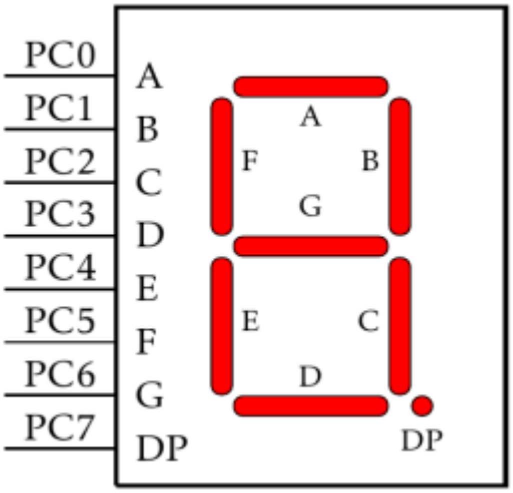
Side Yılmaz

Kağan Cenan

Lab 2 – Interfacing with External Components

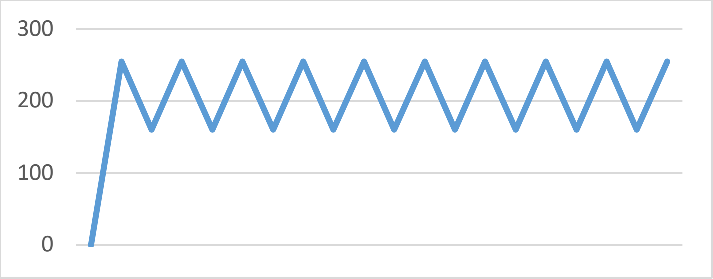
In this lab, we learned to design and implement a stopwatch with using timer interrupt. Firstly we did stopwatch which display seconds (in two digits) and tenths of seconds using three seven segment. Pins of PORTC is connected to seven segment display unit such that:



For all numbers we used the following hexadecimals in terms of the above figure :

Digit	gfedcba	a	b	c	d	e	f	g
0	0$3F	on	on	on	on	on	on	off
1	0$06	off	on	on	off	off	off	off
2	0$5B	on	on	off	on	on	off	on
3	0$4F	on	on	on	on	off	off	on
4	0$66	off	on	on	off	off	on	on
5	0$6D	on	off	on	on	off	on	on
6	0$7D	on	off	on	on	on	on	on
7	0$07	on	on	on	off	off	off	off
8	0$7F	on	on	on	on	on	on	on
9	0$6F	on	on	on	on	off	on	on

To get time, we used timer interrupted. We arranged the numbers, pvalue=1024, system clock of CLK = 1 MHz. Because the timer is 8-bit, max value is 255. Between 0-255 there are 256 numbers. So we changed the graph. We reached 10Hz as counter frequency equal to 0.1 sec



$$Hz = sytemclock \ \% \ (256 - 158) * 1024)$$

$$10 \ Hz = 1MHz \ \% \ (98 * 1024)$$

When the timer finish one loop which means goes from 0 to 255, we incremented one counter which we called ‘bigcounter’. So we reached the real time.

The aim of task 1 is implementing 2 button. When button0 is pressed , the stopwatch start measuring and displaying the elapsed time (starting from 00.0 seconds). Button1 is pressed, the stopwatch stop and freeze its display. For this implementation we used two instruction which are “sei” and “cli”. “sei” is setting global interrupt enables, contrast ”cli” is clearing global interrupt. Starting global interrupt causes that the timer interrupt is able to work, so we could start our “bigcounter”. When we clear global interrupt, possibility of incrementing “bigcounter” became unavailable. So it provide to freeze the elapsed time. Also to initialize interrupt we used the following code:

```
Init_tov0:
ldi r16, (1<<CS02)|(1<<CS00) ;CLK/1024
out TCCR0, r16
ldi r16, (1<<TOV0) ;Clear pending interrupts
out TIFR, r16
ldi r16, (1<<TOIE0) ;Enable T0 overflow interrupt
out TIMSK, r16
ret
T0_OVF:
push r16
in r16, SREG
push r16 ;Save processor status
out TCNT0, r20 ;Initialize T0 with r20
in r16,TIFR
sbrs r16,TOV0
inc bigcounter ;To increment the elapsed time
pop r16
out SREG, r16 ;Recover processor status
pop r16
reti
```

