Khalifa University of Science, Technology and Research

Electronic Engineering Department

ELCE333Microprocessor Systems laboratory

Laboratory Experiment 7

USING HCS12 TIMER AND INTERRUPTS

Lab Partners

Leena ElNeel	100037083
Muna Darweesh	100035522
Shamsah AlNabooda	100036984

Date Experiment Performed: 18/3/2015
Date Lab Report Submitted: 25/3/2015

Lab Instructors:

Mahmoud Khonji

Mohammed Ali Saif Al Zaabi

Spring 2015

Table of Contents

List of Figures and Tables

Abstract

This experiment consists of three tasks which mainly are about writing programs involving using the HCS12 timer and to test them on the Dragon Board. The first task's goal was to measure the toggling rate of PORTB LEDs using an oscilloscope. The second task aimed to use the overflow interrupt and display the counter on the dragon12plus LEDs. The third task revolved around calculating the time delay and the prescalar value and displaying them.

1. Introduction

The timer system includes a free-running 16-bit counter and eight timer channels. The output/input compare/capture is based on timer channels that may be configured in any combination of timer called output compares, or input capture channels, which capture the time when an external event occurs. All these functions have interrupt controls and separate interrupt vectors, the interrupts are enabled or disabled by a bit in a control register. In this experiment the timer overflow and output compare functions along with the timer overflow interrupt.

1.1 Aim

The aim of this experiment was to be introduced to the use of timers and how delays can be implemented using timers and interrupts.

