



รายงานการทดลองที่ Lab3HB05 และ 3HB06

เรื่อง AVR Programming (Assembly/C), H/W and S/W co-design

เสนอ

อ. พัทธี เทพนิมิตร

จัดทำโดย

ชื่อ นายเฟาชัน แซ่หลี่ รหัสนักศึกษา 5635512080 Section 01

ชื่อ นางสาว ชฎาธาร บัวอิน รหัสนักศึกษา 5635512024 (คู่แลบ)

รายงานนี้เป็นส่วนหนึ่งของวิชา

ADVANCED COMPUTER ENGINEERING LABORATORY II

ภาคการศึกษาที่ 2 ปีการศึกษา 2558

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต

Lab3HB05 และ 3HB06

วัตถุประสงค์

- เพื่อให้นักศึกษาได้เรียนรู้เทคนิคการออกแบบฮาร์ดแวร์และซอฟต์แวร์ร่วมกันเพื่อให้ง่ายในการ ทดสอบ ความถูกต้อง

- เพื่อให้นักศึกษาได้ฝึกการเขียนซอฟต์แวร์สำหรับการทดสอบความถูกต้องของฮาร์ดแวร์
- เพื่อให้นักศึกษาได้ฝึกเทคนิคการ Debug โปรแกรม

การทดลองที่ 1 การเขียนโปรแกรมเพื่ออ่านค่าจากสวิตช์และแสดงผลออกทาง 7 segment

โค้ด ภาษา ASSEMBLY และ ภาษา C

```
program.asm" X
.INCLUDE "m32def.inc"
.EQU ALL_PIN_OUT = 0xFF
.EQU ALL_PIN_IN = 0x00
.DCF VAR_A = R16
.DCF TMP = R17
.CSEG
.ORG 0x0000
ldi VAR_A, ALL_PIN_OUT
out DDRC, VAR_A
ldi VAR_A, ALL_PIN_IN
out DORD, VAR_A
ldi TMP, 0x00

MAIN:
in VAR_A, PIND
andi VAR_A, 0x0F
ldi ZL, low(TB_7SEGMENT*2)
ldi ZH, high(TB_7SEGMENT*2)

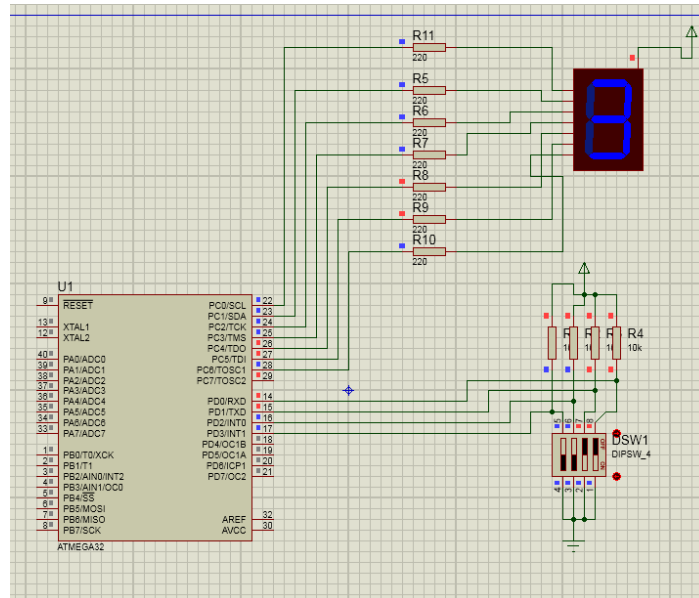
add ZL, VAR_A
adc ZH, TMP
lpm

out PORTC, R0
rjmp MAIN
; hgfdcdba hgfdcdba
TB_7SEGMENT:
.DS 0b00111111, 0b00000110 ; 0 and 1
.DS 0b01011011, 0b01001111 ; 2 and 3
.DS 0b01100110, 0b01101101 ; 4 and 5
.DS 0b01111101, 0b00000111 ; 6 and 7
.DS 0b01111111, 0b01101111 ; 8 and 9
.DS 0b01110111, 0b01111100 ; A and 0
.DS 0b00111001, 0b01011110 ; C and D
.DS 0b0111001, 0b01110001 ; E and F

.DSEG
```

```
#include <avr/io.h>
int main (void)
{
    unsigned char LOOKUPB[] = { 0b00111111,
        0b00000110,
        0b01011011,
        0b01001111,
        0b01100110,
        0b01101101,
        0b01111101,
        0b00000111,
        0b01111111,
        0b01101111,
        0b01101111,
        0b01111100,
        0b00111001,
        0b01011110,
        0b01111001,
        0b01110001 };
    unsigned char DISPLY, SWITCH_V;
    DDRC = 0xFF;
    DORD = 0x00;
    while(1)
    {
        SWITCH_V = PIND;
        SWITCH_V &= 0x0F;
        DISPLY = LOOKUPB[SWITCH_V];
        PORTC = ~(DISPLY);
    }
}
```