Next step in the code was displaying the numbers on the board to show elapsed time. The “check” subroutine which is in the bottom provide to change the stored number of “tmp” which is a defined register that provide PORTC’s output. As you can see, we are checking the recent value of counter and we are implementing the related value on “tmp” register . “c2” and ”c3” also provide to control of numbers which are more than 0.9 and 9.9. It is providing incrementing in the board so we can change the numbers to which comes next.

```
check:
       cpi counter,10 ; Display 9 on seven segment display
       brne cont
       ldi counter,$00
       inc c2

       cpi c2,10
       brne cont
       ldi c2,0
       inc c3

       cpi c3,10
       brne cont
       ldi c2,0
       ldi c3,0

       cont:
       cpi counter,0 ; Display 0 on seven segment display
       breq zero
       cpi counter,1 ; Display 1 on seven segment display
       breq one
       cpi counter,2 ; Display 2 on seven segment display
       breq two
       cpi counter,3 ; Display 3 on seven segment display
       breq three
       cpi counter,4 ; Display 4 on seven segment display
       breq four
       cpi counter,5 ; Display 5 on seven segment display
       breq five
       cpi counter,6 ; Display 6 on seven segment display
       breq six
       cpi counter,7 ; Display 7 on seven segment display
       breq seven
       cpi counter,8; Display 8 on seven segment display
       breq eight
       cpi counter,9 ; Display 9 on seven segment display
       breq nine
       cpi counter,$FF ; Display 9 on seven segment display
       breq zero

       one:
       ldi tmp,$06
       ret
       two:
       ldi tmp,$5B
       ret
       three:
       ldi tmp,$4f
       ret
       four:
       ldi tmp,$66
       ret
       five:
       ldi tmp,$6d
       ret
       six:
       ldi tmp,$7d
       ret
       seven:
       ldi tmp,$07
       ret
       eight:
       ldi tmp,$7f
       ret
       nine:
       ldi tmp,$6F
       ret
       zero:
       ldi tmp,$3f
       ret
```

To display numbers in the code we used the following code, because we called check subroutine we could display numbers what we want on the board.

```
aaa:  ;general display form
       ; Display No#0
       ldi tmp, (1<<0)     ; Make bit#0 one, others zero
       out PORTA, tmp
       cp bigcounter,r26
       brne first
       inc c1
       ldi bigcounter,0
first:
       cpi ispress,1
       brne h
       mov counter,dig0
       rcall check
       mov dig0,counter
       rjmp hh

h:
       mov counter,c1
       mov dig0,c1
       rcall check
       mov c1, counter
hh:
       out PORTC, tmp
       rcall Delay

       ; Display No#1


       ldi tmp, (1<<1)     ; Make bit#1 one, others zero
       out PORTA, tmp
       cpi ispress,1
       brne k
       mov counter,dig1
       rcall check
       mov dig1,counter
       rjmp kk

k:
       mov counter,c2
       mov dig1,c2
       rcall check
       mov c2, counter
kk:    subi tmp,$80
       out PORTC, tmp
       rcall Delay
kkkk:
       ;; Display No#2

       ldi tmp, (1<<2)     ; Make bit#2 one, others zero
       out PORTA, tmp
       cpi ispress,1
       brne p
       mov counter,dig2
       rcall check
       mov dig2,counter
       rjmp pp

p:
       mov counter,c3
       mov dig2,c3
       rcall check
       mov c3, counter
pp:    out PORTC, tmp
       rcall Delay
       rjmp Loop ;goes back to loop to start again code.
```

The aim of the task 2 is implementing a new button which let us to know lap time between the time of push buttons. Firstly we add second counter which we use for measuring lap time and we started to count this counter when button2 is clicked. We could show the laptime between the time of pushing two button on PORTB successfully. You can see it on the board clearly. We incremented value of laptime in the timer interrupt, if button2 is clicked.

```
       cpi ispress,1 ;checking button2 is cliked and we are starting to incrementing
lapcounter
       brne side ;if not continue to side
       inc lapcounter
```

The aim of the task 3 is implementing a new button which let us slow-down functionality, such that timer toggles to 10 times slower or normal operation. For this operation, we changed input of the compare instruction on here :

```
cp bigcounter,r26
```

So that we could check the counter’s value 10 times less. It caused less comparison result. As a result we reached 10 times less toggling which means 10 times slower clock on board. It was not a trivial part for us , we just showed it on board when button3 is clicked