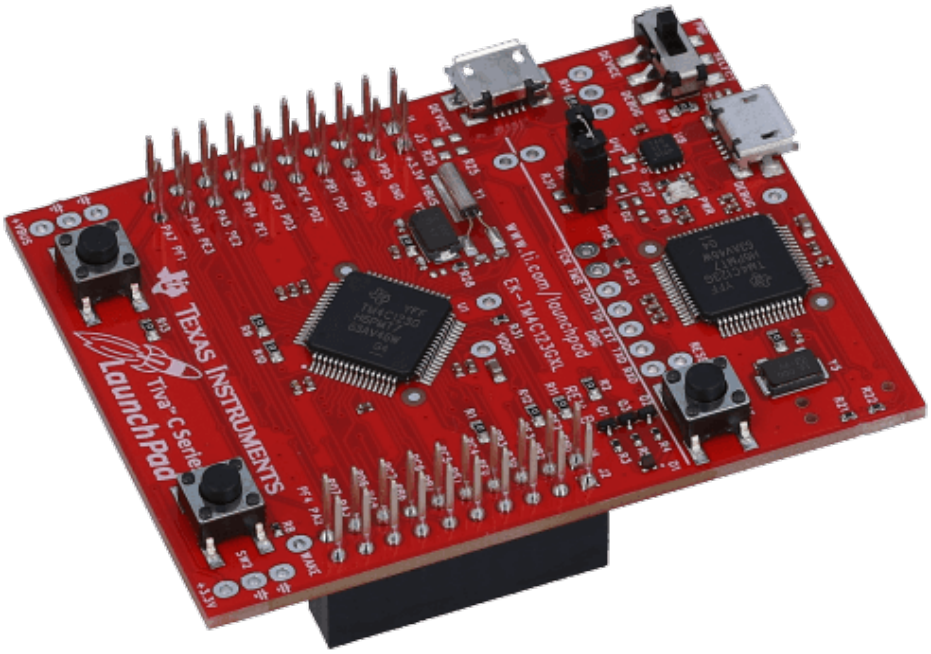
ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

EE447- Introduction to Microprocessors Laboratory With Assembly Programming

(Preliminary Work 4)

EXPERIMENTAL WORK NO: 4

1st Group Member: Uğur SAMANCI - 2398915
2nd Group Member: Barış GÜZEL - 2304764



Question 1) PWM CONSTRUCTION

- In this part of the preliminary work, we created a duty cycle with %20 while designing it we changed the value of the predefined high and low to 0x0A and 0x28 respectively. Because 10/(10+40)=0.2 duty cycle also in this part we changed only TIMER0A handler part of the pulse_init function it can be seen in figure 3. Also one can see designed pwm cycle on figure 1. And main part of the question 1 is available at figure2.