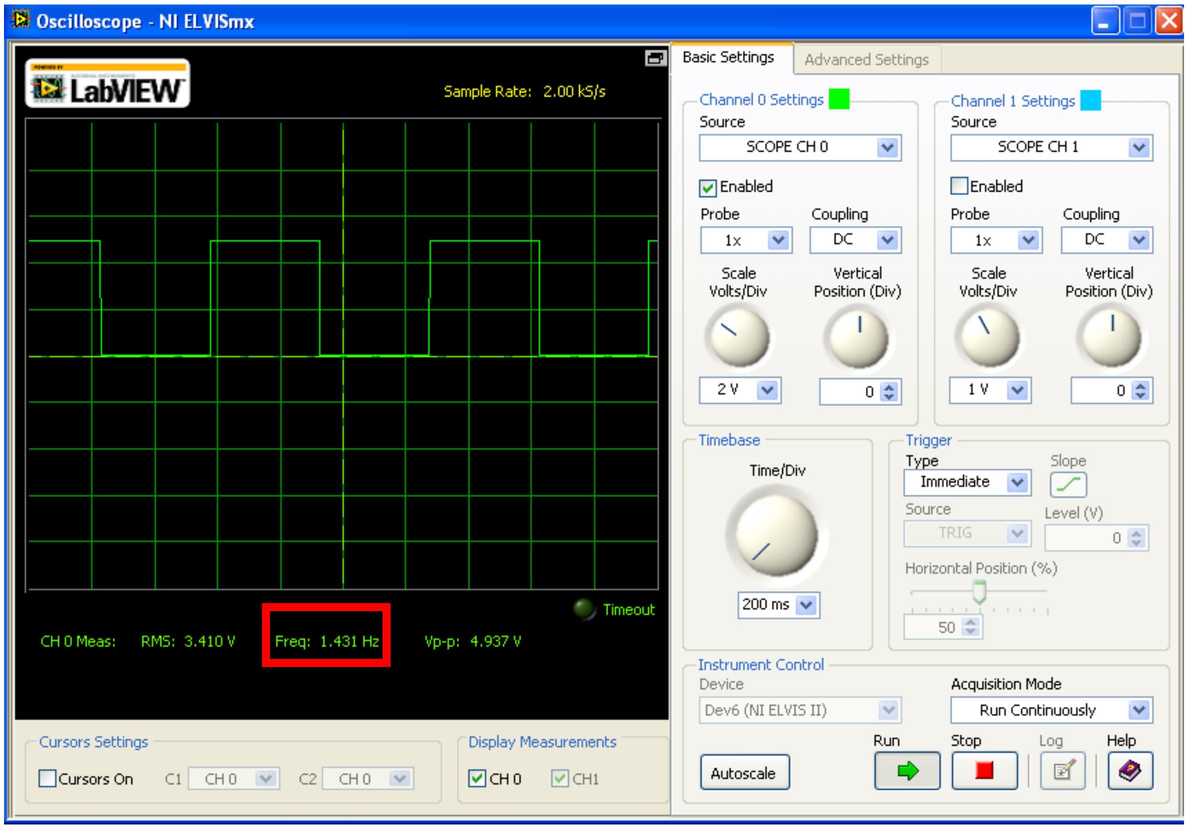
1.2 Objectives

The objectives of this experiment include understanding the concept of timers and writing C programs for using them along with and without interrupts. In addition to using the output compare feature of timer in HCS12. Moreover, the experiment included compiling, downloading, and testing these programs on the Dragon12 Plus Trainer board.

2. Design and Results

Task 1: Timer overflow

The aim of this task was to use the prescalar by running a given code. We were asked to modify the code in order to obtain some values for the prescaler. In the given code, the program toggles the LEDs in the Dragon12 Plus at a rate of 1.431 Hz. Figure 1 below shows the result obtained for the code when it was run.



Sample Rate: 2.00 kS/s

CH 0 Meas: RMS: 3.410 V Freq: 1.431 Hz Vp-p: 4.937 V

Channel 0 Settings: Source SCOPE CH 0; Enabled ✓; Probe 1x; Coupling DC; Scale Volts/Div 2 V; Vertical Position (Div) 0

Channel 1 Settings: Source SCOPE CH 1; Enabled ☐; Probe 1x; Coupling DC; Scale Volts/Div 1 V; Vertical Position (Div) 0

Timebase: Time/Div 200 ms

Trigger: Type Immediate; Source TRIG; Slope; Level (V) 0; Horizontal Position (%) 50

Instrument Control: Device Dev6 (NI ELVIS II); Acquisition Mode Run Continuously

Cursors Settings: ☐ Cursors On; C1 CH 0; C2 CH 0

Display Measurements: ✓ CH 0 ✓ CH1

Figure 1: Frequency measured for prescalar

We modified the given code by changing the prescalar value each time we run the code, the values of the frequencies measuring each prescalar is listed in the Table 1 and the modified code is shown in the following page:

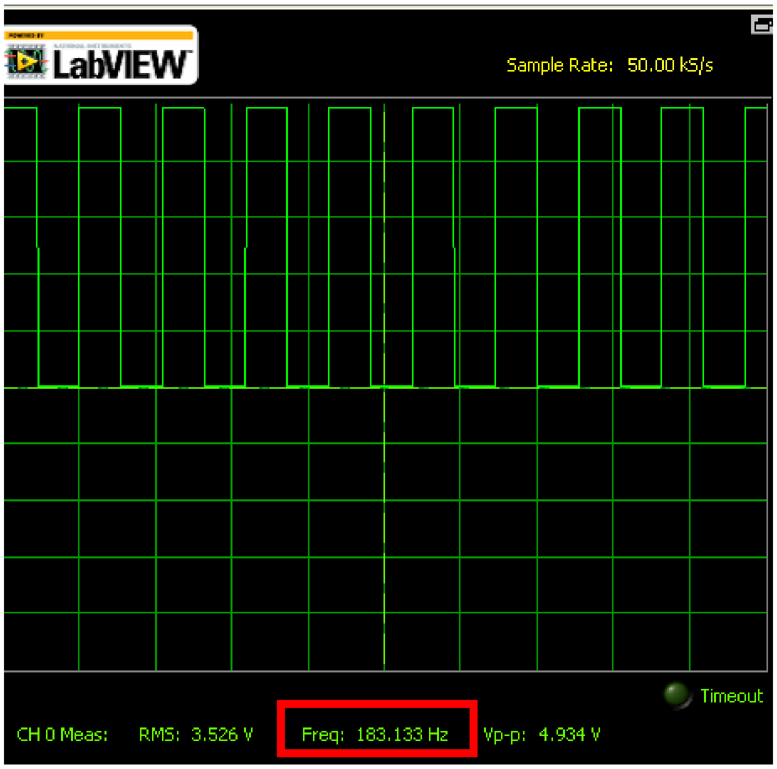
```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
int flag=0;
void init_timer(void)
{TSCR1 = 0x80; // enable timer counter
TSCR2 = 0x86; // enable timer interrupt, set prescaler to 128
TFLG2 = TFLG2_TOF_MASK; } // Reset timer overflow flag
void main(void)
{
 DDRB = 0xFF; //PORTB as output since LEDs are connected to it
DDRJ = 0xFF; //PTJ as output to control Dragon12+ LEDs
DDRP=0xFF;
PTP=0x0F; //Disable 7-seg display
PTJ=0x0; //Allow the LEDs to display data on PORTB pins
PORTB = 0x00;
DDRH &= 0x07;
init_timer();
__asm CLI;
for(;;){

TSCR2_PR0 = PTH_PTH0;
TSCR2_PR1= PTH_PTH1;
TSCR2_PR2 = PTH_PTH2;
if (flag==1)
{
PORTB=PORTB ^ 0xFF;
flag=0;
}} }
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimovf)/2)-1) TIMOVF_ISR(void)
{
flag=1;
TFLG2 =TFLG2_TOF_MASK;
}//Clear Interrupt
```

