

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

***Electronic Engineering Department***

***ELCE333Microprocessor Systems laboratory***

**Laboratory Experiment 6**

**HCS12 Interrupts**

**Lab Partners**

*Anoud Al Shamsi,ID#: 100035514*

*Afra Bin Fares, ID#:100033139*

*Alya Humaid AlAlili ,ID#100037087*

**Lab Instructors:**

*Mohammed Ali Saif Al Zaabi*

*Mahmoud Khonji*

Table of Contents

Abstract 3

1. Introduction 4

1.1 Aim 5

1.2 Objectives 5

2. Design and Results 6

Task 1: Up Counter using C Language 6

Task 2: Interrupt Based Counter 7

Task 3: Interrupt Based Buzzer 8

Task-4: Interrupt Based LED Flashing 9

3. Conclusions and Recommendations 10

## *4. Assignment Questions\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11*

## *Abstract*

The main goal of experiment laboratory six to get familiar with handling interrupt that can occur to microcontrollers. Also, in this lab we will go through various methods to handle microcontrollers’ interrupts. Moreover, this laboratory experiment is divide it into four main tasks. The first task is about 8-bit binary counter that counts from $00 to $0F and repeats counting. The next task is using the IRQ switch to present an interrupt service routine in 8 bits counter. Whereas, the third task is representing the procedure of interrupt based buzzer functionality for different frequencies controlled by PTH. Finally, the fourth task we need to change the pattern of the flashing LEDs when interrupt occur.

## *Introduction*

Some hardware events may generate interrupt , for example interrupt occurs on port H when SW2 is pushed, the CPU need to halt normal operation as well as jump to a specific location in memory , where this situation is called an Interrupt Service Routine (ISR). The ISR acts as a subroutine , also it contains a code that is can be executed once the ISR is entered.After that,control is passed back to the main program when the code in the ISR is completed.

Once an interrupt occurs there are a specific way to locate interrupt service routines in HCS12 .For example , the program jumps to the Interrupt Vector Table when only a particular interrupt is activated.Also, each interrupt vector is located at a unique memory address in the table and it contains the address of the start of the ISR that will run for that interrupt.Moreover, for each type of interrupt only one interrupt vector is valid. In the Program Counter Register (PC) the address of the ISR will be load by CPU for starting the execution . After that , the HCS12 will go to address of the interrupt vector address .

The interrupt vector addresses differ from the ISR addresses that’s the interrupt vector addresses are fixed by the hardware, whereas the ISR addresses in each interrupt vector are changeable values determined by the programmer.

Its important to know that the interrupt can be enabled when it is on a low-to-high event (rising edge) or high-to-low event (falling edge) by setting the polarity in the relevant interrupt register. In addition, the Port H is tied to switch buttons, so a Port H pin tied to a switch will sit at +5V (high or logical 1) until we push it, once we push it, this take the Port H pin to ground (low or logical 0) and thus activate our interrupt.

There are several types of interrupts like the Port H interrupt that it is triggered by peripheral and it is built in to the HCS12 chip. Another type of interrupts which is external interrupts that used by devices external to the HCS12 such as the IRQ pin is an external interrupt that can be used for general purpose cases. Also, if any device connected to the IRQ pin ,the pin will be taken from high to low which that’s mean the IRQ interrupt will be set off , as result the IRQ interrupt service routine will be executed. The non maskable interrupt is type of interruption that it can't be turned off or masked and it is used to service interrupts that have extremely urgent service requests.

***1.1 Aim:***

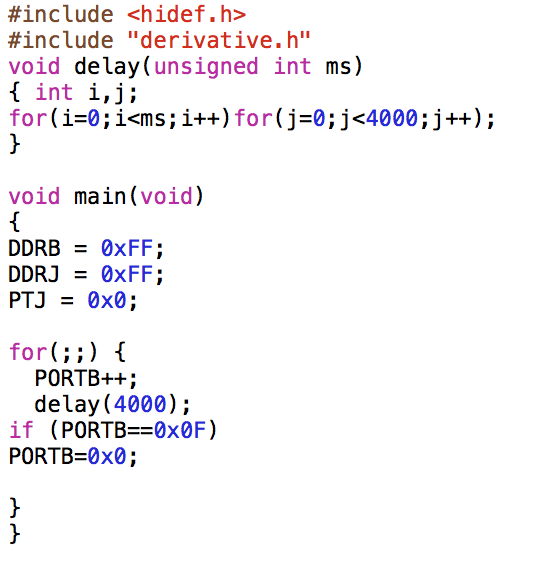
To introduce the concept of interrupt and their purpose in embedded system.

