**Khalifa University of Science, Technology and Research**

**Microprocessor Systems Laboratory ELCE332**

**Using HCS12 Timers and Interrupt**

Laboratory Experiment No. 7

**Lab Partners**

Mohammad Jassem 100037002

Khalid Al-Zarouni 100035499

Khalid Al-Hosani 100037003

**Date Experiment Performed**: 18-3-2015

**Date Lab Report Submitted**: 25-3-2015

**Lab Instructor:**

**Dr. Mahmoud Khonji, and Dr. Mohammad Al Azaabi**

**Spring 2013**

Table of Contents

List of Figures and Tables 3

Summary 4

1. Introduction 4

1.1Aim 5

1.2Objectives 5

2. Design and Results 6

3. Assignment Question: 14

4. Conclusions and Recommendations 16

List of Figures and Tables

List of figures:

Figure 1: Timer Overflow code 6

Figure 2: Frequency of Toggling using oscilloscope with prescaler 128 7

Figure 3: Timer Overflow with Prescaler specified using PTH DIP switches 8

Figure 4: Frequency of Toggling using oscilloscope with prescaler 64 9

Figure 5: generating a signal ( 50% duty cycle) 10

Figure 6: PTT2 pin output ( 50% duty cycle) 11

Figure 7: generating a signal ( 30% duty cycle) 12

**Figure 8: PTT2 pin output ( 30% duty cycle) 12**

Figure 9: input capture result 13

List of tables:

Table 1: Measured frequencies for various prescaler values 8

Summary

This lab report discusses the timer and interrupts of HCS12. This lab includes three main tasks. First, we should write a code that will provide a time delay in switching the LEDs ON and OFF due to the use of timer overflow. Task 2 explains how to write a code using output compare to generate a signal with a certain frequency ( 1KHz in our task). In task 3, we are asked to apply an external signal using the function generator to our program and check the period and frequency values on the LCD.

1. Introduction

Every timer needs a clock pulse to tick. Microcontroller clock source can be internal or external. For the internal clock source, oscillator frequency on the XTAL/EXTAL pins is fed into the timer. Timer is used for time measurement and time delay generation. In the external clock option, pulses feed through one of the HCS12’s pins. This is called a counter since it is being used for event counting. The built-in timer subsystem on the HCS12 has these features:

• 16-bit counter (TCNT) operating from system bus clock (4 MHz on lab trainer). • Upon reset, TCNT starts with zero and counts up with each pulse coming from the oscillator.

• A timer overflow flag to extend the 16-bit capability with 8 MHz clock, overflows are every 8.192 ms = (64K x 125 ns).

• Up to 8 output compare functions that can be used to time intervals shorter than one overflow period.

To apply the overflow, there are some sequences that the programmers should follows as the following:

1. The desired timer frequency should be set using the prescaler. (TSCR2)

2. The timer should be enabling. (TEN in TSCR1)

3. TOF is then cleared by writing a 1 to it. (TFLG2)

4. A loop that counts the number of overflows should be created.

– Wait on TOF.

– Clear TOF by writing a 1 to it.

– Repeat until loop is done.

1.1 Aim

The main aim of this experiment is to get familiar with the use of timers and how delays can be implemented using timers and interrupts

1.2 Objectives

1- Study the concept of timers.

2- To program a code using timers overflow in C.

3- Write C program using timers overflow with interrupts.

4- To program the Output Compare feature of timer in HCS12.

5- Use the CodeWarrior IDE for the development of HCS12 microcontroller C programs.

6- To compile, download and debug/test a C program using CodeWarrior C compiler and Dragon12 Plus Trainer board.

**2.** **Design and Results**

**TASK-1: Timer Overflow**

In the first task it is required to write a program that uses timer overflowto implement a delay. The output of the counter implemented in the program should be displayed on the LEDs connected to PORTB. So the program will be as following:

#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

Figure 1: Timer Overflow code

The program should be downloaded to the Dragon Plus Trainer to verify that it works correctly.

Then, the rate of toggling for PORTB pin is measured with the help of an oscilloscope monitoring the signal at PORTB pin1. The result shown in the oscilloscope is as figure 2.



Figure 2: Frequency of Toggling using oscilloscope with prescaler 128

**Comment**: When prescaler is equal to 128, the frequency obtained of toggling is equal to 1.43Hz

Now the previous code should be modified to use the PTH DIP switches in order to determine the value of the prescaler. The switches SW0, SW1 and SW2 will be used to give the value of the prescaler three bits. Thus the following figure shows the proper written code for that.

