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

**Electronic Engineering Department**

**ELCE332 Microprocessor Systems laboratory**

**Laboratory Experiment 3**

**HCS12 Input and Output Ports**

**Lab Partners**

Mohammad Jassem 100037002

Khalid waleed Alzarouni 100035499

Khalid Alhosani 100037003

**Group: 2**

**Date Experiment Performed**: 04-03-2015

**Date Lab Report Submitted**: 18-03-2015

**Lab Instructor: Mahmoud Khonji**

**Mohammed Alzaabi**

**Spring 2015**

Table of Contents

Abstract 3

1. Introduction 4

1.1 Aim 6

1.2 Objectives 6

2. Design and Results 7

Task 1: Up Counter using C Language 7

Task 2: Interrupt Based Counter 8

Task 3: Interrupt Based Buzzer 9

Task-4: Interrupt Based LED Flashing 10

3. Assignment Questions 11

4. Conclusions and Recommendations 14

Abstract

In this report we successfully meet the main aim of this laboratory experiment which to introduce the students to the concept of interrupt and their usage in embedded system. This report divided into three main divisions. Firstly, it has a brief introduction about the laboratory experiment and discusses the aims, objectives, and introduction. The second part has all the four tasks and their experiment codes. Also some comments about the results are attached. At the third part we have the assignment questions and their solution and the report will be concluded by a brief discussion on the main achievements.

1. Introduction

What is an interrupt? It is a special event occurs to get the attention of the CPU and makes it stop its normal execution sequence to provide and perform some services for itself. There are two types of interrupts, internal and external and all of them are caused by the IRQ (which is interrupt request). The IRQ works by requesting the running program to perform an action called Interrupt service routine (ISR). Moreover, there are several functions of the interrupts like; organizing the I/O activities, CPU tied up prevention, and error detections and handler. Furthermore, the interrupts are either maskable or nonmaskable. Maskable interrupts are the ones that the CPU can ignore, and they need to be enabled by setting an enable flag before it can interrupt the CPU. While the nonmaskable interrupts are the ones the CPU cannot ignore. Also we have another two types of interrupt and they would be; level trigged interrupts and edged trigged interrupts. As an example of the level trigged interrupts in HCS12 is the PTH, while an example of the edged trigged interrupts is the IRQ. What happens when an interrupt occurs to a process? It will execute the following steps:

1- Complete the currently executing instruction

2- Turn off the interrupt system

3- Transfer the control to the interrupt service routine

4- Clear the interrupt source

5- Store important registers to stack

6- Perform interrupt specific actions (ISR)

7- Restore important registers from stack

8- Turn on interrupt system

9- Return control to prior running process

The memory address of an interrupt handler called the interrupt vector, or an index into an array called an interrupt vector table. The interrupt vector table contains the memory addresses of interrupt handlers. We can determine interrupt vectors by following these steps:

1. Predefined locations (Microchip PIC18, 8051 variants)

2. Fetching the vector from a predefined memory location (HCS12, Atmel avr)

3. Executing an interrupt acknowledge cycle to fetch a vector number in order to locate the interrupt vector (68000 and x86 families)

The interrupt must be enabled by software and in order for the microcontroller to respond them there is a register called IE. The IE register is responsible for enabling and disabling the interrupts. In the reset mechanism there is an establishment of the initial values of some CPU registers, flip-flops and the control registers in I/O interface chips that must be established in order for the computer to function properly. Resets is divided into two types, the first type called the power-on reset, and the second type called the manual reset. The power-on reset establishes the initial values of registers and I/O control registers. The manual reset without power-down allows the computer to get out of most error conditions if hardware doesn’t fail.

1.1 Aim

To introduce the students to the concept of interrupt and their usage in embedded system.

1.2 Objectives

On completion of this experiment the student should be able to:

1- Understand interrupts and their use in embedded system.

2- Write a program that handles external interrupts.

3- Develop simple programs for an embedded system.

4- 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 Plus Trainer board.

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

## Task 1: Up Counter using C Language

In this task it is required to write an 8-bit binary counter that counts from $00 to $0F and repeats. It should increment at about ¼ Hz and the count should be displayed on LED7-LED0.