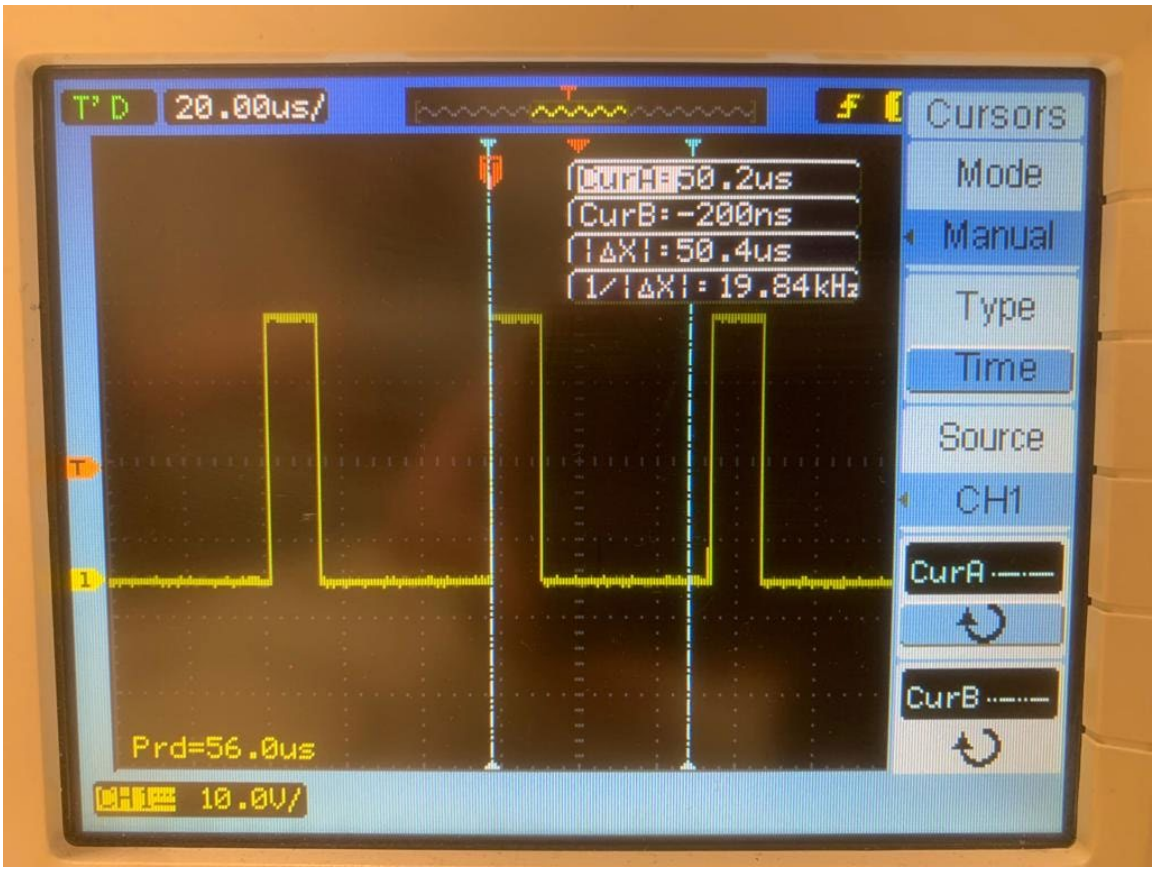


Figure 1: PWM module on the screen

```
1  #include "Pulse_init.h"
2  #include  "TM4C123GH6PM.h"
3
4  int main()
5  {
6    pulse_init();
7      while (1)
8    {
9      //do nothing here
10   }
11 }
12
```

Figure 2:Mainone function

```
57
58  void TIMER0A_Handler (void){
59    GPIOF->DATA  ^= 8;  //toggle PF3 pin
60
61    if(TIMER0->TAILR == LOW){
62      TIMER0->TAILR = HIGH;
63    }
64    else if(TIMER0->TAILR == HIGH) {
65    TIMER0->TAILR =LOW;
66    }
67
68    TIMER0->ICR |= 0x01;
69
70  }
71
```

Figure 3: Changed Part of the pulse_init function

Question 2) Two Clock Cycle

- In this part of the preliminary work, we initialized two clocks namely TIMER0A and TIMER3A. We initialized both timers succesfully but we could not handle capture mode interrupts. Also we use GPIOB_PIN2 as input pin we initialized that too. In figure 4 we add new main function with little changes and we add new pulse_init.h screen in figure 5 ,figure 6 and figure7 respectively.

```
1   #include  "TM4C123GH6PM.h"
2   #include "Pulse_init.h"
3   #include <stdio.h>
4
5   extern void OutStr(char*);
6   extern void delay(int);
7   char info[150];
8
9   int main (void){
10    pulse_init();
11    while(1){
12
13    }
14  }
```

Figure 4:Maintwo part of the preliminary work

```
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  #include <stdio.h>
9  #define LOW   0X09
10 #define HIGH  0x27
11 #define OUT (*((volatile unsigned long *)(GPIOB_BASE + 0x010UL)))
12 void pulse_init(void);
13 void TIMER0A_Handler (void);
14 void TIMER3A_Handler (void);
15 volatile double period;      // Global variables that used in ISR should be volatile
16 volatile double width;       // Global variables that used in ISR should be volatile
17 volatile double dc;
18 uint32_t rising_current_value = 0;
19 uint32_t old_rising_value  = 0;
20 uint32_t falling_current_value = 0;
21
22
23 void pulse_init(void){
24   volatile int *NVIC_EN0 = (volatile int*) 0xE000E100;
25   volatile int *NVIC_EN1 = (volatile int*) 0xE000E104;
26   volatile int *NVIC_PRI4 = (volatile int*) 0xE000E410;
27   volatile int *NVIC_PRI8 = (volatile int*) 0xE000E420;
28
29   SYSCTL->RCGCGPIO |= 0x22; // turn on bus clock for GPIOF
30   __ASM("NOP");
31   __ASM("NOP");
32   __ASM("NOP");
33   __ASM("NOP");
34 // PORT B AND F INITIALIZATION
35   GPIOF->DIR      |= 0x04;          // set PF2 as output
36   GPIOB->DIR      &= ~(1 << 2);     // set PB2 as input
37   GPIOF->AFSEL    &= (0xFFFFFFFB); // Regular port function
38   GPIOB->AFSEL    |= 0x04;          // use PB2 alternate function
39   GPIOF->PCTL     &= 0xFFFFF0FF;    // No alternate function
40   GPIOB->PCTL     &= ~(1 << 2);     // clear out bit-field for pin 2
41   GPIOB->PCTL     |= 0x700;         // set bit-field 7 for pin 2 using T3CCP0
42   GPIOF->AMSEL     = 0;             // Disable analog
43   GPIOB->AMSEL     = 0;             // Disable analog
44   GPIOF->DEN      |=0x04;           // Enable port digital
45   GPIOB->DEN      |=0x04;           // Enable port digital
46 // TIMER0 AND TIMER3 INITIALIZATION
47   SYSCTL->RCGCTIMER |=0x09;         // Start timer0 AND timer3
```

