

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

**Electronic Engineering Department**

**ELCE332 Microprocessor Systems laboratory**

**Laboratory Experiment 6**

**HCS12 Interrupts**

**Lab Partners**

Hamad Eisa Alhazami 100036985

Jasim Alhammadi 100035510

**Date Experiment Performed**: 4-3-2015

**Date Lab Report Submitted**: 18-3-2015

**Lab Instructor: Mr. Mohammed Alzaabi/ Mr. Mahmoud Khounji**

**Spring 2015**

# Summary

The purpose of this report is to discuss the results obtained after conducting the four main tasks in the laboratory session. The first task required to implement a counter using C language. The second task is a slight modification on task 1 to introduce Interrupt Service Routine (ISR) in the program. The third task was about introducing different buzzer sounds to the interrupt. The fourth task required creating an interrupt for two push buttons, the first one to change the pattern of LEDs, and the second one to turn all the LEDs off.

Contents

[Summary 2](#_Toc414446810)

[1. Introduction 4](#_Toc414446811)

[1.1. Aim 4](#_Toc414446812)

[1.2. Objectives 4](#_Toc414446813)

[2. Design and results 5](#_Toc414446814)

[2.1. Task 1: Up counter Using C language: 5](#_Toc414446815)

[2.2. Task 2: Interrupt Based Counter 6](#_Toc414446816)

[2.3. Task 3: Interrupt Based Buzzer 8](#_Toc414446817)

[2.4. Task 4: Interrupt based LED flashing: 9](#_Toc414446818)

[3. Analysis and Interpretation 11](#_Toc414446819)

[4. Assignment Questions 12](#_Toc414446820)

[5. Conclusion and Recommendation 14](#_Toc414446821)

# Introduction

An interrupt is a special subroutine that requires the CPU to stop the execution of the current program, and handle the cause of the interrupt event. Interrupts are used to coordinate and handle errors or urgent activities. There are several types of interrupts including: I/O completion; timer time-out; illegal op-codes; arithmetic overflow; and divide-by-0.

There are two types of interrupts, maskable and non-maskable interrupts. Maskable interrupts are interrupts that can be ignored by the CPU; thus, they need to be enabled first in order to let the CPU handle them. Non-maskable interrupts are the interrupts that CPU cannot be ignored.

The CPU should have a program to handle interrupts called Interrupt Service Routine (ISR). When an interrupt occurs, the ISR saves the current running program counter value and the CPU status in the stack. Then, it identifies the cause of the interrupt and resolves it. After that, it executes the interrupt service routine. Once the ISR is finished, it restores the CPU status and the program counter from the stack.

## Aim

## 

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

## 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.

# Design and results

## Task 1: Up counter Using C language:

In the first 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 program is written using C language, and run on Dragon 12 Plus board using Code Warrior. 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++);**

**} void main(void) {**

**int x=0x00;**

**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**

**DDRH = 0x00; //PTH as input**

**PIEH = 0xFF; //enable PTH interrupt**

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

**\_\_asm CLI; //Enable interrupts globally**

**for(;;)**

**{ //do something**

**PORTB = x; //Toggle PB0 while waiting for Interrupt**

**delay (4000);**

**if (x==0x0F) x=0x00;**

**else x++;**

**} }**

**#pragma CODE\_SEG NON\_BANKED**

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

**{ unsigned char x;**

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

**{ PTT = PTT ^ 0b00100000; //toggle PT5 for Buzzer**

**delay (10);**

**} PIFH = PIFH | 0xFF;}**

Dragon board registers needed to be set in order to let the program function correctly. Registers DDRB, DDRJ and DDRT are set to HEX value 0xFFin order to set them up as output for LEDs. Register PTJ for Port J is set to HEX value 0x00 to Let Port B – for LEDs - show data. Register DDRH is set to 0x00 to let Port H – for push buttons - serve as input. In order to enable interrupts, the line “\_\_asm CLI” is written to enable interrupts globally, and register PPSH is set to value 0x00 to make it Falling Edge-Trigger. And finally register PIEH is set to the 0xFF enable PTH interrupt.

The counter is implemented by a loop that increment an integer value and set the value into LEDs registers. The interrupt will introduce a delay with a buzzer sound once the push button clicked for a period. As observed, the LEDs were showing the counter and once it reached the value $FF, it goes back to zero and increment again.