รูปการต่อวงจร



Concept :-

เริ่มต้นด้วยการกำหนดตัวแปร ชนิด unsigned ขึ้นมา 2 ตัว แล้วทำการกำหนดให้ PORTC เป็น Output และ PORTD เป็น Input (PIND คือ เป็นการรับข้อมูลเข้ามาจากสวิตช์) แล้วภายในลูปก็ทำการเช็คเฉพาะ 4 บิตล่าง แล้วนำไปชี้ที่ตาราง 7-Segment โดยแล้วแสดงผลว่ามีค่าเท่าไรออกมาทาง 7-Segment

จากการทดลองจะได้ ไฟล์ .hex ของภาษาแอสเซมบลี ขนาด 157 bytes และไฟล์ .hex ของภาษาซี ขนาด 627 bytes ดังนั้นจะเห็นได้ว่า ไฟล์ .hex ของภาษาแอสเซมบลีมีขนาดน้อยกว่า

การทดลองที่ 2 เขียนโปรแกรมเพื่อนับ logic low จาก dip switch

โปรแกรมจะทำการอ่านค่าจาก dip switch จำนวน 8 ตัว ที่ port D จากนั้นทำการตรวจสอบบิตที่มีค่า logic low และแสดงผลออกทาง 7 segment ที่พอร์ต C

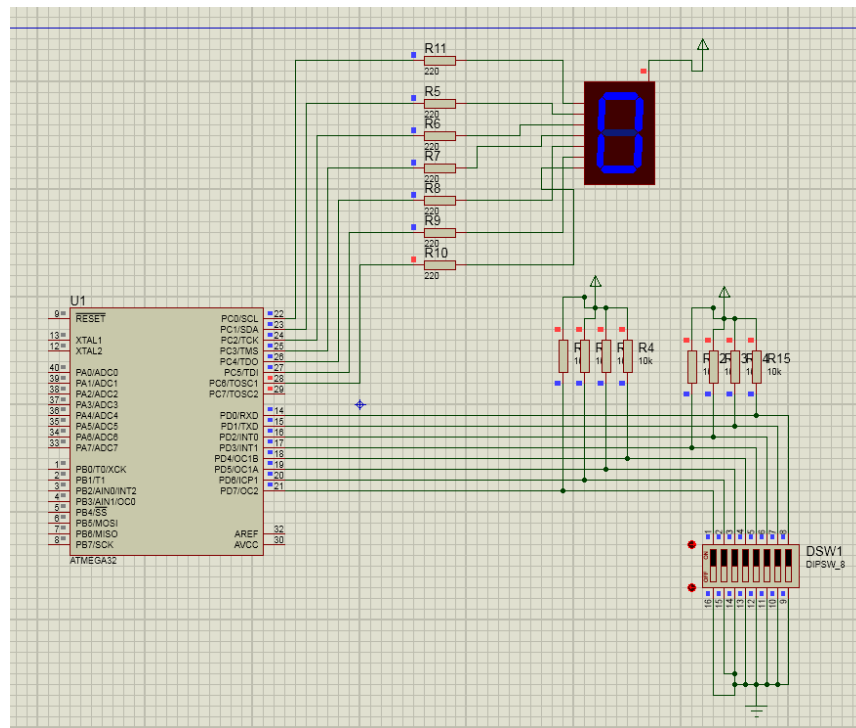
โค้ด ภาษา C

```
#include <avr/io.h>
int main(void)
{
    DDRD = 0x00; // set port B as input
    DDRC = 0xFF; // set port C as output
    unsigned char SWITCH_V, DISP, i, count, mask, test_bit;
    unsigned char LOOKUPTB[] = { 0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111101,
    0b00000111,
    0b01111111,
    0b01101111,
    0b01110111,
    0b01111100,
    0b00111001,
    0b01011110,
    0b01111001,
    0b01110001 };
    while (1)
    {
        //---read input switch via portB
        SWITCH_V = PIND;
        count = 0;
        mask = 0x01;

        for (i=0;i<8;i++)
        {
            test_bit = SWITCH_V & (mask << i);

            if (test_bit)
                count++;
        }
        count = 8-count;
        DISP = LOOKUPTB[count];
        PORTC = DISP;
    }
}
```

รูปการต่อวงจร



Concept :-

เริ่มต้นด้วยการกำหนดให้ PORTC เป็น Output และ PORTD เป็น Input (PIND คือ เป็นการรับข้อมูลเข้ามาจากสวิต) และจะมีตัวนับ 1 ตัวที่ชื่อว่า count โดยใน loop for ในการวนนับ โดยจะทำการ shift แล้วเช็คว่าเป็น high หรือไม่ หากเป็น high ให้เพิ่มค่า count ไปหนึ่ง แล้ววน for และทำการ shift ไปเช็คบิตถัดไปจนครบ 8 บิต จากนั้นนำค่า count ที่ได้ ไป ลบกับ 8 ก็จะได้เป็นจำนวนของ logic low แล้วนำไปชี้ที่ตาราง 7-Segment (LookUpTable) โดยแล้วแสดงผลว่ามีค่าเท่าไรออกมาทาง 7-Segment

