รายงาน

I^2C และการต่อหลายอุปกรณ์ใน Bus เดียวกัน

จัดทำโดย

ธนวัฒน์ กะโห้ทอง 57-010126-2020-9

จิรายุส วิริยบัณฑร 57-010126-3005-1

คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ชั้นปีที่ 3 ปีการศึกษา 2/2559

เสนอ

อาจารย์ ดนุชา ประเสริฐสม

อาจารย์ คทาเทพ สวัสดิพิศาล

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้า พระนครเหนือ

Register ที่ใช้ในการทำงาน

1. TWBR (TWI Bit Rate Register) ใช้กำหนดค่าสัญญาณนาฬิกา (SCL) ที่ใช้ในการเชื่อมต่อ

TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

ซึ่งสามารถกำหนดได้ตั้งแต่ 100KH จนถึง 400KHz โดยสามารถคำนวนได้จาก

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2\,(\text{TWBR}) \times 4^{\text{TWPS}}}$$

- SCL frequency คือ ค่า clock ที่ต้องการจะนำไปใช้
- CPU Clock Frequency คือ ค่าสัญญาณนาฬิกาของ CPU(Arduino เป็น 16MHz)
- TWBR คือค่าใน TWBR Register
- TWPS คือ ค่า prescaler bits ใน TWSR (TWI status register)

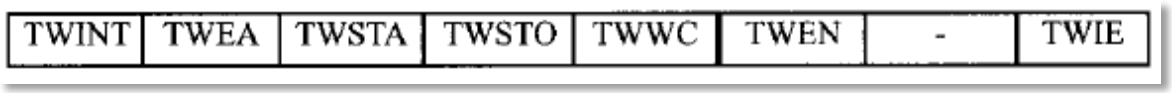
2. TWSR (TWI Status Register) เป็น Register ที่บอกสถานะต่างๆของการเชื่อมต่อ

TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0

- TWS7 – TWS3 (bit7-bit3) เป็นสถานะของการควบคุม TWI และ การเส้นทางเชื่อมต่อ
- TWPS1-TWPS0 (bit1-bit0) เป็นสถานะการ prescale clock มีค่าตั้งแต่ 0-3

3. TWDR (TWI Data Register) เก็บข้อมูลที่ต้องการจะส่งหรือรับ

4. TWCR (TWI Control Register) ใช้ควบคุมการเชื่อมต่อของ I2C เช่น การเริ่ม การหยุด และการส่ง Acknowledge



TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

- TWINT (TWI Interrupt) บิทจะถูกเซ็ตเมื่อจบการทำงาน หากต้องการ clear ให้เขียน 1 ลงไป
- TWEA (TWI Enable Acknowledge) ใช้สร้าง acknowledge bit
- TWSTA (TWI START condition Bit) ถ้าการเชื่อมต่อว่างอยู่ การเซ็ตเป็น high จะเริ่มการติดต่อ
- TWSTO (TWI STOP condition Bit) การเซ็ตเป็น high จะสร้างการหยุดการเชื่อมต่อ และบิทจะถูก clear เมื่อเงื่อนไขการหยุดถูกส่งไป
- TWWC (TWI Write Collision Flag) บิทจะถูกเซ็ตเป็น High เมื่อมีการเข้าถึง TWDR ขณะที่ TWINT เป็น Low และสามารถ clear ได้ด้วยการเขียน TWDR ขณะที่ TWINT เป็น High
- TWEN (TWI Enable) บิทเปิดการทำงาน TWI Module
- TWIE (TWI Interrupt Enable) บิท enable Interrupt เมื่อเป็น High

การเชื่อมต่อหลายอุปกรณ์

ในการระบุอุปกรณ์ที่ต้องการสื่อสารด้วยนั้น จะอ้างอิง Address Device ซึ่งต้องทำให้มีค่าที่ไม่ซ้ำกัน เนื่องจากจะได้ส่งไปหลายอุปกรณ์ได้ สำหรับการทำงานได้เลือกใช้ LCD ขนาด 16x2 จำนวน 2 ตัวในการทดลอง

