How to use I²C for W7100A

Version 1.1

For more information, visit our website at http://www.wiznet.co.kr

Table of Contents

1 Introduction

Inter-Intergrated Circuit (I^0C) is a series computer bus, and is used to connect with other electrical devices like the mother-board and embedded systems. This document will explain how to implement I^2C by using iMCU7100EVB. The EEPROM 24C02 (Microchip product) is used.

Note: This document is based on W7100A 100pin package. But the W7100A 64pin package has limited number of GPIO pins. So this application note may cannot work properly on the W7100A 64pin package. Please check the GPIO pins for W7100A 64pin package.

2 I2C

The features of I^2C-bus are as followed.

- Two-bus lines: serial data line (SDA) and serial clock line (SCL)
- Standard mode (100kbit/s), low-speed mode (10kbit/s), and high-speed mode can be used.

3 Hardware Configuration.

This section will explain the Port Expansion Pin Header Hole (J4) and the EEPRORM (24C02) PIN.
(Refer to the Appendix for more details on MCU Port Expansion Pin Header Hole : 32Pin (8pin * 4) 2.54mm Pitch.)
<CAUTION> Since 24C02 EEPROM is used in this document, if the user uses a different device, the hardware composition should be modified.

The EEPROM Address (A0~A2) in this example is set to '000.' Also, WP is deactivated. SDA and SCL is each set to P3_0 and P3_1 of iMCU7100EVB. These ports have pull-up register internally. Hence, external pull-up is not necessary.

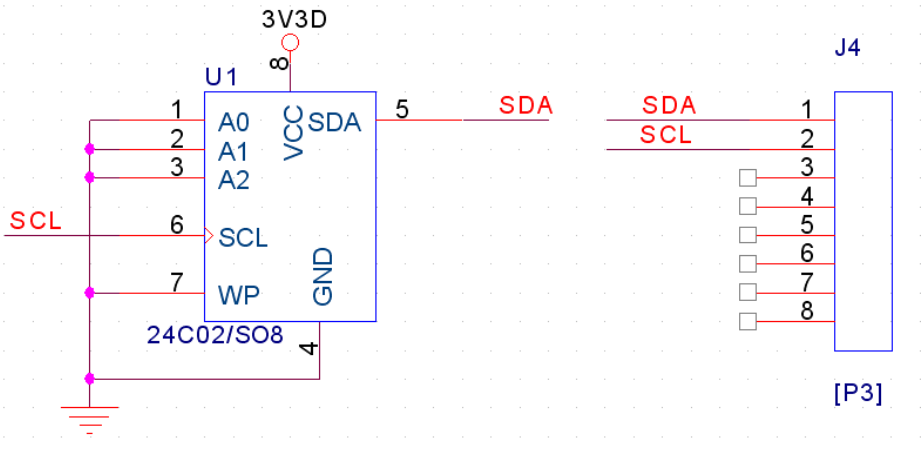

Fig. 1 Hardware Configuration

PIN#	PIN NAME	Description
1~3	A0~A3	Address for device
4	GND	Ground
5	SDA	Serial Data
6	SCL	Serial Clock
7	WP	Write Protect
8	Vcc	Voltage for EEPROM

Table 1 EEPROM Pin Description

4 Software

4.1 Initialization

The Register definition to control the port is defined as shown below.

```
W7100.h
/* Defined registers */
50: sfr P3                  = 0xB0;     /* Port 3 */

/* BIT Register */
137: sbit P3_1        = P3^1;
138: sbit P3_0        = P3^0;
```

Code 1 Register definitions

The status of the port (high/low) and the clock unit is defined in eeprom.c.

```
eeprom.c
 7: #define High 1
 8: #define Low 0
11:#define SCL_DUTY    1          // UNIT 1us
```

Code 2 Status and Clock Unit definition

4.2 Byte READ

The SDA of READ from the I²C-bus is shown in Fig.2. The device address is described as total of 8 bits. The top 4bits have a value of A. Among the lower 4bits, set the address with 3bits and use the remaining 1bit to set WRITE(0)/READ(1).