Table 1: the frequency corresponding to prescaler

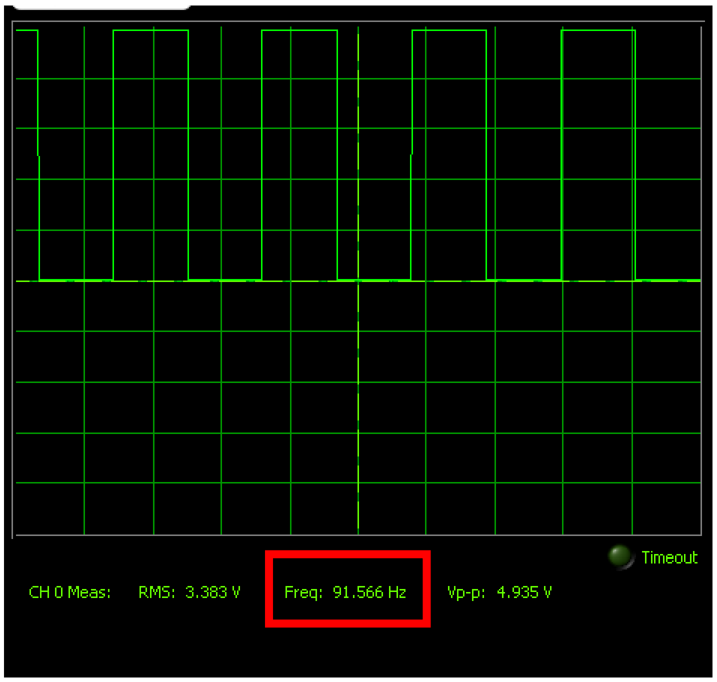
Prescaler	Frequency
000	183
001	91.5
010	45.7
011	41.1
100	23
101	11
110	4.23
111	1.430

Figure 2 and 3 below show cases of some prescalars' frequencies:



