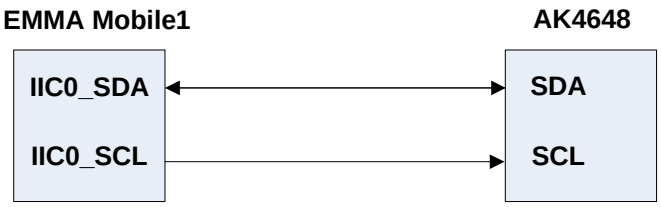

Figure 3-1 Hardware Connection of Write and Read as a Master

The AK4648 is a stereo CODEC. It supports the IIC interface (max: 400KHz). EMMA Mobile1 write data to AK4648 registers and read data from AK4648 registers with master mode by IIC interface.
About the details of AK4648, please refer to the AK4648 user's manual.

3.1.1 Operation Flow

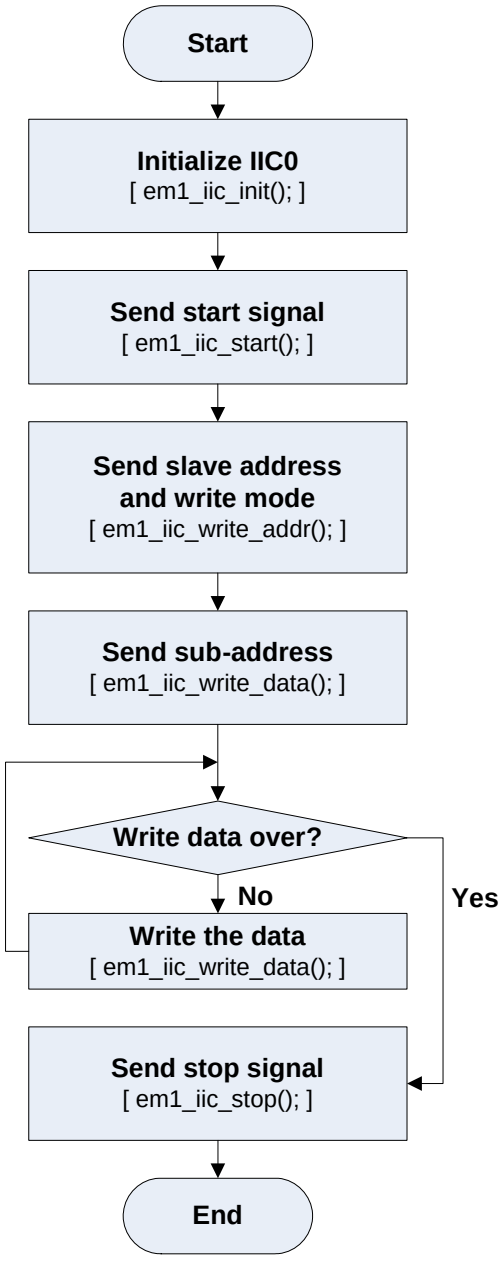

Figure 3-2 Example of Write as a Master

About the IIC function, please refer to the "**Appendix A IIC Driver Function**".

3.1.2 Operation Detail

(1). Initialize IIC0.
The process calls the em1_iic_init().
The em1_iic_init() finishes the following functions for IIC0:

- Open IIC_CLK.
  ASMU_GCLKCTRL2ENA[3] = 1;
  ASMU_GCLKCTRL2[3] = 1;
  ASMU_GCLKCTRL2ENA[3] = 0;

- Open IIC_SCLK.
  ASMU_GCLKCTRL2ENA[4] = 1;
  ASMU_GCLKCTRL2[4] = 1;
  ASMU_GCLKCTRL2ENA[4] = 0;

- Reset IIC0.
  ASMU_RESETREQ1ENA[26] = 1;
  ASMU_RESETREQ1[26] = 1;
  ASMU_RESETREQ1ENA[26] = 0;

- Reset release IIC0.
  ASMU_RESETREQ1ENA[26] = 1;
  ASMU_RESETREQ1[26] = 0;
  ASMU_RESETREQ1ENA[26] = 0;

- Set divisor.
  ASMU_DIVIICSCLK[7:0] = Divisor;

Divisor is an input value and it should be set according to I2C user's manual. Details about how to set the Divisor please refer to the "**Chapter 3.2.43 IIC_SCLK frequency division setting register**" of ASMU/GIO user's manual.

- Switch port to IIC functions.
  CHG_PINSEL_G80[7:6] = 0x1; // GIO_P83 --> IIC_SCL.
  CHG_PINSEL_G80[9:8] = 0x1; // GIO_P84 --> IIC_SDA.

- Disable input and disable pull up/down.
  CHG_PULL_G80[14:12] = 0x1;
  CHG_PULL_G80[18:16] = 0x1;

- Set driver capability: 4mA.
  CHG_DRIVE0[31:30] = 0x1;

- Set wait control time and read mode.

```
ASMU_AB1_IICWAITCTRL = 0xF1F0F;
ASMU_AB1_IICREADCTRL = 0;
```

- Set high speed mode.

```
IIC_IICCL0[3] = 0x1;          // Set the high speed mode;
IIC_IICCL0[2] = 0x1;          // Enable digital filtering;
IIC_IICCL0[1:0] = 0x0;        // Set the clock frequency 24 divisor;
```

(2). Send start signal.

The process calls the em1_iic_start().

The em1_iic_start() finishes the following functions for IIC0:

- Issue start signal without stop detection.

```
IIC_IICF0[1] = 0x1;
```

- Enable communication reservation.

```
IIC_IICF0[0] = 0x0;
```

- Enable IIC0.

```
IIC_IICC0[7] = 0x1;
```

- Enable stop interrupt.

```
IIC_IICC0[4] = 0x1;
```

- Set interrupt generated at the 9th falling edge of the clock.

```
IIC_IICC0[3] = 0x1;
```

- Wait IIC bus free.

If IIC_IICF0[6] is 0, the IIC0 bus is free.

- Issue start signal.

```
IIC_IICC0[1] = 0x1;
```

- Wait start signal issued;

If IIC_IICF0[7] is 0, the start signal is issued.

(3). Send slave address and write mode.

The process calls the em1_iic_write_addr().

The em1_iic_write_addr() finishes the following functions for IIC0:

- Set interrupt generated at the 9th falling edge of the clock.

```
IIC_IICC0[3] = 0x1;
```

- Disable ACK signal.
  IIC_IICC0[2] = 0x0;

- Send address.
  IIC_IIC0 = address;

- Detect ACK signal.
  If the IIC_IICSE0[10] is 1, detect an ACK signal.

(4). Send sub-address.

The process calls the em1_iic_write_data().

The em1_iic_write_data() finishes the following functions for IIC0:

- Check the transmission state.
  If the IIC_IICSE0[11] is 1, the IIC is in transmission state.

- Set interrupt generated at the 9th falling edge of the clock.
  IIC_IICC0[3] = 0x1;

- Disable the ACK signal.
  IIC_IICC0[2] = 0x0;

- Send the address.
  IIC_IIC0 = data;

- Detect the ACK signal.
  If the IIC_IICSE0[10] is 1, detect an ACK signal.