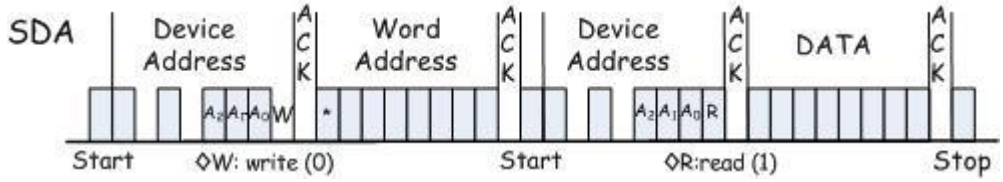

Fig.2 Random READ in 24C02 EEPROM

The READ function of I²C-bus is shown in Code 3. The first eep_wirtedata is a DUMMY. Since the EEPROM address is '000,' the device address of the second eep_wirtedata is '0000.' Refer to I²C F/W for the unit functions.

```
uint8 EEP_Read(uint8 ee_addr) //Random READ in 24C02 EEPROM
{
        uint8 ee_data;
        eep_sendstart();
        eep_writedata(0xa0);
        eep_checkack();
        eep_writedata(ee_addr);
        eep_checkack();
        eep_sendstart();
        eep_writedata(0xa1);
        eep_checkack();
        ee_data = eep_readdata();
        eep_sendnoack();
        eep_sendend();

        return ee_data;
}
```

Code 3 Receive process

4.3 Byte Write

I²C-bus에서 WRITE의 SDA (Serial Data)는 Fig.3와 같이 정의 되어 있다. Device address에 wirte 하기 위해 W값을 0으로 설정한다. I²C-bus에서 WRITE함수의 구현은 아래와 같다.

The SDA of WRITE from the I²C-bus is shown in Fig.3. Set the value of W to 0 in order to write in the device address. Code 4 shows how the WRITE function is used in the I²C-bus.

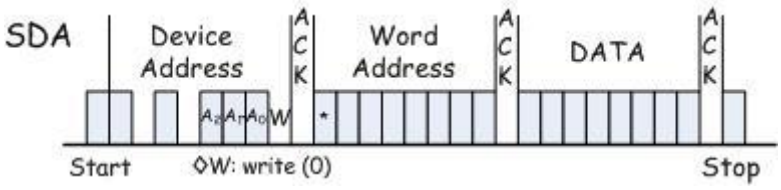

Fig.3 Random Write in 24C02 EEPROM

```
void EEP_Write(uint8 ee_addr,uint8 ee_data) //Random Write in 24C02 EEPROM
{
     eep_sendstart();
       eep_writedata(0xa0); //   H/W address
       eep_checkack();
       eep_writedata(ee_addr); // addr
       eep_checkack();
       eep_writedata(ee_data); //data
       eep_checkack();
       eep_sendend();
       wait_1us(SCL_DUTY*3);
}
```

Code 4 Receive process

5 I²C READ/WRITE Demonstrations

5.1 Terminal setting

The settings of Hyper terminal for demonstration and the execution of I²C F/W is as shown below.