CH 0 Meas: RMS: 3.526 V Freq: 183.133 Hz Vp-p: 4.934 V

Figure 2: Frequency measured for prescalar 000



CH 0 Meas: RMS: 3.383 V Freq: 91.566 Hz Vp-p: 4.935 V

Figure 3: Frequency measured for prescalar 001

Task 2: Output compare

1.In this task, the output compare is used to generate digital wave form. To do so, we are required to execute the program shown in the introduction. Next, we monitored the signal at PTT2 pin using the oscilloscope and measured the signal period and frequency. The code below produces a square signal of 1kHz with 50% duty cycle on port (PT2). In this code, first, it is required to program the HCS12 timer. This is done through TSCR1 and TSCR2 registers. TSCR1 bit number 7 is set to 1, since 0x80 is 10000000 in binary because this bit corresponds to timer enable, the command is TSCR1 = 0x80;. The lower 3 bits of TSCR2 are set to %011 or 0x08 and this corresponds to prescaler, the command is TSCR2 = 0x03 . After that, the second channel of the Timer Interrupt Enable Register (TIER) is enabled by setting it to 1, the command is TIE_C2I=1. The clear interrupt flags are cleared and the Timer Input/Output Select (TIOS) second channel (IOS2) is set to 1 so the channel act as output compare, the command is TIOS_IOS2=1. Then, toggle OC2 pin by setting the output level (OL2) to 1. TC2 register, which is a 16 bit register for Each of the 8 compare channels, is set to a desired value which is based at the number of cycle. Then, the OC2pin is toggled. The TFLG1 second channel interrupt is reset.

```
#include "derivative.h" /* derivative-specific definitions */
void init_timer(void)
{ TSCR1 = 0x80; // enable timer counter
TSCR2 = 0x03; // disable timer interrupt, set prescaler to 8
TIE_C2I=1; // enable channel 2 interrupt
TFLG1 |= TFLG1_C2F_MASK; // Clear interrupt flags
TIOS_IOS2=1; // enable output compare channel 2
TCTL2_OL2=1; } // Toggle OC2 pin
void main(void)
{ init_timer();
__asm CLI;
for(;;){}
}
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch2)/2)-1) TIMCH2_ISR(void)
{TC2 += 1500; //start a new OC2 operation
//(No. of cycles (ON/OFF)= (24MHz*period(ON/OFF))/prescaler)
//period(ON/OFF)=1/(1KHz/2)
//(24MHz*0.0005)/8=1500
TCTL2_OL2=1; //Toggle OC2 pin
TFLG1 =TFLG1_C2F_MASK; } //reset Ch 2 interrupt
```

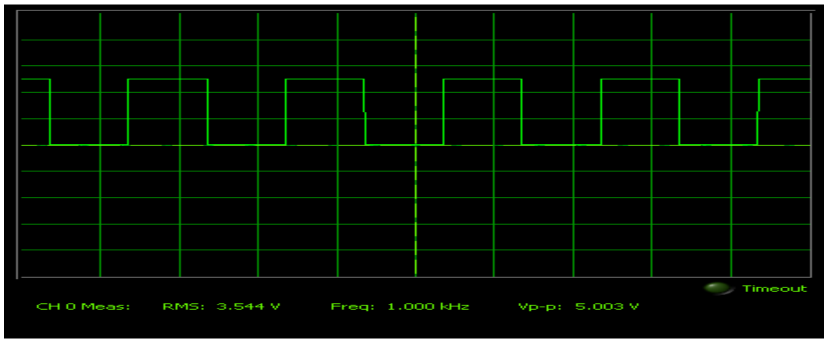


Figure 4: Square signal of 1kHz with 50% duty cycle

The code above produces square wave with 50% duty cycle at 1kHz. Actually, the number of cycle is calculated as the following:

$$(No. of\ cycles\ (ON/OFF)) = \ (24MHz * period(ON/OFF))/prescaler) \qquad (1)$$

The period is the inverse of the frequency:

$$Period = 1/(1kHz) = 1ms$$

Since, it is required to have 50% duty cycle, this means that part of the period has 50 % duty cycle, the signal is ON half the period (i.e $1ms/2 = 0.0005$)

$$(24MHz * 0.0005)/8 = 1500$$

This is the desired value that the TC2 is set to 1500. Using the oscilloscope, the result is as the following. The frequency is 1kHz and the duty cycle is 50% which is clear by having identical shape for the signal in the ON and OFF cycle.