(5). Write the data.

The process calls the em1_iic_write_data().

The em1_iic_write_data() finishes the functions for IIC0 as above process (4).

(6). Send stop signal.

The process calls the em1_iic_stop().

The em1_iic_stop() finishes the following functions for IIC0:

- Enable the IIC.
  IIC_IICC0[7] = 0x1;

- Enable stop interrupt.
  IIC_IICC0[4] = 0x1;

- Set interrupt generated at the 9th falling edge of the clock.

  IIC_IICC0[3] = 0x1;

- Issue the stop signal.

  IIC_IICC0[0] = 0x1;

- Detect the stop signal.

  If the IIC_IICSE0[8] is 1, the stop signal is detected.

3.2 Example of Read as a Master

3.2.1 Operation Flow

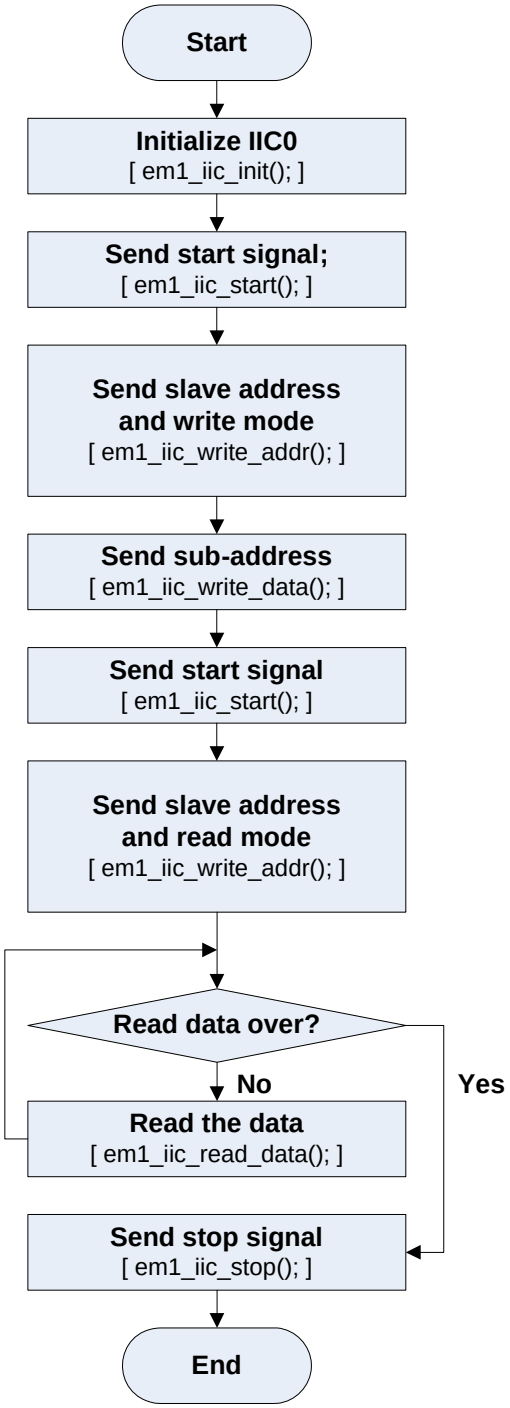

Figure 3-3 Example of Read as a Master

About the IIC function, please refer to the "**Appendix A IIC Driver Function**".

3.2.2 Operation Detail

(1). Initialize IIC0.

The process calls the em1_iic_init().

The em1_iic_init() finishes the functions for IIC0 as process (1) of "**Chapter 3.1.2 Operation Detail**".

But in the read example, IIC0 is stander mode.

- Set standard mode:

```
IIC_IICCL0[3] = 0x0;          // Set the standard mode.
IIC_IICCL0[1:0] = 0x0;        // Set the clock frequency 44 divisor.
```

(2). Send start signal.

The process calls the em1_iic_start().

The em1_iic_start() finishes the functions for IIC0 as process (2) of "**Chapter 3.1.2 Operation Detail**".

(3). Send slave address and write mode.

The process calls the em1_iic_write_addr().

The em1_iic_write_addr() finishes the functions for IIC0 as process (3) of "**Chapter 3.1.2 Operation Detail**".

(4). Send sub-address.

The process calls the em1_iic_write_data().

The em1_iic_write_data() finishes the functions for IIC0 as process (4) of "**Chapter 3.1.2 Operation Detail**".

(5). Send start signal.

The process calls the em1_iic_start().

The em1_iic_start() finishes the functions for IIC0 as above process (2).

(6). Send slave address and read mode.

The process calls the em1_iic_write_addr().

The em1_iic_write_addr() finishes the functions for IIC0 as above process (3).

(7). Read the data.

The process calls the em1_iic_read_data().

The em1_iic_read_data() finishes the following functions for IIC0:

- Check the reception state.

  If the IIC_IICSE0[11] is 0, the IIC is in reception state.

- Set Interrupt is generated at the 8th falling edge of the clock.

  IIC_IICC0[3] = 0x0;

- Enable the ACK signal.

  IIC_IICC0[2] = 0x1;

- Cancel the wait state.

  IIC_IICC0[5] = 0x1;

- Receive the data.

  data = IIC_IIC0;

(8). Send stop signal.

The process calls the em1_iic_stop().

The em1_iic_stop() finishes the functions for IIC0 as process (6) of "**Chapter 3.1.2 Operation Detail**".

3.3 Example of Write as a Slave

The hardware connection of write and read as a slave is as follow figure.

