M480 EEPROM Access Using I2C

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	EEPROM Access Using Software I2C
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 Ver 3.0 + M487 Advance Ver 4.0

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

1.1 Introduction

It is possible to use GPIO to emulate I2C function when the system needs more I2C multi-function pins than M480 can provide. This example code uses GPIO to emulate I2C to access an EEPROM.

1.2 Principle

There are four phases in an I2C protocol, start, read, write and stop. The sample code is implemented, the following function to match the I2C protocol
void I2C_start(void) => start stage packet
unsigned char I2C_write(unsigned char data) => data stage packet.
unsigned char I2C_read(unsigned char ack) => data stage packet.
void I2C_stop(void) => stop stage packet
If you want to detail EEPROM, please refer to the EEPROM technical manual for EEPROM protocol.

1.3 Demo Result

```
    ack=0x0,
    ack=0x0,
    ack=0x0,
    ack=0x0,
    ack=0x0,
    ack=0x0,
    ack=0x0,
    ack=0x0,
    data=0x55,
```

2 Code Description

Initial I2C to access EEPROM

```
void I2C_init(void)
{
    GPIO_SetMode(PA, BIT10, GPIO_MODE_OPEN_DRAIN);/* I2C_CLK */
    GPIO_SetMode(PA, BIT11, GPIO_MODE_OPEN_DRAIN);/* I2C_DAT */
    I2C_CLK = 1;
    I2C_DAT = 1;
}
```

Function for Implement I2C start

```
void I2C_start(void)
{

    /* I2C start sequence is defined as
     * a High to Low Transition on the data
     * line as the CLK pin is high */
    I2C_DAT = 1;
    I2C_CLK = 1;
    CLK_SysTickDelay(I2C_DELAY_TIME);
    I2C_DAT = 0;
    I2C_CLK = 0;
    CLK_SysTickDelay(I2C_DELAY_TIME);
}
```

Function for Implement I2C stop

```
void I2C_stop(void)
{
    /* I2C stop sequence is defined as
     * data pin is low, then CLK pin is high,
     * finally data pin is high. */
    I2C_DAT = 0;
    I2C_CLK = 1;
    I2C_DAT = 1;
}
```

Function for Implement I2C write

```
unsigned char I2C_write(unsigned char data)
{
    /* An I2C output byte is bits 7-0
     * (MSB to LSB).  Shift one bit at a time
     * to the MDO output, and then clock the
     * data to the I2C Slave */
    unsigned char i, temp;
    /* Write to slave */
    for (i = 0; i < 8; i++)
    {
        /* Send data bit */
        if ((data & 0x80) == 0x80)
            I2C_DAT = 1;
        else
            I2C_DAT = 0;

        data <<= 1;  /* Shift one bit */
        I2C_CLK = 1; /* SCL: High */
        CLK_SysTickDelay(I2C_DELAY_TIME);
        I2C_CLK = 0; /* SCL: Low */
        CLK_SysTickDelay(I2C_DELAY_TIME);
```

```
    }
    I2C_DAT = 1;
    I2C_CLK = 1; /* SCL: High */
    CLK_SysTickDelay(I2C_DELAY_TIME / 2);
    temp = I2C_DAT; /* Read ACK bit from slave */
    CLK_SysTickDelay(I2C_DELAY_TIME / 2);
    I2C_CLK = 0; /* SCL: Low */
    CLK_SysTickDelay(I2C_DELAY_TIME);
    return temp;
}
```

Function for Implement I2C read

```
unsigned char I2C_read(unsigned char send_ack)
{
    unsigned char i, data;
    data = 0x00;
    /* Read from slave */
    for (i = 0; i < 8; i++)
    {
        data <<= 1;   /* Shift one bit */
        data |= I2C_DAT; /* Read data bit */
        I2C_CLK = 1; /* SCL: High */
        CLK_SysTickDelay(I2C_DELAY_TIME);
        I2C_CLK = 0; /* SCL: Low */
        CLK_SysTickDelay(I2C_DELAY_TIME);
    }
    /* Send ACK bit to slave */
    I2C_DAT = send_ack;
    I2C_CLK = 1; /* SCL: High */
    CLK_SysTickDelay(I2C_DELAY_TIME);
    I2C_CLK = 0; /* SCL: Low */
    CLK_SysTickDelay(I2C_DELAY_TIME);
    return data;
}
```