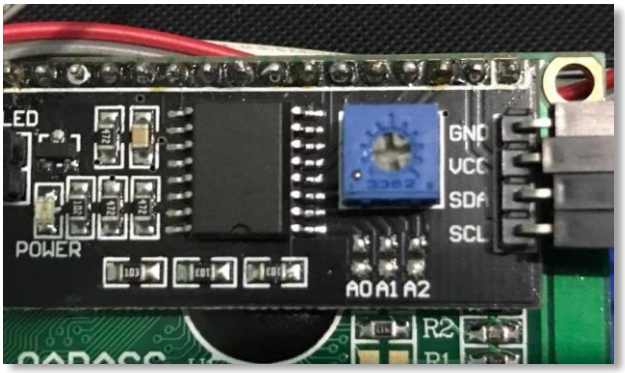
การกำหนด Address Device ของ LCD

A0	A1	A2	HEX Address
1	1	1	0x27
0	1	1	0x26
1	0	1	0x25
0	0	1	0x24
1	1	0	0x23
0	1	0	0x22
1	0	0	0x21
0	0	0	0x20

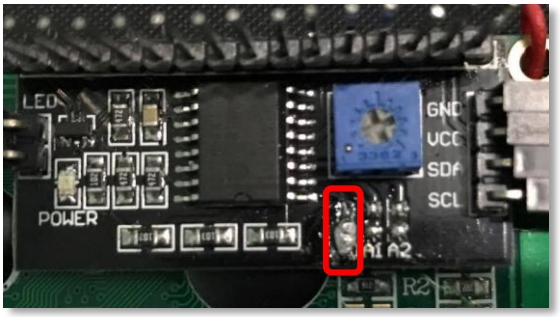
Credit. : https://arduino-info.wikispaces.com/LCD-Blue-I2C

วิธีการกำหนด Address Device

การกำหนด Address สามารถทำได้โดยตรงที่อุปกรณ์ LCD ด้วยวิธีการบัดกรี

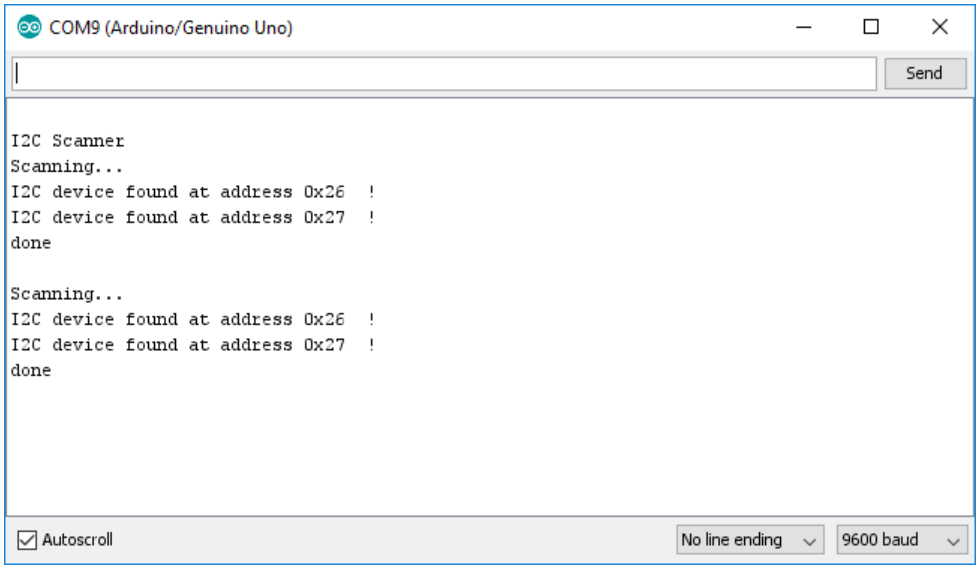


LCD ตัวที่ 1 ต้องการกำหนด address 0x27 โดยทั่วไป LCD ส่วนใหญ่เป็น 0x27 อยู่แล้ว จึงไม่ต้องบัดกรี



LCD ตัวที่ 2 ต้องการกำหนด address 0x26 ให้ทำการบัดรี ระหว่างจุด A0 กับจุดด้านบน A0

การตรวจอบ Address Device ได้ทดลองใช้โค้ดจาก http://playground.arduino.cc/Main/I2cScanner



