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

```
/*Pulse init.h file
       Function for creating a pulse train using interrupts
Uses Channel 0, and a 1Mhs Timer clock (_TAPR = 15)
Uses Timer0A to create pulse train on PF2
      #include "TM4C123GH6PM.h"
       void pulse_init(void);
 9
      void TIMEROA_Handler (void);
10
      void detect_init (void);
      #define LOW 60
#define HIGH 15
12
13
14
15
       void pulse init(void) {
         volatile int *NVIC ENO = (volatile int*) 0xE000E100;
volatile int *NVIC PRI4 = (volatile int*) 0xE000E410;
SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF
16
          ASM("NOP");
ASM("NOP");
19
20
           ASM("NOP");
22
                                |= 0x04; //set PF2 as output

6= (0xFFFFFFFB); // Regular port function

6= 0xFFFFF0FF; // No alternate function
23
         GPIOF->DIR
         GPIOF->AFSEL
24
          GPIOF->PCTL
26
         GPIOF->AMSEL
                                 =0; //Disable analog
         GPIOF->DEN
                                 |=0x04; // Enable port digital
28
                                     |= 0x08; //set GREEN pin as a digital output pin
|= 0x08; // Enable PF3 pin as a digital pin
29
         //GPIOF->DIR
         //GPIOF->DEN
30
32
         SYSCTL->RCGCTIMER |=0x01; // Start timer0
         __ASM("NOP");
33
         __ASM("NOP");
34
         ASM("NOP");
TIMERO->CTL
35
                                 &=0xFFFFFFFF; //Disable timer during setup
=0x04; //Set 16 bit mode
=0x02; // set to periodic, count down
26
          TIMERO->CFG
37
          TIMERO->TAMR
                                =LOW; //Set interval load as LOW
=15; // Divide the clock by 16 to get lus
39
         TIMERO->TAILR
          TIMERO->TAPR
40
         TIMERO->IMR
                                 =0x01; //Enable timeout intrrupt
41
42
          //TimerOA is interrupt 19
         //Interrupt 16-19 are handled by NVIC register PRI4
//Interrupt 19 is controlled by bits 31:29 of PRI4
44
         *NVIC PRI4 &=0x00FFFFFF; //Clear interrupt 19 priority
*NVIC_FRI4 |=0x40000000; //Set interrupt 19 priority to 2
46
47
48
49
          //NVIC has to be neabled
         //Interrupts 0-31 are handled by NVIC register EN0 //Interrupt 19 is controlled by bit 19 *NVIC_EN0 |=0x00080000;
50
51
53
          //Enable timer
54
         TIMERO->CTL
                                  |=0x03; // bit0 to enable and bit 1 to stall on debug
56
         return;
57
      void TIMEROA_Handler (void) {
   GPIOF->DATA ^= 4; //toggle PF2 pin
59
60
61
62
         if(TIMERO->TAILR==LOW)
            TIMERO->TAILR=HIGH;
62
64
         else
            TIMERO->TAILR=LOW;
65
66
         TIMERO->ICR |=0x01;
67
         return:
68
70
71
         pulse init():
          while(1){}
73
74
```

```
/*Pulse_init.h file
      Function for creating a pulse train using interrupts
     Uses TimerOA to create pulse train on PF2 */
 3
      #include "TM4C123GH6PM.h"
     void pulse_init(void);
      void TIMEROA_Handler (void);
     void detect_init (void);
10
11
     #define LOW 60
#define HIGH 15
12
13
14
15
     void pulse init(void) {
       volatile int *NVIC ENO = (volatile int*) 0xE000E100;
volatile int *NVIC PRI4 = (volatile int*) 0xE000E410;
SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF
16
17
18
        ASM("NOP");
19
          ASM("NOP");
20
         ASM("NOP");
21
22
       GPIOF->DIR
                         |= 0x04; //set PF2 as output

6= (0xFFFFFFB); // Regular port function

6= 0xFFFFF0FF; // No alternate function
23
24
        GPIOF->AFSEL
        GPIOF->PCTL
        GPIOF->AMSEL
                           =0; //Disable analog
26
        GPIOF->DEN
                           |=0x04; // Enable port digital
28
                              |= 0x08; //set GREEN pin as a digital output pin
|= 0x08; // Enable PF3 pin as a digital pin
        //GPIOF->DIR
29
       //GPIOF->DEN
30
31
       SYSCTL->RCGCTIMER |=0x01; // Start timer0
32
       __ASM("NOP");
33
        ASM("NOP");
34
        ASM("NOP");
TIMERO->CTL
35
                           &=OxFFFFFFFE; //Disable timer during setup
36
                           =0x04; //Set 16 bit mode
=0x02; // set to periodic, count down
        TIMERO->CFG
        TIMERO->TAMR
38
39
        TIMERO->TAILR
                          =LOW; //Set interval load as LOW
40
        TIMERO->TAPR =15; // Divide the clock by 16 to get lus
41
        TIMERO->IMR
                          =0x01; //Enable timeout intrrupt
42
43
        //TimerOA is interrupt 19
44
        //Interrupt 16-19 are handled by NVIC register PRI4
45
        //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
46
47
48
        //{\tt NVIC} has to be neabled
49
        //Interrupts 0-31 are handled by NVIC register ENO
50
        //Interrupt 19 is controlled by bit 19
51
        *NVIC ENO |=0x00080000;
52
        //Enable timer
        TIMERO->CTL
                           |=0x03; // bit0 to enable and bit 1 to stall on debug
55
        return;
56
     1
57
58
     void TIMEROA Handler (void) {
   GPIOF->DATA ^= 4; //toggle PF2 pin
59
60
61
       if (TIMERO->TAILR==LOW)
62
63
          TIMERO->TAILR=HIGH;
        else
64
          TIMERO->TAILR=LOW;
65
        TIMER0->ICR |=0x01;
67
        return:
68
69
70
71
        pulse init();
72
        while(1){}
73
74
     1
```