Main function

```
int main(void)
{
    SYS_Init();
    UART0_Init();
    I2C_init();
    I2C_start();
    printf("ack=0x%x,\n\r", I2C_write(I2C_ADDRESS_w));
    printf("ack=0x%x,\n\r", I2C_write(0x00)); /* eeprom address high */
    printf("ack=0x%x,\n\r", I2C_write(0x00)); /* eeprom address low */
    printf("ack=0x%x,\n\r", I2C_write(0x55)); /* write 0x55 */
    I2C_stop();
    CLK_SysTickDelay(10000); /* delay 10ms */
    I2C_start();
    printf("ack=0x%x,\n\r", I2C_write(I2C_ADDRESS_w));
    printf("ack=0x%x,\n\r", I2C_write(0x00)); /* eeprom address high */
    printf("ack=0x%x,\n\r", I2C_write(0x00)); /* eeprom address low */
    I2C_start();
    printf("ack=0x%x,\n\r", I2C_write(I2C_ADDRESS_r));
    printf("data=0x%x,\n\r", I2C_read(NO_ACK));
    I2C_stop();

    while (1);
}
```

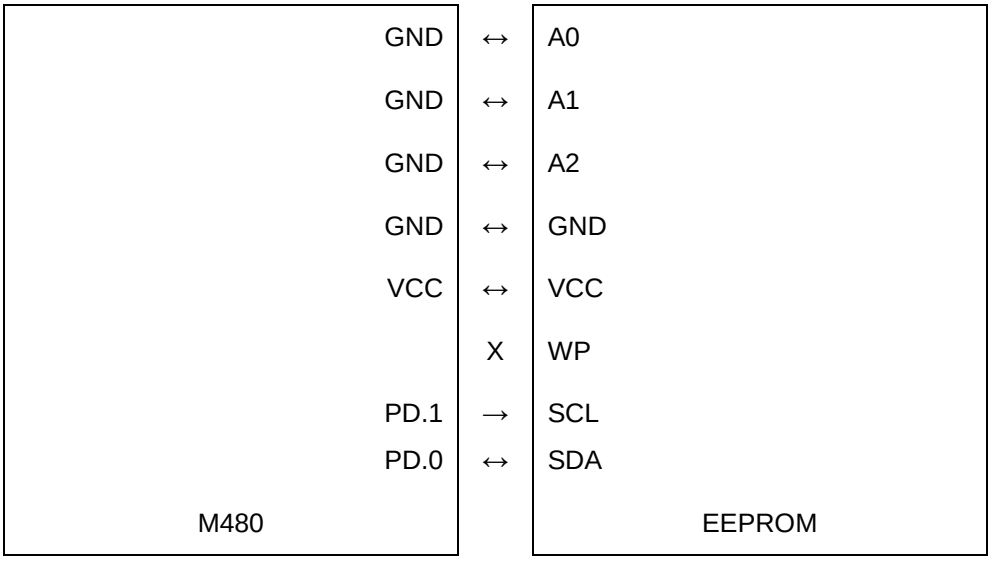
3 Software and Hardware Environment

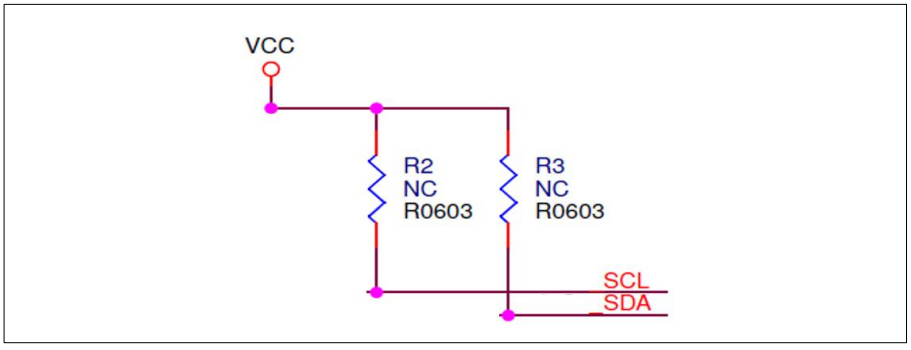
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.001
- IDE version
  - Keil uVersion 5.22

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 + M487 Advance Ver4.0
- Diagram





4 Directory Information

- EC_M480_EEPROM_Access_Using_Software_I2C_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click GPIO_I2C_EEPROM.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
June. 17, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.