The code of this task is shown below:

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

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

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

DDRJ = 0xFF; //PTJ as output

DDRT = 0xFF; //PORTT as output

PTJ = 0x0; //Let PTB show data. Needed by Dragon12+ board

//PORTH interrupt setup

DDRH = 0x00; //PTH as input

PIEH = 0xFF; //enable PTH interrupt

PPSH = 0x0; //Make it Falling Edge-Trig.

\_\_asm CLI; //Enable interrupts globally

for(;;)

{

delay(4000);

if (PORTB == 0x0F){

PORTB= 0x00;

delay(4000);

}

PORTB++;

//do something

//PORTB = PORTB ^ 0b00000001; //Toggle PB0 while waiting for Interrupt

//delay (100);

} //stay here until interrupt come.

Comment: This code will make the LED to light from $00 to $0F and repeat.

## Task 2: Interrupt Based Counter

In this task, it is required to write an interrupt service routine using the IRQ switch. When the IRQ switch is pressed, the current contents of switches SW4-SW1 must be used to initialize the starting count of the counter done in part 1. For example, if the switches are set to $06, the counter should go from $06 $07 . . . to $0F and repeat $06 $07 . . . $0F. The counter must be immediately set to the new initial starting value when the IRQ switch is pressed and there must not be any noticeable change in the rate at which the counter increments. The code is shown below:

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

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

void delay(unsigned int ms)

{ int i,j;

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

}

int var;

void main(void)

{

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

DDRJ = 0xFF; //PTJ as output

DDRT = 0xFF; //PORTT as output

PTJ = 0x0; //Let PTB show data. Needed by Dragon12+ board

INTCR = 0xC0;

//PORTH interrupt setup

DDRH = 0x00; //PTH as input

PIEH = 0xFF; //enable PTH interrupt

PPSH = 0x0; //Make it Falling Edge-Trig.

\_\_asm CLI; //Enable interrupts globally

for(;;)

{

delay(4000);

if(0 ){

var = PORTB;

}

if (PORTB == 0x0F){

if(0) {

PORTB= var;

}

else PORTB= 0x00;

delay(4000);

}

PORTB++;

//do something

//PORTB = PORTB ^ 0b00000001; //Toggle PB0 while waiting for Interrupt

//delay (100);

} //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 6 void lollll(void)

// interrupt 25 void PORTH\_ISR(void)

{

unsigned int x = PTH;

x = x & 0x0f;

PORTB = x;

}

Comment: The previous code will display on the LED the counter but this time and by using IRQ, the counter will be shown according to the switches. As we press IRQ and when we use the second switch for example the counter will initialize from $02 until $0F.

## Task 3: Interrupt Based Buzzer

In this task it is required to modify the ISR shown in the introduction to sound the buzzer (PT5) for a short period of time at different frequency for each bit of the PTH. Each bit of PTH can represent a door (or window) and as any door gets opened, the buzzer will make a unique sound. The code for this task is shown below:

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

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

void delay(unsigned int ms)

{ int i,j;

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

}

int var;

void main(void)

{

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

DDRJ = 0xFF; //PTJ as output

DDRT = 0xFF; //PORTT as output

PTJ = 0x0; //Let PTB show data. Needed by Dragon12+ board

INTCR = 0xC0;

//PORTH interrupt setup

DDRH = 0x00; //PTH as input

PIEH = 0xFF; //enable PTH interrupt

PPSH = 0x0; //Make it Falling Edge-Trig.

\_\_asm CLI; //Enable interrupts globally

}

#pragma CODE\_SEG NON\_BANKED

interrupt 25 void PORTH\_ISR(void)

{

unsigned int x;

//Upon PTH Interrupt (any of DIP SWitch going from H-to-L) will sound the //buzzer for a short period

for (x=0;x<100;x++)

{

//toggle PT5 for Buzzer

//how long the Buzzer should sound. During the buzzer sound PB0 stops //toggling. Why?

PTT = PTT ^ 0b00100000;

delay (PTH);

//clear PTH Interupt Flags for the next round. Writing HIGH will clear the //Interrupt flags

PIFH = PIFH | 0xFF;

}

}

Comment: The previous code will display on the LED the number which we press on PTH and we will here different sound(buzzer) for different period for each PTH.

