Microcomputer Control Systems I
2018/2019 Laboratory Tasks Report

In all of the tasks, a microcomputer controller based on ATMEL AT89S52 is used.

Task 1 – Blinking of LEDs

This task consisted of 2 main parts, which are programming LEDs module to switch on and off constantly, and implementing a delay function to decrease the switching speed down to a value, that can be oberved by naked human eye.

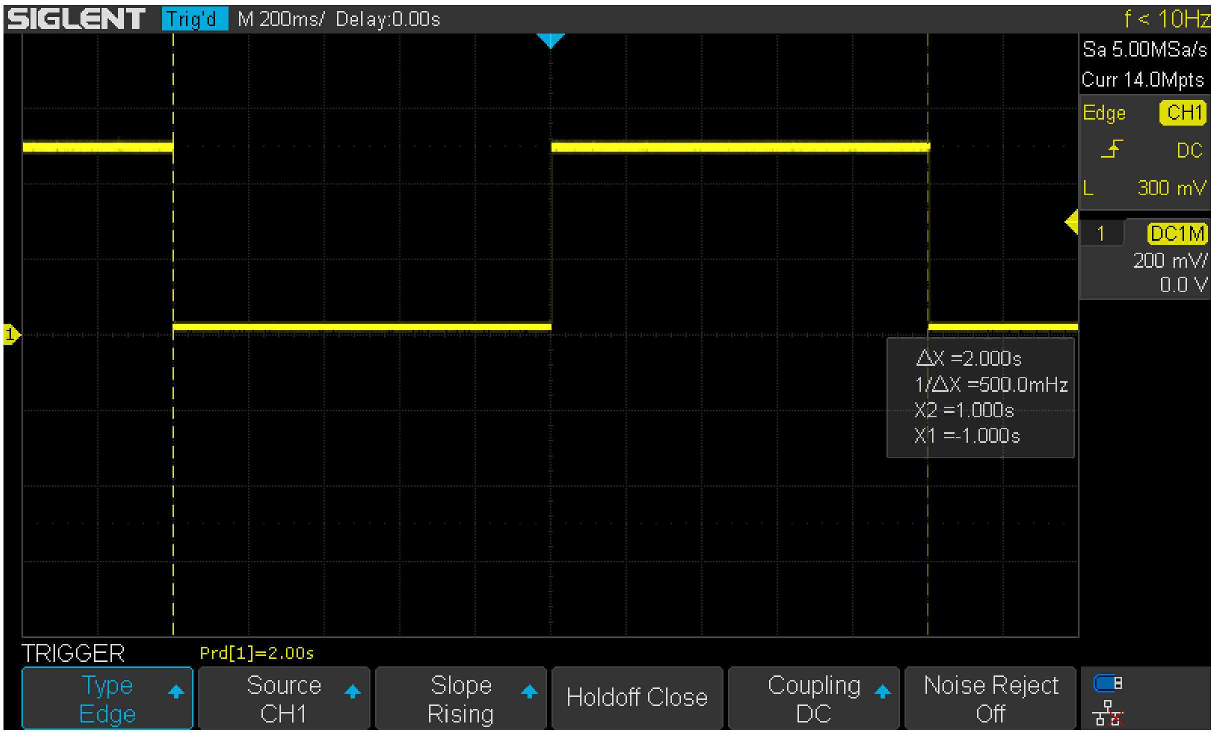
In order to achieve these main tasks, our group has first activated LEDs module on the controller, and implemented the delay function to wait approximately 2 seconds between every cycle of the loop, which switched the LEDs on and off. Please find required codes and oscilloscope measurements below.