<u>2</u>. In this part, it is required to modify the program to produce the signal with 30% duty cycle. To do so, it is required to find the values for TC2 to match the condition. Using the equation(1), and taking into consideration, that it is required to have 30% duty cycle, this means that part of the period has 30 % duty cycle, the signal is ON for 0.3 of the period, i.e, $1ms * (30/100) = 0.0005$)

$$(24MHz * 0.0003)/8 = 900$$

What is left of the 100% is 70% and this is considered to the OFF part of signal i.e, $1ms * (70/100) = 0.0007$

$$(24MHz * 0.0007)/8 = 2100$$

The code of the previous part is modified so that, it will account for 30% duty cycle. Thus, TC2 is modified according to the values above. Also, we need to conduct an if statement to check

TCTL2_OL2 bit is it is 0 or 1 because OL and OM of TCTL2 will be sit depending on this value. Hence, when OL2 is 1, the signal at the ON cycle, so the OM2 bit should be 1 and OL2 should be 0 and TC2 will be 900 and OL2 is 0, the signal is at the OFF cycle, so the both bits will be set to 1 and TC2 =2100.

```
#include <hidef.h>

#include "derivative.h"

 void init_timer(void)

{

TSCR1 = 0x80;

TSCR2 = 0x03;

TIE_C2I=1;

TFLG1 |= TFLG1_C2F_MASK;

TIOS_IOS2=1;

TCTL2_OL2=1;
```

```
void interrupt (((0x10000-Vtimch2)/2)-1)
TIMCH2_ISR(void)

{



if ( TCTL2_OL2==1)

{

TC2 +=900;

TCTL2_OL2=0;

TCTL2_OM2=1;  // clear OC2 pin to 0

TFLG1= TFLG1_C2F_MASK; //reset Ch 2
interrupt
```

Using the oscilloscope, the result is the following. Where, the ON cycle is less than the OFF cycle.

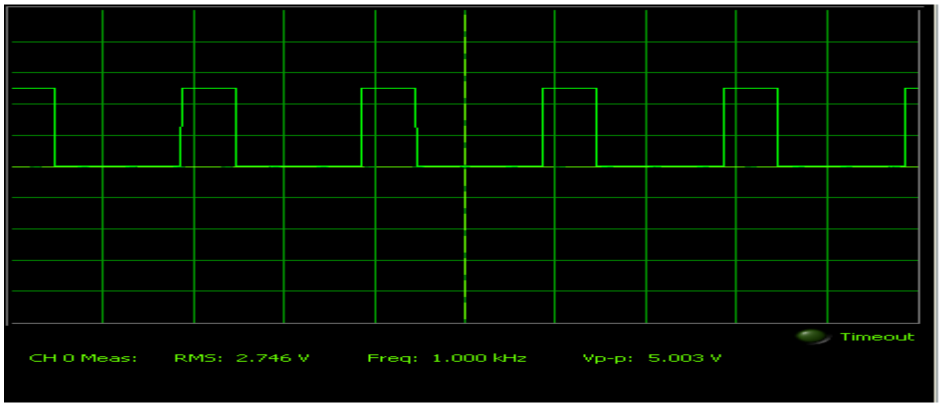


Figure 5: Square signal of 1 kHz with 30% duty cycle

Task 3: Input Capture

1.In the first part of this task we were asked to simulate a given code for input capture function that is used for measuring the time of outside events in which the registers that are used for selecting the channel and the type of the edge detection.

The following program uses input capture to measure the period and frequency of a square wave signal on pin PTT0:

```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
#include "lcd.h"
int edge1,flag; // this has to be int not float
float freq, period;
void init_timer(void)
{ TSCR1 = 0x80; // enable timer counter
TSCR2 = 0x03; // disable timer interrupt, set prescaler to 8
TIE_C0I=1; // enable channel 0 interrupt
TFLG1 |= TFLG1_C0F_MASK; // Clear interrupt flags
TIOS_IOS0=0; // enable input capture channel 0
TCTL4 = 0x01; // capture the rising edge of the PT0 pin
}
void main(void)
{ init_timer();
LCD_Init();
__asm CLI;
flag=0;

for(;;){
LCDWriteLine(2, "period= ");
LCDWriteInt(period);
LCDWriteChar(' ');LCDWriteChar('u'); LCDWriteChar('s');
LCDWriteLine(1, "freq= ");
LCDWriteFloat(freq);
LCDWriteChar(' ');LCDWriteChar('K');LCDWriteChar('H');LCDWriteChar('z');
delay(100);
LCD_clear_disp();}}
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1) TIMCH0_ISR(void)
{if(flag==0) { edge1= TC0; // save the first captured edge
flag=1;}
else{period=TC0 - edge1; // calculates period in cycles (second captured edge - first captured
edge))
period = period * 0.33; // calculates period in us (period in cycles*(1/(24/prescaler)))
freq=(1/period)*1000; // calculates frequency in KHz
flag=0;}
TFLG1 =TFLG1_C0F_MASK; } //reset Ch 0 interrupt
```

After establishing a code warrior project for the code above we used function generator to apply a 1KHz and 50 % duty cycle on PTT0.