Figure 3-4 Hardware Connection of Write and Read as a Slave

3.3.1 Operation Flow

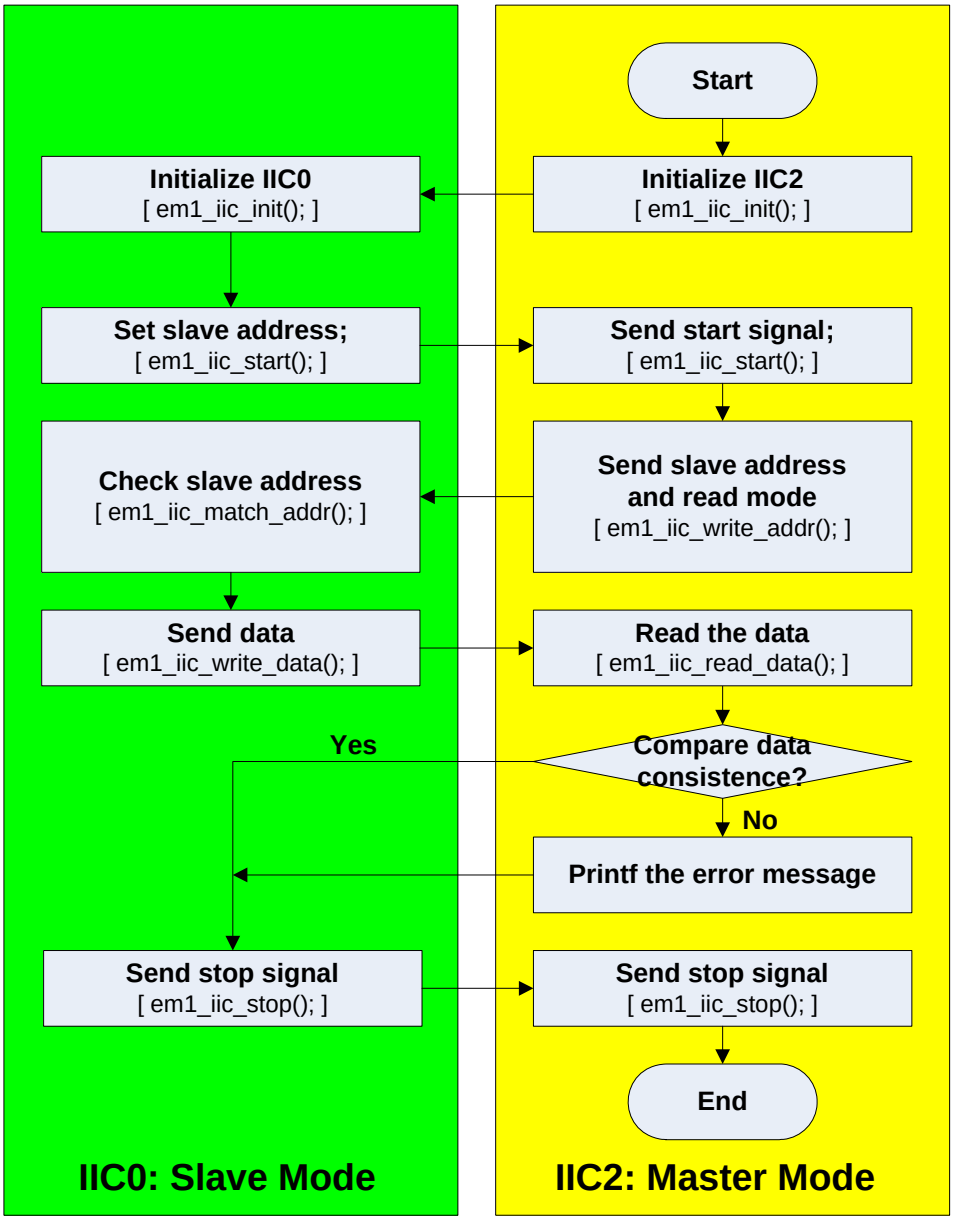

Figure 3-5 Example of Write as a Slave

In the above figure, green part is slave mode of IIC0, yellow part is master mode of IIC2.

The following contents only describe slave mode. About the master mode, please refer to the "**Chapter 3.2 Example of Read as a Master**".

About the IIC function, please refer to the "**Appendix A IIC Driver Function**".

3.3.2 Operation Detail

(1). Initialize IIC0.

The process calls the em1_iic_init().

The em1_iic_init() finishes the functions for IIC0 as process (1) of "**Chapter 3.2.2 Operation Detail**".

(2). Set slave address.

The process calls the em1_iic_start().

The finishes the functions for IIC0 as process (4) of "**Chapter 3.1.2 Operation Detail**".

- Set Interrupt is generated at the 8th falling edge of the clock.

  IIC_IICC0[3] = 0x0;

- Enable the ACK signal.

  IIC_IICC0[2] = 0x1;

- Cancel the wait state.

  IIC_IICC0[5] = 0x1;

- Set the slave address.

  IIC_SVA0 = address;

- Enable IIC0.

  IIC_IICC0[7] = 0x1;

(3). Check slave address.

The process calls the em1_iic_match_addr().

The em1_iic_match_addr() finishes the following functions for IIC0:

- Check the transmission state.

  If the IIC_IICSE0[12] is 1, an address match.

  If the IIC_IICSE0[13] is 1, an extension code match.

(4). Write the data.

The process calls the em1_iic_write_data().

The em1_iic_write_data() finishes the functions for IIC0 as process (4) of "**Chapter 3.1.2 Operation Detail**".

(5). Send stop signal.

The process calls the em1_iic_stop().

The em1_iic_stop() finishes the following functions for IIC0:

- End the communication and set wait state.

  IIC_IICC0[6] = 0x1;