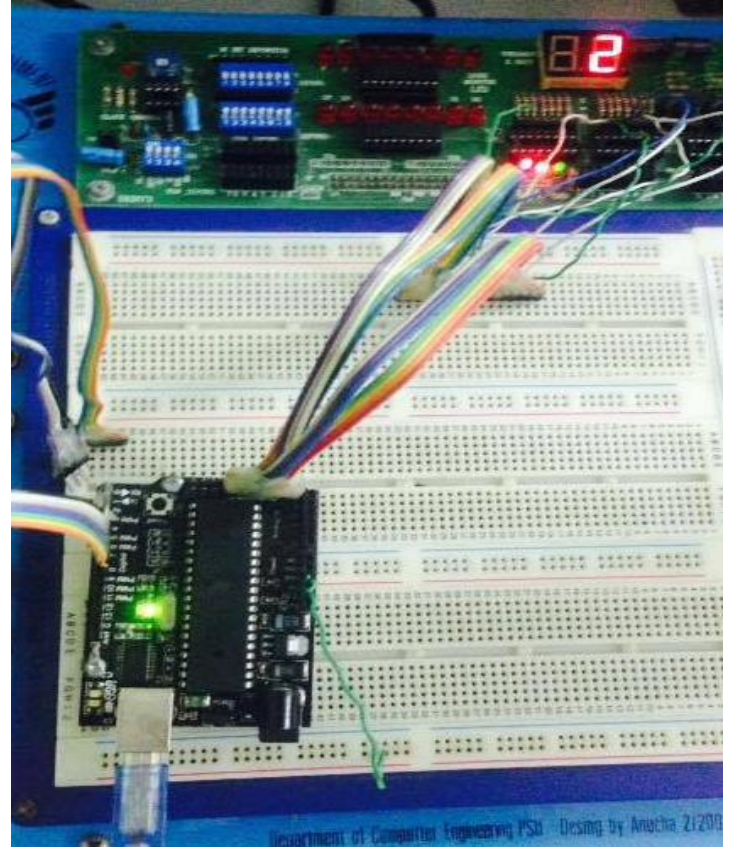
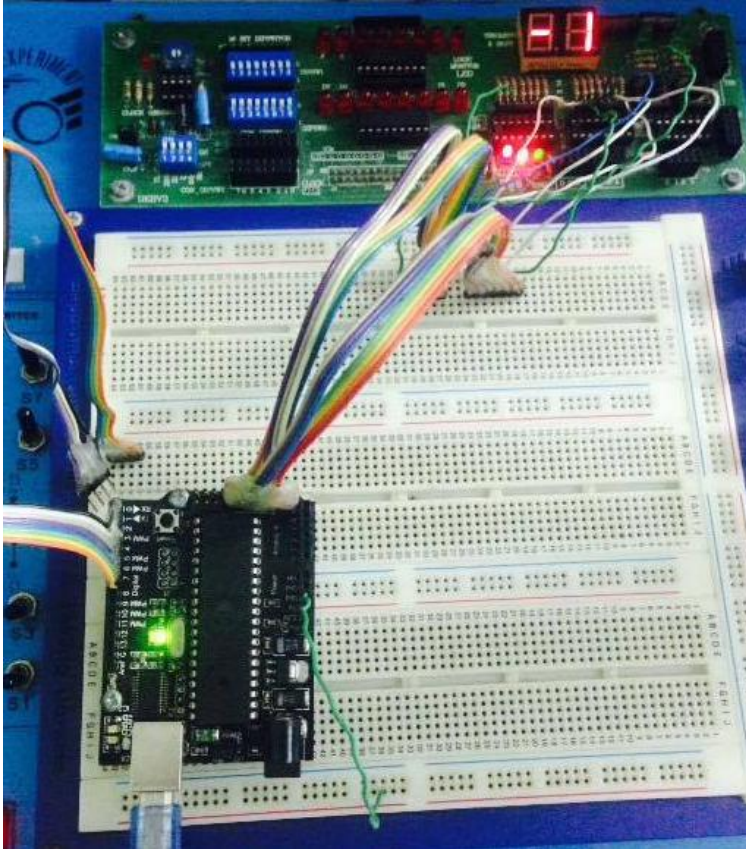
การทดลองที่ 3 เขียนโปรแกรมเพื่ออ่านค่าจาก switch โดยข้อมูลเป็น signed number

จงเขียนโปรแกรมเพื่อแปลงค่าจากสวิตช์เป็นเลขฐานสิบแบบมีเครื่องหมายซึ่งมีค่าอยู่ระหว่าง -8...+7 ออก
แสดงผลทาง LED แบบ 7 segment จำนวน 2 ตัว

```
sketch_mar10a
#include <avr/io.h>
int main (void)
{
    unsigned char LOOKUPTB[] = {
        0b00111111,
        0b00000110,
        0b01011011,
        0b01001111,
        0b01100110,
        0b01101101,
        0b01111101,
        0b00000111,
        0b11111111,
        0b10000111,
        0b11111101,
        0b11101101,
        0b11100110,
        0b11001111,
        0b11011011,
        0b10000110 };
    unsigned char DISPLY, SWITCH_V;

    DDRC  = 0xFF;
    DDRD = 0x00;
    while(1)
    {
        SWITCH_V = PIND;
        SWITCH_V &= 0xF0;
        SWITCH_V = SWITCH_V >> 4;
        DISPLY = LOOKUPTB[SWITCH_V];
        PORTC = DISPLY;
    }
}
```