## Task-4: Interrupt Based LED Flashing

In this task it is required to write an interrupt based program that flashes all LEDs with a delay of 100 ms and changes the flashing pattern displayed with a different pattern each time you press SW5 and stops the flashing each time you press SW2.

The code of this task is shown below:

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

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

int x=0xFF;

void delay(unsigned int ms)

{

unsigned int i; unsigned int j;

for(i=0;i<ms;i++)

for(j=0;j<4000;j++);

}

void main(void)

{

DDRB = 0xFF; // Set PORTB as output since LEDs are connected to it

DDRJ = 0xFF; // Set PORTJ as output to control Dragon12+ LEDs

DDRP = 0xFF; // Set PORTP as output

DDRT = 0xFF; // Set PORTT as output

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

PTJ = 0x00; // Turn off PTJ1 to allow the LEDs to show data on PORTB

//PORTH interrupt setup

DDRH = 0x00; // Set PORTH as input

PIEH = 0x09; // Enable PTH interrupt

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

EnableInterrupts;

for(;;)

{ //do something

PORTB =0x00;

delay(400);

PORTB = x;

delay(400);

} //stay here until interrupt come.

}

#pragma CODE\_SEG NON\_BANKED // to access the interrupt vector table

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

interrupt 25 void PORTH\_ISR(void)

{

if(PIFH\_PIFH0==1)

{

x=x+10;

}

if(PIFH\_PIFH3==1) {

x=0;

}

PIFH = PIFH | 0xFF;

}

Comment: The previous code will light the LED with delay 100ms. In addition, as we press switch 5 different pattern will be displayed and as we press switch 2 the flash will stop.

# 3. 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> /\* common defines and macros \*/

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

#include "lcd.h"

int var;

void main(void)

{

DDRB = 0xFF; // Set PORTB as output since LEDs are connected to it

DDRJ = 0xFF; // Set PORTJ as output to control Dragon12+ LEDs

DDRP = 0xFF; // Set PORTP as output

DDRT = 0xFF; // Set PORTT as output

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

PTJ = 0x00; // Turn off PTJ1 to allow the LEDs to show data on PORTB

//PORTH interrupt setup

DDRH = 0x00; // Set PORTH as input

PIEH = 0xFF; // Enable PTH interrupt

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

\_\_asm CLI;

for(;;)

{

PORTB++;

delay (4000);

if ( PORTB == 0x0F)

PORTB = 0x00 ;

} //stay here until interrupt come.

}

#pragma CODE\_SEG NON\_BANKED // to access the interrupt vector table

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

interrupt 25 void PORTH\_ISR(void)

{

LCD\_Init();

if(PIFH\_PIFH2==1)

{

for(;;){

LCDWriteLine(1, "the number= ");

LCDWriteFloat(PORTB);

}

PIFH=PIFH\_PIFH2\_MASK;

}

if(PIFH\_PIFH1==1) {

for(;;)

{

PORTB--;

delay (4000);

if ( PORTB == 0x00)

PORTB = 0x0F ;

}

PIFH=PIFH\_PIFH1\_MASK;

}

}

**Comment:**

**This code will count from 0 until 15. If we press on SW3, the number that the counter arrive will be shown in LCD as shown in the below figure while if we press on SW4, the counter will be reversed.**