3.4 Example of Read as a Slave

3.4.1 Operation Flow

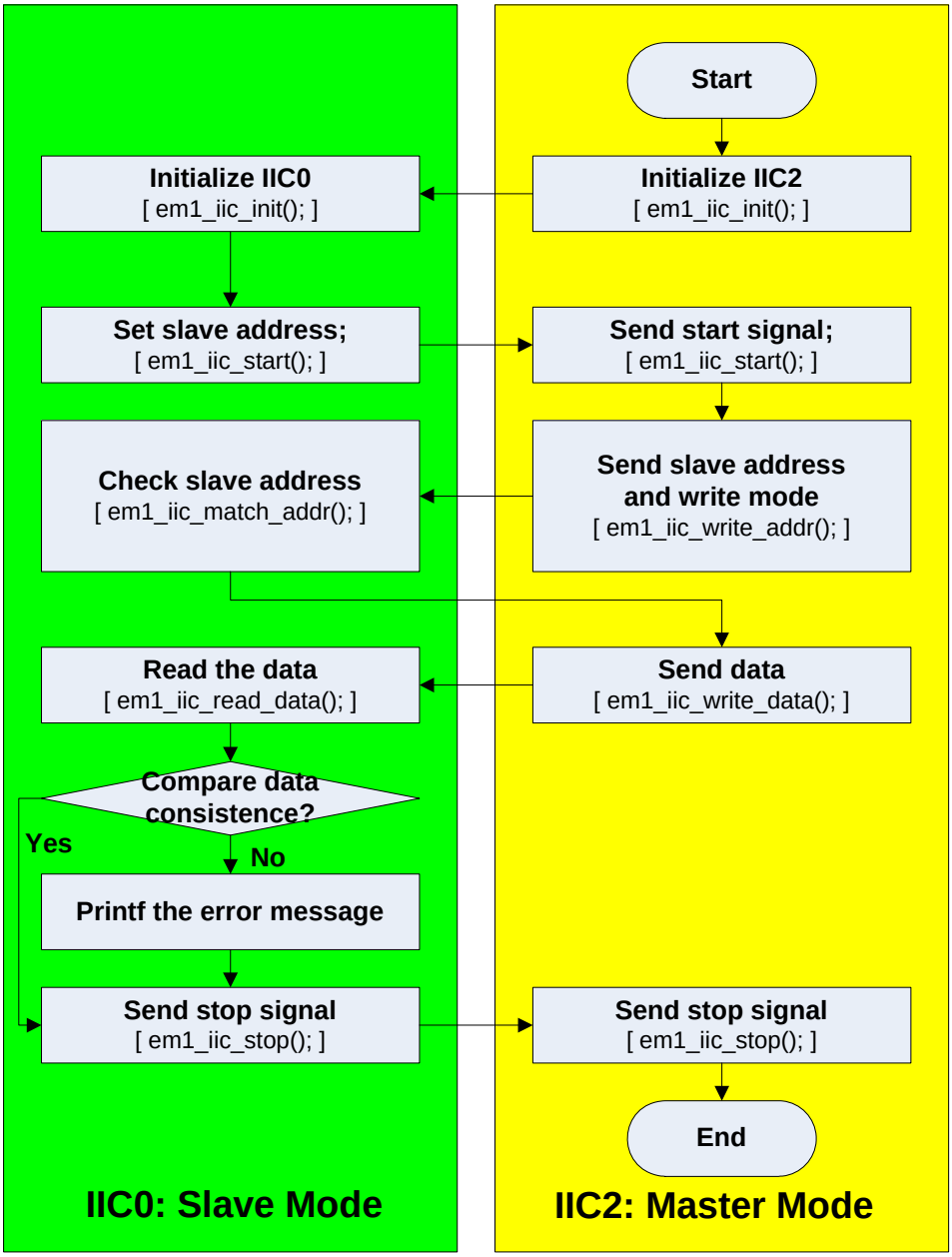

Figure 3-6 Example of Read as a Slave

In the above figure, green part is slave mode of IIC0, yellow part is master mode of IIC2.

The following contents only describe slave mode. About the master mode, please refer to the "**Chapter 3.1 Example of Write as a Master**".

About the IIC function, please refer to the "**Appendix A IIC Driver Function**".

3.4.2 Operation Detail

(1). Initialize IIC0.

The process calls the em1_iic_init().

The em1_iic_init() finishes the functions for IIC0 as process (1) of "**Chapter 3.2.2 Operation Detail**".

(2). Set slave address.

The process calls the em1_iic_start().

The em1_iic_start() finishes the functions for IIC0 as process (2) of "**Chapter 3.3.2 Operation Detail**".

(3). Check the received address.

The process calls the em1_iic_match_addr().

The em1_iic_match_addr() finishes the functions for IIC0 as process (3) of "**Chapter 3.3.2 Operation Detail**".

(4). Read the data.

The process calls the em1_iic_read_data().

The em1_iic_read_data() finishes the functions for IIC0 as process (7) of "**Chapter 3.2.2 Operation Detail**".

(5). Send stop signal.

The process calls the em1_iic_stop().

The em1_iic_stop() finishes the functions for IIC0 as process (5) of "**Chapter 3.3.2 Operation Detail**".

Appendix A IIC Driver Function

A.1 IIC Driver Function List

The following table shows the IIC driver interface functions.

Table A-1 IIC Driver Function List

Class	Function Name	Function Detail
Driver Function	em1_iic_init;	Initialize the iic.
	em1_iic_start;	Send the start signal.
	em1_iic_stop;	Send the stop signal.
	em1_iic_write_addr;	Write address.
	em1_iic_write_data;	Write data.
	em1_iic_read_data;	Read data.
	em1_iic_match_addr;	Check slave address.

A.2 IIC Driver Function Detail

A.2.1 Initialize

[Function Name]

em1_iic_init

[Format]

DRV_RESULT em1_iic_init(uchar num, uchar mode);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number 0: IIC0 device 2: IIC2 device
mode	uchar	I	Standard mode or high speed mode

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

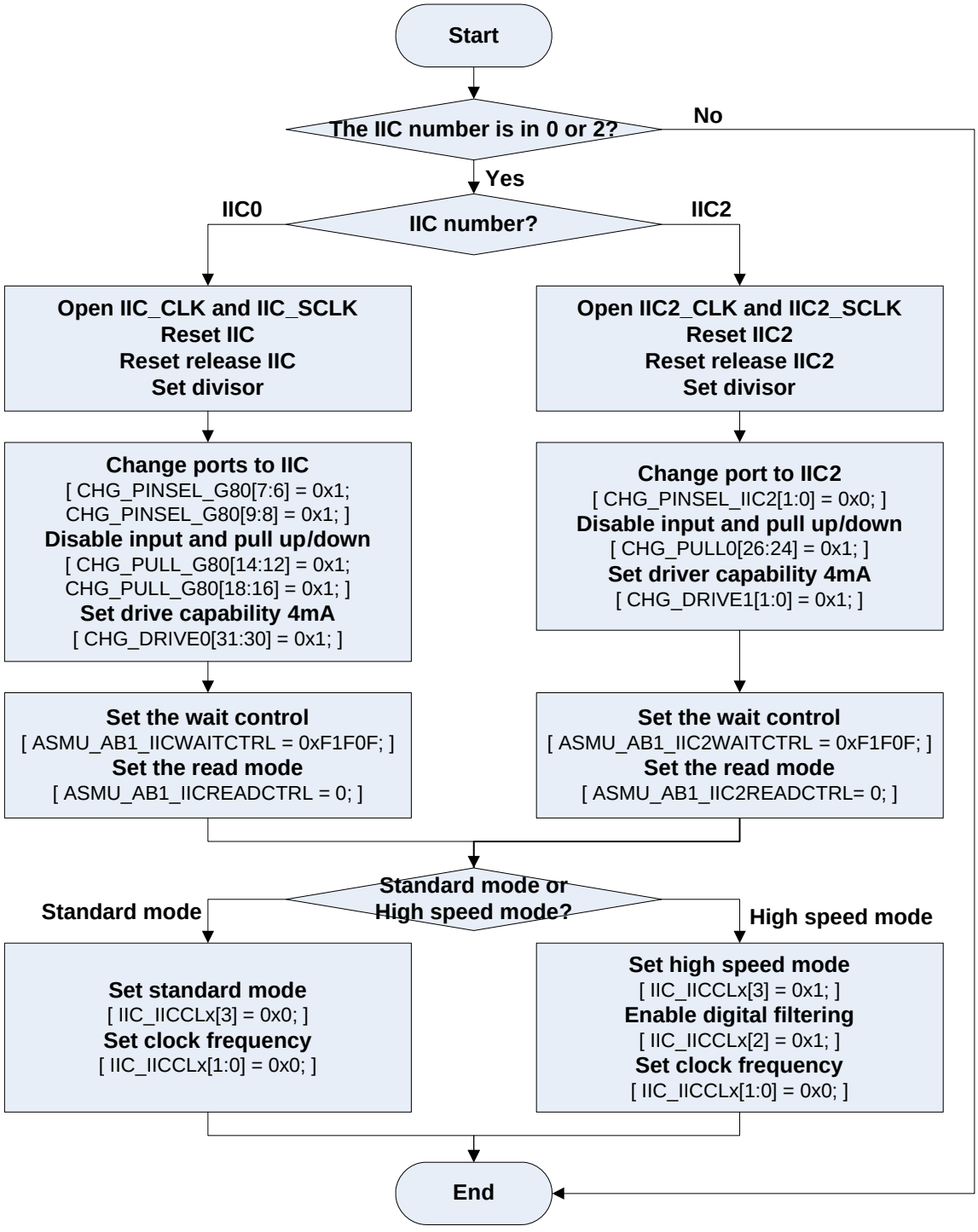

Figure A-1 IIC Initialize

Note: x is 0 or 2.

