**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 Alzarouni 100035499

Khalid Alhosani 100037003

**Group: 2**

**Date Experiment Performed**: 11-2-2015

**Date Lab Report Submitted**: 18-2-2015

**Lab Instructor: Mahmoud Khonji**

**Mohammed Alzaabi**

**Spring 2015**

Table of Contents

[List of Figures and Tables 3](#_Toc412027970)

[Summary 4](#_Toc412027971)

[Introduction 5](#_Toc412027972)

[1.1 Aim 5](#_Toc412027973)

[1.2 Objectives 5](#_Toc412027974)

[2. Design and Results 6](#_Toc412027975)

[Task 1: Deriving Output Lines 6](#_Toc412027976)

[Table 1: The content of Accumulator Register A during execution time . 7](#_Toc412027977)

[Task 2: Reading Input Lines 8](#_Toc412027978)

[Table 2: The content of Accumulator A during the execution time. 8](#_Toc412027979)

[Task 3: Reading DIP Switches and Writing them to LEDs 9](#_Toc412027980)

[Task 4: Using Subroutine to Implement Delay 10](#_Toc412027981)

[3. Assignment Questions 12](#_Toc412027982)

[4. Conclusions and Recommendations 15](#_Toc412027983)

[Extra Task: TASK 4 UPDATED to 10 second: 16](#_Toc412027984)

List of Figures and Tables

Table 1: The content of Accumulator Register A during execution time . 7

Table 2: The content of Accumulator A during the execution time. 8

Summary

This report illustrates the tasks that are made in third lab of microprocessor. The HCS12 assembly program, Dragon Pulse Trainer board and Code Warrior are used the do perform the tasks. Four tasks are performed successfully in this lab. In the first task, we read the data from the output port depends on the light of LEDs weather on or off. Then, we programed the code that reads input data from the dragon 12 plus board. After that, we combined the two codes of reading data from the input ports and writing them to the output ports immediately. Finally, we applied a delay on the output of the code of the third task.

Introduction

One of the features of the HCS12 microcontroller is a number of input/output pins that are used for connection with external I/O. Most microcontrollers will have at least 8 digital I/O lines. Some have 12, 32 and 96 I/O lines. A common way to describe a grouping of I/O lines is as a port, since they are not equal. There are different ports that can have different functions such as the following:

* PORTA is split between input and output.
* PORT B is output only.
* PORTC is called a bi-directional port, and can be programmed as input or output.
* PORTD is also programmable as input or output, but only 6 of its bits are useful to the programmer.
* PORTE is input only.

1.1 Aim

To introduce the students to read and write data processes from input and output ports and the use of delays while implementing loops.

1.2 Objectives

1. Understand microcontroller IO ports.
2. Configure the ports as either inputs or outputs.
3. Read and write data from input and output ports.
4. Examine DIP switches of PTH for IO programming on Dragon-12 Plus board.
5. Perform IO bit programming in HCS12 assembly language.
6. Create binary counter on Dragon-12 Plus board.
7. Download, run and test code on Dragon-12 Plus board.

2. Design and Results

## Task 1: Deriving Output Lines

In the following code segment, the CodeWarrior is now used to control outside hardware. This is exactly the practical usage of the assembly language. However, in this program, we will learn how to make the proper setting for 8 LEDs connected to port B such that they are set as outputs and controlled with respect to the entered hex number. In the following program, we will see how easily assembly language is used in such field.

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

; export symbols

XDEF Entry

ORG $4000 ;Flash ROM address for Dragon12+

Entry:

LDAA #$FF

STAA DDRB ;Make PORTB output

;PTJ1 controls the LEDs connected to PORTB

In the previous code segment, we see how the DDRB is used to set the LEDs as output controlled. Meaning that, we see the results out of it. Now, Port J is also set as output, we will see in the following code the reason behind it.

LDAA #$FF

STAA DDRJ ;Make PORTJ output

STAA DDRP ;Make PORTP output

LDAA #$00

STAA PTJ ;Turn off PTJ1 to allow the LEDs to show data

LDAA #$0F

STAA PTP ; Disable the 7-segment display

;-------Switch on LEDs connected to PORTB based on the Acc A value

LDAA #$55

STAA PORTB ;Store A into PORTB

Notice that PORT (port name) or PT(port name) is used to set the desired value. We would like to represent 55hex in the LEDs, that’s why we load the value #$55 into the LEDs. Note that PTJ is set to zero. This is a very important step, because we want to close the circuit (set to the ground) to obtain the output. The following table explains in details the situation occurring in each code segment.