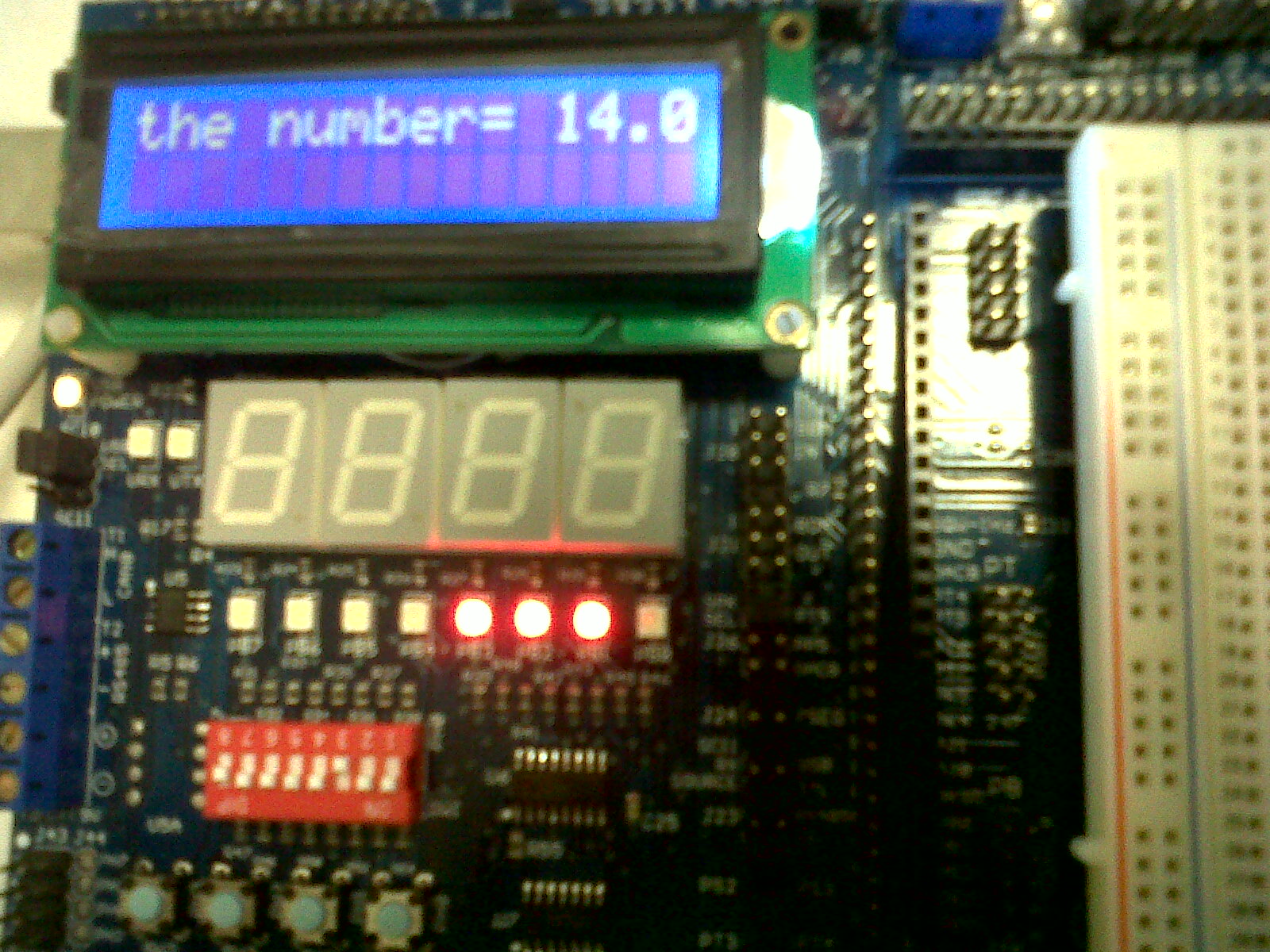
****

Figure 1: assignment question

**As seen in the figure, the counter is stopped at 14 when we press switch3 and the result displayed on LCD.**

**2) Define interrupt latency and give at least two components of interrupt latency in HCS12.**

Interrupt latency is the time that elapses from when an [interrupt](http://en.wikipedia.org/wiki/Interrupt) is generated to when the source of the interrupt is serviced. This duration can be affected depending upon:

1. Source of the interrupt.
2. The complexity of current instruction and the number of clock cycles needed to execute it.
3. Time needed to save current microcontroller status and registers on the stack.

Interrupt latency also refer to the time taken from the leading edge of an external interrupt request signal to the processor to the fetch of the first interrupt service instruction by the processor. Interrupt latency consists of the sum of two components: processor interrupt latency and software interrupt latency.

4. Conclusions and Recommendations

As a conclusion, the tasks of the second laboratory experiment are fulfilled. It is obvious that the objectives are successfully reached. understanding the interrupts and their use in embedded system, writing program that have an external interrupts and we developed a simple program for an embedded system. Furthermore we were able to compile, download and test a C program using CodeWarrior C compiler and Dragon12 Plus Trainer board. There were not any problems or risks faced in this practical session. Moreover, serios practising and paying attention to the lab instructor helped us in the laboratory session especially in avoiding errors.