[Note]

(1). Set clock and release reset.

- Open IIC_CLK.

For IIC0:

```
ASMU_GCLKCTRL2ENA[3] = 1;
ASMU_GCLKCTRL2[3] = 1;
ASMU_GCLKCTRL2ENA[3] = 0;
```

For IIC2:

```
ASMU_GCLKCTRL2ENA[1] = 1;
ASMU_GCLKCTRL2[1] = 1;
ASMU_GCLKCTRL2ENA[1] = 0;
```

- Open IIC_SCLK.

For IIC0:

```
ASMU_GCLKCTRL2ENA[4] = 1;
ASMU_GCLKCTRL2[4] = 1;
ASMU_GCLKCTRL2ENA[4] = 0;
```

For IIC2:

```
ASMU_GCLKCTRL2ENA[2] = 1;
ASMU_GCLKCTRL2[2] = 1;
ASMU_GCLKCTRL2ENA[2] = 0;
```

- Reset IIC0.

For IIC0:

```
ASMU_RESETREQ1ENA[26] = 1;
ASMU_RESETREQ1[26] = 1;
ASMU_RESETREQ1ENA[26] = 0;
```

For IIC2:

```
ASMU_RESETREQ1ENA[25] = 1;
ASMU_RESETREQ1[25] = 1;
ASMU_RESETREQ1ENA[25] = 0;
```

- Reset release IIC0.

For IIC0:

```
ASMU_RESETREQ1ENA[26] = 1;
ASMU_RESETREQ1[26] = 0;
ASMU_RESETREQ1ENA[26] = 0;
```

For IIC2:

```
ASMU_RESETREQ1ENA[25] = 1;
ASMU_RESETREQ1[25] = 0;
ASMU_RESETREQ1ENA[25] = 0;
```

- Set divisor.

For IIC0:

ASMU_DIVIICSCLK[7:0] = Divisor;

For IIC2:

ASMU_DIVIICSCLK[23:16] = Divisor;

Divisor is an input value and it should be set according to I2C user's manual. Details about how to set the Divisor please refer to the "**Chapter 3.2.43 IIC_SCLK frequency division setting register**" of ASMU/GIO user's manual.

A.2.2 Start

[Function Name]

em1_iic_start

[Format]

DRV_RESULT em1_iic_start(uchar num, uchar ms, uchar slave_addr);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number
ms	uchar	I	Master mode or slave mode
slave_addr	uchar	I	Slave mode address

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

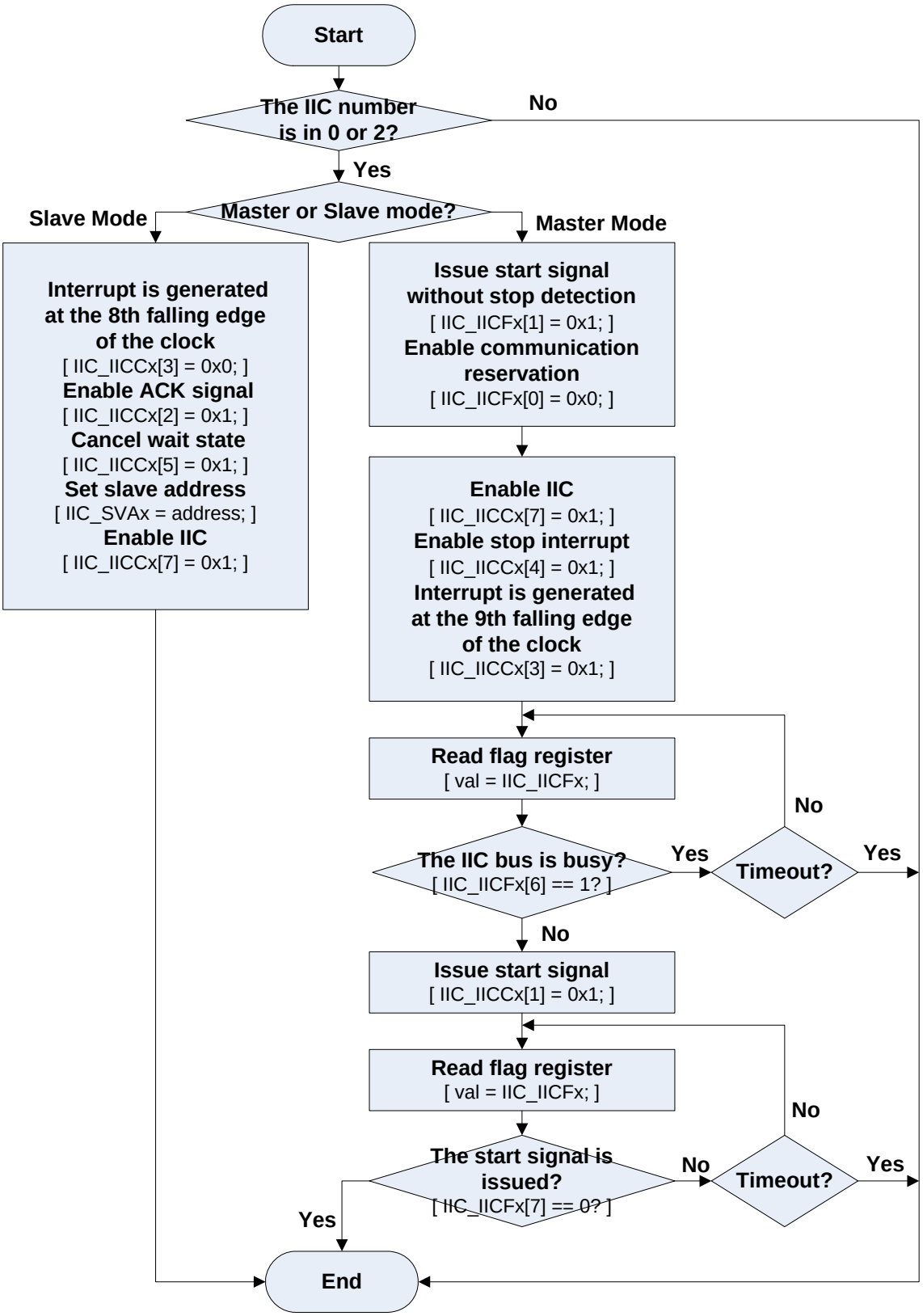

Figure A-2 IIC Start

Note: x is 0 or 2.