## Table 1: The content of Accumulator Register A during execution time .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Instruction* | *Accumulator A* | *DDRB (0x03)* | *DDRJ(0x026A)* | *PJT* | *PORTB* | *Comment* |
| *LDAA #$FF* | $FF( 255) | \_\_ | \_\_ | \_\_ | \_\_ | Load A with #$FF |
| *STAA DDRB* | $FF( 255) | $FF( 255) | \_\_ | \_\_ | \_\_ | Store the value of A in DDRB |
| *LDAA #$FF* | $FF( 255) | \_\_ | \_\_ | \_\_ | \_\_ | Load A with #$FF |
| *STAA DDRJ* | #$FF | \_\_ | $FF( 255) | \_\_ | \_\_ | Store the value of A in DDRJ |
| *LDAA #$0* | #$00 | \_\_ | \_\_ | \_\_ | \_\_ | Load A with #$00 |
| *STAA PTJ* | #$00 | \_\_ | \_\_ | #$00 | \_\_ | Turn off PTJ by storing 0 on it |
| *LDAA #$55* | #$55 | \_\_ | \_\_ | \_\_ | \_\_ | Load A with #$55 |
| *STAA PORTB* | #$55 | \_\_ | \_\_ | \_\_ | #$55 | Make POTB the output |

## Task 2: Reading Input Lines

Moving on, a physical method is used to control the contents of DDRB. This is similar to the on/off switch button. This will control the status of the LEDs. By using DIP switches on Dragon12+ board associated to PTH and lining them to port B is the correct method to do so. The corresponding code shows the mechanism.

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

; export symbols

XDEF Entry

ORG $4000 ;Flash ROM address for Dragon12+

Entry:

LDAA #$0

STAA DDRH

LDAA PTH

----------------------------------------

ANDA #$08 ;if we want to read bit 4 of PTH only

----------------------------------------

BRA Entry

The code has been tested and the content of A, DDRH and PTH was recorded in Table 2.

## Table 2: The content of Accumulator A during the execution time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Accumulator A | PTH | DDRH | Comment |
| LDAA #$0 | #$0 | - | - | Load A with 0 |
| STAA DDRH | #$0 | - | #$0 | Switch DDRH off and make PTH as input |
| LDAA PTH | Whatever in DIP switches will appear on A, depending on the user choice | - | - | Store the value of stored in A |
| ANDA #$08 | Whatever in DIP switches will appear on A, depending on the user choice | - | #$08 | to read bit 4 of PTH |

## Task 3: Reading DIP Switches and Writing them to LEDs

In this interesting task, the previous two codes will be combined together such that the switches control the output of the LED. The related code is shown below:

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

; export symbols

XDEF Entry

ORG $4000 ;Flash ROM address for Dragon12+

Entry:

LDAA #$0

STAA DDRH

LDAA PTH

ANDA #PTH\_PTH2

LDAA #$FF

STAA DDRB ;Make PORTB output

;PTJ1 controls the LEDs connected to PORTB

LDAA #$FF

STAA DDRJ ;Make PORTJ output

STAA DDRP ;Make PORTP output

LDAA #$00

STAA PTJ ;Turn off PTJ1 to allow the LEDs to show data

LDAA #$0F

STAA PTP ; Disable the 7-segment display

;-------Switch on LEDs connected to PORTB based on the Acc A value

LDAA PTH

STAA PORTB ;Store A into PORTB

BRA Entry

The previous code has been loaded into the board for verification. To verify, the value that has been entered using DIP switches has been shown on the LEDs. The loop section of the code enabled us to continually changing the DIP switches.

## Task 4: Using Subroutine to Implement Delay

In some cases delays are required to fit the exact practical desire. to get an idea about how delays work, the previous task has been written but with a delay in the LED activation time. Given the values of R1, R2 and R3 and based on provided formula the delay is decided as follows. The formula that is used for this purpose is:

***Delay = 1 / (24 Mhz) X R1 X R2 X R3***

Depending on the values of R1, R2 and R3, an effect on the delay will occur. The following code that implements the desired operation is shown below:

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

XDEF Entry

R1 EQU $1000

R2 EQU $1001

R3 EQU $1002

Entry:

LDAA #$0

STAA DDRH

LDAA PTH

LDAA #$FF

STAA DDRB ;Make PORTB output

;PTJ1 controls the LEDs connected to PORTB

LDAA #$FF

STAA DDRJ ;Make PORTJ output

STAA DDRP ;Make PORTP output

LDAA #$00

STAA PTJ ;Turn off PTJ1 to allow the LEDs to show data

LDAA #$0F

STAA PTP ; Disable the 7-segment display

;-------Switch on LEDs connected to PORTB based on the Acc A value

LDAA PTH

JSR DELAY

STAA PORTB ;Store A into PORTB

BRA Entry

DELAY

PSHA ;Save Reg A on Stack

LDAA #100 ;Change this value to see

STAA R3 ;how fast LEDs shows data coming from PTH DIP switches

;--10 msec delay. The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board

;Freq. for Instruction Clock Cycle is 24MHz (1/2 of 48Mhz).

