Beste Öztop 2375624
Deniz Karakay 2443307 05.12.2022

EE447 Introduction to Microprocessors

Laboratory-4 Preliminary Work

Question-1

TIMER0A Configuration Code Explanation:

```
TIMER0->CTL         &=0xFFFFFFFE; //Disable timer during setup
```

Before making the initializations fort he Timer0 A , the timer must be disabled.

```
TIMER0->CFG             =0x04;  //Set 16 bit mode
```

Mode is set to be seperate timer 16-bit.

```
TIMER0->TAMR        =0x02; // set to periodic, count down
```

Timer A Mode Register (_TAMR): Used to set the function of the timer.

- TAMR [1:0]: One-shot, Periodic or Capture Mode

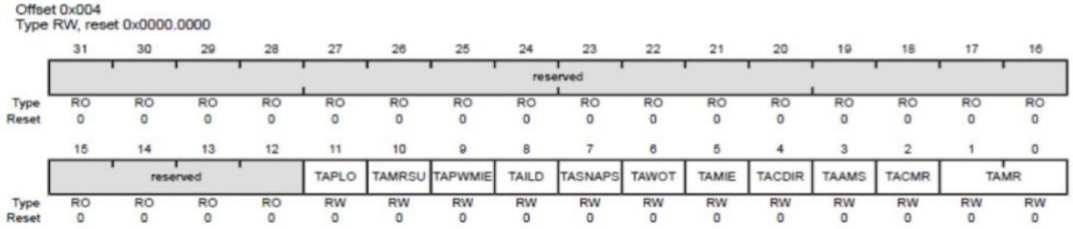
1	One-Shot
2	Periodic
3	Capture

- TACMR [2]: Type of Capture mode

0	Edge-Count
1	Edge-Time

- TACDIR [4]:

0	Count-Down
1	Count-Up



Setting only the 1st bit results in periodic, edge-count, count-down mode for the TIMER0A.

```
TIMER0->TAILR       =LOW; //Set interval load as LOW
```

TAILR value determines the value from which to count down when we are using the periodic mode. When the counter hits 0, interrupt bit is enabled.

```
TIMER0->TAPR        =15; // Divide the clock by 16 to get 1us
```

The actual clock is 16 MHz (1/16 microseconds). Therefore, a prescaler with a value of 16 must be used to have 1 microseconds.

```
TIMER0->IMR             =0x01; //Enable timeout interrupt
```

IMR bit, therefore the timout interrupt is enabled. The interrupt subroutine is executed when the counter hits 0.

Beste Öztop 2375624
Deniz Karakay 2443307 05.12.2022

In the given code where a pulse-width-modulated signal is produced, we made the following changes on LOW and HIGH values. Furthermore, we have written the following section of code into the interrupt subroutine to have the 20% duty cycle. The results are observed via the Oscilloscope and confirmed.