ผลที่ได้บนบอร์ดทดลอง

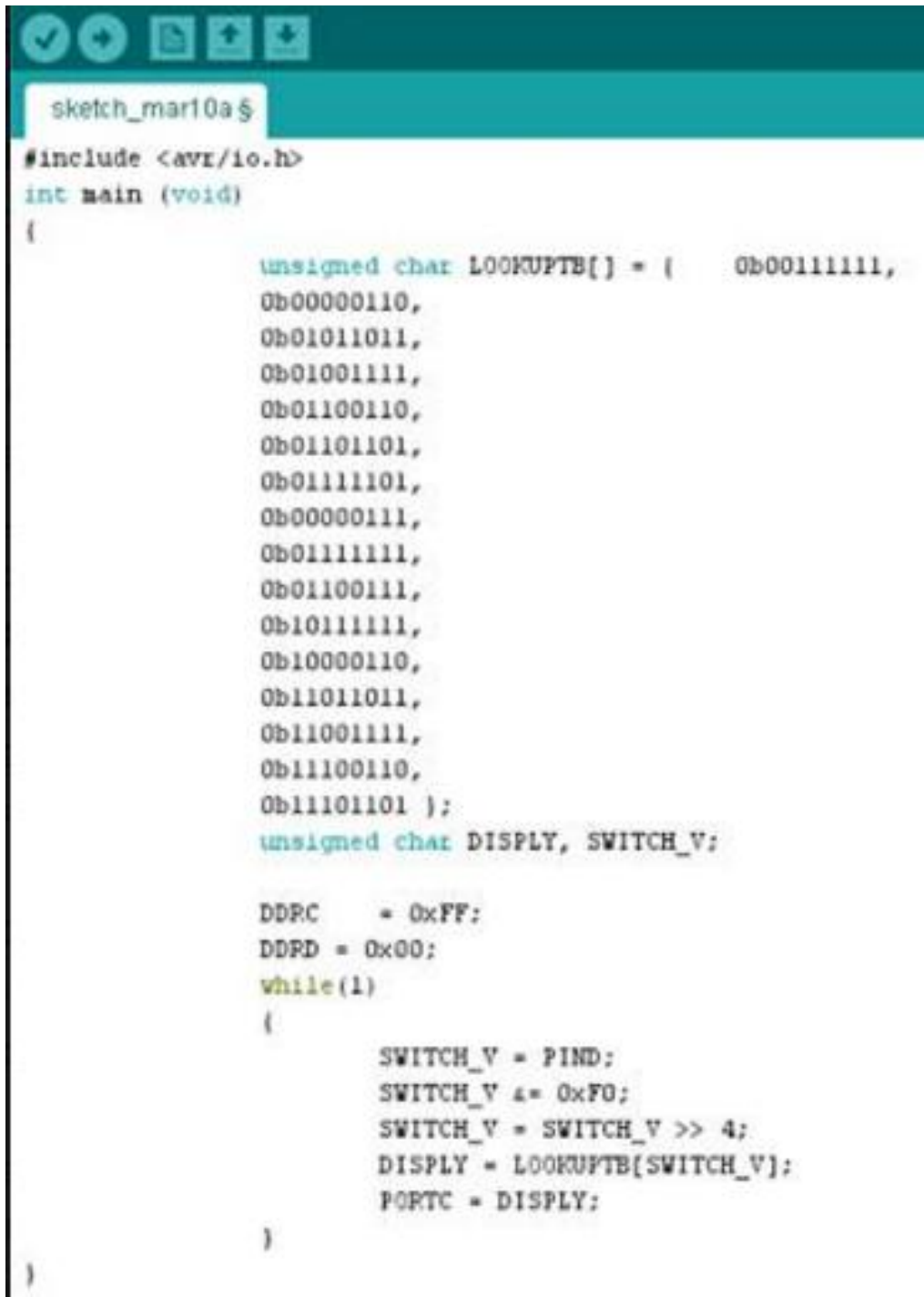


Concept :-

เริ่มต้นด้วยการกำหนดให้ DDRC เป็น Output และ DDRD เป็น Input จะใช้แค่ 4 บิตล่าง โดยแล้วแล้ว จะการ Shift บิตไป 4 บิต จากนั้นก็ไปชี้ตำแหน่งในตาราง 7-Segment โดยในโค้ดนี้ส่วนสำคัญก็คือโค้ดในส่วน ของตาราง 7-Segment (LookUpTable) โดยตารางดังกล่าวจะมีค่าระหว่าง -8 ถึง +7 โดยส่วนที่เป็นลบ ของ ตารางดังกล่าวก็จะกำหนด signed bit ให้มีค่า เป็น 1 หากไม่มีก็กำหนดให้มีค่าเป็น 0 แล้วแสดงผลว่ามีค่าเท่าไร ออกมาทาง 7-Segment