A.2.3 Stop

[Function Name]

em1_iic_stop

[Format]

DRV_RESULT em1_iic_stop(uchar num, uchar ms);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number
ms	uchar	I	Master mode or slave mode

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

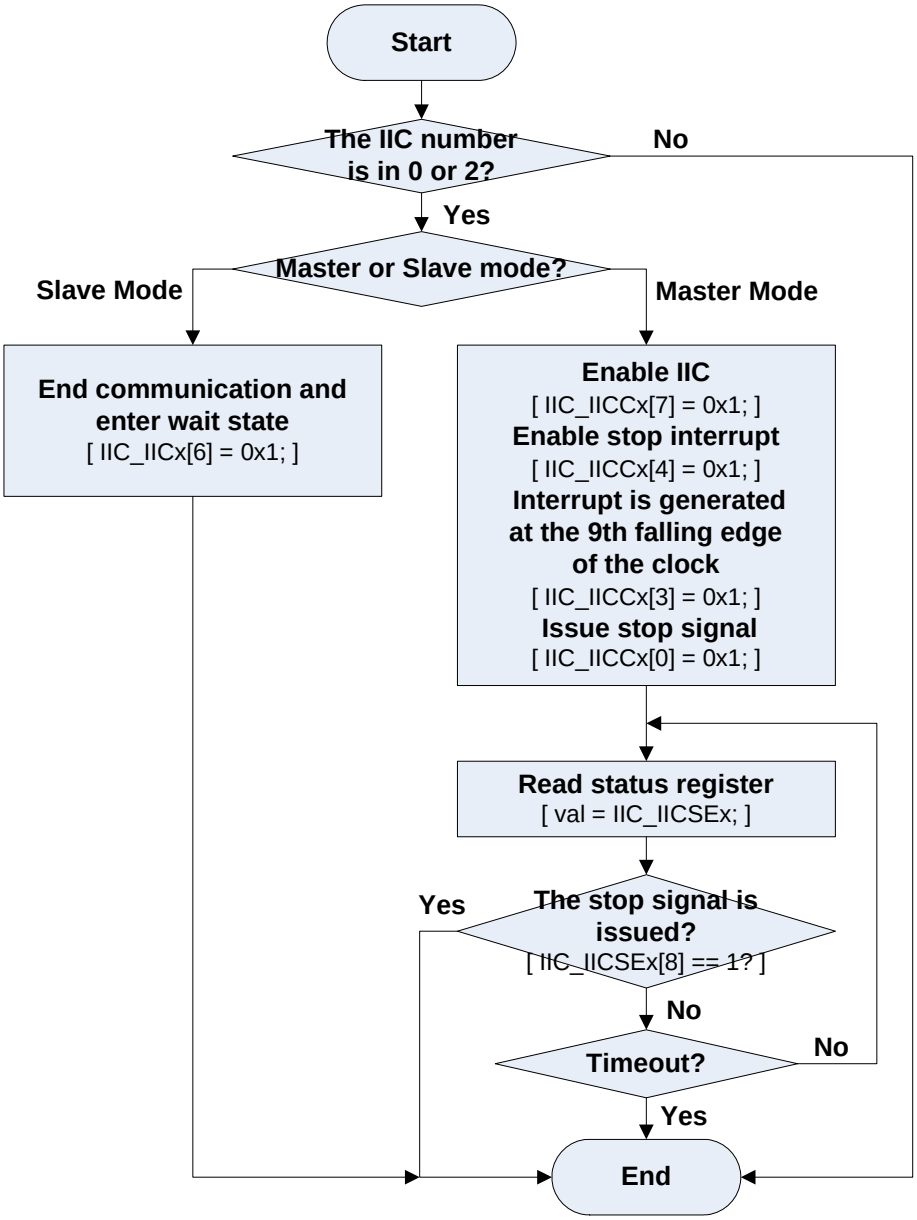

Figure A-3 IIC Stop

Note: x is 0 or 2.

A.2.4 Write Address

[Function Name]

em1_iic_write_addr

[Format]

DRV_RESULT em1_iic_write_addr(uchar num, uchar addr);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number
addr	uchar	I	The address to be send

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

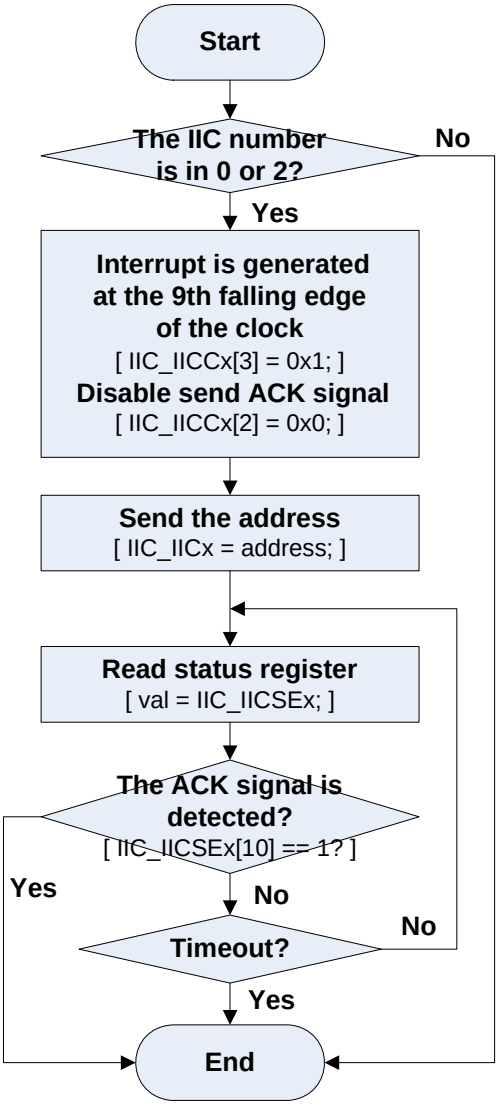

Figure A-4 Write Address

Note: x is 0 or 2.

A.2.5 Write Data

[Function Name]

em1_iic_write_data

[Format]

DRV_RESULT em1_iic_write_data(uchar num, uchar data);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number
data	uchar	I	The data to be send