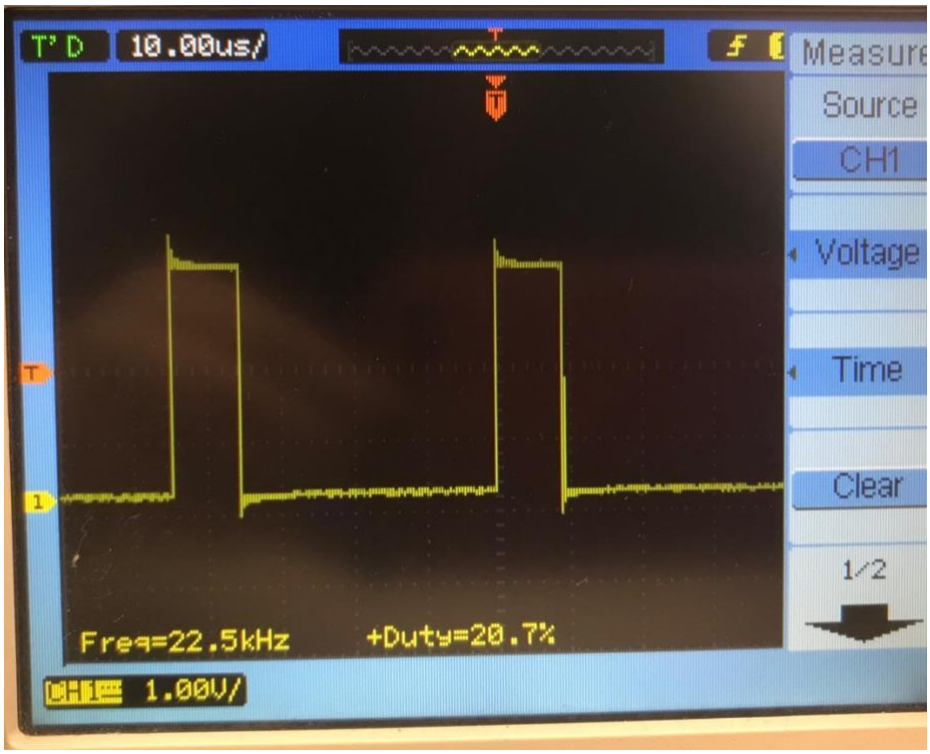


Figure 1: The Oscilloscope result for the 20% duty cycle signal with 20 kHz

```
#define LOW 0x0000006C //108 in decimal
#define HIGH 0x0000001B //27 in decimal

Interrupt subroutine:

void TIMER0A_Handler (void){
       if(TIMER0 -> TAILR == HIGH){
                                  TIMER0 -> TAILR = LOW; //Set interval load as
LOW, defined above
       }else{
                        TIMER0 -> TAILR =HIGH; //Set interval load as LOW,
defined above
       }

       GPIOF -> DATA ^= 4;
       TIMER0 -> ICR |= 0x01;
       return;
}
```

Beste Öztop 2375624
Deniz Karakay 2443307 05.12.2022

Question-2

We have written the following section of code fort his part and obtained the results in Figure 2 and Figure 3.