;(1/24MHz) x 100 Clk x240x100=10 msec. Overheads are excluded in this calculation.

L3 LDAA #100

STAA R2

L2 LDAA #240

STAA R1

L1 NOP ;1 Intruction Clk Cycle

NOP ;1

NOP ;1

DEC R1 ;4

BNE L1 ;3

DEC R2 ;Total Instr.Clk=100

BNE L2

DEC R3

BNE L3

;--------------

PULA ;Restore Reg A

RTS

;-------------------

;-------Switch on LEDs connected to PORTB based on the Acc A value

From implementation perspective, we feed inputs by using DIP switches and the output on the LEDs was shown after a period of time. In addition, when we enter another value the delay becomes more observed when the LEDs switch to the new value that has been entered.

# 3. Assignment Questions

***1. Create a program that implements a binary counter. Use the delay subroutine given in Task-4 to display the counting sequence on the LEDs connected to PORT B.***

***Answer:***

The following mechanism is followed, where the code is modified by incrementing the value of register-B and storeing it in PORTB which is included in a loop.

ABSENTRY Entry ; for absolute assembly: mark this as application entry point

INCLUDE 'mc9s12dg256.inc'

R1 EQU $1000

R2 EQU $1001

R3 EQU $1002

ORG $4000 ;Flash ROM address for Dragon12+

Entry:

LDAA #$FF

STAA DDRB ;Make PORTB output

STAA DDRJ ;Make PORTJ output, (Needed by Dragon12+)

LDAA #$0

STAA DDRH ;PTH as Input

STAA PTJ ;Turn off PTJ1 to allow the LEDs on PORTB ;to show data (Needed by Dragon12+)

;PTJ1 controls the LEDs connected to PORTB (For Dragon12+ ONLY)

CLRB

;-------Get data from DIP switches connected to PORTH

LOOP1 ; LDAA PTH

INCB

BSR DELAY

STAB PORTB

BRA LOOP1

DELAY:

PSHA ;Save Reg A on Stack

LDAA #120 ;Change this value to see

STAA R3 ;how fast LEDs shows data coming from PTH DIP switches

;--10 msec delay. The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board

;Freq. for Instruction Clock Cycle is 24MHz (1/2 of 48Mhz).

;(1/24MHz) x 10 Clk x240x100=10 msec. Overheads are excluded in this calculation.

L3 LDAA #100

STAA R2

L2 LDAA #240

STAA R1

L1 NOP ;1 Intruction Clk Cycle

NOP ;1

NOP ;1

DEC R1 ;4

BNE L1 ;3

DEC R2 ;Total Instr.Clk=10

BNE L2

DEC R3

BNE L3

;--------------

PULA ;Restore Reg A

RTS

;-------------------

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Interrupt Vectors \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ORG $FFFE

DC.W Entry ; Reset Vector

***2. Create a program that flash all the LED’s with a delay of 0.1 sec in between.***

;export symbols

XDEF Entry

DDR: EQU $0002

PTA: EQU $0000

R1: EQU $1000

R2: EQU $1001

R3: EQU $1002

Entry:

bset DDR,%11111111

bset PTA,%11111111

BSR DELAY

bclr PTA,%00000001

BSR DELAY

bclr PTA,%00000010

BSR DELAY

bclr PTA,%00000100

BSR DELAY

bclr PTA,%00001000

BSR DELAY

bclr PTA,%00010000

BSR DELAY

bclr PTA,%00100000

BSR DELAY

bclr PTA,%01000000

BSR DELAY

bclr PTA,%10000000

DELAY

PSHA ;Save Reg A on Stack

LDAA #100 ;Change this value to see

STAA R3 ;how fast LEDs shows data coming from PTH DIP switches

;--10 msec delay. The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board

;Freq. for Instruction Clock Cycle is 24MHz (1/2 of 48Mhz).

;(1/24MHz) x 10 Clk x240x100=10 msec. Overheads are excluded in this calculation.