## Task 2: Interrupt Based Counter

In the second task, it is required to write an interrupt service routine using the IRQ switch. If 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. 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 program is written using C language, and run on Dragon 12 Plus board using Code Warrior. 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 num=0x00;**

**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.**

**INTCR = 0b11000000;**

**\_\_asm CLI; //Enable interrupts globally**

**for(;;)**

**{**

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

**delay (1000);**

**if (num==0x0F) num=0x00;**

**else num++;**

**} }**

**#pragma CODE\_SEG NON\_BANKED**

**interrupt 6 void loool(void)**

**// interrupt 25 void PORTH\_ISR(void)**

**{**

**unsigned x = PTH;**

**x = x & 0x0F;**

**PORTB = x;**

**}**

The same setup for registers in task 1 is used for the task 2. The program algorithm consists of a loop to implement the counter and interrupt function. The interrupt is set to the value 6 – the line code **“interrupt 6 void loool(void)”** in order to enable the IRQ interrupt. The initial value of counter when the interrupt is enabled is set by reading the value from PTH port – the switches. As observed, the counter is started from the value set by the switched with the IRQ is enabled.

## Task 3: Interrupt Based Buzzer

In the third task, it is required to modify the ISR to enable the buzzer at a different frequency for each bit of the PTH. The program is written using C language, and run on Dragon 12 Plus board using Code Warrior. 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++);**

**} void main(void)**

**{ int x=0x00;**

**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**

**DDRH = 0x00; //PTH as input**

**PIEH = 0xFF; //enable PTH interrupt**

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

**INTCR = 0b11000000;**

**\_\_asm CLI; //Enable interrupts globally**

**for(;;)**

**{ //do something**

**PORTB = 0x00; //Toggle PB0 while waiting for Interrupt**

**delay (500);**

**} }**

**#pragma CODE\_SEG NON\_BANKED**

**interrupt 25 void PORTH\_ISR(void)**

**{unsigned char x;**

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

**{ PTT = PTT ^ 0b00100000; //toggle PT5 for Buzzer**

**delay (PTH);}**

**PIFH = PIFH | 0xFF;}**

In order to set the sounds for the buzzer, the delay will be set by the value of Port H (the switches). With this implementation, each bit in the switch will produce a unique buzzer sound. As observed, the buzzer had different sounds for every bit when it is turned on.

## Task 4: Interrupt based LED flashing:

In the last task, it is required to write an interrupt based program that flashes all LEDs with a delay of 100 msec. It will change the flashing pattern displayed each time the SW5 is pressed, and stops the flashing each time SW2 is pressed. The program is written using C language, and run on Dragon 12 Plus board using Code Warrior. The code is shown below:

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

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

**int num=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**

**DDRH = 0x00; // Set PORTH as input**

**PIEH = 0x09; // Enable PTH interrupt**

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

**\_\_asm CLI;**

**EnableInterrupts;**

**for(;;) {**

**PORTB =0x00;**

**delay(100);**

**PORTB = num;**

**delay(100);**

**} }**

**interrupt 25 void PORTH\_ISR(void)**

**{ if(PIFH\_PIFH0==1) { num=num-1; }**

**if(PIFH\_PIFH3==1) { num=0; }**

**PIFH = PIFH | 0xFF;**}

The program algorithm consists of a loop and an interrupt function. The loop is used to set the LEDs value. The interrupt has two “if” conditions to check if either push button 0 or 3 is pressed. For Push button 0, the pattern will be changed by decrementing the value saved in Port B (the LEDs). While for button 3, it will set Port B to 0 i.e. all LEDs will be of – zeros. As observed, the patterns of LEDs change each time button zero is pressed, and all LEDs go off each time button 3 is pressed.

# Analysis and Interpretation

In the four tasks, the interrupt is used in order to save the cycles consumed by the subroutines. The program will jump to the interrupt subroutine when in the occurrence of an event when pressing or enabling a button on the board. There are several interrupts for several buttons pressed on the board. For all tasks except task 2, interrupt 25 is used for Port H. While in the second task, interrupt 6 is used for IRQ.

# Assignment Questions

**1- Write an interrupt based binary counter that will display the reached count on the LCD if SW3 is pressed and reverses the counter if SW4 is pressed.**

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

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

#include "lcd.h"

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

DDRH = 0x00; // Set PORTH as input

PIEH = 0xFF; // Enable PTH interrupt

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

\_\_asm CLI;

EnableInterrupt;

for(;;)

{PORTB++;

delay (100\*30);

if ( PORTB == 0x0F)

PORTB = 0x00 ;

} }

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

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 (1000);

if ( PORTB == 0x00)

PORTB = 0x0F ; }

PIFH=PIFH\_PIFH1\_MASK;} }

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

Interrupt latency is the time interval between interrupt activation till the CPU starts to execute Interrupt Subroutine code.

Several factors can affect the interrupt latency:

1. The interrupt source.
2. Interrupt time complexity (required time to save the current state of microcontroller, time to execute the interrupt cycles.

# Conclusion and Recommendation

To conclude, the lab session was very useful in teaching the students how to implement the interrupts in programming. In doing so, new skills were obtained as the programming was done in C. In addition, it introduced the students to how to generate different unique sounds by the buzzer using a function of an interrupt. The concept of interrupts is clear and can be implemented for use in future programs.