Preliminary Work

Deniz Soysal 2305332

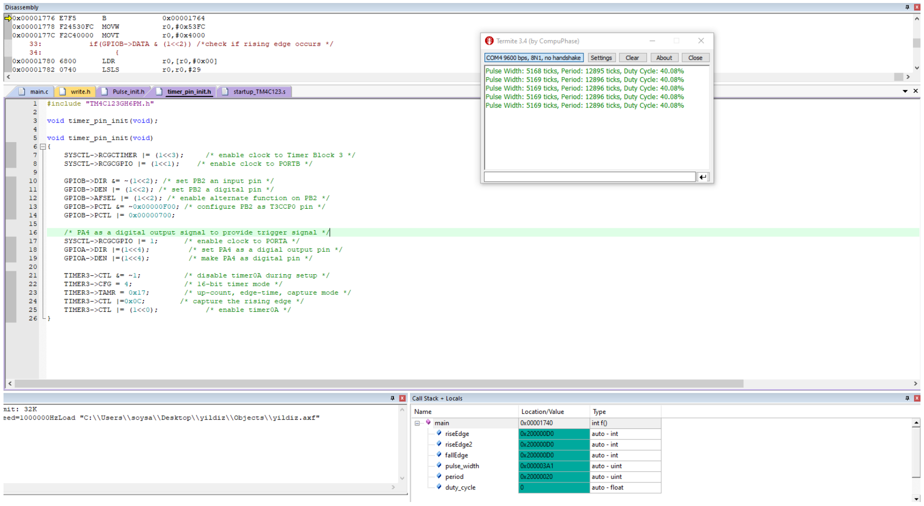
November 2024

1)

value of TAMR is set to 0x2 thus it is in periodic mode.

```
    /*Pulse_init.h file
Function for creating a pulse train using interrupts
Uses Channel 0, and a 1Mhz Timer clock (_TAPR = 15)
Uses Timer0A to create pulse train on PF2
*/

#include "TM4C123GH6PM.h"
void pulse_init(void);
void TIMER0A_Handler (void);

#define LOW  0x000001E0
#define HIGH  0x00000140

void pulse_init(void){
 volatile int *NVIC_EN0 = (volatile int*) 0xE000E100;
 volatile int *NVIC_PRI4 = (volatile int*) 0xE000E410;
 SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF
 __ASM("NOP");
 __ASM("NOP");
 __ASM("NOP");


  GPIOF->DIR   |= 0x04; //set PF2 as output
  GPIOF->AFSEL  &= (0xFFFFFFFB);  // Regular port function
 GPIOF->PCTL   &= 0xFFFFF0FF;  // No alternate function
 GPIOF->AMSEL  =0; //Disable analog
 GPIOF->DEN   |=0x04; // Enable port digital

 GPIOF->DIR       |= 0x08; //set GREEN pin as a digital output pin
  GPIOF->DEN       |= 0x08;  // Enable PF3 pin as a digital pin
 SYSCTL->RCGCTIMER |=0x01; // Start timer0
 __ASM("NOP");
 __ASM("NOP");
 __ASM("NOP");
 TIMER0->CTL  &=0xFFFFFFFE; //Disable timer during setup
 TIMER0->CFG   =0x04;  //Set 16 bit mode
 TIMER0->TAMR  =0x02; // set to periodic, count down
 TIMER0->TAILR  =LOW; //Set interval load as LOW
 TIMER0->TAPR  =15; // Divide the clock by 16 to get 1us
 TIMER0->IMR   =0x01; //Enable timeout intrrupt

 //Timer0A is interrupt 19
 //Interrupt 16-19 are handled by NVIC register PRI4
 //Interrupt 19 is controlled by bits 31:29 of PRI4
 *NVIC_PRI4 &=0x00FFFFFF; //Clear interrupt 19 priority
 *NVIC_PRI4 |=0x40000000; //Set interrupt 19 priority to 2

 //NVIC has to be neabled
 //Interrupts 0-31 are handled by NVIC register EN0
 //Interrupt 19 is controlled by bit 19
 *NVIC_EN0 |=0x00080000;

 //Enable timer
 TIMER0->CTL    |=0x01; // bit0 to enable and bit 1 to stall on debug
 return;
}

void TIMER0A_Handler(void) {
    static int state = 0; // State variable to alternate HIGH/LOW
    TIMER0->ICR = 0x01;  // Clear the timeout interrupt flag

    if (state == 0) {
        // Set to HIGH state
        GPIOF->DATA |= 0x04;      // Turn PF2 ON
        TIMER0->TAILR = HIGH;   // Set HIGH duration (20 µs)
        state = 1;                // Switch state
    } else {
        // Set to LOW state
        GPIOF->DATA &= ~0x04;     // Turn PF2 OFF
        TIMER0->TAILR = LOW;   // Set LOW duration (30 µs)
        state = 0;                // Switch state
    }
}
```