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

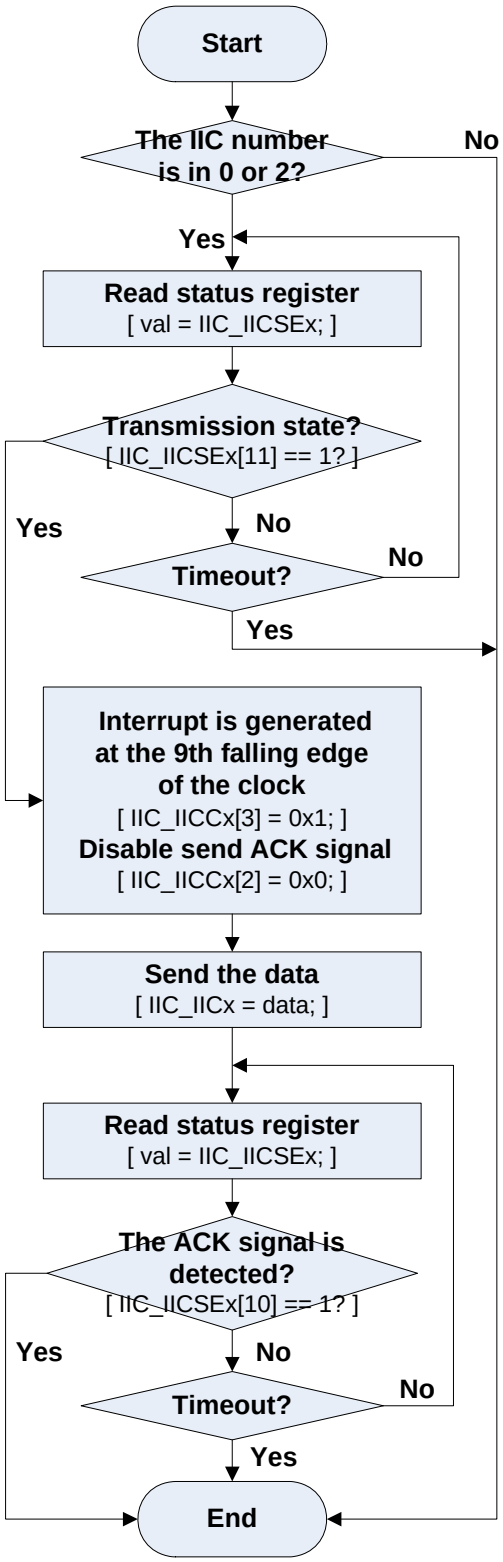

Figure A-5 Write Data

Note: x is 0 or 2.

A.2.6 Read Data

[Function Name]

em1_iic_read_data

[Format]

DRV_RESULT em1_iic_read_data(uchar num, uchar *data);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number
data	uchar *	O	The received data

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

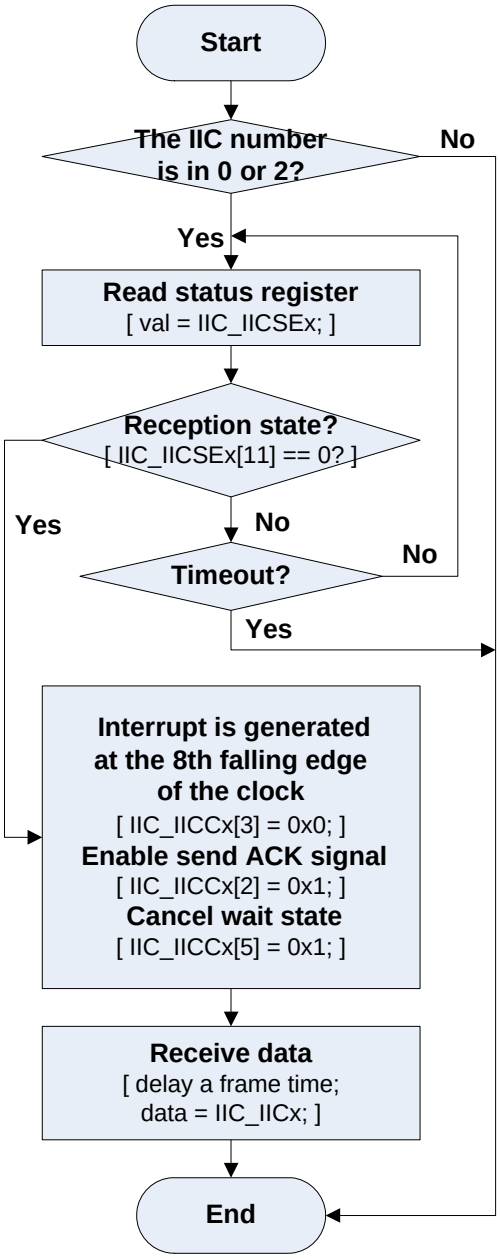

Figure A-6 Read Data

Note: x is 0 or 2.

A.2.7 Check slave address

[Function Name]

em1_iic_match_addr

[Format]

DRV_RESULT em1_iic_match_addr(uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	IIC number

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_TIMEOUT: The execution is timeout.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

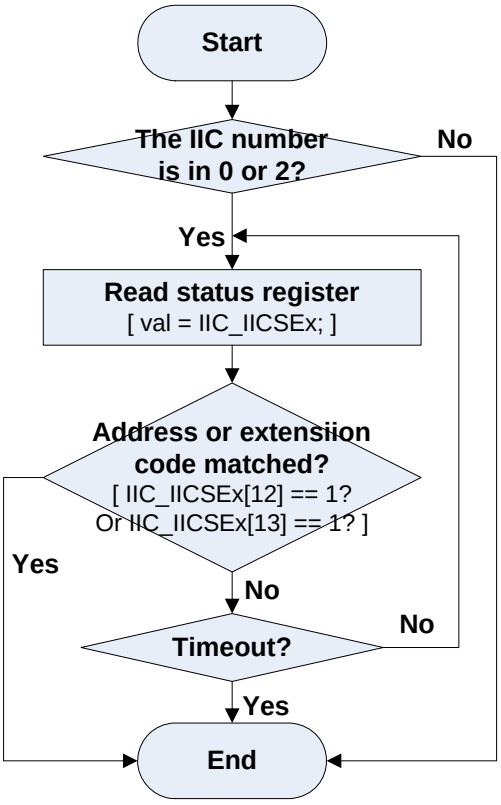

Figure A-7 Check slave address

Note: x is 0 or 2.

Appendix B IIC Transmission Format

The data transfer includes the start condition, slave address and R/W transmission, ACK and data transmission/reception, and the stop condition.
Serial data transfer timing of IIC bus is shown below.

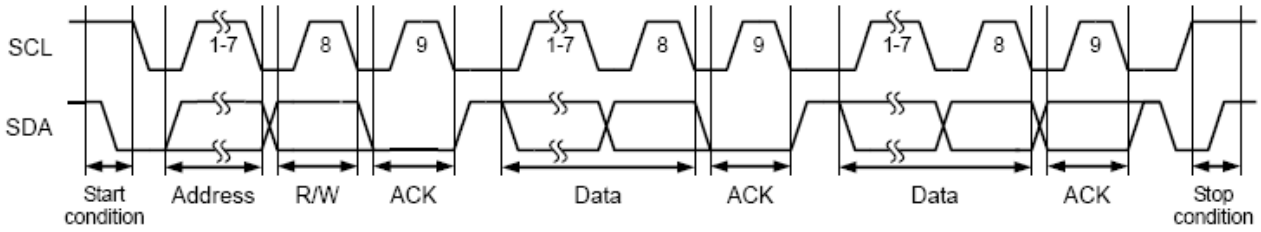

Figure B-1 Serial Data Transfer Timing of IIC Bus

1). A start condition is generated when the SCL line level is high and SDA line level is changed from high to low.
2). After the start condition, 7-bit address data is outputted by master to select one slave from multiple slaves connected to IIC bus.
3). After 7-bit address data, 1-bit Read/Write data is outputted by master. This bit sets the data transfer direction on the IIC bus.
4). After 8-bit data, there must be an ACK signal outputted by the receiver. If an ACK is received normally, the next transfer continues. Otherwise, there is no need for more data or something wrong in the communication.
5). A stop signal is outputted by master when the SCL line is high and SDA line level is changed from low to high.

IIC supports both master operation and slave operation, so there are two modes of IIC operation:

Master-Transmitter and Slave-Receiver.

Master-Receiver and Slave-Transmitter.