ผลลัพธ์ที่ได้จากการสแกน Address ของอุปกรณ์ที่เชื่อมต่อแบบ I2C

ขั้นตอนการทำงานของโค้ด

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <string.h>
#define LCD_RS  0
#define LCD_RW  1
#define LCD_LED 3
#define LCD_EN  2
#define LCD_Clear  0b00000001
#define F_CPU 16000000L
#define LCD_SetCursor   0b10000000  // set cursor position

void I2C_Start();
void I2C_Write(uint8_t dat);
void I2C_Stop(void);
void I2C_AddressR_W(uint8_t addr, uint8_t r_w);
void LCD_CMNDWRT41(uint8_t addr, uint8_t cmnd);
void LCD_CMNDWRT42(uint8_t addr, uint8_t cmnd);
void LCD_DATWRT42(uint8_t addr, uint8_t dat);
void LCD_CLS(uint8_t addr);
uint8_t swap(uint8_t cmnd);
void LCD_gotoxy(uint8_t addr,uint8_t col, uint8_t row);
void LCD_Print(uint8_t addr, char * str);
```

เริ่มต้นด้วยการประกาศใช้ library และกำหนดตัวแปรต่างๆ จากนั้นสร้างฟังก์ชั่นการทำงานสำหรับการติดต่อแบบ I2C โดยอ้างอิงจาก textbook

```
void I2C_Init(void) {
    TWSR = 0x00;  //set prescaler to zero
    TWBR = 12;    //set clock 16 MHz
    TWCR = 0x04;  //enable TWI module
}
```

ฟังก์ชั่น I2C_Init กำหนดค่า clock ไม่ทำการ prescale , เซ็ต clock 16MHz และเปิดใช้ module TWI(two wire interface)

```
void I2C_Start(void) {
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
    while ((TWCR & (1 << TWINT)) == 0);
}
void I2C_Stop(void)
{
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);
    while ((TWCR & (1 << TWINT)) == 0x80);
}

void I2C_Write(uint8_t dat)
{
    TWDR = dat ;
    TWCR = ((1<< TWINT) | (1<<TWEN));
    while (!(TWCR & (1 <<TWINT)));
}

void I2C_AddressR_W(uint8_t addr, uint8_t r_w){
    addr = (addr << 1)| r_w;
    TWDR = addr;
    TWCR = ((1<< TWINT) | (1<<TWEN));
    while (!(TWCR & (1 <<TWINT)));
}
```

สร้างฟังก์ชั่นอื่นๆตาม textbook

ต่อมาเป็นส่วนของ LCD เริ่มต้นด้วยการสร้างฟังก์ชั่นสำหรับตำแหน่งแสดงผลในแต่ละบรรทัด

```
void LCD_gotoxy(uint8_t addr, uint8_t col, uint8_t row)
{
    if(row ==2){
        LCD_CMNDWRT42(addr,0xC0);
    }
    for(int i=1;i<col;i++){
        LCD_DATWRT42(addr,LCD_SetCursor);
    }
}
```

และฟังก์ชั่นสำหรับเริ่มต้นการทำงานของ LCD

```
void Init_LCD4D(uint8_t addr){
    _delay_ms(50);
    LCD_CMNDWRT41(addr, 0x30);
    _delay_ms(4.1);

    LCD_CMNDWRT41(addr, 0x30);
    _delay_us(100);

    LCD_CMNDWRT41(addr, 0x30);
    _delay_us(100);

    LCD_CMNDWRT41(addr, 0x20);
    _delay_us(100);

    LCD_CMNDWRT42(addr, 0x28);
    _delay_us(100);

    LCD_CMNDWRT42(addr, 0x0C);
    _delay_us(100);

    LCD_CLS(addr);
    _delay_us(100);

    LCD_CMNDWRT42(addr, 0x06);
    _delay_us(100);

}
```

ฟังก์ชั่นการเขียนข้อมูลให้ LCD ภายในมีรูปแบบการส่งข้อมูลแบบ I2C

```
void LCD_DATWRT42(uint8_t addr, uint8_t dat){
    uint8_t data = dat;
    data = data & 0xF0;
    data |= (1<<LCD_RS);    // led,en,rw,rs = 1101
    data |= (1<<LCD_EN);    // led,en,rw,rs = 1101
    data |= (1<<LCD_LED);   // led,en,rw,rs = 1101
    I2C_Start();            // Start I2C communication
    I2C_AddressR_W(addr, 0);
    I2C_Write(data);
    _delay_us(1);
    data &= 0xF0;
    data &= ~(1<<LCD_EN);//set EN = 0// led,en,rw,rs = 1001
    I2C_Write(data);
    I2C_Stop();

    data = swap(dat);
    data = data & 0xF0;
    data |= (1<<LCD_RS);    // led,en,rw,rs = 1101
    data |= (1<<LCD_EN);    // led,en,rw,rs = 1101
    data |= (1<<LCD_LED);   // led,en,rw,rs = 1101
    I2C_Start();            // Start I2C communication
    I2C_AddressR_W(addr, 0);
    I2C_Write(data);
    _delay_us(1);
    data &= ~(1<<LCD_EN);//set EN = 0 // led,en,rw,rs = 1001
    I2C_Write(data);
    I2C_Stop();
    _delay_us(100);
}
```

ฟังก์ชั่นสำหรับเคลียหน้าจอ LCD เมื่อข้อมูลเปลี่ยน