Task I - Code

```
#include <REGX52.H>
void wait(void){
unsigned int i;
for (i=0;i<10000;i++);
}
void main(void)
{
        unsigned int k;
        P3_7 = 0; // ON
        P3_6 = 0; // OFF
        P1_4 = 0; // OFF
        P1_3 = 0; // OFF
        while(1)
            {
                P2 = ~P2;
for (k=0; k<40; k++){
wait();

        }
    }
    }
```

Task I – Oscilloscope Measurement



ΔX =2.000s
1/ΔX =500.0mHz
X2 =1.000s
X1 =-1.000s

Prd[1]=2.00s

Task II – Blinking of LEDs Using Interrupt

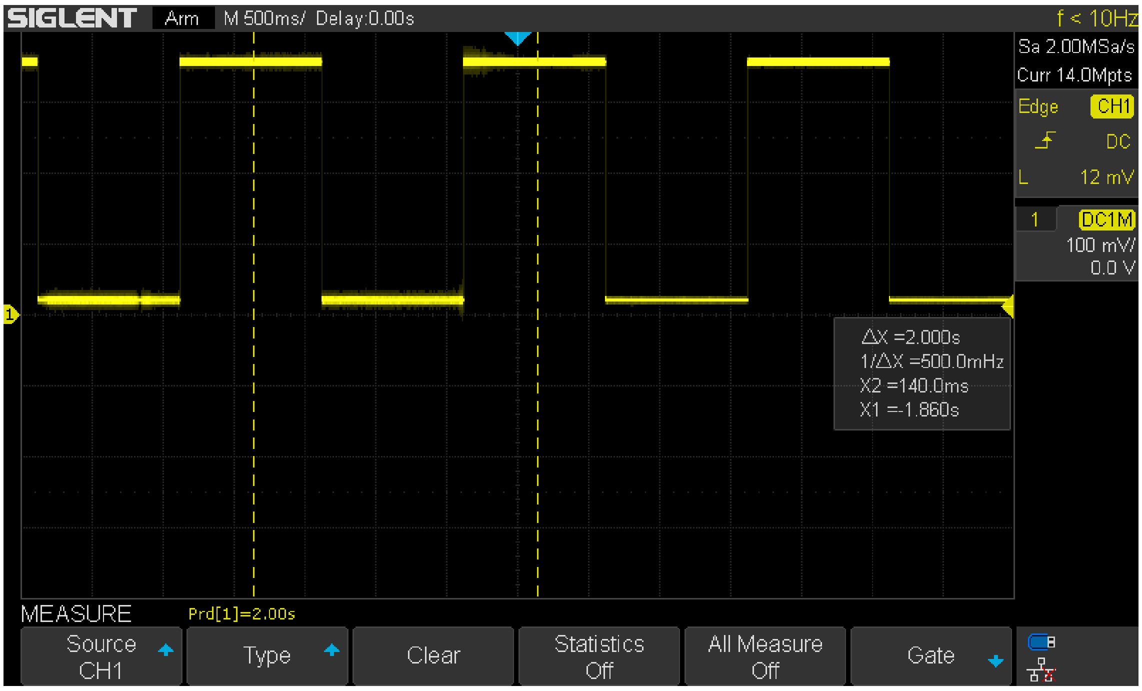
Making an improvement over the first task, we've implemented blinking LEDs via usage of Timer0, instead of our own delay function. For this purpose, we've determined preset values for Timer0 registers, and set the blinking time between lights to 2 seconds.

The main advantages of using the Timer and Interrupt cycles compared to the previous work are our availability to set and measue the time correctly, therefore avoiding further complications in case of creating a more complex project on the same basis. Please find required codes and oscilloscope measurements below.

Task II - Code

```
#include <REGX52.H>
unsigned int k=0;
void main(void)
{
        P3_7 = 0; // ON
        P3_6 = 0; // OFF
        P1_4 = 0; // OFF
        P1_3 = 0; // OFF
        TMOD=0x01;
        EA=1;
        ET0=1;
        TL0=0xC0;
        TH0=0x63;
        TR0=1;
        while(1){
        }
}
void it_timer0(void) interrupt 1{
        k++;
            TL0=0xC0;
        TH0=0x63;
        TF0=0;
    if(k==50){
        P2=~P2;
        k=0;
        }
}
```

