Khalifa University of Science, Technology and Research

Electronic Engineering Department

ELCE333: Microprocessor Systems Laboratory

Laboratory Experiment No. 3

HCS12 Input and Output Ports

Laboratory Partners

Name: Afra Bin Fares,ID#:100033139

Name: Alya Humaid AlAlili ,ID#100037087

Name:Anoud Alshamsi ,ID#100035514

Laboratory Instructor:

Mohammed Ali Saif Al Zaabi

Mahmoud Khonji

Spring 2015

Table of Contents

List of Illustrations

Abstract

The Laboratory experiment 3 is presenting how data is read and written from the input and output ports and also how delays can be implemented in the board using loop instructions.The main goal of this lab is to be familiar with the input/output ports of the board, and it is important to know that LEDs are controlled by ports which are B, J , P and the Dip Switches where the push buttons controlled by port H.

This experiment is divided into four tasks.The first task is about editing the program in way we control the value of 8 bit accumulator A to obtain the output which means we write a program to control the hardware. The second task is to set the switches as inputs and result need to displayed in accumulator A.Whereas the third task is about using combination program of task 1 and task 2 in order to read the conditions of the DIP switches on the board . Finally, task 4 is about implementing a Delay .

1. *Introduction*

An important feature of any microcontroller is a number of input/output pins utilized for connection with peripherals. for the MC9S12DP256B micro controller it has seven 8 bit ports, that can be used for simple parallel I/O if they aren't being used. Parallel I/O is when transmitted binary data of information are exchanged in the meantime over particular wires. The processor has the data at the register which then show up at the port pins during parallel output.

The term serial refers to data sent through a single wire the bits are sent one after the other and takes place asynchronously, so no clock is needed. the data may be sent at random intervals.thats why each character is preceded by a START bit and followed by a STOP bit. These control bits, which are needed for serial transmission, waste 20% of the bandwidth.

Parallel I/O is the transmission of binary data where several bits of data are transferred at the same time over separate wires. the processor writes the data into a output port, and the data appears at the port pins. parallel output is suitable for relays and the processor reads the data on the pins of an input port. the figure below demonstrates the difference between parallel I/O and serial I/O

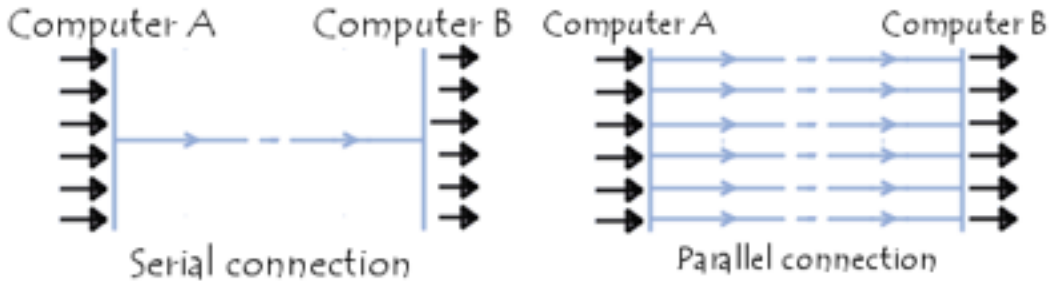

Figure 1: serial and parallel connection

On the Dragon12-Plus boards,port B is used as a parallel output port for the LED display and port H is used as a parallel input port for the DIP switches. When used as an 8 bit parallel port,the pins of each port are programmed as both input or output pins. The port direction can be changed at any time. The PTIX register is used to read data of an input port, while the PTX register is the data register used to write data to an output port

1.1 Aim:

This experiment's aim is to be more familiar in how data is read and written from the input and output ports on a Dragon12+ Board, along with implementing delays using loops.

1.2 Objectives:

1. Recognize the input and output ports of a microcontroller.
2. Construct the ports either as inputs or output.
3. Read and write data from input and output ports.
4. Observe the DIP switches of PTH for I/O programming on Dragon12+ Board.
5. Perform I/O bit programming in HCS12 Assembly language.
6. Generate binary counter on Dragon12+ Board.
7. Download, run, and test code on a Dragon12+ Board.

2. Design, Results and Analysis

In this part of the report the tasks performed in the experiment will be discussed by explaining the steps of these tasks, listing, explaining and analyzing the results obtained from this experiment. The experiment is basically divided into four tasks. All four tasks will be performed using code warrior software and the Dragon Plus Trainer Board.

2.1. TASK-1: Deriving Output Lines

In this part of the experiment a program will be used to switch the LED's on or off according to the state bits in accumulator A. the instructions and steps in lab script 3 were followed and the following program was assembled and downloaded to the microcontroller (comments were added to explain the code; the values chosen for each port were explained in lab script 3):

```
INCLUDE 'derivative.inc'
 XDEF Entry, _Startup, main
XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack
MY_EXTENDED_RAM: SECTION
Counter     ds.w 1
FiboRes     ds.w 1
MyCode:     SECTION
main:
_Startup:
  ORG $4000 ;Flash ROM address for Dragon12+
Entry:
  LDAA #$FF ; load the immediate value $FF to accumulator A
  STAA DDRB ; load the value of accumulator A into DDRB to Make PORTB output
  ;PTJ1 controls the LEDs connected to PORTB
  LDAA #$FF; load the immediate value $FF to accumulator A
  STAA DDRJ ; load the value of accumulator A into DDRJ to Make PORTJ output
  STAA DDRP ; load the value of accumulator A into DDRP to Make PORTP output
  LDAA #$00; load the immediate value $00 to accumulator A
  STAA PTJ ; load the value of accumulator A to Turn off PTJ1 to allow the LEDs to show data
  LDAA #$0F;  load the immediate value $0F to accumulator A
  STAA PTP ; load the value of accumulator A to Disable the 7-segment display
  ;-------Switch on LEDs connected to PORTB based on the Acc A value
  LDAA #$55;  oad the immediate value $0F to accumulator A
  STAA PORTB ;Store A into PORTB (the leds)
  BRA Entry
```