2)



```
0x00001776 E7F5      B             0x00001764
0x00001778 F24530FC  MOVW          r0,#0x53FC
0x0000177C F2C40000  MOVT          r0,#0x4000
    33:          if(GPIOB->DATA & (1<<2)) /*check if rising edge occurs */
    34:             {
0x00001780 6800      LDR           r0,[r0,#0x00]
0x00001782 0740      LSLS          r0,r0,#29
```

```
#include "TM4C123GH6PM.h"

void timer_pin_init(void);

void timer_pin_init(void)
{
    SYSCTL->RCGCTIMER |= (1<<3);     /* enable clock to Timer Block 3 */
    SYSCTL->RCGCGPIO |= (1<<1);    /* enable clock to PORTB */

    GPIOB->DIR &= ~(1<<2); /* set PB2 an input pin */
    GPIOB->DEN |= (1<<2); /* set PB2 a digital pin */
    GPIOB->AFSEL |= (1<<2); /* enable alternate function on PB2 */
    GPIOB->PCTL &= ~0x00000F00; /* configure PB2 as T3CCP0 pin */
    GPIOB->PCTL |= 0x00000700;

    /* PA4 as a digital output signal to provide trigger signal */
    SYSCTL->RCGCGPIO |= 1;      /* enable clock to PORTA */
    GPIOA->DIR |=(1<<4);         /* set PA4 as a digial output pin */
    GPIOA->DEN |=(1<<4);         /* make PA4 as digital pin */

    TIMER3->CTL &= ~1;          /* disable timer0A during setup */
    TIMER3->CFG = 4;            /* 16-bit timer mode */
    TIMER3->TAMR = 0x17;        /* up-count, edge-time, capture mode */
    TIMER3->CTL |=0x0C;        /* capture the rising edge */
    TIMER3->CTL |= (1<<0);           /* enable timer0A */
}
```

COM4 9600 bps, 8N1, no handshake

```
Pulse Width: 5168 ticks, Period: 12895 ticks, Duty Cycle: 40.08%
Pulse Width: 5169 ticks, Period: 12896 ticks, Duty Cycle: 40.08%
Pulse Width: 5169 ticks, Period: 12896 ticks, Duty Cycle: 40.08%
Pulse Width: 5169 ticks, Period: 12896 ticks, Duty Cycle: 40.08%
Pulse Width: 5169 ticks, Period: 12896 ticks, Duty Cycle: 40.08%
```

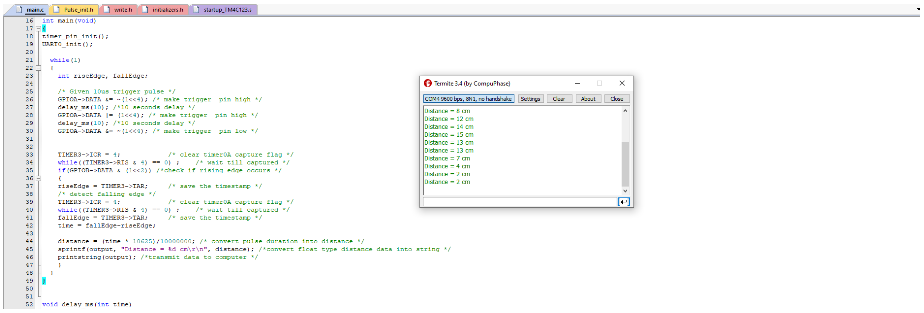
```
mit: 32K
eed=1000000HzLoad "C:\\Users\\soysa\\Desktop\\yildiz\\Objects\\yildiz.axf"
```

Name	Location/Value	Type
main	0x00001740	int f()
riseEdge	0x200000D0	auto - int
riseEdge2	0x200000D0	auto - int
fallEdge	0x200000D0	auto - int
pulse_width	0x000003A1	auto - uint
period	0x20000020	auto - uint
duty_cycle	0	auto - float