Figure 5: Pulse_Init function part 1

```
47    SYSCTL->RCGCTIMER |=0x09;         // Start timer0 AND timer3
48    __ASM("NOP");
49    __ASM("NOP");
50    __ASM("NOP");
51    __ASM("NOP");
52    TIMER0->CTL     &=0xFFFFFFFE;     // Disable timer during setup
53    TIMER3->CTL     &=0xFFFFFFFE;     // Disable timer during setup
54    TIMER0->CFG      =0x04;           // Set 16 bit mode
55    TIMER3->CFG      =0x04;           // Set 16 bit mode
56    TIMER0->TAMR     =0x02;           // Set to periodic, count down
57    TIMER3->TAMR    |=0x03;           // Capture mode enabled
58    TIMER3->TAMR    |=(1<<2)|(1<<4);  // Edge Time mode and counts up enabled
59    TIMER0->TAILR    =LOW-1;          // Set interval load as LOW. This is 16-bit, max of 1-65535.
60    TIMER0->TAPR     =15;             // Divide the clock by 16 to get 1us. 8-bit Prescaler can reduce the frequency (16MHz) by 1-255.
61    TIMER3->TAPR     =15;             // Divide the clock by 16 to get 1us. 8-bit Prescaler can reduce the frequency (16MHz) by 1-255.
62    TIMER0->IMR      =0x01;           // Enable timeout interrupt
63    TIMER3->IMR      =0x4;            // Enable capture mode event interrupt
64    TIMER3->CTL     |=0x0C;           // Enable capture rising and falling edges interrupts
65
66  // Timer0A is interrupt 19
67  // Interrupt 16-19 are handled by NVIC register PRI4
68  // Interrupt 19 is controlled by bits 31:29 of PRI4
69    *NVIC_PRI4 &=0x00FFFFFF;         // Clear interrupt 19 priority
70    *NVIC_PRI4 |=0x40000000;         // Set interrupt 19 priority to 2
71  // Interrupt 35 is controlled by bits 31:29 of PRI8
72    *NVIC_PRI8 &=0xFFFFFF;          //Clear interrupt 35 priority
73    *NVIC_PRI8 |=0x60000000;        //Set interrupt 35 priority to 3
74  // NVIC has to be neabled
75  // Interrupts 0-31 are handled by NVIC register EN0
76  // Interrupt 19 is controlled by bit 19
77  // Interrupt 35 is controlled by bit 35
78    *NVIC_EN0 |=0x00080000;
79    *NVIC_EN1 |=0x00000008;
80
81  //Enabling TIMER0 and TIME3
82    TIMER0->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
83    TIMER3->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
84
85    return;
86  }
87
88 void TIMER0A_Handler (void){
89
90    if(TIMER0->TAILR == LOW){
91      TIMER0->TAILR = HIGH;
92      GPIOF->DATA  ^= 4;  //toggle PF3 pin
93    }
```

Figure 6: Pulse_Init function part 2

```
81  //Enabling TIMER0 and TIME3
82    TIMER0->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
83    TIMER3->CTL      |=0x03; // bit0 to enable and bit 1 to stall on debug
84
85    return;
86  }
87
88  void TIMER0A_Handler (void){
89
90    if(TIMER0->TAILR == LOW){
91      TIMER0->TAILR = HIGH;
92      GPIOF->DATA  ^= 4;  //toggle PF3 pin
93    }
94    //TIMER0->ICR |=0x01; //Clear the interrupt
95
96    else {
97    TIMER0->TAILR = LOW;
98      GPIOF->DATA  ^= 4;  //toggle PF3 pin
99    }
100
101   TIMER0->ICR |= 0x01;
102
103 }
104 void TIMER3A_Handler (void){
105   if(OUT & 0x04) // Check value at the PB2
106   {
107     rising_current_value = TIMER3->TAR;
108     while (rising_current_value > old_rising_value){
109     period = ((rising_current_value - old_rising_value) & 0x00FFFFFF )* 0.0625;
110   }
111
112   }
113   else //falling edge
114   {
115     falling_current_value = TIMER3->TAR;
116     if (falling_current_value > old_rising_value){
117     width = ((falling_current_value - old_rising_value) & 0x00FFFFFF )* 0.0625;
118     }
119   }
120
121   old_rising_value = rising_current_value;
122   TIMER3->ICR |=0x04; //Clear the interrupt
123 }
124
125
126
```

Figure 7:Pulse_Init function part3