#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;

DDRH = 0x00;

PTP=0x0F; //Disable 7-seg display

PTJ=0x0; //Allow the LEDs to display data on PORTB pins

PORTB = 0x00;

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

Figure 3: Timer Overflow with Prescaler specified using PTH DIP switches

After that, for each value of prescaler, the frequency should be measured using the oscilloscope. And the measured frequencies will be as in the following table.

Table 1: Measured frequencies for various prescaler values

|  |  |
| --- | --- |
| **Prescaler** | **Freq. of PORTB0 (Hz)** |
| 000 | 183.09 |
| 001 | 91.5 |
| 010 | 45.7 |
| 011 | 22.9 |
| 100 | 11.45 |
| 101 | 9.9 |
| 110 | 2.86 |
| 111 | 1.44 |

**Comment**: as it is obvious from the measured values that when the prescaler value is increased, the rate of toggling will be decreased.

The following figures show the oscilloscope when the prescaler is 64



Figure 4: Frequency of Toggling using oscilloscope with prescaler 64

**TASK-2: Output compare**

In the second task, it is supposed to write a code that generates a signal with a certain frequency and using the oscilloscope we should monitor it at PTT2 pin after that, we have to measure the signal period and frequency. For this task, the following code has been written:

#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

Figure 5: generating a signal ( 50% duty cycle)

The program should be downloaded to the Dragon Plus Trainer to verify that it works correctly.

The following figure shows the output of PTT2 pin in which the frequency and period can be measured:



Figure 6: PTT2 pin output ( 50% duty cycle)

**Comment:** its clearly seen that the signal has a frequency of 1 KHz and period of 1/f which is 1 ms with duty cycle equals to 50%.

After that, we have been asked to modify the program to generate a signal with duty cycle equals to 30% ( 30 % ON and 70% off)

To calculate the TC2 for the high period:

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

**period(ON)=1/(1KHz)\* 0.3 = 0.0003**

**🡪TC2 (high) = (24MHz\*0.0003)/8=900**

* **TC2 (low) = 3000-900 = 2100**

The following code shows the full 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;

TCTL2\_OM2=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)

{ //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

if(TCTL2\_OL2 == 1){

TC2 +=900 ;

TCTL2\_OL2 = 0;

TFLG1 = TFLG1\_C2F\_MASK;

} else if(TCTL2\_OL2 == 0){

TC2 +=2100;

TCTL2\_OL2 = 1;

TFLG1 = TFLG1\_C2F\_MASK;

}

} //reset Ch 2 interrupt

Figure 7: generating a signal ( 30% duty cycle)

The following figure shows the output of PTT2 pin

****

**Figure 8: PTT2 pin output ( 30% duty cycle)**

**Comment:** the signal has the same frequency of the previous one but the off-period is greater than the ON-period due to the 30% duty cycle.

**TASK-3: Input Capture**

In the final task it is required to write a program that capture the input signal at PTT0 and displays its frequency and period on the LCD. The code of the program has been taken from the lab script and downloaded to the Dragon Plus Trainer to ensure that it works correctly. After running the program and connecting the input with PTT0 the following result has been captured. Then, we should add the code of the second task part 2 to this task, to use 30% duty cycles. Figure 9 below is the result once we compile the project in the first part.

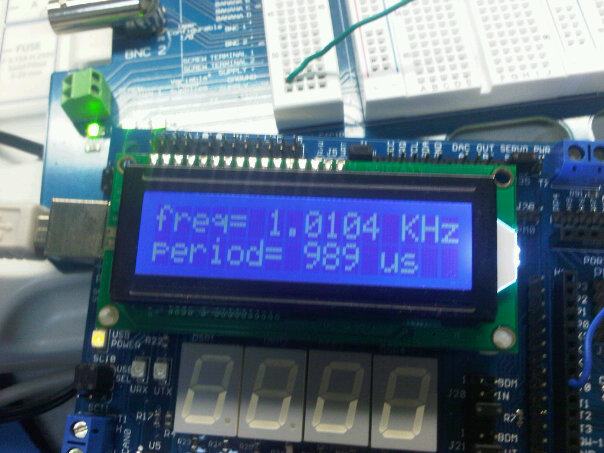


Figure 9: input capture result

**Comment**: the LCD shows that the signal has a frequency of 1.01 KHz and period of 0.989 ms.

# Assignment Question:

**Q.1 )** XTAL freqeucny = 48MHz , Bus clk = = 24 MHz

Prescale = 0x03, 🡪 For f=50Hz, (24MHz/2^3)/(50) = 60000

For f=300, (24MHz/2^3)/(270) = 10000

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

#include "derivative.h" /\* derivative-specific definitions \*/

float DC ;