Figure 1: termite screen after running the code for a few seconds.

```
#include "TM4C123GH6PM.h"
#include "write.h"
#include "Pulse_init.h"
#include "timer_pin_init.h"
#include <stdio.h>

/*Function prototype for Timer0A and UART module initialization */

void delay(unsigned long counter);

/* global variables to store and display distance in cm */
uint32_t time; /*stores pulse on time */
uint32_t distance; /* stores measured distance value */
char output[100];  /* string format of distance value */

/* main code to take distance measurement and send data to UART terminal */
int main(void)
{
timer_pin_init();
UART0_init();
pulse_init();

 while(1)
 {
    int riseEdge;
    int riseEdge2;
  int fallEdge;


    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0);    /* wait till captured */
   if(GPIOB->DATA & (1<<2)) /*check if rising edge occurs */
  {
    riseEdge = TIMER3->TAR;     /* save the timestamp */
  /* detect falling edge */
    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0);    /* wait till captured */
    fallEdge = TIMER3->TAR;     /* save the timestamp */
  TIMER3->ICR = 4;
  while((TIMER3->RIS & 4) == 0);    /* wait till captured */
    riseEdge2 = TIMER3->TAR;     /* save the timestamp */
  time = fallEdge-riseEdge;
  }


    uint32_t pulse_width = fallEdge - riseEdge;
    uint32_t period = riseEdge2 - riseEdge;
    float duty_cycle = ((float)pulse_width / (float)period) * 100.0;

    // Prepare the output message
    sprintf(output, "Pulse Width: %u ticks, Period: %u ticks, Duty Cycle: %.2f%%\r\n", pulse

    printstring(output);
    delay(2000);
 }
}


void delay(unsigned long counter)
{
 unsigned long i = 0;

 for(i=0; i< counter*1000; i++);
}
```

3)



```
int main(void)
{
timer_pin_init();
UART0_init();

  while(1)
  {
    int riseEdge, fallEdge;

    /* Given 10us trigger pulse */
    GPIOA->DATA &= ~(1<<4); /* make trigger  pin high */
    delay_ms(10); /*10 seconds delay */
    GPIOA->DATA |= (1<<4); /* make trigger  pin high */
    delay_ms(10); /*10 seconds delay */
    GPIOA->DATA &= ~(1<<4); /* make trigger  pin low */


    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0) ;    /* wait till captured */
    if(GPIOB->DATA & (1<<2)) /*check if rising edge occurs */
    {
    riseEdge = TIMER3->TAR;     /* save the timestamp */
    /* detect falling edge */
    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0) ;    /* wait till captured */
    fallEdge = TIMER3->TAR;     /* save the timestamp */
    time = fallEdge-riseEdge;

    distance = (time * 10625)/10000000; /* convert pulse duration into distance */
    sprintf(output, "Distance = %d cm\r\n", distance); /*convert float type distance data into string */
    printstring(output); /*transmit data to computer */
    }
  }
}


void delay_ms(int time)
```

```
Distance = 8 cm
Distance = 12 cm
Distance = 14 cm
Distance = 15 cm
Distance = 13 cm
Distance = 13 cm
Distance = 7 cm
Distance = 4 cm
Distance = 2 cm
Distance = 2 cm
```

