Khalifa University for Sciences Technology and Research

Department of electrical and electronic engineering

ELCE333 :Microprocessor Systems Laboratory

Laboratory Experiment No. 7

Experiment Title: Using HCS12 Timers and Interrupts

Group Members

Afra I. Bin Fares 100033139

Alia Alalili 100037087

Anoud Alshamsi 100035514

Submitted to

Dr. Mahmoud Khonji

Dr. Mohammed Al Zaabi

Spring 2015

Abstract

The laboratory experiment seven is about getting more familiar with the design and execution of HCS12 timers and delays.Also, this laboratory experiment is divide it into three tasks . The first task is about examining and measuring the frequency's signal at any of PORTB pins at the oscilloscope with various prescaler values. Next, is the second task which aiming for monitoring the signal at PTT2 pin for measuring period and frequency with generating a certain duty cycle pulse signal. The final task which is task three is to modify a given C code to apply a certain delay on PTT0 by using the function generator and display its frequency and period on HCS12 LCD screen.

Table of Contents

List of tables and figures

1. Introduction

It is very difficult and impossible to implement time delay creation and measurement, period and pulse width measurement and frequency measurement applications without a timer function:The HCS12 has a complex standard timer module (TIM) that consists of eight channels of multiplexed input capture and output compare functions, 16-bit pulse accumulator A to count external events or act as a gated timer counting internal clock pulses, and 16-bit timer counter. these functions have interrupt controls and separate interrupt vectors, the interrupts are enabled or disabled by a bit in a control register (IRQ). The figure below shows a TIM block diagram.

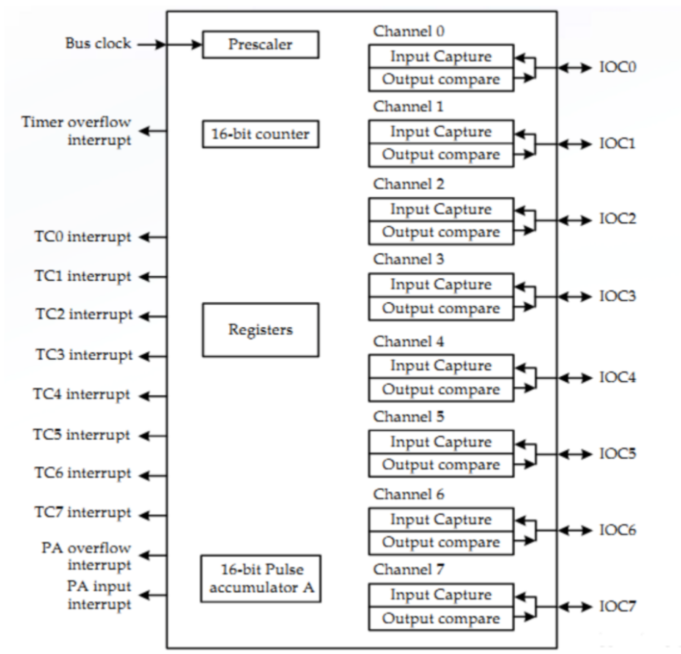


Figure 1 HSC12 standard timer (TIM) block diagram

The TIM shares the eight Port T pins (IOC0…IOC7) and the Timer Counter Register (TCNT) is required for input capture and output compare functions which must be accessed in one 16-bit operation in order to obtain the correct value. Three other registers related to the operation of the TCNT: TSCR1, TSCR2, TFLG2.

Aim:

To introduce the students to use of timers and how delays can be implemented using timers and interrupts.

Objectives:

the objective of this experiment is to understand the concept of timers and write a C program using timers overflow with interrupts. In addition to programming the output Compare feature and Input Capture feature of timer in HCS12. CodeWarrior IDE will be used for the development of HCS12 microcontroller C programs. oreover, the experiment included compiling, downloading, and testing these programs on the Dragon12 Plus Trainer board.

2. Design, results and analysis

In this part of the report, the tasks performed in the experiment will be discussed by explaining the steps of these tasks, listing, explaining and analyzing the results obtained from this experiment. The experiment is basically divided into three tasks. All three tasks were performed using C language written in the code warrior software, and the results were obtained using the oscilloscope.