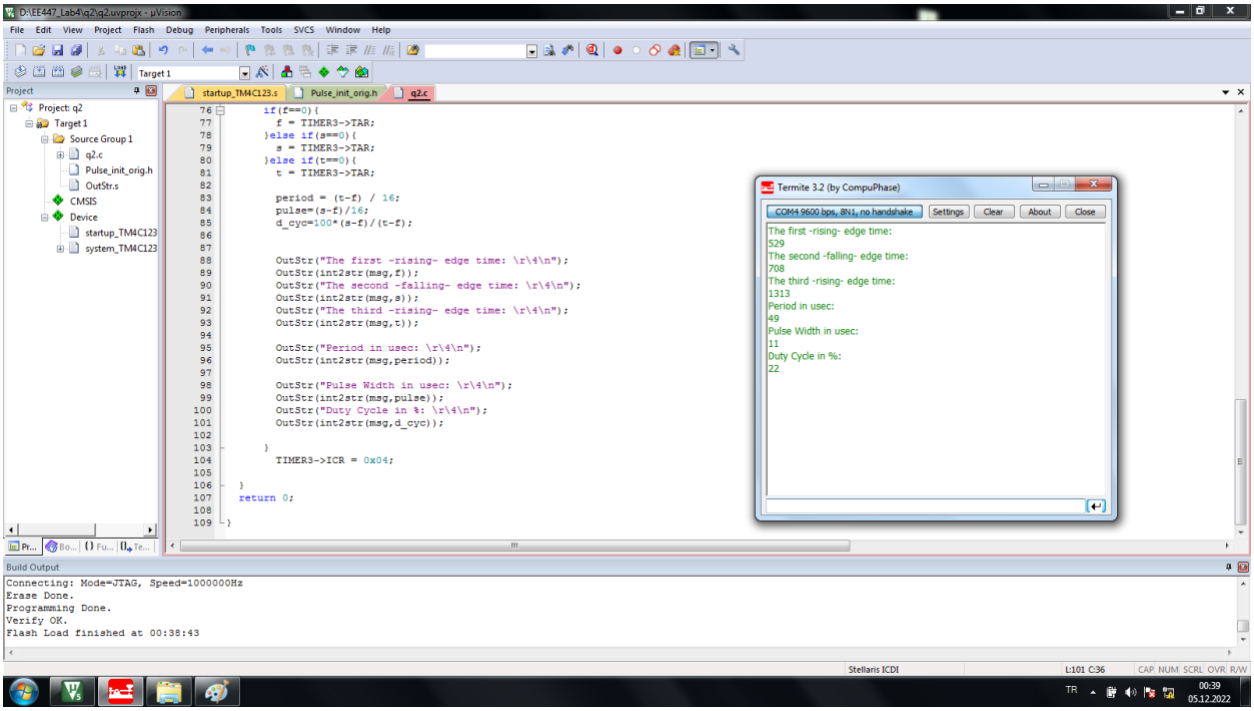


Figure 2: Pulse width, period and duty cycle results from the Termite

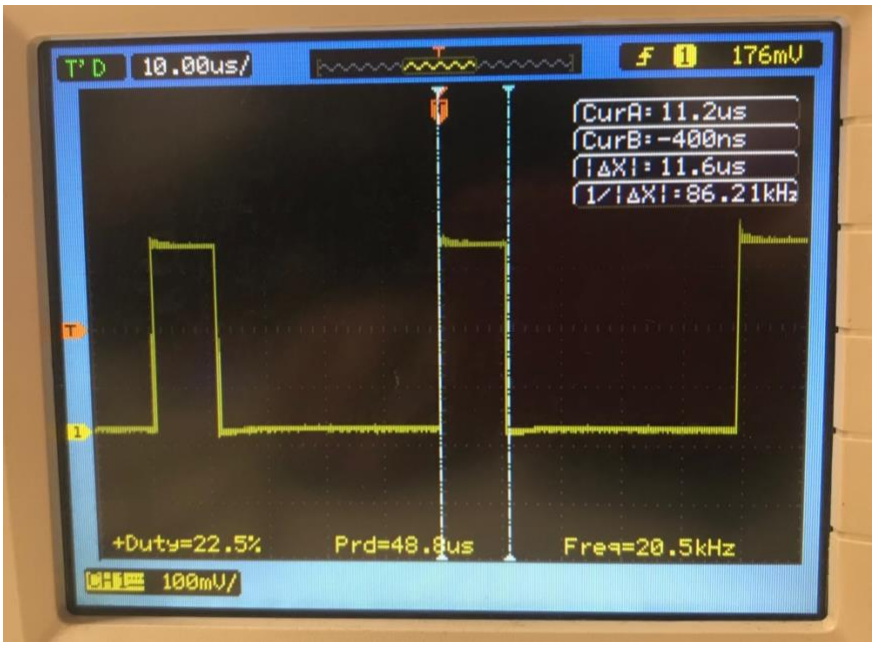


Figure 3: The Oscilloscope result for the 20% duty cycle signal with 20 kHz measured from PB2

Beste Öztop 2375624
Deniz Karakay 2443307 05.12.2022

```
#include "TM4C123GH6PM.h"
#include "Pulse_init_orig.h"

int f=0,s=0,t=0;
int period=0;
int freq=0;
int d_cyc=0;
int pulse=0;
char msg[32];

void initRead(){
       SYSCTL->RCGCGPIO |= 0x02; // turn on bus clock for GPIOB
       SYSCTL->RCGCTIMER |= 0x08; // turn on bus TIMER3

       GPIOB->DIR                    &= ~(1<<2); //set PB2 as input
       GPIOB->DEN                    |= (1<<2); // Enable port digital
  GPIOB->AFSEL        |= (1<<2);  // Enable AFSEL for PB2
       GPIOB->PCTL                   &= ~0x00000F00;  // Setting the
alternate function to 7th one
       GPIOB->PCTL                   |=  0x00000700;  //

       TIMER3->CTL                    =0; //Disable timer during setup
       TIMER3->CFG                    =0x04;  //Set 16 bit mode
       TIMER3->TAMR           =0x17; // set to capture mode, count up
       TIMER3->TAMATCHR = 0xFFFF;
       TIMER3->TAPR          =15; // Divide the clock by 16 to get 1us
       TIMER3->ICR      = 0x1;
       TIMER3->CTL      |= (1<<3); //Both edges
       TIMER3->CTL      |= (1<<2); //Both edges
       //TIMER3->CTL     = 12; //Both edges
       TIMER3->CTL        |= (1<<0); //Enable timer during setup
}

#define        OFFSET         0x10
#define        LENGTH         0x10

extern void OutStr(char*);
char *int2str(char* ms,int number) {
    int i = 0;
    int div = 1;
    int cmp = number;
               int j=0;
               for(j=0; j<10; j++){