Task II – Oscilloscope Measurement



ΔX =2.000s
1/ΔX =500.0mHz
X2 =140.0ms
X1 =-1.860s

Prd[1]=2.00s

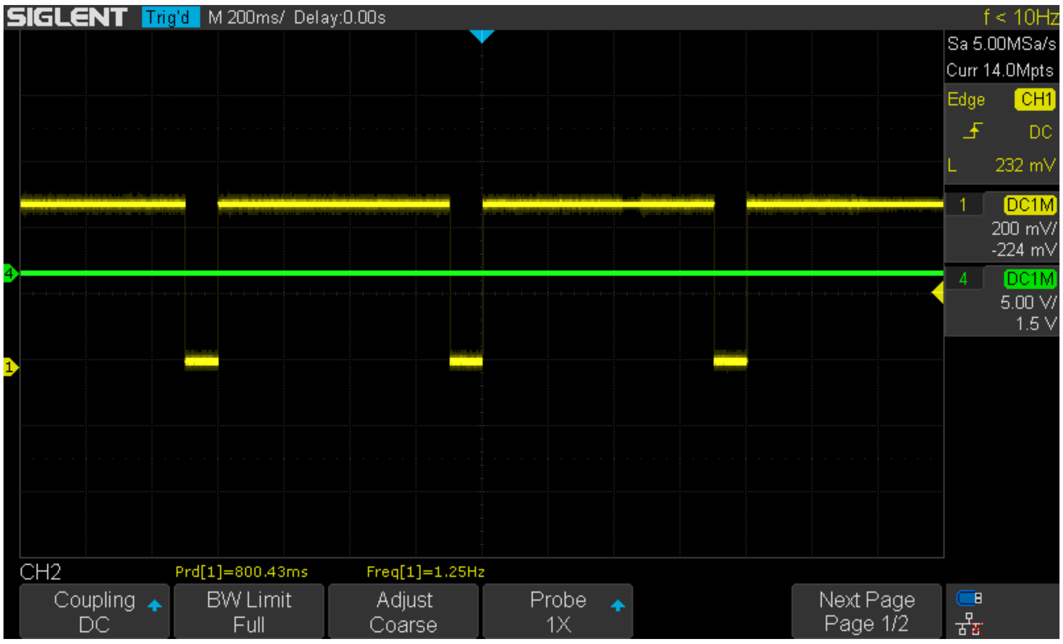
Task III – Rotation of Turned-On One Led

Based on the previous tasks, we've modified the project to allow one LED light to be turned on, and move to the next LED by turning it off, and turning the next LED on at the same time. In the implementation, we've given P2 a Hexadecimal value to turn on the right most LED, and shifted this one bit at the end of each interrupt cycle, which is again modified to delay the shiftings 100ms in-between. Please find the required code and oscilloscope measurements below.

Task III - Code

```
#include <REGX52.H>
unsigned int k=0;
unsigned int Port2 = 0x1;
unsigned int a=1;
void main(void)
{
            P3_7 = 0; // ON
            P3_6 = 0; // OFF
            P1_4 = 0; // OFF
            P1_3 = 0; // OFF
            TMOD=0x01;
            EA=1;
            ET0=1;
            TL0=0xC0;
            TH0=0x63;
            TR0=1;
            while(1){
            }
}
void it_timer0(void) interrupt 1{
            k++;
                  TL0=0xC0;
            TH0=0x63;
            TF0=0;
            if(k>a){
            k=0;
            }
      if(k==a){
                  Port2=Port2<<1;
                  P2=~Port2;
                  if(Port2==0x100){
                        while(Port2!=0x01){
                              Port2=Port2>>1;
                              P2=~Port2;
                        }
            }
                  k=0;
}
            }
```

Task III – Oscilloscope Measurement



Task IV – Different Rotation Speed of Turned-On One Led

Over Task III, we've implemented buttons to adjust the time interval between rotation of turned-on LEDs. After activating button controllers on the microcomputer board, we've set buttons P1.0, P1.1, P1.2 to adjust time interval between shifts to 1s, 100ms, 20ms accordingly. Please find the required codes below, and oscilloscope measuremets on the page of previous task .