```
void LCD_CLS(uint8_t addr){
    LCD_CMNDWRT42(addr,LCD_Clear);
    _delay_ms(2);
}
```

และฟังก์ชั่นแสดงผลแต่ละตัวอักษร LCD_Print

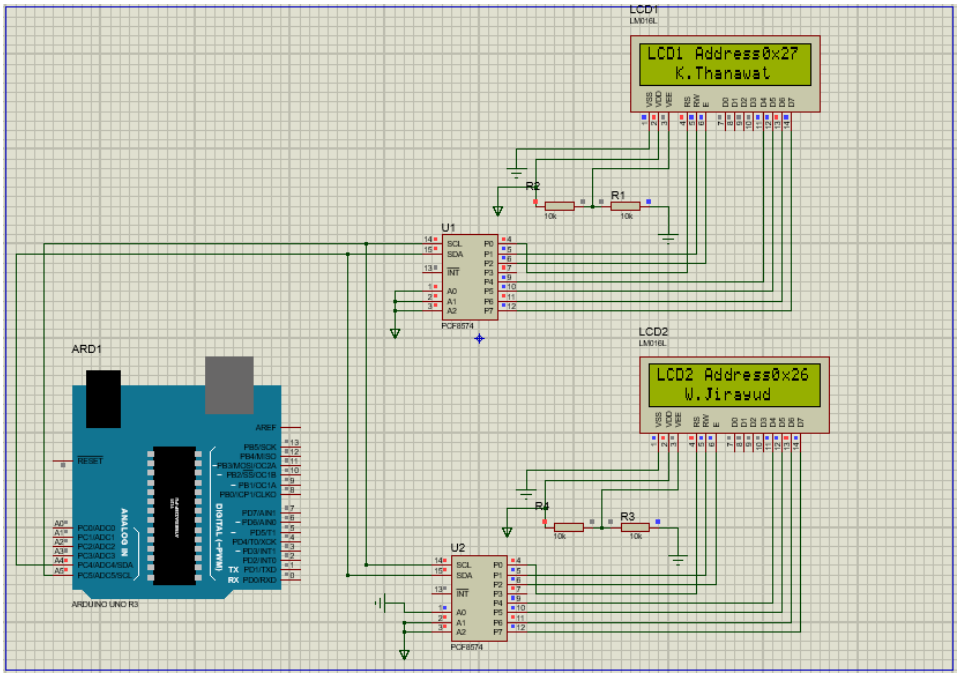
```
void LCD_Print(uint8_t addr, char * str)
{
    while(*str)
    {
        LCD_DATWRT42(addr, *str);
        str++;
    }
}
```

```c
int main(void)
{
    // lcd 1
    uint8_t lcd1_addr = 0x27;
    I2C_Init();             // Initialize the I2c
    Init_LCD4D(lcd1_addr);
    LCD_CLS(lcd1_addr);
    LCD_gotoxy(lcd1_addr, 1, 1);
    LCD_Print(lcd1_addr, "LCD1 Address0x27");
    LCD_gotoxy(lcd1_addr, 4, 2);
    LCD_Print(lcd1_addr, "K.Thanawat");

    //lcd 2
    uint8_t lcd2_addr = 0x26;
    Init_LCD4D(lcd2_addr);
    LCD_CLS(lcd2_addr);
    LCD_gotoxy(lcd2_addr, 1, 1);
    LCD_Print(lcd2_addr, "LCD2 Address0x26");
    LCD_gotoxy(lcd2_addr, 4, 2);
    LCD_Print(lcd2_addr,"W.Jirayud");

    while (1){}
}
```

การทำงานหลักในส่วนของฟังก์ชั่น main คือ เริ่มต้นกำหนด Address Device แรกที่ต้องการติดต่อแล้วเตรียมพร้อมการติดต่อแบบ I2C ต่อไปเตรียมพร้อมสำหรับ LCD และทำการเคลียข้อมูลบนหน้าจอก่อน จากนั้นกำหนดตำแหน่งที่ต้องการแสดงผลตามด้วยข้อมูล



ภาพจำลองการทำงานด้วยโปรแกรม Proteus

ผลการทำงานของโปรแกรม

ตัวอย่างโค้ด : https://github.com/ThanwatCprE/I2C-to-LCD-s-/blob/master/main.c