                      ms[j]=32;
               }

    while (cmp/10 != 0){
        div = div * 10;
                              cmp /= 10;
               }
               while (div > 0) {
        ms[i++] = number / div + 48;
        number = number % div;
```

```
        div /= 10;
    }

    ms[i]   = '\r';
    ms[i+1] = '\4';
              ms[i+2] = '\n';
    return ms; }

int main(){
       pulse_init();
       initRead();


       TIMER3->CTL |=1;
       while(1){

              while(TIMER3->RIS != 0x04){}

                     if(f==0){
                            f = TIMER3->TAR;
                     }else if(s==0){
                            s = TIMER3->TAR;
                     }else if(t==0){
                            t = TIMER3->TAR;

                            period = (t-f) / 16;
                            pulse=(s-f)/ 16;
                            d_cyc=100*(s-f)/(t-f);


                            OutStr("The first -rising- edge time:
\r\4\n");
                            OutStr(int2str(msg,f));
                            OutStr("The second -falling- edge time:
\r\4\n");
                            OutStr(int2str(msg,s));
                            OutStr("The third -rising- edge time:
\r\4\n");
                            OutStr(int2str(msg,t));

                            OutStr("Period in usec: \r\4\n");
                            OutStr(int2str(msg,period));

                            OutStr("Pulse Width in usec: \r\4\n");
                            OutStr(int2str(msg,pulse));
                            OutStr("Duty Cycle in %: \r\4\n");
                            OutStr(int2str(msg,d_cyc));

                     }
                            TIMER3->ICR = 0x04;

       }
       return 0;
```