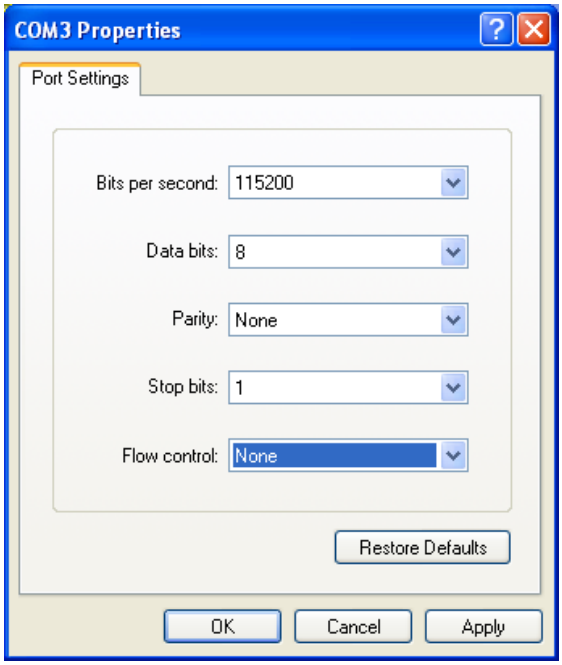

Fig.4 Hyper Terminal

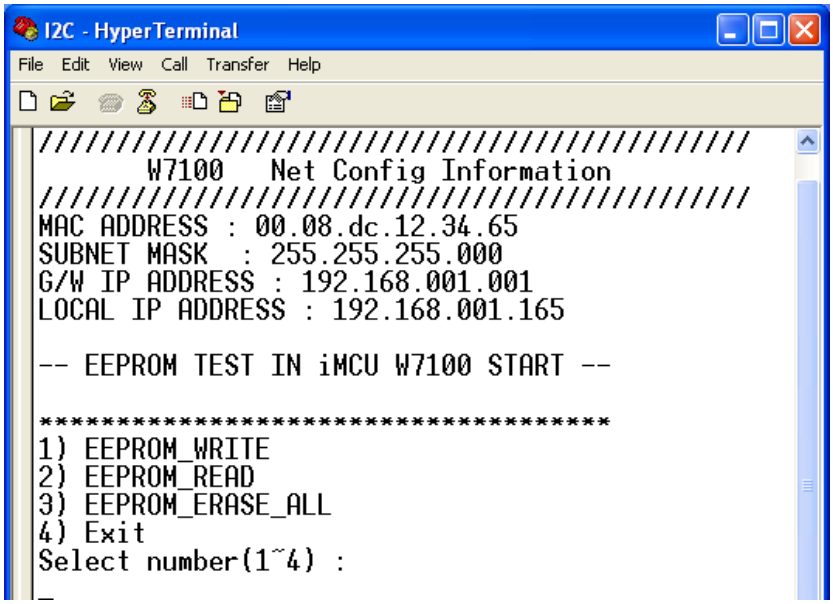

Fig.5 I²C F/W Execution

5.2 Bytes READ

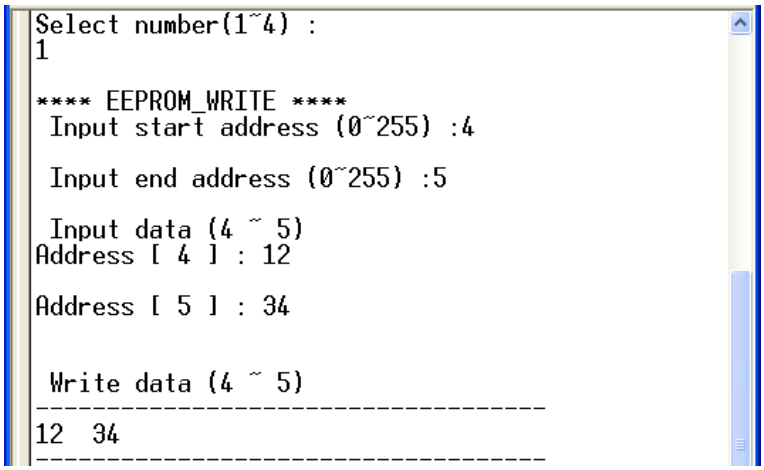

Fig.6 Bytes READ

5.3 Bytes WRITE

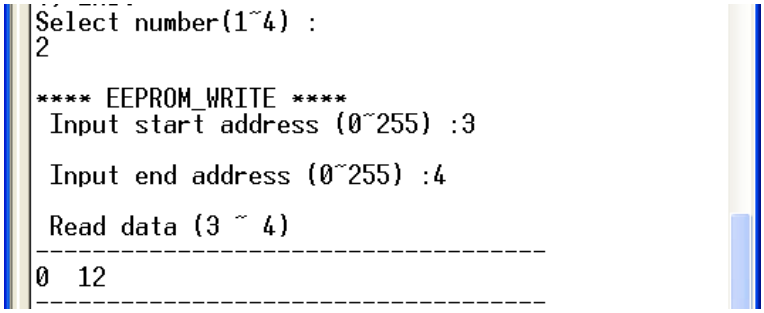

Fig.7 Bytes WRITE

5.4 All Byte ERASE

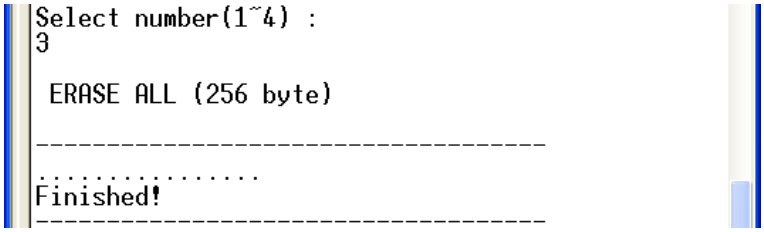

Fig.8 All Byte ERASE

6 Appendix iMCU7100 Board Layout

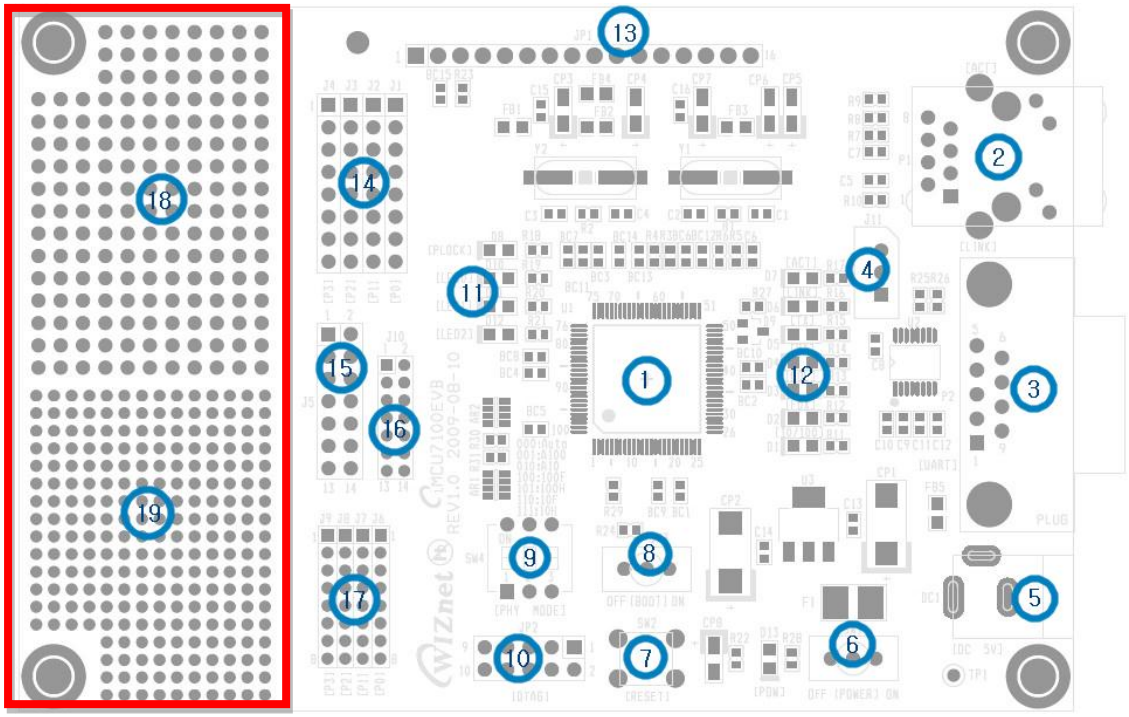

Fig.9 Layout of iMCU7100EVB Board

- The description of each Part shown in Fig.1 is as below.

No	Description	No	Description
1	WIZnet iMCU W7100	2	RJ-45 Jack (integrated transformer)
3	RS-232C DB9 Connector	4	RS-232C 3Pin Header Hole (TTL)
5	DC 5V / 2A Adapter Jack	6	Power Switch
7	Reset Switch	8	Enable Boot Switch
9	PHY mode selection Switch	10	W7100 Debugger Connector
11	User LED * 3Ea	12	Network Status Indicate LED * 8Ea
13	Character LCD Connector	14	MCU Port Expansion Pin Header Hole 32Pin (8pin * 4) 2.54mm Pitch
15	MCU Port Expansion Pin Header Hole 14Pin (7pin * 2) 2.54mm Pitch	16	MCU Port Expansion Pin Header Hole 14Pin (7pin * 2) 2.00mm Pitch
17	MCU Port Expansion Pin Header Hole 32Pin (8pin * 4) 2.00mm Pitch	18	Dummy Pin Header Hole 236Pin 2.00mm Pitch Dummy Hole
19	Dummy Pin Header Hole 167Pin 2.54mm Pitch Dummy Hole		

Table 2 Parts Description of iMCU7100EVB

Document History Information

Version	Date	Descriptions
Ver. 0.9βeta	Nov, 2009	Release
Ver. 1.0	Mar, 2011	Modified for W7100A 64pin QFN package

Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at http://www.wiznet.co.kr