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 m, edge=0; int edgel, edge2, edge3; int period, pulse width, duty cycle, new number; 10 char msgl[100],msg2[100]; 11 void print_number(int number) 13 14 int i=0,j=0; while(number) { new number=number/10; 15 16 msgl[i]=number-(new number*10)±48; 17 18 number=new number; 20 for(i=i-1;i>=0;i--){ 22 23 msg2[j]=msg1[i]; j++; 24 25 msg2[j]='\r'; msg2[j+1]='\4'; Out3tr(msg2); 26 29 int main(){ 30 pulse_init(); 31 32 detect init(); while (1) { 34 x=TIMER1->RIS&4; //Seperating CAERIS bit 35 36 if(x==4){ if(edge==0) 37 edgel=TIMER1->TAR; //Get timer register value 29 edge=edge+1; TIMER1->ICR |=0x04;//Clear ICR 40 42 continue: 44 45 else if(edge==1) edge2=TIMER1->TAR; //Get timer register value 47 edge=edge+1; TIMER1->ICR |=0x04;//Clear ICR 49 continue; 51 else if(edge==2) 52 edge3=TIMER1->TAR;//Get timer register value edge=edge+1; TIMER1->ICR |=0x04; //Clear ICR 54 55 56 continue; 58 else period=edgel-edge3; //PERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns] pulse_width=edgel-edge2;//PULSE WIDTH (FIRST EDGE- SPOOND EDGE) (YV COLOR EDGE) 59 period=edge1-edge2; //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"); nrint_number(pulse_width/16); 61 62 63 64 66 print_number(period/16); } 68 69 70 71 while(1){} } 72 } 76 }

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

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

```
#include "Pulse_init.h"
#include "TM4C123GH6PM.h"
       #include <stdio.h>
       extern void OutStr(char*);
       void print_number(int number);
       int m, edge=0;
       int edge1, edge2, edge2;
      int period, pulse width, duty cycle, new number; char magl[100], mag2[100];
10
11
12
       void print_number(int number)
13
14
15
            int i=0,j=0;
            while (number) {
16
              new number=number/10;
17
18
               msgl[i]=number-(new number*10)±48;
               number=new number;
19
               i++;
20
            for(i=i-1;i>=0;i--){
22
23
            msg2[j]=msg1[i];
24
25
           msg2[j]='\r';
msg2[j+1]='\4';
OutStr(msg2);
26
28
30
         int main(){
           pulse_init();
detect_init();
31
32
33
            while (1) {
35
                x=TIMER1->RIS&4; //Seperating CAERIS bit
36
               if (n==4) {
                  if(edge==0)
38
                     edgel=TIMER1->TAR; //Get timer register value
39
                     edge=edge+1;
TIMER1->ICR |=0x04;//Clear ICR
41
42
                     continue:
43
                  else if(edge==1)
44
45
46
                    edge2=TIMER1->TAR; //Get timer register value
47
                     edge=edge+1;
TIMER1->ICR |=0x04;//Clear ICR
49
50
                     continue:
51
52
                  else if(edge==2)
53
54
                      edge3=TIMER1->TAR;//Get timer register value
55
                     edge=edge+1;
TIMER1->ICR |=0x04; //Clear ICR
56
57
                     continue;
59
60
61
                    period=edgel-edge3; //PERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns]
                   period=edged=edge3; //FERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns]
pulse_width=edge2;//FULSE WIDTH (FIRST EDGE- SECOND EDGE) [IN CYCLE UNIT, NOT IN ns]
duty_cycle=(pulse_width*100)/period; //Fulse 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(pulse_width*0.34/32);
62
63
65
66
67
68
                print number(period/16);*/
}
69
70
71
72
73
74
75
76
                        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)