Task IV - Code

```
#include <REGX52.H>
unsigned int k=0;
unsigned int Port2 = 0x1;
unsigned int a=1;
void main(void)
{
        P3_7 = 0; // ON
        P3_6 = 0; // OFF
        P1_4 = 0; // OFF
        P1_3 = 0; // OFF
        TMOD=0x01;
        EA=1;
        ET0=1;
        TL0=0xC0;
        TH0=0x63;
        TR0=1;
        while(1){
                if (P1_0 == 0 && P1_1 == 1 && P1_2 == 1){
                        a=50;
                }
                if (P1_0 == 1 && P1_1 == 0 && P1_2 == 1){
                        a=5;
                }
                if (P1_0 == 1 && P1_1 == 1 && P1_2 == 0){
                        a=1;
                }
        }
}
void it_timer0(void) interrupt 1{
        k++;
                TL0=0xC0;
        TH0=0x63;
        TF0=0;
        if(k>a){
        k=0;
        }
    if(k==a){
                Port2=Port2<<1;
                P2=~Port2;
                if(Port2==0x100){
                        while(Port2!=0x01){
                                Port2=Port2>>1;
                                P2=~Port2;
                        }
        }
                k=0;
}
        }
```

Task V – Multiplexed Display

This task is built over Task no. 2, and aims to achieve displaying number “1234” on multiplexed display module of the board. To implement this task, after activating the display module, we’ve followed the principle of switching between numbers in an interval of 2.5ms, so that human eye could not capture the switching and recognize the display as a whole number. The reason behind is us, having just enough bits to represent display place and number to display, therefore we needed to set digit places one by one accordingly, to create the number of “1234” . Having an interval of 2.5ms let us create the illusion of constantly lightning numbers.

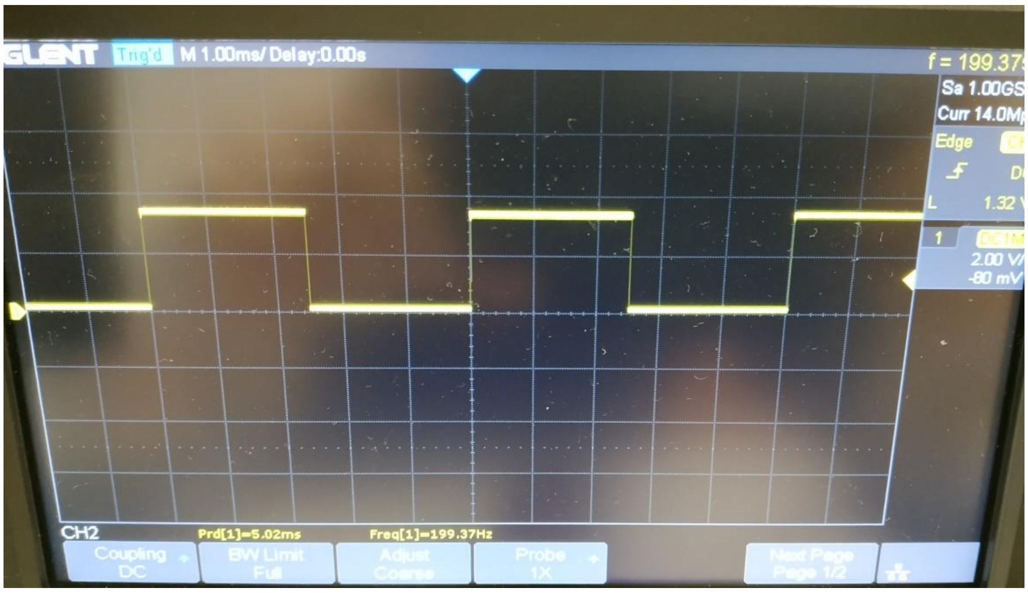
This principle also makes it possible to change the displayed number, since it goes through the cylce of checking each number over and over again, which is going to be helpful to us in the next tasks. Please find the required code below, there is no oscilloscope measurement done for this task.