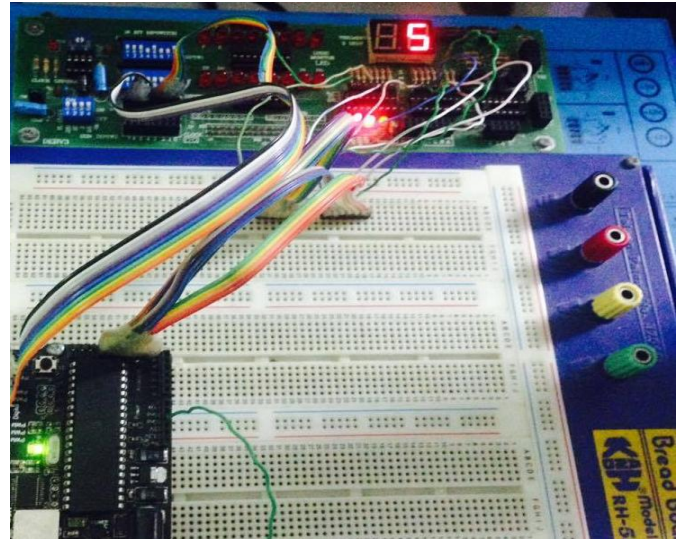
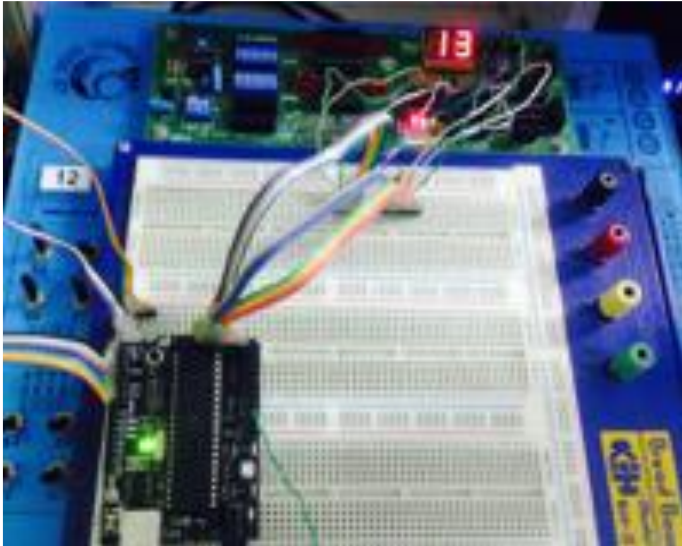
การทดลองที่ 4 เขียนโปรแกรมเพื่ออ่านค่าจาก switch โดยข้อมูลเป็นเลขฐานสิบ

จงเขียนโปรแกรมเพื่อแปลงค่าจากสวิตช์เป็นเลขฐานสิบแบบไม่มีเครื่องหมาย ซึ่งมีค่าอยู่ระหว่าง 0-15 ออกแสดงผลทาง LED แบบ 7 segment จำนวน 2 ตัว



```
sketch_mar10a $  
#include <avr/io.h>  
int main (void)  
{  
    unsigned char LOOKUPTB[] = { 0b00111111,  
    0b00000110,  
    0b01011011,  
    0b01001111,  
    0b01100110,  
    0b01101101,  
    0b01111101,  
    0b00000111,  
    0b01111111,  
    0b01100111,  
    0b10111111,  
    0b10000110,  
    0b11011011,  
    0b11001111,  
    0b11100110,  
    0b11101101 };  
    unsigned char DISPLY, SWITCH_V;  
  
    DDRC = 0xFF;  
    DDRD = 0x00;  
    while(1)  
    {  
        SWITCH_V = PIND;  
        SWITCH_V &= 0xF0;  
        SWITCH_V = SWITCH_V >> 4;  
        DISPLY = LOOKUPTB[SWITCH_V];  
        PORTC = DISPLY;  
    }  
}
```


ผลที่ได้บนบอร์ดทดลอง



Concept :-

เริ่มต้นด้วยการกำหนดให้ DDRC เป็น Output และ DDRD เป็น Input จะใช้แค่ 4 บิตล่าง โดยแล้วแล้ว จะการ Shift บิตไป 4 บิต จากนั้นก็ไปชี้ตำแหน่งในตาราง 7-Segment โดยในโค้ดนี้ส่วนสำคัญก็คือโค้ดในส่วนของ ตาราง 7-Segment (LookUpTable) คล้ายกับโค้ดข้อ 3 โดยตารางดังกล่าวจะมีค่าระหว่าง 0 ถึง 15 โดยส่วนที่เป็นเลข 1 ข้างหน้า ของตารางดังกล่าวก็จะกำหนด signed bit ให้มีค่า เป็น 1 แต่จะมีการต่อของสายที่จุด b กับ c ของ 7-Segment ให้มีการต่อกับบิตในส่วนของ signed bit ดังกล่าว หากไม่มีก็กำหนดให้มีค่าเป็น 0 แล้ว แสดงผลว่ามีค่าเท่าไรออกมาทาง 7-Segment