* 1. ***Objectives:***

1. To understand interrupts and their function in embedded system.
2. To write a program that deal with external interrupts.
3. To develop simple programs for an embedded system.
4. To use the CodeWarrior IDE for the development of HCS12 microcontroller C programs.
5. To compile, download and debug/test a C program using CodeWarrior C compiler and Dragon12+ Trainer board.
6. ***Design and Results***

**TASK-1: Up Counter using C Language**

In the first task we were required to write a code that has a 8-bit binary counter which counts from $00 to $0F and repeats counting again. The frequency of the counter should be 1⁄4 Hz which means there is a 4000 ms delay. The value of the count is displayed on LED7-LED0. The code is as following:



First PORTB is incremented with a delay of 1/4 Hz \* delay(4000) \* between each increment, until it reaches 0x0F and then start from the beginning \* PORTB=0x0 \*.

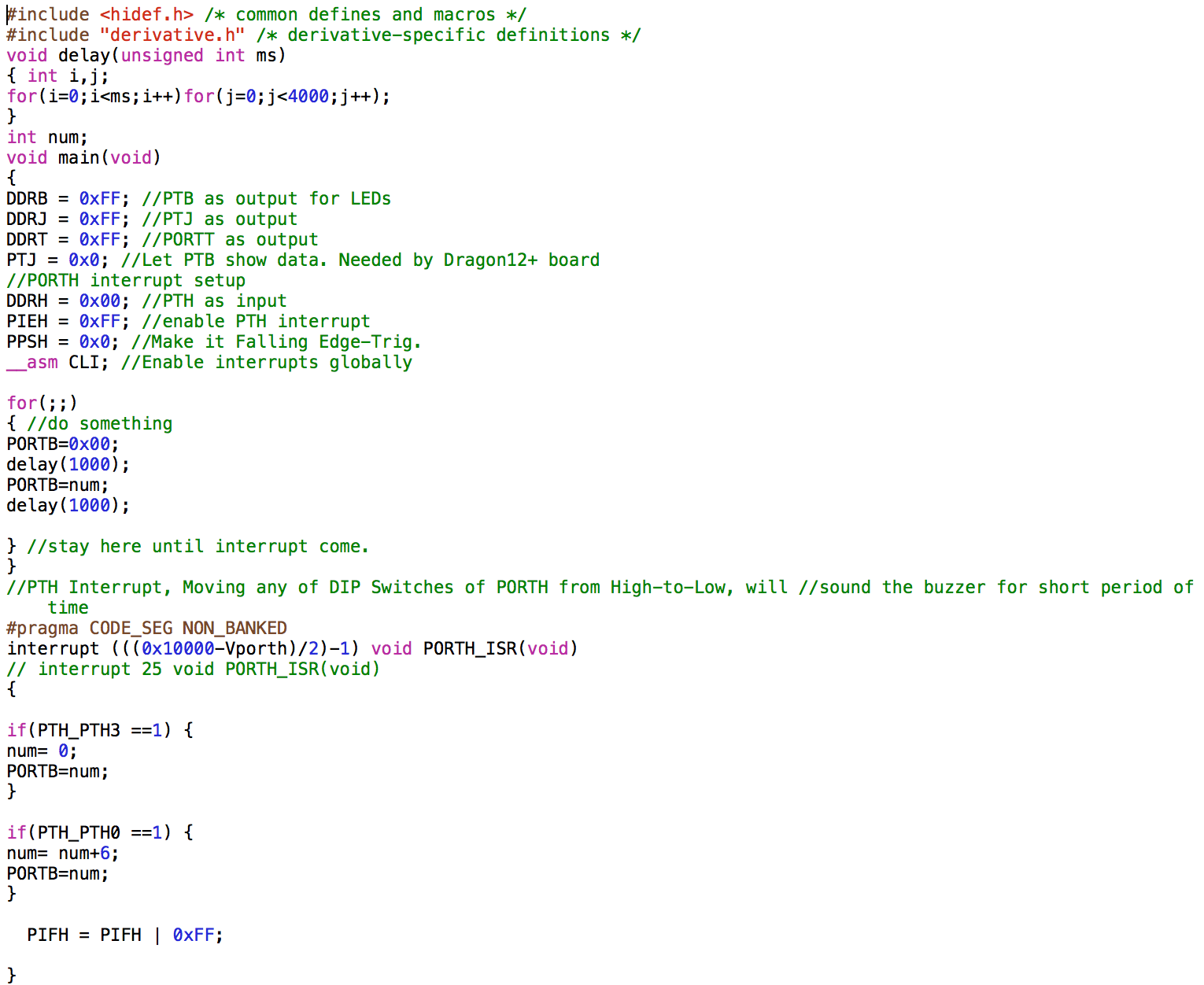
**TASK-2: Interrupt Based Counter**

In this task, an interrupt routine modifies task 1’s counter using the IRQ switch. So, the previous counter will be modified to be interrupted when the IRQ switch is pressed, the current contents of switches SW3-SW0 must be used to initialize the starting count of the counter. For example, if the switches are set to $06, the counter should go from $06 $07 . . . to $0F and repeat $06 $07 . . . $0F. In the for loop there is first a delay of 1000 then PORTB will be incremented, when it reaches 0x0F , PORTB while be equal to the number pressed.

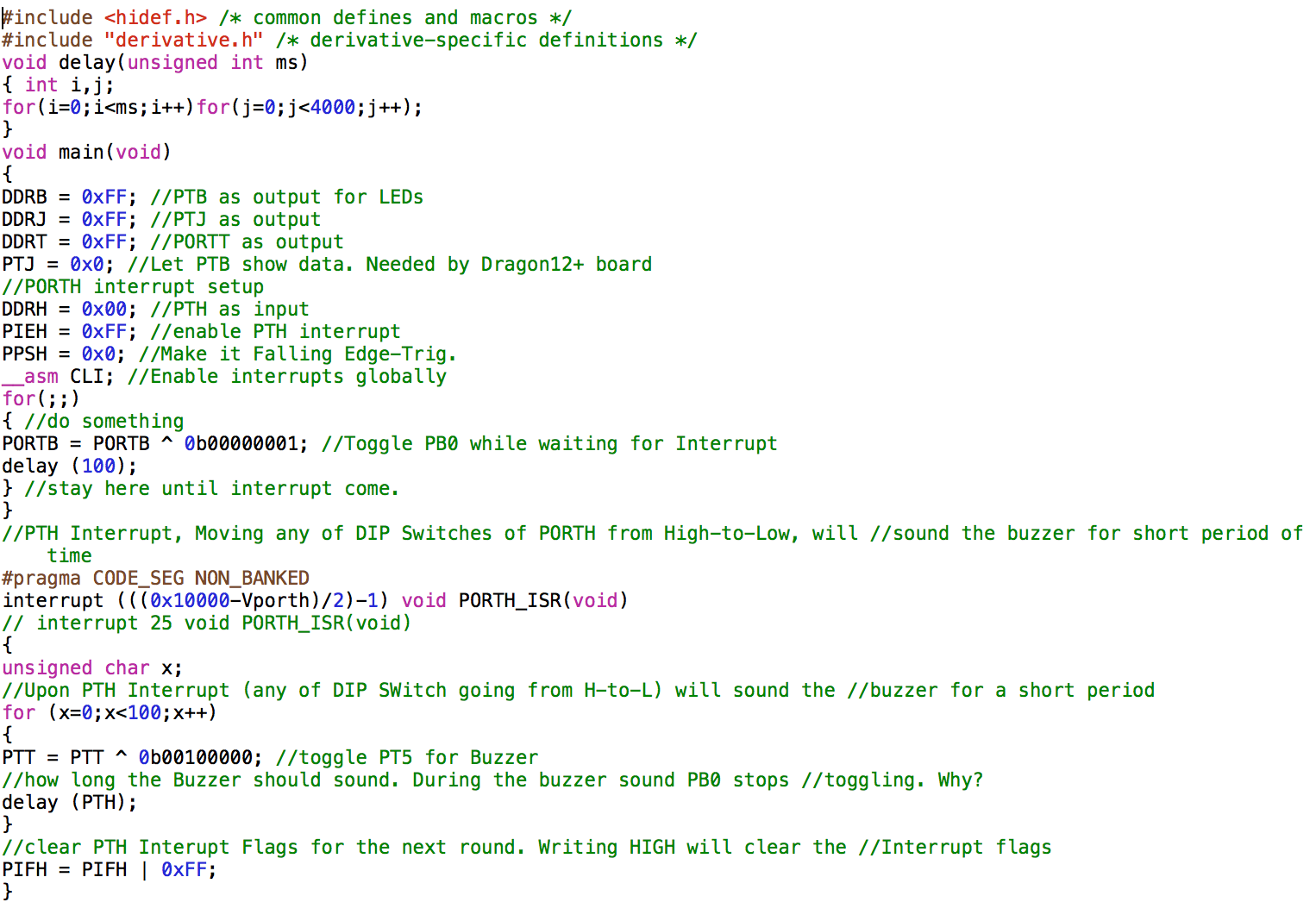
The code is implemented as follows:



**Task-3: Interrupt Based Buzzer**

In this task the interrupt service routine given in the introduction was modified in order to generate a sound the buzzer (PT5) for a short period of time at different frequency for each bit of the PTH (SW5-SW4). the buzzer will make a different sound each time PTH changed. The ms delay function was modified by using this formula msdelay(( PTH $00011000) \*3 +100), this function made the buzzer sound change according to the change of bits 4,5 of PTH, the frequency of the buzzer will change accordingly the sound of the buzzer will change. a delay which equals to the value of the PTH is set for how long the buzzer will sound. the program as follows:

**Task-4: Interrupt Based LED Flashing**

In the final task, a program was written that flashes all the LEDs with a delay of 100 ms and changes the flashing pattern displayed with a different pattern each time SW5 is pressed, and stops the flashing each time SW2 is pressed. When SW2 is pressed, the value of PTH will equal to zero in order to stop the flashing. When SW5 is pressed the value of PORTB will increment by a random value which we chose here as 6.The code used for this program is the following:

3.Conclusion

To sum up, we dealt with lab experiment with HCS12 interrupts, the objectives were successfully met and the tasks were performed and implemented correctly. A good realization of the different types of interrupts in embedded systems was gained. And the lab team members learned how to write a program that handles external interrupts through these tasks. All tasks and concepts were cleas and easy to be implemented however it is recommended that the students should come prepared well and get a good understanding of the theoretical concepts in order to improve the performance of the laboratory experiment. No problems or risks were faced in this lab experiment.

Assignment Questions

1. Write an interrupt based binary counter that will display the reached count on the LCD if SW 3 is pressed and reverses the counter if SW4 is pressed (keep your interrupt routine optimized).

#include <hidef.h>

#include "derivative.h"

#include "lcd.h"

int Num=0;

void delay(unsigned int ms)

{ int i,j;

for(i=0;i<ms;i++)for(j=0;j<4000;j++); }

void main(void) {

DDRB = 0xFF; //PTB as output for LEDs

DDRP = 0xFF; //PTP as output

PTJ = 0x0F;

DDRJ = 0xFF; //PTJ as output

DDRT = 0xFF; //PORTT as output

PTP = 0x0F; // Disable the 7-segment display

PTJ = 0x00; //Let PTB show data. Needed by Dragon12+ board

//PORTH interrupt setup

DDRH = 0x00; //PTH as input

PIEH = 0xFF; //enable PTH interrupt

PPSH = 0x00; //Make it Falling Edge-Trig.

\_\_asm CLI; //Enable interrupts globally

LCD\_Init();

for(;;) { //do something

Num++;

PORTB = num; //Toggle PB0 while waiting for Interrupt

delay (1000); } //stay here until interrupt come. }

//PTH Interrupt, Moving any of DIP Switches of PORTH from High-to-Low, will

//sound the buzzer for short period of time

#pragma CODE\_SEG NON\_BANKED

interrupt 25 void PORTH\_ISR(void) {

if(PIFH\_PIFH0==1) {

for(;;){

LCDWriteInt(num);

PIFH= PIFH\_PIFH0\_MASK;

LCD\_Init() } }

if(PIFH\_PIFH2==1)

{ { for(;;){

Num--;

delay (1000);

PORTB=num;}

PIFH= PIFH\_PIFH3\_MASK; }}

1. Define interrupt latency and give at least two components of interrupt latency in HCS12.

Interrupt latency, also called interrupt response time, is the length of time that it takes for a microprocessor interrupt to be acted on after it has been generated. It is basically a signal that tells the controller to stop and decide what to do next. There are many factors that affect the interrupt latency like the microprocessor architecture, the microprocessor clock speed and the type of interrupt controller used. Components of interrupt latency in HCS12 include the following:

* The source of the interrupt
* The complexity of the instruction
* The number of clock cycles needed to execute the instruction
* Time to complete the current instruction.
* Time to push the registers on the stack.
* Time to fetch the ISR address.