Task V - Code

```
#include <REGX52.H>
unsigned int PortDisplay=0xE1;
unsigned int PortDisplay2=0xD2;
unsigned int PortDisplay3=0xB3;
unsigned int PortDisplay4=0x74;
char a = 4;
char b = 3;
char c = 2;
char d = 1;
char pos = 0;
unsigned int k=0;
unsigned int Display = 0x1;
void main(void)
{
        P3_7 = 1; // OFF
        P3_6 = 1; // ON
        P1_4 = 0; // OFF
        P1_3 = 0; // OFF
        TMOD=0x01;
        EA=1;
        ET0=1;
        TL0=0x78;
        TH0=0xEC;
        TR0=1;
        while(1){
        }
}
void it_timer0(void) interrupt 1{
        k++;
            TL0=0x78;
        TH0=0xEC;
        TF0=0;
        P2=~P2;
        switch(pos){
            case 0:
```

a. P2=P0/1000;
b. pos=1;
2. break;
3. case 1:
a. P2=P0/100;
4. pos=2;
5. break;
6. case 2:
a. P2=P0/10;
7. pos=3;
8. break;
9. case 3:
a. P2=P0;
b. pos=0;
10. break;
vi. }
17. }

Task V – Oscilloscope Measurement



Task VI – Analog to Digital Converter with Multiplexed Display

We've modified last task's implementations to have the potentiometer attached to the board, change the number displayed on the multiplexed display module. We benefit from the advantage of the principle we've chosen to use, which allowed us check the displayed number between every cycle. In this implementation, we send the number to be displayed not by hardcoding, but by potentiometer, which is represented by 8 bits, therefore indicating numbers between 0 and 255 without further modifications. Please find the required code below.

Task VI - Code

```
#include <REGX52.H>
unsigned int PortDisplay=0xE1;
unsigned int PortDisplay2=0xD2;
unsigned int PortDisplay3=0xB3;
unsigned int PortDisplay4=0x74;
int PortDisp;
char a = 4;
char b = 3;
char c = 2;
char d = 1;
char pos = 0;
unsigned int k=0;
unsigned int Display = 0x1;
void main(void)
{
        P3_7 = 0;
        P3_6 = 1;
        P1_4 = 1;
        P1_3 = 1;
        TMOD=0x01;
        EA=1;
        ET0=1;
        TL0=0x78;
        TH0=0xEC;
        TR0=1;
        while(1){
        }
}
void it_timer0(void) interrupt 1{
        k++;
            TL0=0x78;
        TH0=0xEC;
        TF0=0;
        P2=~P2;
        PortDisp = P0;
        a=PortDisp/10;
        b=(PortDisp/10)%10;
        c=PortDisp%10;
        d=PortDisp;
```

```
                x.   switch(pos){
                        1.   case 0:
                                a.   P2= 0x70 | (c&0x0F);
                                b.   pos=1;
                        2.   break;
                        3.   case 1:
                                a.   P2= 0xB0 | (b&0x0F);
                        4.   pos=2;
                        5.   break;
                        6.   case 2:
                                a.   P2= 0xD0 | (a&0x0F);
                        7.   pos=3;
                        8.   break;
                        9.   case 3:
                                a.   P2= 0xE0 | (d&0x0F);
                                b.   pos=0;
                        10.  break;
                xi.  }
        18.  }
```

Task VII – Counter Using Multiplexed Display

Over the Task no.5, we've implemented a counter that increases by 1 per 100ms, goes from 0 to 9999, and displays this number on the multiplexed display module.

Task VII - Code

```
#include<REGX52.H>
int i;
int x;
char a=4;
char b=3;
char c=2;
char d=0;
int value;
char position=0;
int count=0;
void main (void){
P3_6=1;
P3_7=0;
P1_3=0;
P1_4=1;
x=0x01;
TMOD=0x01;
EA=1;
ET0= 1;
TL0= 0x78;  //highest value - 5000=EC78
TH0=0xEC ;
TR0=1;
while(1) {
}
}
void it_timer0 (void) interrupt 1 {
TL0= 0x78;
TH0=0xEC;
TF0=0;
i++;
P2 = ~P2;
if(i==40) {
count++;
value=count;
d=(value/1000);
c= (value/100)%10;
b=(value/10)%10;
a=value%10;
i=0;
if(count==10000){
i=0;
}
}
switch(position){
case 0:
P2=0x70 | (a & 0x0F); // ONES
position=1;
break;
case 1:
    P2=0xB0 | (b & 0x0F); // TENS
position=2;
    break;
case 2:
    P2=0xD0 | (c & 0x0F); // Hundreds
position=3;
    break;
case 3:
    P2=0xE0 | (d & 0x0F); // Thouands
position=0;
    break;
}
}
```