2. In the second part of this task we were asked to combine the codes of task 2 and task 3 together to combine the functionality of the input capture and output compare. After combining the codes we were asked to measure the frequency and the duty cycle when connecting PTT0 and PTT2 together.

We start the code with enabling the timer counter an disabling the timer interrupt. Next, since we are using both PTT 0 and PTT2 we enabled the interrupts from corresponding channels, clear the flags and set PTT0 for input capture and PPT2 for output compare. Then, we initialize LCD to display the output (i.e. frequency and duty cycle). Frequency is measured when both rising and falling edges are captured and the signal is high in which its value will be $f = \frac{1}{period}\ Hz$. In addition to that duty is measured under the same circumstances in which it is depending on falling and rising edges in which it calculated as following equal to $duty = \frac{falling\ edge - rising\ edge}{last\ rising\ edge - rising\ edge}$. Moreover, we were required to make a duty cycle of 30 % as in task 2 in which we added the duty cycle controlling section to this current code which is shown as following.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
int flag1=0;
float edge1, edge2, edge3;
float freq, period, duty;
void init_timer(void){
TSCR1 = 0x80;
TSCR2 = 0x03;
TIE_C0I=1;
TIE_C2I=1;
TFLG1 |= TFLG1_C0F_MASK;
TFLG1 |= TFLG1_C2F_MASK;
TIOS_IOS0=0;
TIOS_IOS2=1;
TCTL4 = 0x03;
TCTL2_OL2=1;
TCTL2_OM2=1; }
void main(void){
LCD_Init();
__asm CLI;
init_timer();
for(;;){
LCDWriteLine(2, "Duty Cycle= ");
LCDWriteInt(duty);
LCDWriteChar(' ');LCDWriteChar('%');
LCDWriteLine(1, "Freq= ");
LCDWriteFloat(freq);
LCDWriteChar('
');LCDWriteChar('K');LCDWriteChar('H');LCDWrite
Char('z');
delay(100);
LCD_clear_disp();   }      }
```

```
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1)
TIMCH0_ISR(void){
if(flag==0 && PTT_PTT2==1){
edge1= TC0;
flag=1;}
else if(flag==1 && PTT_PTT2==0){
edge2= TC0;
flag=2;}
else if(flag==2 && PTT_PTT2==1){
edge3= TC0;
period=edge3-edge1;
period=period*0.33;
freq=(1/period)*1000;
duty=((edge2-edge1)/(edge3-edge1));*100;
flag==0;
edge1=0;
edge2=0;
edge3=0;}
TFLG1=TFLG1_C0F_MASK;}
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1)
TIMCH2_ISR(void){
if(TCTL2_OL2==1){
TC2+=900;
TCTL_OL2=0;
TCTL_OM2=1;
}
else if (TCTL2_OL2==0){
TC2+=2100;
TCTL_OL2=1;
TCTL_OM2=1;
}
TFLG1=TFLG1_C2F_MASK;
}
```

3. Assignment Questions

1) When using two output compare channels we can generate two different pulse waveforms. Write a program that produces two signals with different frequencies. The program should output a 50-Hz square wave on Port T bit 7 with a duty cycle based of 5% and a 300-Hz square wave on Port T bit 0 with a duty cycle of 5%. The duty cycles should be incremented by a 5% each time you press SW5 till it reaches 95% and then resets to 5% (use a PTH based interrupt) (Show your register settings and calculations).

The number of cycle is calculated as the following:

(No. of cycles (ON/OFF))= (24MHz*period(ON/OFF))/prescaler) (1)

The period is the inverse of the frequency:

Period=1/(*f*)

<u>For 50-Hz square wave:</u>

Using the equation(1), and taking into consideration, that it is required to have 5% duty cycle, this means that part of the period has 5 % duty cycle, the signal is ON for 0.05 of the period (1/50 s), i.e, 20ms*(5/100)=1ms

(24MHz*1m)/8=3000

What is left of the 100% is 95% and this is considered to the OFF part of signal i.e, 20ms *(95/100)=19ms

(24MHz*19ms)/8=57000

Thus, TC2 is modified according to the values above.

<u>For 300-Hz square wave:</u>

Using the equation(1), and taking into consideration, that it is required to have 5% duty cycle, this means that part of the period has 5 % duty cycle, the signal is ON for 0.05 of the period (1/300 s), i.e, 3.33ms*(5/100)

(24MHz*3.33ms*(5/100))/8=500

What is left of the 100% is 95% and this is considered to the OFF part of signal i.e, 3.33ms *(95/100)

(24MHz*3.33ms *(95/100))/8=95000

Thus, TC2 is modified according to the values above.