L3 LDAA #100

STAA R2

L2 LDAA #240

STAA R1

L1 NOP ;1 Intruction Clk Cycle

NOP ;1

NOP ;1

DEC R1 ;4

BNE L1 ;3

DEC R2 ;Total Instr.Clk=10

BNE L2

DEC R3

BNE L3

PULA ;Restore Reg A

RTS

***3. Re-examine the toggle program in assignment question 2 which flashes the LEDs PORTB with 0.1 sec delay. Now, modify that program to get the byte of data from PTH switches and give it to R3 register of the DELAY loop. Run the program to show how you can set the time delay size using the PTH switches.***

***Answer:***

For this question, there are 8 DIP switches and they are going to be used in a way that each switch corresponds to a certain delay level. The code that implements this behavior is shown below:

ABSENTRY Entry ; for absolute assembly: mark this as application entry point

INCLUDE 'mc9s12dg256.inc'

R1 EQU $1000

R2 EQU $1001

R3 EQU $1002

ORG $4000 ;Flash ROM address for Dragon12+

Entry:

LDAA #$FF

STAA DDRB ;Make PORTB output

STAA DDRJ ;Make PORTJ output, (Needed by Dragon12+)

LDAA #$0

STAA DDRH ;PTH as Input

STAA PTJ ;Turn off PTJ1 to allow the LEDs on PORTB ;to show data (Needed by Dragon12+)

;PTJ1 controls the LEDs connected to PORTB (For Dragon12+ ONLY)

;-------Get data from DIP switches connected to PORTH

LOOP1 BSR DELAY

LDAA #$55

STAA PORTB

BSR DELAY

LDAA #$AA ;toggling the sitched ON LEDs.

STAA PORTB

BRA LOOP1

DELAY:

PSHA ;Save Reg A on Stack

LDAA #100 ;Change this value to see

LDAB PTH

STAB R3 ;how fast LEDs shows data coming from PTH DIP switches

;--10 msec delay. The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board

;Freq. for Instruction Clock Cycle is 24MHz (1/2 of 48Mhz).

;(1/24MHz) x 10 Clk x240x100=10 msec. Overheads are excluded in this calculation.

L3 LDAA #100

STAA R2

L2 LDAA #240

STAA R1

L1 NOP ;1 Intruction Clk Cycle

NOP ;1

NOP ;1

DEC R1 ;4

BNE L1 ;3

DEC R2 ;Total Instr.Clk=10

BNE L2

DEC R3

BNE L3

;--------------

PULA ;Restore Reg A

RTS

;-------------------

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Interrupt Vectors \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ORG $FFFE

DC.W Entry ; Reset Vector

4. Conclusions and Recommendations

All the tasks in this experiment clarified the use of dip switches and LEDs on the board and helped understanding them. Firstly, in Task 1 it has been explained the use of PORTB to control the LEDs, and how to turn the PJT off using accumulator A. Secondly, in Task 2 the code of Dip switches was provided from the previous task and we added PTH to control Dip switches. Thirdly, combining the codes of both Task1 and Task2 show the process of reading Dip switches and writes them to the LEDs. And, in order to display the output continuously on the LEDs we used a loop. Eventually, we implemented time delay to the displayed output and whenever there is a delay in input there is a time delay in output.

To sum up, all the tasks were successfully applied. Full understanding of the input/output lines was achieved simply. The group members were able to download and test the code on the dragon 12+. Reading data from IO ports have been done. The time delay effect on LEDs were shown and understood. No problems or risks faced in this experiment. Also, writing a code and check it by the hardware become easier after paying attention to the lab instructor which helped us to overcome some errors.

Extra Task: TASK 4 UPDATED to 10 second:

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

XDEF Entry

R1 EQU $1000

R2 EQU $1001

R3 EQU $1002

Entry:

LDAA #$0

STAA DDRH

LDAA PTH

LDAA #$FF

STAA DDRB ;Make PORTB output

;PTJ1 controls the LEDs connected to PORTB

LDAA #$FF

STAA DDRJ ;Make PORTJ output

STAA DDRP ;Make PORTP output

LDAA #$00

STAA PTJ ;Turn off PTJ1 to allow the LEDs to show data

LDAA #$0F

STAA PTP ; Disable the 7-segment display

;-------Switch on LEDs connected to PORTB based on the Acc A value

LDAA PTH

JSR DELAY

STAA PORTB ;Store A into PORTB

BRA Entry

DELAY

PSHA ;Save Reg A on Stack

LDAA #255 ;Change this value to see

STAA R3 ;how fast LEDs shows data coming from PTH DIP switches

;--10 msec delay. The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board

;Freq. for Instruction Clock Cycle is 24MHz (1/2 of 48Mhz).

;(1/24MHz) x 100 Clk x240x100=10 msec. Overheads are excluded in this calculation.

L3 LDAA #255

STAA R2

L2 LDAA #255

STAA R1

L1 NOP ;1 Intruction Clk Cycle

NOP ;1

NOP ;1

NOP ;1

NOP ;1

NOP ;1

NOP ;1

NOP ;1

DEC R1 ;4

BNE L1 ;3

DEC R2 ;Total Instr.Clk=100

BNE L2

DEC R3

BNE L3

;--------------

PULA ;Restore Reg A

RTS

;-------------------

;-------Switch on LEDs connected to PORTB based on the Acc A value