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\_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; }

DC = 0.05;

void main(void)

{

init\_timer();

\_\_asm CLI;

for(;;){}

}

#pragma CODE\_SEG NON\_BANKED

void interrupt (((0x10000-Vtimch2)/2)-1) TIMCH2\_ISR(void)

{

// for f= 50Hz

if(TCTL2\_OL0 == 1) {

TC2 += 10000\*DC;

TCTL2\_OL0=0;

TFLG1 =TFLG1\_C2F\_MASK;

}

else if(TCTL2\_OL0 == 0) {

TC2 += 10000\*(1-DC);

TCTL2\_OL0=1;

TFLG1 =TFLG1\_C2F\_MASK;

// for f=300 Hz

if(TCTL1\_OL7 == 1) {

TC1 += 60000\*DC;

TCTL1\_OL7=0;

TFLG1 =TFLG1\_C2F\_MASK;

}

else if(TCTL1\_OL7 == 0) {

TC1 += 60000\*(1-DC);

TCTL1\_OL7=1;

TFLG1 =TFLG1\_C2F\_MASK;

}

}

interrupt (((0x10000-Vporth)/2)-1) void PORTH\_ISR(void)

{

DC=DC+0.05;

if(DC>=0.95)

duty=0.05;

PIFH = PIFH | 0xFF;

}

**Q.2)**

To do the following task, the following code should be addedto the previous one

float on , off, time;

int edge1,edge2,flag;

void main(void)

{

init\_timer();

\_\_asm CLI;

LCD\_Init();

DDRJ=0xFF;

PTJ=0x00;

PIEH = 0x01;

PPSH = 0x0;

flag=0;

for(;;){

LCDWriteLine(1, "ON period=");

LCDWriteFloat(on);

LCDWriteChar(' ');

LCDWriteChar('m');

LCDWriteChar('s');

LCDWriteLine(2, "OFF period=");

LCDWriteFloat(off);

LCDWriteChar(' ');

LCDWriteChar('m');

LCDWriteChar('s');

delay(300);

LCD\_clear\_disp();

}

}

#pragma CODE\_SEG NON\_BANKED

void interrupt (((0x10000-Vtimch2)/2)-1) TIMCH2\_ISR(void)

{

if((PTT\_PTT2==1) &&(flag==0)){

edge1= TC2;

flag=1; }

else if((PTT\_PTT2==0) &&(flag==1))

{

edge2=TC2;

flag=2;

} else if((PTT\_PTT2==1) &&(flag==2)){

time=TC2-edge1;

on=((edge2-edge1)\*5.33)/1000;

off=((time-(edge2-edge1))\*5.33)/1000;

flag=0;

}

TFLG1 =TFLG1\_C2F\_MASK;

}

5. Conclusions and Recommendations

To sum up, In the first task, it is noticeable that the code of the overflow provides a delay in time between each ON and OFF using a set of registers such as:

* TSCR1 : enable the timer when it is 1
* TSCR2 : set the prescale factor

Also the rate of toggling (1/T) was measured using the oscilloscope when applying different prescale. This step allows us to observe that the rate toggling changes due to the change of the value of the prescale. In other words, when increasing the pre-scale, the values of the rate toggling decrease. In the second task, it has been obviously seen that output compare is used to generate a signal with a desired frequency. In the third task, we have used the input capture to monitor and display the period and frequency of the input signal connected to PTT0.

The tasks of the seventh laboratory experiment are fulfilled. It is obvious that the objectives are successfully reached. The ability of understanding the use of the timer overflow in different codes is obviously is achieved. Also, it is understood that the interrupt and timer overflow can be used in the same code. Moreover, reading the recommended text book helps us to understand the concept of timer overflow. There were not any problems or risks faced in this practical session. Moreover, it seemed easier to write a code program and check it by the hardware implementation because of the serious practice and paying attention to the lab instructor which helped to avoid errors.