Figure 2: termite screen after running the code for a few seconds[2]

```
#include "TM4C123GH6PM.h"
#include "write.h"
#include "initializers.h"
#include <stdio.h>

/*Function prototype for Timer0A and UART module initialization */

void delay_ms(int time);

/* global variables to store and display distance in cm */
uint32_t time; /*stores pulse on time */
uint32_t distance; /* stores measured distance value */
char output[100];  /* string format of distance value */

/* main code to take distance measurement and send data to UART terminal */
int main(void)
{
timer_pin_init();
UART0_init();

 while(1)
 {
  int riseEdge, fallEdge;

   /* Given 10us trigger pulse */
   GPIOA->DATA &= ~(1<<4); /* make trigger  pin high */
   delay_ms(10); /*10 seconds delay */
   GPIOA->DATA |= (1<<4); /* make trigger  pin high */
   delay_ms(10); /*10 seconds delay */
   GPIOA->DATA &= ~(1<<4); /* make trigger  pin low */
```

```
    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0) ;    /* wait till captured */
   if(GPIOB->DATA & (1<<2)) /*check if rising edge occurs */
  {
    riseEdge = TIMER3->TAR;     /* save the timestamp */
  /* detect falling edge */
    TIMER3->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER3->RIS & 4) == 0) ;    /* wait till captured */
    fallEdge = TIMER3->TAR;     /* save the timestamp */
  time = fallEdge-riseEdge;

  distance = (time * 10625)/10000000; /* convert pulse duration into distance */
    sprintf(output, "Distance = %d cm\r\n", distance); /*convert float type distance data in
    printstring(output); /*transmit data to computer */
  }
 }
}


void delay_ms(int time)
{
    int i;
    SYSCTL->RCGCTIMER |= 2;     /* enable clock to Timer Block 1 */
    TIMER1->CTL = 0;            /* disable Timer before initialization */
    TIMER1->CFG = 0x04;         /* 16-bit option */
    TIMER1->TAMR = 0x02;        /* periodic mode and down-counter */
    TIMER1->TAILR = 16;  /* TimerA interval load value reg */
    TIMER1->ICR = 0x1;          /* clear the TimerA timeout flag */
    TIMER1->CTL |= 0x01;        /* enable Timer A after initialization */

    for(i = 0; i < time; i++)
    {
        while ((TIMER1->RIS & 0x1) == 0) ;      /* wait for TimerA timeout flag */
        TIMER1->ICR = 0x1;      /* clear the TimerA timeout flag */
    }
}
```

HELPER FUNCTIONS

write.h

```
    #include "TM4C123GH6PM.h"

void UART0_init(void);
void UART0_Transmitter(unsigned char data);
void printstring(char *str);

void UART0_init(void)
{
    SYSCTL->RCGCUART |= 0x01;  /* Enable clock to UART0 */
    SYSCTL->RCGCGPIO |= 0x01;  /* Enable clock to PORTA for PA0/Rx and PA1/Tx */

    UART0->CTL = 0;            /* Disable UART0 module during configuration */
    UART0->IBRD = 104;         /* Integer part for 9600 baud rate */
    UART0->FBRD = 11;          /* Fractional part for 9600 baud rate */
    UART0->CC = 0;             /* Use system clock */
    UART0->LCRH = 0x60;        /* 8-bit data, no parity, 1 stop bit */
    UART0->CTL = 0x301;        /* Enable UART0 module, Rx and Tx */

    GPIOA->DEN = 0x03;         /* Enable digital functions for PA0 and PA1 */
    GPIOA->AFSEL = 0x03;       /* Enable alternate functions for PA0 and PA1 */
    GPIOA->PCTL = 0x11;        /* Configure PA0 and PA1 for UART */
}

void UART0_Transmitter(unsigned char data)
{
    while ((UART0->FR & (1 << 5)) != 0); /* Wait until Tx buffer is not full */
    UART0->DR = data;                    /* Transmit the data */
}

void printstring(char *str)
{
    while (*str)
    {
        UART0_Transmitter(*(str++));    /* Send characters one by one */
    }
}
```

timer_pin_init.h

```
#include "TM4C123GH6PM.h"

void timer_pin_init(void);

void timer_pin_init(void)
{
    SYSCTL->RCGCTIMER |= (1<<3);     /* enable clock to Timer Block 3 */
    SYSCTL->RCGCGPIO |= (1<<1);    /* enable clock to PORTB */

    GPIOB->DIR &= ~(1<<2); /* set PB2 an input pin */
    GPIOB->DEN |= (1<<2); /* set PB2 a digital pin */
  GPIOB->AFSEL |= (1<<2); /* enable alternate function on PB2 */
  GPIOB->PCTL &= ~0x00000F00; /* configure PB2 as T3CCP0 pin */
  GPIOB->PCTL |= 0x00000700;

   /* PA4 as a digital output signal to provide trigger signal */
   SYSCTL->RCGCGPIO |= 1;      /* enable clock to PORTA */
   GPIOA->DIR |=(1<<4);         /* set PA4 as a digial output pin */
   GPIOA->DEN |=(1<<4);         /* make PA4 as digital pin */

    TIMER3->CTL &= ~1;          /* disable timer0A during setup */
    TIMER3->CFG = 4;            /* 16-bit timer mode */
    TIMER3->TAMR = 0x17;        /* up-count, edge-time, capture mode */
    TIMER3->CTL |=0x0C;        /* capture the rising edge */
    TIMER3->CTL |= (1<<0);           /* enable timer0A */
}
```