การทดลองที่ 5

การทดลองที่ 5.1 ออกแบบและทดสอบระบบสวิตช์สัมผัสสำหรับเปิดปิดเครื่องใช้ไฟฟ้า

- เมื่อเริ่มจ่ายไฟให้กับโปรแกรม หลอด L1 จะดับอยู่ และ 7-segment แสดงคำว่า of
- เมื่อกดสวิตช์ หลอด L1 จะติด และ 7-segment แสดงคำว่า on
- เมื่อกดสวิตช์อีกครั้ง หลอด L1 จะดับ และ 7-segment แสดงคำว่า of

```
#include <avr/io.h>

unsigned char TS7SEG[] = {
    0b01110001,
    0b01010100,
    0b01011100
};

unsigned char mode, tmp, current;

int main()
{
    // Write your code here
    DDRA = 0x00;
    DDRC = 0xFF;
    DDRD = 0xFF;

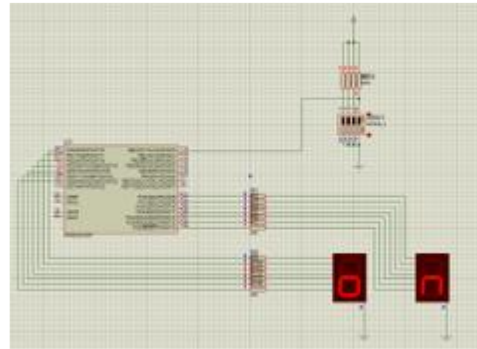
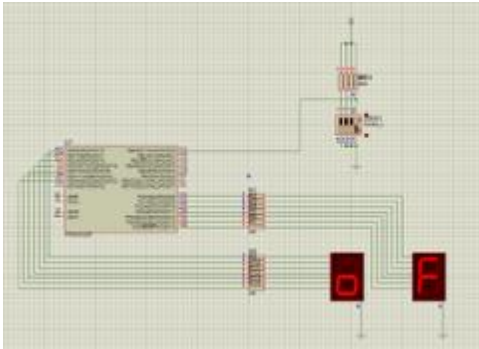
    PORTD = TS7SEG[2];
    mode = 0x00;
    PORTC = TS7SEG[0];
    tmp = 0x00;
    current = 0x00;

    while (1){
        current = tmp;
        tmp = PINB;
        tmp <= 0x0F;

        if(tmp != current){
            mode = mode ^ 0x01;
            PORTC = TS7SEG[mode];
        }
    }

    return 0;
}
```