2.1. TASK-1: Timer Overflow

In this part of the experiment the following program was used to make a CodeWarrior project:

```
#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 = 0x87; // 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;
init_timer();
__asm CLI;
for(;;){
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
```

This program was executed in the Dragon12 plus trainer, and then the signal at PORTB pin was monitored with the help of the oscilloscope. The figure below shows the result obtained in the oscilloscope:

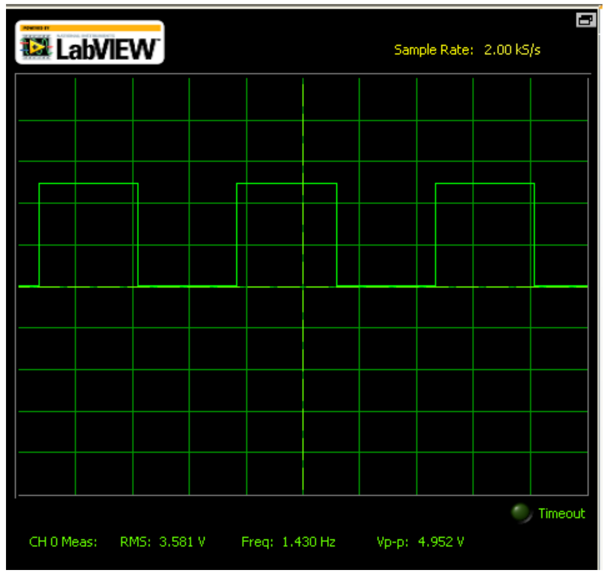


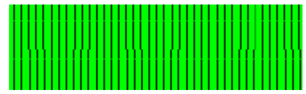
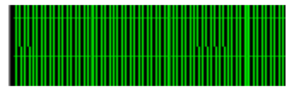
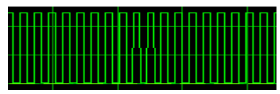
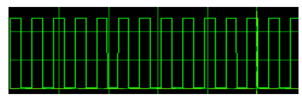
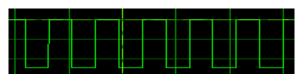
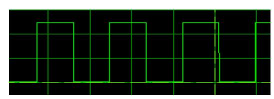
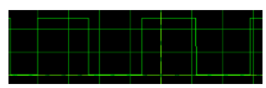
Figure 2 measured frequency

as it is observed from first part the figure above the sample rate is 2K/s and the frequency is about 1.43

The program was modified to set the prescaler value using the PTH DIP switches as follows:

```
#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 = 0x87; // 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 = 0x80|PTH;
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
```

Then, the frequency of any of PORTB pins was measured using the Scope for the following prescaler values shown in table1 :

Table1: prescaler output		
Prescaler	Frequency of PORTB pin (Hz)	Wave form
0	183.09	
1	91.546	
10	45.773	
11	22.886	
100	11.444	
101	5.722	
110	2.861	
111	1.432	

Observation and analysis: the main aim of this task was to investigate the timer overflow and the effect of the prescaler value in dividing the clock bus. The first Program was used to flash PORT B LEDs using a delay created with the timer overflow interrupt (which is explained in lab script7). In the second part the program was modified by adding the prescaler which is controlled using the PTH DIP Switches. It was observed that as the prescalar increases, the frequency decreases by half and the timer overflow takes longer to be introduced or executed.

2.2. TASK-2: Output Compare

In this part of the experiment it was asked to make a CodeWarrior project for the following program:

```
#include <hidef.h> /* common defines and macros */
#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
```

This program was executed in the Dragon12 plus trainer, and then the signal at PTT2 pin was monitored with the help of the oscilloscope. The figure below shows the result obtained in the oscilloscope:

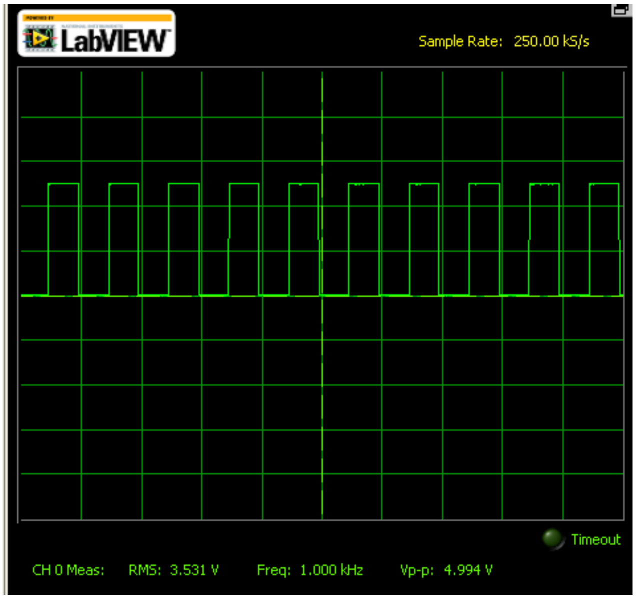


Figure 3 measured frequency for 50% duty cycle

The program was modified to produce a 1 KHz 30% duty cycle pulse signal as follows:

```
#include <hidef.h> /* common defines and macros */
#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) {
    if(TCTL2_OL2==1)
  {TC2 += 900; //start a new OC2 operation
 //(No. of cycles (ON)= (24MHz*period(ON))/prescaler)
 //period(ON)=1/(1KHz/2)
 //(24MHz*0.0003)/8=900
  TCTL2_OL2 =0;
  TCTL2_OM2 =1;
  TFLG1 = TFLG1_C2F_MASK;
} else if(TCTL2_OL2==0) //at the lower edge
 {
  TC2 += 2100; //start a new OC2 operation
 //(No. of cycles (OFF)= (24MHz*period(OFF))/prescaler)
 //period(OFF)=1/(1KHz/2)
 //(24MHz*0.0007)/8=2100
TCTL2_OL2 =1;
TCTL2_OM2 =1;
TFLG1 = TFLG1_C2F_MASK;
 }
 } //reset Ch 2 interrupt
```

This program was executed in the Dragon12 plus trainer, and then the signal at PTT2 pin was monitored with the help of the oscilloscope. The figure below shows the result obtained in the oscilloscope:

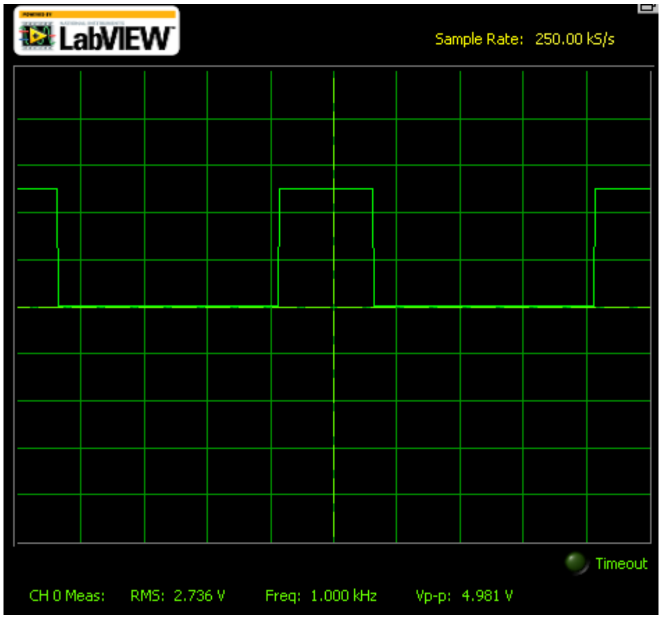


Figure 4 measured frequency for 30% duty cycle

Observation and analysis: the first program used output compare to produce a 1KHz square pulse with 50% duty cycle as it was observed in figure 3 where 50% of the 1Khz was on and 50% of it was off. The following calculation was performed in order to initialize the 24 MHz to the desired signal which is 1Hz: 24MHz*0.0005)/8=1500 , where 0.0005 represent the On/Off period and 8 represents the prescaler value. the second program was handled differently since the on/off periods do not equal to each other. The program needed to be modified in order to produce a 1KHz square pulse with a 30% duty cycle. Which means the 1Khz has to be on for 30% of the cycle and off for the 70% remaining of it. The OL and OM bits decide the action (as mentioned in the introduction) where used for the success of this program and an If-else conditions were used as it is shown in the program. The following calculations were performed in order to initialize30% of the 24 MHz signal to be on : (24MHz*0.0003)/8=900 . and (24MHz*0.0007)/8=2100 was used to make the 70% of the signal to be off.

2.3. Task-3: Input Capture

In this part of the experiment it was asked to make a CodeWarrior project for the following program:

```
#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
```

This program was executed in the Dragon12 plus trainer, and then using a function generator a 1 KHz 50% duty cycle signal on PTT0 was applied

Observation and analysis: the program was executed successfully and the following figure shows the output obtained on the LCD screen.