}

Question-3

Timer 1 is used for 100 msec delay.
PB3 port generates trigger signal for the sensor.
PB2 port is used to measure the ECHO signal.
The results in Figure 4 are obtained.

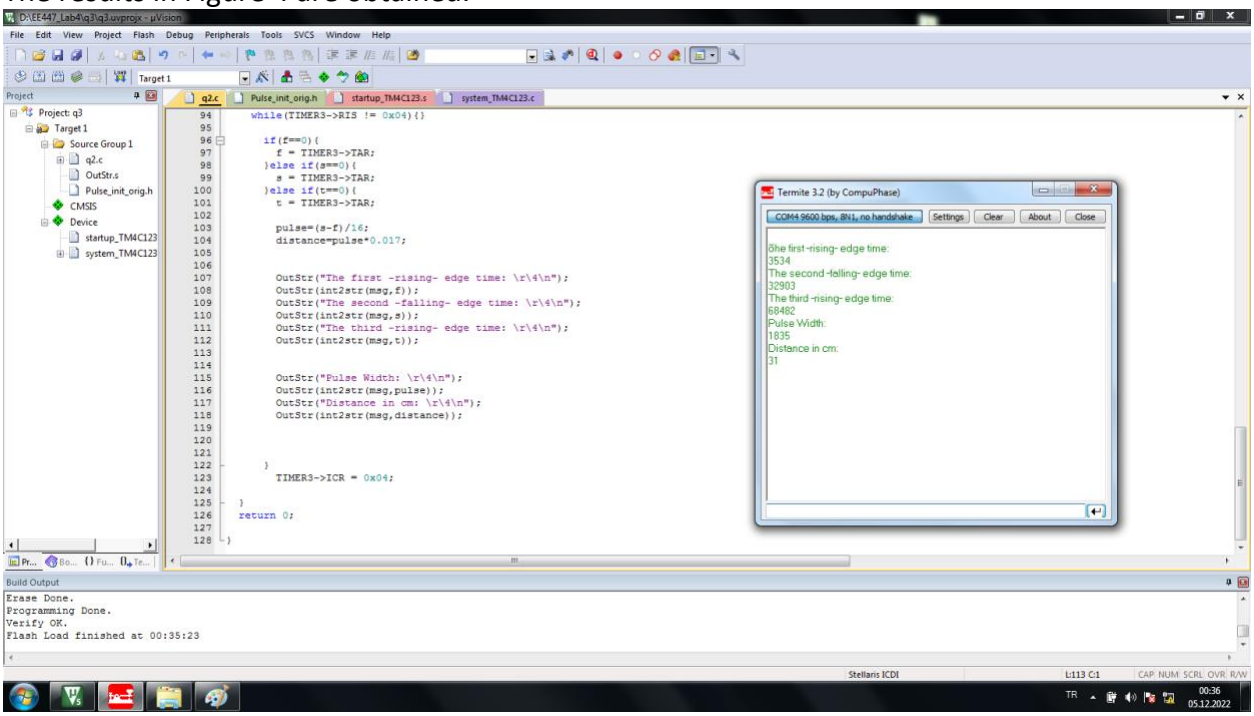


Figure 4: Pulse width result from Termite

The following section of code is written to get the distance from the HC-SR04 Ultrasonic Distance Sensor.

```
#include "TM4C123GH6PM.h"
#include "Pulse_init_orig.h"

int f=0,s=0,t=0;
int pulse=0;
int distance=0;
char msg[32];


void time1_delay(){
       SYSCTL->RCGCTIMER |= 0x02; // turn on bus TIMER1
  TIMER1->CTL                =0; //Disable timer during setup
       TIMER1->CFG                  =0x04;  //Set 16 bit mode
       TIMER1->TAMR          =0x02; // set to periodic, count down
       TIMER1->TAILR         =6400-1; //Set interval load as LOW
       TIMER1->TAPR          =249; // Divide the clock by 16 to get 1us
       TIMER1->ICR                  =0x1; //Ena
       TIMER1->CTL            |=0x01; //Ena
```

```
        while((TIMER1->RIS&0x1)==0){};
        TIMER1->ICR =0x1;
}


void initRead(){
        SYSCTL->RCGCGPIO |= 0x02; // turn on bus clock for GPIOB
        SYSCTL->RCGCTIMER |= 0x08; // turn on bus TIMER1


        GPIOB->DIR                    |= (1<<3);  //set PB3 as OUTPUT
        GPIOB->DEN                    |= (1<<3); // Enable port digital
  GPIOB->AFSEL        &= ~(1<<3);  // Enable AFSEL for PB3

        GPIOB->DIR                    &= ~(1<<2); //set PB2 as input
        GPIOB->DEN                    |= (1<<2); // Enable port digital
  GPIOB->AFSEL        |= (1<<2);  // Enable AFSEL for PB2
        GPIOB->PCTL                   &= ~0x00000F00;  // Setting the
alternate function to 7th one
        GPIOB->PCTL                   |=  0x00000700;  //

        pulse_init();
        time1_delay();
        //GPIOB->PCTL                 &= ~0x00000F00;

        TIMER3->CTL                    =0; //Disable timer during setup
        TIMER3->CFG                    =0x04;  //Set 16 bit mode
        TIMER3->TAMR           =0x17; // set to capture mode, count up
        TIMER3->TAMATCHR = 0xFFFF;
        TIMER3->TAPR          =15; // Divide the clock by 16 to get 1us
        TIMER3->ICR      = 0x1;
        TIMER3->CTL      |= (1<<3); //Both edges
        TIMER3->CTL      |= (1<<2); //Both edges
        //TIMER3->CTL     = 12; //Both edges
        TIMER3->CTL        |= (1<<0); //Enable timer during setup
}

#define        OFFSET         0x10
#define        LENGTH         0x10

extern void OutStr(char*);

char *int2str(char* ms,int number) {
    int i = 0;
    int div = 1;
    int cmp = number;
               int j=0;
               for(j=0; j<10; j++){
                      ms[j]=32;
               }

    while (cmp/10 != 0){
        div = div * 10;
```

```
                                cmp /= 10;
                }
                while (div > 0) {
         ms[i++] = number / div + 48;
         number = number % div;
         div /= 10;
    }


    ms[i]   = '\r';
    ms[i+1] = '\4';
                ms[i+2] = '\n';
    return ms;
}

int main(){
        initRead();

        TIMER3->CTL |=1;
        while(1){


                while(TIMER3->RIS != 0x04){}

                        if(f==0){
                                f = TIMER3->TAR;
                        }else if(s==0){
                                s = TIMER3->TAR;
                        }else if(t==0){
                                t = TIMER3->TAR;

                                pulse=(s-f)/16;
                                distance=pulse*0.017;


                                OutStr("The first -rising- edge time:
\r\4\n");
                                OutStr(int2str(msg,f));
                                OutStr("The second -falling- edge time:
\r\4\n");
                                OutStr(int2str(msg,s));
                                OutStr("The third -rising- edge time:
\r\4\n");
                                OutStr(int2str(msg,t));


                                OutStr("Pulse Width: \r\4\n");
                                OutStr(int2str(msg,pulse));
                                OutStr("Distance in cm: \r\4\n");
                                OutStr(int2str(msg,distance));}
                                TIMER3->ICR = 0x04;

        }
        return 0;}
```