```
include <hidef.h> /* common defines and
macros */
#include "derivative.h" /* derivative-specific
definitions */
  float duty ;
void init_timer(void)
{
TSCR1 = 0x80; // enable timer counter
TSCR2 = 0x03; // disable timer interrupt, set
prescaler to 8
TIE_C0I=1; // enable channel 2 interrupt
TFLG1 = TFLG1_C0F_MASK; // Clear
interrupt flags
TIE_C7I=1; // enable channel 2 interrupt
TFLG1 = TFLG1_C7F_MASK; // Clear
interrupt flags

TIOS_IOS0=1; // enable output compare
channel 0
TCTL2_OL0=1;  // PTT0
TCTL2_OM0=1;

TIOS_IOS7=1; // enable output compare
channel 7
TCTL1_OL7=1;  // PTT7
TCTL1_OM7=1;
 }
void main(void)
{
DDRH = 0x00;   // Set PORTH as input
PIEH = 0x09;   // Enable PTH interrupt
PPSH = 0x00;   // Make it Edge-Trig.
init_timer();
__asm CLI;
for(;;){}
}
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1)
TIMCH0_ISR(void)
{
                              // for f= 300Hz
if(TCTL2_OL0 == 1)
{TC0 += 500;
TCTL2_OL0=0;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK;
}
else if(TCTL2_OL0 == 0) {
TC0 += 95000;
TCTL2_OL0=1;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK; }
```

```
}
void interrupt (((0x10000-Vtimch7)/2)-1)
TIMCH7_ISR(void)
{
// for f=50 Hz
if(TCTL1_OL7 == 1) {
TC7 += 3000;
TCTL1_OL7=0;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;
}
else if(TCTL1_OL7 == 0) {
TC7 += 57000;
TCTL1_OL7=1;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;
}
}
   float DC=0.05 ;
interrupt (((0x10000-Vporth)/2)-1) void
PORTH_ISR(void)
  {
   if(PIFH_PIFH0==1) {
   for(;;) {
     float DC1=DC+0.05;
    if (DC=0.95)
    DC=0.05;

      if(TCTL2_OL0 == 1)
{TC0 += 500;
TCTL2_OL0=0;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK;
}