Figure 5 LCD output

The program was combined with task 2 second program as the following:

```
#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

  void interrupt (((0x10000-Vtimch2)/2)-1) TIMCH2_ISR(void) {
    if(TCTL2_OL2==1)
  {TC2 += 900; //start a new OC2 operation
 //(No. of cycles (ON)= (24MHz*period(ON))/prescaler)
 //period(ON)=1/(1KHz/2)
 //(24MHz*0.0003)/8=900
  TCTL2_OL2 =0;
  TCTL2_OM2 =1;
  TFLG1 = TFLG1_C2F_MASK;
} else if(TCTL2_OL2==0) //at the lower edge
 {
  TC2 += 2100; //start a new OC2 operation
 //(No. of cycles (OFF)= (24MHz*period(OFF))/prescaler)
 //period(OFF)=1/(1KHz/2)
 //(24MHz*0.0007)/8=2100
TCTL2_OL2 =1;
TCTL2_OM2 =1;
TFLG1 = TFLG1_C2F_MASK;
 }
 } //reset Ch 2 interrupt
```

Observation and analysis: the modified program was executed and the following figure shows the output obtained on the LCD screen. In this task the program was not modified the way it is supposed to , therefore the output obtained does not considered correct and this part of the experiment was not performed successfully.

Figure 6 LCD output

3. Conclusions and Recommendations

In conclusion, we believe that the objectives of the experiment were successfully achieved and we are now familiar with the use of timers as well as the implementation of delays using timers and interrupts. Also, the first task represents the idea of timers with overflow and the idea of prescaler. Whereas,from the second task we learned how to deal with timer interrupt by executing a written code .However, the third task represented the input capture feature by executing a written code that’s we combined task 2 with task 3 for applying the signal generated in task3.

4. 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 (use a PTH based interrupt) (Show your register settings and calculations).**

Calculations for 50-Hz square wave:
TC= 3MHz/300 Hz= 10 000
When ON:
TC*0.05= 500
When OFF:
TC*0.95= 9500

Calculations for 300-Hz square wave:
TC= 3MHz/50 Hz= 60 000
When ON:
TC*0.05 = 3000
When OFF:
TC*0.95= 57000

```
#include <hidef.h>
#include "derivative.h"

int inc=0.05;

void init_timer(void)
{ TSCR1 = 0x80; // enable timer counter
    TSCR2 = 0x03;

    TIE_C0I=1; // enable channel 0 interrupt
    TIE_C7I=1; // enable channel 7 interrupt

    TFLG1 |= TFLG1_C0F_MASK; // Clear interrupt flags
    TFLG1 |= TFLG1_C7F_MASK; // Clear interrupt flags


    TIOS_IOS0=1; // enable output compare channel 0 of port T
    TIOS_IOS7=1; // enable output compare channel 7 of port T



    TCTL2_OL0=1;
    TCTL2 OM0=1; // OC0 pin


    TCTL1_OL7=1;
    TCTL1_OM7=1;} // OO7 pin


void main(void)
{ init_timer();
    asm CLI;
    for(;;){}
}

#pragma CODE_SEG NON_BANKED
void interrupt (((0x10000-Vtimch0)/2)-1) TIMCH0_ISR(void)
{

    int TC= 10000;



    if(TCTL2_OL0==1){

        TC0 += (TC*inc);
        TCTL2_OL0=0; // OL0 pin low
        TCTL2_OM0=1;// OM0 pin high
        TFLG1 =TFLG1_C0F_MASK; }
```

```
    else if(TCTL2_OL0==0){


        TC0 += (TC*(1-inc));
        TCTL2_OL0=1;
        TCTL2_OM0=1;
        TFLG1 =TFLGl_COF_MASK;
    }
}
void interrupt (((0x10000-Vtimch7)./2)-1) TINCH7_ISR(void)
{

    int TC= 60000;

    if(TCTL1_OL7==1){

        TC7 += (TC*inc);
        TCTI1_OL7=0; // 007 pin low
        TCTI1_OM7=1;
        TFLG1 =TFLG1_C7F_MASK; }

    else if(TCTL1_OL7==0){
        TC7 += (TC*(1-inc));
        TCTL1_OL7=1 // 007 pin high
        TCTL1_OM7=1;
        TFLG1 =TFLG1_C7F_MASK;


#pragma CODE_SEG NON_BANKED
        interrupt (((0x10000-Vporth)/2)-1) void PORTH_ISR(void)
        {
            if(FIFH_PIFH0 ==1)
                for(;;) {
                    inc= inc+0.05;
                    if(inc=0.95) {
                        inc= 0.05;
                    }break;}}
```