Q1

C:\Users\alper\Desktop\EE\EE4-1\EE 447\Lab4\Example1\Pulse_init.c

```
1   /*Pulse_init.h file
2   Function for creating a pulse train using interrupts
3   Uses Channel 0, and a 1Mhz Timer clock (_TAPR = 15)
4   Uses Timer0A to create pulse train on PF2
5   */
6
7   #include "TM4C123GH6PM.h"
8   void pulse_init(void);
9   void TIMER0A_Handler (void);
10  void detect_init (void);
11
12  #define LOW   60
13  #define HIGH  15
14
15  void pulse init(void){
16    volatile int *NVIC EN0 = (volatile int*) 0xE000E100;
17    volatile int *NVIC PRI4 = (volatile int*) 0xE000E410;
18    SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF
19      ASM("NOP");
20      ASM("NOP");
21      ASM("NOP");
22
23    GPIOF->DIR      |= 0x04; //set PF2 as output
24    GPIOF->AFSEL    &= (0xFFFFFFFB);  // Regular port function
25    GPIOF->PCTL     &= 0xFFFFF0FF;  // No alternate function
26    GPIOF->AMSEL    =0; //Disable analog
27    GPIOF->DEN      |=0x04; // Enable port digital
28
29    //GPIOF->DIR       |= 0x08; //set GREEN pin as a digital output pin
30    //GPIOF->DEN       |= 0x08;  // Enable PF3 pin as a digital pin
31
32    SYSCTL->RCGCTIMER |=0x01; // Start timer0
33    __ASM("NOP");
34    __ASM("NOP");
35    __ASM("NOP");
36    TIMER0->CTL     &=0xFFFFFFFE; //Disable timer during setup
37    TIMER0->CFG     =0x04;  //Set 16 bit mode
38    TIMER0->TAMR    =0x02; // set to periodic, count down
39    TIMER0->TAILR   =LOW; //Set interval load as LOW
40    TIMER0->TAPR    =15; // Divide the clock by 16 to get 1us
41    TIMER0->IMR     =0x01; //Enable timeout intrrupt
42
43    //Timer0A is interrupt 19
44    //Interrupt 16-19 are handled by NVIC register PRI4
45    //Interrupt 19 is controlled by bits 31:29 of PRI4
46    *NVIC PRI4 &=0x00FFFFFF; //Clear interrupt 19 priority
47    *NVIC_PRI4 |=0x40000000; //Set interrupt 19 priority to 2
48
49    //NVIC has to be neabled
50    //Interrupts 0-31 are handled by NVIC register EN0
51    //Interrupt 19 is controlled by bit 19
52    *NVIC_EN0 |=0x00080000;
53
54    //Enable timer
55    TIMER0->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
56    return;
57  }
58
59  void TIMER0A_Handler (void){
60    GPIOF->DATA  ^= 4;  //toggle PF2 pin
61
62    if(TIMER0->TAILR==LOW)
63      TIMER0->TAILR=HIGH;
64    else
65      TIMER0->TAILR=LOW;
66    TIMER0->ICR |=0x01;
67    return;
68  }
69  int main()
70  {
71    pulse init();
72    while(1){}
73
74  }
```

C:\Users\alper\Desktop\EE\EE4-1\EE 447\Lab4\Example1\Pulse_init.c

```
1   /*Pulse_init.h file
2   Function for creating a pulse train using interrupts
3   Uses Channel 0, and a 1Mhz Timer clock (_TAPR = 15)
4   Uses Timer0A to create pulse train on PF2
5   */
6
7   #include "TM4C123GH6PM.h"
8   void pulse_init(void);
9   void TIMER0A_Handler (void);
10  void detect_init (void);
11
12  #define LOW   60
13  #define HIGH  15
14
15  void pulse init(void){
16    volatile int *NVIC EN0 = (volatile int*) 0xE000E100;
17    volatile int *NVIC PRI4 = (volatile int*) 0xE000E410;
18    SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF
19      ASM("NOP");
20      ASM("NOP");
21      ASM("NOP");
22
23    GPIOF->DIR      |= 0x04; //set PF2 as output
24    GPIOF->AFSEL    &= (0xFFFFFFFB);  // Regular port function
25    GPIOF->PCTL     &= 0xFFFF0FF;  // No alternate function
26    GPIOF->AMSEL    =0; //Disable analog
27    GPIOF->DEN      |=0x04; // Enable port digital
28
29    //GPIOF->DIR       |= 0x08; //set GREEN pin as a digital output pin
30    //GPIOF->DEN       |= 0x08;  // Enable PF3 pin as a digital pin
31
32    SYSCTL->RCGCTIMER |=0x01; // Start timer0
33    __ASM("NOP");
34    __ASM("NOP");
35    __ASM("NOP");
36    TIMER0->CTL     &=0xFFFFFFFE; //Disable timer during setup
37    TIMER0->CFG     =0x04;  //Set 16 bit mode
38    TIMER0->TAMR    =0x02; // set to periodic, count down
39    TIMER0->TAILR   =LOW; //Set interval load as LOW
40    TIMER0->TAPR    =15; // Divide the clock by 16 to get 1us
41    TIMER0->IMR     =0x01; //Enable timeout intrrupt
42
43    //Timer0A is interrupt 19
44    //Interrupt 16-19 are handled by NVIC register PRI4
45    //Interrupt 19 is controlled by bits 31:29 of PRI4
46    *NVIC PRI4 &=0x00FFFFFF; //Clear interrupt 19 priority
47    *NVIC_PRI4 |=0x40000000; //Set interrupt 19 priority to 2
48
49    //NVIC has to be neabled
50    //Interrupts 0-31 are handled by NVIC register EN0
51    //Interrupt 19 is controlled by bit 19
52    *NVIC_EN0 |=0x00080000;
53
54    //Enable timer
55    TIMER0->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
56    return;
57  }
58
59  void TIMER0A_Handler (void){
60    GPIOF->DATA  ^= 4;  //toggle PF2 pin
61
62    if(TIMER0->TAILR==LOW)
63      TIMER0->TAILR=HIGH;
64    else
65      TIMER0->TAILR=LOW;
66    TIMER0->ICR |=0x01;
67    return;
68  }
69  int main()
70  {
71    pulse init();
72    while(1){}
73
74  }
```