else if(TCTL2_OL0 == 0) {
TC0 += 95000;
TCTL2_OL0=1;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK;
}
    if(TCTL1_OL7 == 1) {
TC7 += 3000;
TCTL1_OL7=0;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;
}
else if(TCTL1_OL7 == 0) {
TC7 += 57000;
TCTL1_OL7=1;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;
}
    PIFH=PIFH_PIFH0_MASK;
     }
   }
  }
```

2) Using Input Capture on channel PTT2 write a program that will measure the ON time and OFF time of the signals produced in the previous question and displays those on the LCD. Connect the signals to PTT2 and provide snap shots for the LCD screen.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
int flag1=0;
  float duty, edge1, edge2, onn, off ;
void init_timer(void)
{ TSCR1 = 0x80; // enable timer counter
TSCR2 = 0x03; // disable timer interrupt, set
prescaler to 8
TIE_C0I=1; // enable channel 2 interrupt
TFLG1 = TFLG1_C0F_MASK; // Clear interrupt flags
TIE_C7I=1; // enable channel 2 interrupt
TFLG1 = TFLG1_C7F_MASK; // Clear interrupt flags
TIOS_IOS0=1; // enable output compare channel 0
TCTL2_OL0=1;  // PTT0
TCTL2_OM0=1;
TIOS_IOS7=1; // enable output compare channel 7
TCTL1_OL7=1;  // PTT7
TCTL1_OM7=1;
 }
void main(void)
{ DDRH = 0x00;   // Set PORTH as input
PIEH = 0x09;   // Enable PTH interrupt
PPSH = 0x00;   // Make it Edge-Trig.
init_timer();
LCD_Init();
__asm CLI;
for(;;){
LCDWriteLine(2, "Duty Cycle= ");
LCDWriteInt(duty);
LCDWriteChar(' ');
LCDWriteChar('%');
LCDWriteLine(1, "Freq= ");
LCDWriteFloat(freq);
LCDWriteChar(' ');
LCDWriteChar('K');
LCDWriteChar('H');
LCDWriteChar('z');
delay(100);
LCD_clear_disp();}}
#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1)
TIMCH0_ISR(void){
if(flag==0 && PTT_PTT2==1){
edge1= TC0;//rising edge
flag=1;}
else if(flag==1 && PTT_PTT2==0){
edge2= TC0; //falling edge
flag=2;}
```

```
else if(flag==2 && PTT_PTT2==1){
edge3= TC0;
onn=edge2-edge1; // onn time = falling - rising
off= edge1-edge2; // off time = rising - falling
flag==0;
edge1=0;
edge2=0;
edge3=0;}
TFLG1=TFLG1_C0F_MASK;}
void interrupt (((0x10000-Vtimch0)/2)-1)
TIMCH0_ISR(void)
{   // for f= 300Hz
if(TCTL2_OL0 == 1)
{TC0 += 500;
TCTL2_OL0=0;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK;}
else if(TCTL2_OL0 == 0) {
TC0 += 95000;
TCTL2_OL0=1;
TCTL2_OM0=1;
TFLG1 =TFLG1_C0F_MASK; }}
void interrupt (((0x10000-Vtimch7)/2)-1)
TIMCH7_ISR(void)
{ // for f=50 Hz
if(TCTL1_OL7 == 1) {
TC7 += 3000;
TCTL1_OL7=0;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;}
else if(TCTL1_OL7 == 0) {
TC7 += 57000;
TCTL1_OL7=1;
TCTL1_OM7=1;
TFLG1 =TFLG1_C7F_MASK;}}
  float DC=0.05 ;
interrupt (((0x10000-Vporth)/2)-1) void
PORTH_ISR(void)
 { if(PIFH_PIFH0==1) {
  for(;;) {
  float DC1=DC+0.05;
   if (DC=0.95)
   DC=0.05;
     if(TCTL2_OL0 == 1)
```

See the source code for the rest of the code

4. Conclusion

This lab session mainly aimed to introduce the use HCS12 Timers and Interrupts and the idea of implementing delays using timers and interrupts. In Task 1, the design of the timer over flow is presented. The LED is flashed using a delay with timer overflow using TSCR (1 and2) registers to enable and program the timer. The oscilloscope is used to monitor with the help of an oscilloscope monitor in order to observe and measure the output frequency. Also, using the Dipswitches, we have done a manual modification on prescale values to observe the frequency. In the second task, it was required to understand the output compare function by simulating output compare code, using one of its 8 channel and one of this channel register which is TC2. TC can be modified to desired duty cycle and thus, the timer register bits, OL and OM, are modified. By using the oscilloscope, the square signal will be observed along with its frequency and the period and the duty cycle can be estimated.

In the third task, it is required to understand the input capture function of the microprocessor. Thus, the TCTL4 is used to capture the rising and falling edge of port T, so the applied signal, with 50% duty cycle, frequency and period is checked then the period and frequency values are displayed on the LCD.