รูปการต่อวงจร ขณะที่แสดง off และ on



Concept :-

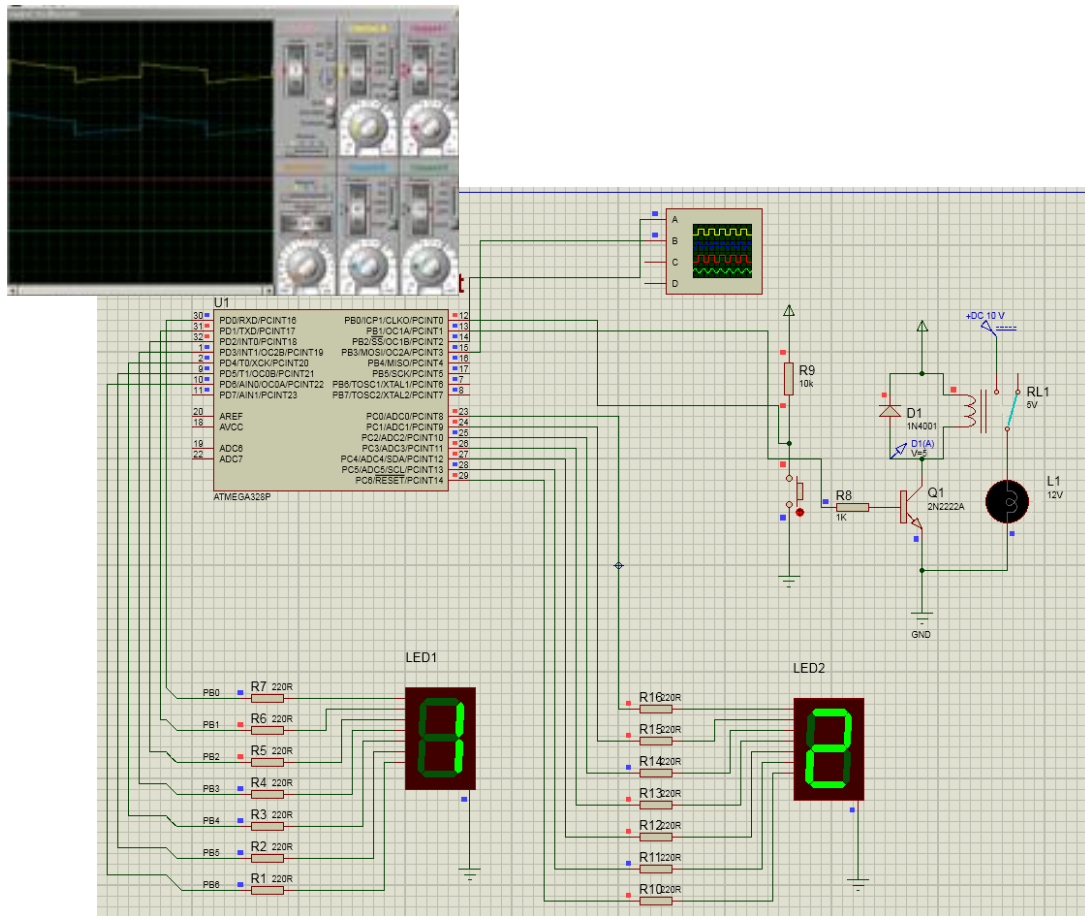
เริ่มต้นด้วยการกำหนดให้ PORTB เป็น Input และ PORTC,D เป็น Output โดยให้ PORTD อ่านค่าจากตาราง ซึ่งจะเป็นการแสดงออกทาง 7 segment เป็นตัวอักษร o และ port C อ่านค่าจากตาราง TB7Segment เช่นกัน ซึ่งจะเป็นการแสดงออกทาง 7 segment เป็นตัวอักษร F เริ่มกดสวิตช์ แล้วจะมี input เข้ามาทาง PINB จากนั้นนำมา AND กับ 0x0F เพราะจะดูแค่บิตต่ำ จากนั้นเช็คเงื่อนไข if ให้ทำงานในเงื่อนไขเพื่อเป็นการเช็คการกดสวิตช์โดยการเช็คก็จะเช็ค ค่า 2 ตัวคือ ขณะปัจจุบันเป็นอะไร และ mode อะไร หากมีการกดสวิตช์ 7segment ของ port C แล้วแสดงผลว่ามีค่าเท่าไรออกมาทาง 7-Segment

```
ISR (TIMER1_OVF_vect)
{
    TCNT1 = VALUE_T1;
    PORTB = PORTB ^ 0x0C; // 0000 1100

    count++;

    if(count == 0)
    {
        count = 20;
    }
    PORTC = LOOKUPTB[count%10];
    PORTD = LOOKUPTB[count/10];
}
```




รูปการต่อวงจร



Concept :-

กำหนดให้ port B, C และ D เป็น input เริ่มต้นกำหนดให้ count เริ่ม 20 เมื่อเริ่มนับถอยหลังจะให้ port C แสดงตัวเลขของหลักหน่วย โดยการนำค่า count mod กับ 10 ส่วนหลักสิบจะแสดงออกทาง port D ทำโดยการนำค่า count หารกับ 10 และการ set timer จาก TCCR1A= 0x00 เป็นการบอกว่าเป็น Timer ขนาด 16 บิต แบบ Normal และ TCCR1B = 0x04 คือใช้ prescaler เท่ากับ 256 และในส่วนของ ISR เคลียบิตที่ 0 กับ 1 โดยการทำการ OR กัน แล้วลดค่า count ลง 1 เรื่อย จน มีค่าเป็น 0 ก็ให้ count มีค่าเป็น 20 อีกครั้ง โดยแต่ละรอบจะแสดงว่ามีค่าเท่าไรออกมาทาง 7-Segment