- Master-Transmitter and Slave-Receiver Mode

In this mode, the "R/W" bit after 7-bit slave address is set to 0, and then the data will be transmitted from the master to the IIC bus and received by the slave from IIC bus.

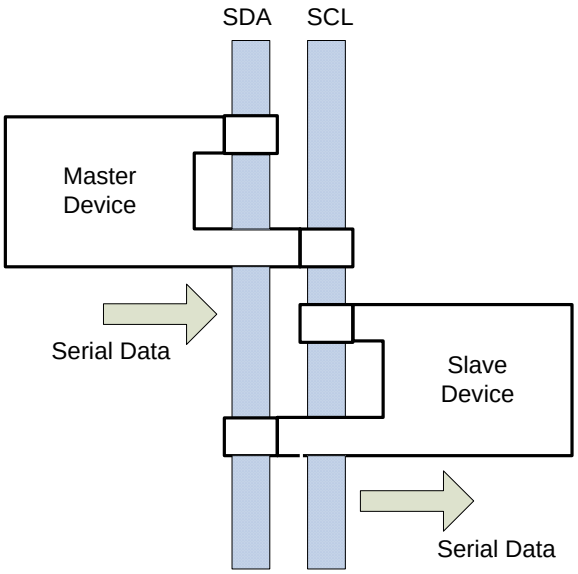

Figure B-2 Data Transfer Process of Master-Transmitter and Slave-Receiver Mode

In this mode, if the slave device doesn't have sub-address, such as Texas Instruments' (TI) PCF8574 and PCF8574A (remote 8-bit I/O expander for IIC bus), the serial data transmit timing of IIC bus is shown as below:

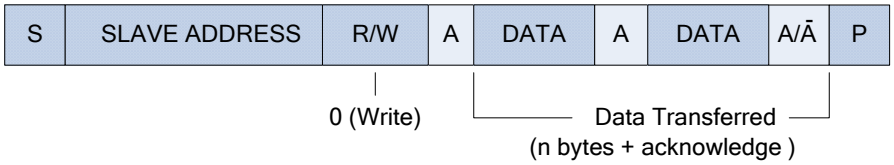

Figure B-3Transmit Timing of Master-Transmitter (Slave Device without Sub-address)

If the slave device has sub-address, serial data transmit timing of IIC bus is shown as below, and the sub-address can be more than one byte.

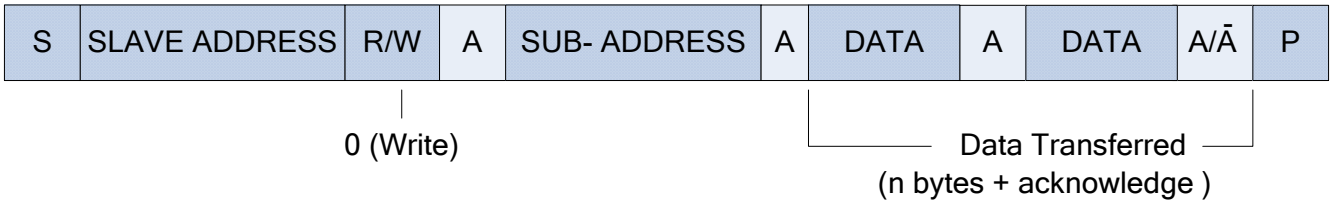

Figure B-4 Transmit Timing of Master-Transmitter (Slave Device with Sub-address)

- Slave-Transmitter and Master-Receiver Mode

In this mode, if the slave doesn't have sub-address, the "R/W" bit is set to 1 and the data will be transmitted from the slave to the IIC bus and received by the master from IIC bus.

If the slave has sub-address, the master should transmit the slave address and sub-address to the slave address with the "R/W" bit set to 0, then restart the IIC with the "R/W" bit set to 1, the data can be transmitted from the slave to the IIC bus and received by the master from IIC bus.

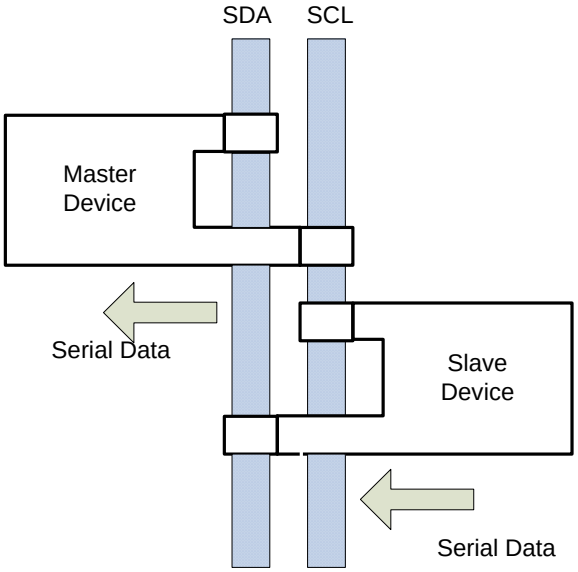

Figure B-5 Data Transfer Process of Slave-Transmitter and Master-Receiver Mode

In this mode, if the slave doesn't have sub-address, serial data receive timing of IIC bus is shown as below:

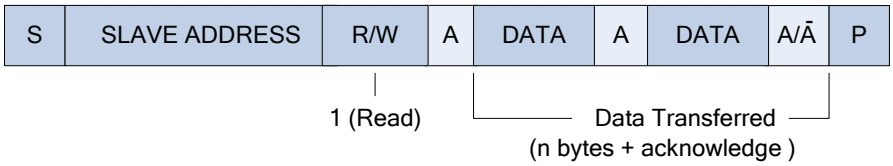

Figure B-6 Transfer Timing of Master-Receiver (Slave Device without Sub-address)

If the slave has sub-address, serial data receive timing of IIC bus is shown as below and the sub-address can be more than one byte.

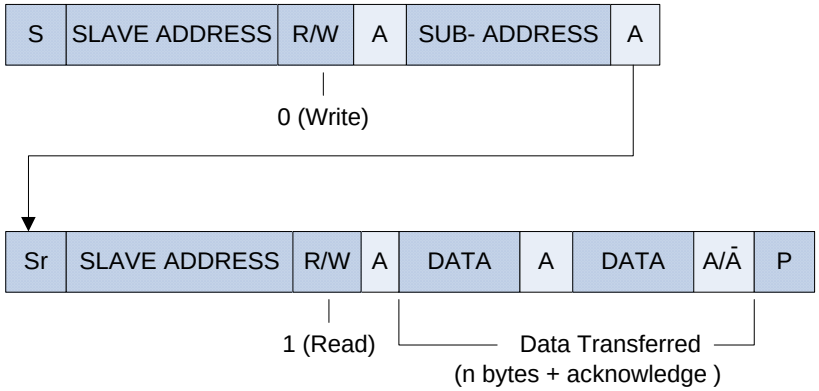

Figure B-7 Transfer Timing of Master-Receiver (Slave Device with Sub-address)

ANNEX Modification History

Number	Modification Contents	Author	Date
Ver 1.00	New version		Aug,4,2009