Q2

C:\Users\alper\Desktop\EE\EE4-1\EE 447\Lab4\Q2\q2.c

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

extern void OutStr(char*);
void print_number(int number);
int x,edge=0;
int edge1,edge2,edge3;
int period,pulse_width,duty_cycle,new_number;
char msg1[100],msg2[100];

void print_number(int number)
{
    int i=0,j=0;
    while(number){
      new_number=number/10;
      msg1[i]=number-(new_number*10)+48;
      number=new_number;
      i++;
    }
    for(i=i-1;i>=0;i--){
    msg2[j]=msg1[i];
    j++;
    }

    msg2[j]='\r';
    msg2[j+1]='\4';
    OutStr(msg2);
}
  int main(){
    pulse_init();
    detect_init();
    while(1){

      x=TIMER1->RIS&4;  //Seperating CAERIS bit
      if(x==4){
        if(edge==0)
        {
          edge1=TIMER1->TAR;  //Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04;//Clear ICR
          continue;
        }
        else if(edge==1)
        {
          edge2=TIMER1->TAR;  //Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04;//Clear ICR
          continue;
        }
        else if(edge==2)
        {
          edge3=TIMER1->TAR;//Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04; //Clear ICR
          continue;
        }
        else
        {
          period=edge1-edge3; //PERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns]
          pulse_width=edge1-edge2;//PULSE WIDTH (FIRST EDGE- SECOND EDGE) [IN CYCLE UNIT, NOT IN ns]
          duty_cycle=(pulse_width*100)/period; //Pulse Width*100 / PERIOD = DUTY CYCLE
          OutStr("Duty Cycle (%): \r\4");
          print_number(duty_cycle);
          OutStr("Pulse Width (us): \r\4");
          print_number(pulse_width/16);
          OutStr("Period (us): \r\4");
          print_number(period/16);
        }
            while(1){}

        }

      }

}
```

Duty Cycle (%):
22
Pulse Width (us):
18
Period (us):
81

Q3

C:\Users\alper\Desktop\EE\EE4-1\EE 447\Lab4\Q3\q2.c

```c
#include "Pulse_init.h"
#include "TM4C123GH6PM.h"
#include <stdio.h>

extern void OutStr(char*);
void print_number(int number);
int x,edge=0;
int edge1,edge2,edge3;
int period,pulse_width,duty_cycle,new_number;
char msg1[100],msg2[100];

void print_number(int number)
{
    int i=0,j=0;
    while(number){
      new_number=number/10;
      msg1[i]=number-(new_number*10)+48;
      number=new_number;
      i++;
    }
    for(i=i-1;i>=0;i--){
    msg2[j]=msg1[i];
    j++;
    }

    msg2[j]='\r';
    msg2[j+1]='\4';
    OutStr(msg2);
}
  int main(){
    pulse_init();
    detect_init();
    while(1){

      x=TIMER1->RIS&4;  //Seperating CAERIS bit
      if(x==4){
        if(edge==0)
        {
          edge1=TIMER1->TAR;  //Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04;//Clear ICR
          continue;
        }
        else if(edge==1)
        {

          edge2=TIMER1->TAR;  //Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04;//Clear ICR
          continue;
        }
        else if(edge==2)
        {
          edge3=TIMER1->TAR;//Get timer register value
          edge=edge+1;
          TIMER1->ICR |=0x04; //Clear ICR
          continue;
        }
        else
        {
          period=edge1-edge3; //PERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns]
          pulse_width=edge1-edge2;//PULSE WIDTH (FIRST EDGE- SECOND EDGE) [IN CYCLE UNIT, NOT IN ns]
          duty_cycle=(pulse_width*100)/period; //Pulse Width*100 / PERIOD = DUTY CYCLE
          /*OutStr("Duty Cycle (%): \r\4");
          print_number(duty_cycle);*/
          OutStr("Pulse Width (us):\4");
          print_number(pulse_width*0.34/32);
          /*OutStr("Period (us): \r\4");
          print_number(period/16);*/
        }
            edge=0;

        }

      }

}
```

```
Pulse Width (us):(/0/
Pulse Width (us):1721
Pulse Width (us):1744
Pulse Width (us):2490
Pulse Width (us):((+(
Pulse Width (us):1739
Pulse Width (us):2458
Pulse Width (us):1717
Pulse Width (us):2537
Pulse Width (us):1751
Pulse Width (us):2445
Pulse Width (us):(0('
Pulse Width (us):1748
Pulse Width (us):2466
Pulse Width (us):1749
Pulse Width (us):1731
Pulse Width (us):2448
Pulse Width (us)
```