การทดลองที่ 5.4 รวมโปรแกรม

- เมื่อเริ่มจ่ายไฟให้กับเครื่อง เครื่องใช้ไฟฟ้าจะดับอยู่ และแอลอีดีแสดงสถานะ 
- เมื่อกดสวิตช์ 1 ครั้งจะเป็นการกดเปิดเครื่องใช้ไฟฟ้าโดยมีแอลอีดี 7 เซกเมนต์ 2 ตัวแสดงสถานะของวงจรเป็น 
- เมื่อกดสวิตช์อีก 1 ครั้งจะเป็นการสั่งปิดเครื่องใช้ไฟฟ้า แต่เครื่องใช้ไฟฟ้าจะไม่ถูกตัดไฟในทันที แต่จะมีการหน่วงเวลาออกไป 20 วินาที พร้อมทั้งแสดงการนับลงที่แอลอีดีทั้งสอง เมื่อครบ 20 วินาทีจึงดับเครื่องใช้ไฟฟ้าแล้วแสดงสถานะที่แอลอีดีเป็น 

```
#include <avr/io.h>
#define VALUE 34284
unsigned char count = 20;
unsigned char zero = 0;
unsigned char zero1 = 0x01010100;
unsigned char zero2 = 0x01011000;
unsigned char zero3 = 0x01110001;
unsigned char zero4 = 0x01011000;
unsigned char LOCKSTEP[] = {
    0x00111111,
    0x00001111,
    0x01111111,
    0x01001111,
    0x01101111,
    0x01111111,
    0x00000111,
    0x00111111,
    0x01001111,
    0x01101111,
    0x01111111,
    0x00011111,
    0x00111111,
    0x01111111,
    0x01111111,
    0x01111111
};

int main()
{
    //set output
    DDRA = 0xFF; // Set pin (bit 0) as output 1111 1110
    DDRB = 0xFF;
    DDRD = 0xFF;

    PORTB = PORTB & 0xF3; // 1111 0011

    PORTC = LOCKSTEP[count*10]; // unit
    PORTD = LOCKSTEP[count/10]; // ten

    //set timer
    TCCR1A = 0x00;
    TCCR1B = 0x00;
    TIMSK1 = 0x01;

    PCICR = 0x01; //enable check port B (14/8) Pin change interrupt
    PCMSK0 = 0x00000001; //use port B pin 0123

    //Starts OFF
    PORTC = zero0;
    PORTD = zero4;
    TIMSK1 = 0x00;

    cli();
    TCCR1A = VALUE;
    sei();
    while (1);
    return 0;
}

ISR(TIMSK1_OVF_vect)
{
    TCCR1A = VALUE;
    PORTB = PORTB ^ 0x0E; // 0000 1110 back bit

    if(count == 0)
    {
        PORTC = zero3; // unit
        PORTD = zero4; // ten
        TIMSK1 = 0x00;
        PORTB = 0x00; //LED OFF
    }
    else if(count == 1)
    {
        PORTB = 0x02; //LED on
        count--;
        PORTB = PORTB ^ 0x0C; // 0000 1100 back bit
        PORTC = LOCKSTEP[count*10]; // unit
        PORTD = LOCKSTEP[count/10]; // ten
    }
}
```

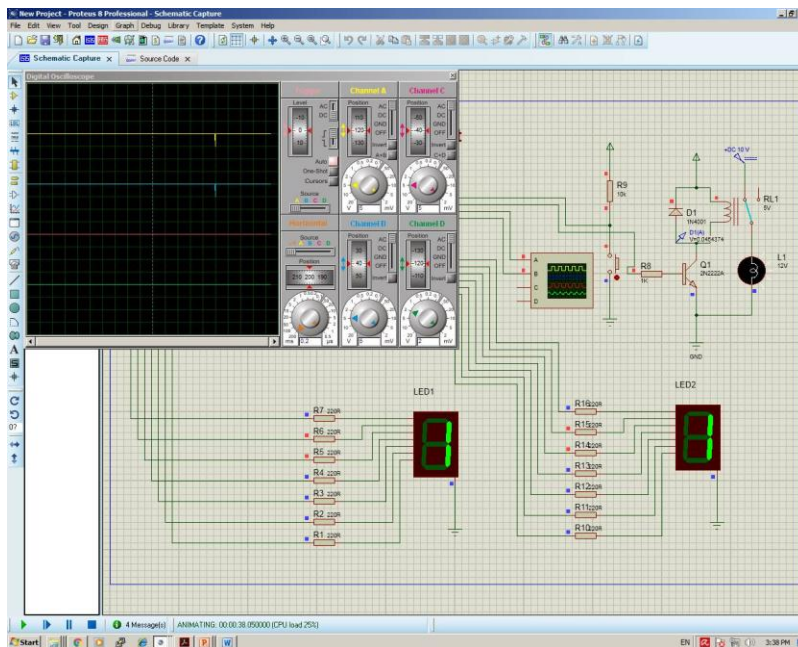
```

else
{
    PORTB = 0x02; //LED on
    count--;
    PORTB = PORTB ^ 0x0C; // 0000 1100 back bit
    PORTC = LOOKUPTB[count%10]; // unit
    PORTD = LOOKUPTB[count/10]; // ten
}

ISR (PCINT0_vect){
    TIMER1 = 0x01;
    PORTC = zero1; // unit
    PORTD = zero2; // ten
    count = 21;
}

```

รูปการต่อวงจร



Concept :-

โดยในส่วนนี้ก็คือจะเป็นนำข้อ 5.1 กับ 5.3 มารวมกัน โดยอธิบายของโค้ดก็เหมือนกับก่อนหน้านี้ แล้วที่มีเพิ่มเติมเข้ามาคือ เริ่มต้นให้ 7segment แสดงว่า 0F ก่อนและเมื่อมีการกดสวิทซ์จะการทำงานให้ ISR (TIMER1_OVF_vect) ซึ่งในตอนเริ่มต้นจะเข้าเงื่อนไข else โดย 7segment จะแสดงคำว่า on และ LED จะติด แล้วค่า count จะเริ่มนับที่ 20 แล้วลดลงเรื่อยๆ และหาก count ลดลงเหลือ 0 จะแสดงคำว่า 0F และ LED ก็ระดับหกกดสวิตช์อีกครั้งก็จะเกิดรูปแบบเดิมอีกครั้ง