Observations:

1) As the program was run, and the line (STAA PORTB ;) was executed; the LED were emitted and the value appears on it which is 01010101 reflects the bits of accumulator A which is $55 (01010101 in binary).
2) When the value of accumulator A was changed from $55 to $FF (11111111 in binary) , all LED turned on.
3) When the line STAA DDRP was commented out, the 7 segment display started to work, the reason for that is Port B is associated for both LED and 7 segment display, and if we don't specify the DDRP as an out put (the pin associated with the 7 segment) it will not be able to control the 7 segment display (whether to turn it on or off) and there will be no use of adding STAA PTP to the code to turn off the 7 segment.

2.2. TASK-2: Reading Input Lines

In this part of the experiment it is required to 1) follow the instructions and steps described in lab script 3 and 2) assemble and download the following program into the microcontroller. this program basically reads the value asserted in the switches and store this value into accumulator A(comments were added to explain the code; the values chosen for each port were explained in lab script 3):

```
INCLUDE 'derivative.inc'
 XDEF Entry, _Startup, main
XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack
MY_EXTENDED_RAM: SECTION
Counter     ds.w 1
FiboRes     ds.w 1
MyCode:     SECTION
main:
_Startup:
  ORG $4000 ;Flash ROM address for Dragon12+
Entry:
LDAA #$0; load the immediate value $FF to accumulator A
 STAA DDRH; load the value of accumulator A into DDRH to assign it as an input
 LDAA PTH; load the data register PTH value (the switches connection) into Accumulator A
BRA Entry
```

Observations:

1) As the program was run, and the line (LDAA PTH) was executed; the value the switches appears in accumulator A register.

2) When the program was modified from (LDAA PTH;) to :

LDAA PTH

ANDA #$04

The value of accumulator A represented the logic operation (ANDA #$04) , which is basically an And operation between the value 00000100 and the switches value.

2.3. TASK-3: Reading DIP Switches and Writing them to LEDs

In this part of the experiment it was required a program that will read the states of the DIP switches of the Dragon Plus Trainer board and reflect them immediately to the LED continuously. The programs in tasks 1 and 2 were combined to perform this task as the following:

```
INCLUDE 'derivative.inc'

          XDEF Entry, _Startup, main
          XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack
MY_EXTENDED_RAM: SECTION
Counter     ds.w 1
FiboRes     ds.w 1
MyCode:     SECTION
main:
_Startup:
  ORG $4000 ;Flash ROM address for Dragon12+
Entry:
  LDAA #$FF ; load the immediate value $FF to accumulator A
  STAA DDRB ;Make PORTB output ; load the value of accumulator A into DDRB to Make PORTB output
  ;PTJ1 controls the LEDs connected to PORTB
  LDAA #$FF ; load the immediate value $FF to accumulator A
  STAA DDRJ ; load the value of accumulator A into DDRJ to Make PORTJ output
  STAA DDRP ; load the value of accumulator A into DDRP to Make PORTP output
  LDAA #$00 ; load the immediate value $00 to accumulator A
  STAA PTJ ; load the value of accumulator A to Turn off PTJ1 to allow the LEDs to show data
  STAA DDRH ; load the value of accumulator A into DDRH to assign it as an input
  LDAA #$0F ;  load the immediate value $0F to accumulator A
  STAA PTP ;  load the value of accumulator A to Disable the 7-segment display
  ;-------Switch on LEDs connected to PORTB based on the Acc A value
  LDAA PTH ; load the value of the switches into accumulator A
  STAA PORTB ;Store the value of Accumulator A into PORTB
  ;-------------------
  BRA Entry
```

This program was assembled, downloaded, run and performed successfully. Where the values of the LED represented the values inserted in the switches.

2.4. TASK-4: Using Subroutine to Implement Delay

In this task it is required to introduce a delay subroutine into the previously used program in task 3. The code to implement a delay was provided in the lab script where it performs the following formula:

$$Delay = \frac{1}{24MHz} R_1 \times R_2 \times R_3$$

Based on the values of R1, R2, R3 the delay will be decided.

```
  ORG $4000 ;Flash ROM address for Dragon12+
Entry:
  LDAA #$FF ; load the immediate value $FF to accumulator A
  STAA DDRB ;Make PORTB output ; load the value of accumulator A into DDRB to Make PORTB output
  ;PTJ1 controls the LEDs connected to PORTB
  LDAA #$FF ; load the immediate value $FF to accumulator A
  STAA DDRJ ; load the value of accumulator A into DDRJ to Make PORTJ output
  STAA DDRP ; load the value of accumulator A into DDRP to Make PORTP output
  LDAA #$00 ; load the immediate value $00 to accumulator A
  STAA PTJ ; load the value of accumulator A to Turn off PTJ1 to allow the LEDs to show data
  STAA DDRH ; load the value of accumulator A into DDRH to assign it as an input
  LDAA #$0F ;  load the immediate value $0F to accumulator A
  STAA PTP ;  load the value of accumulator A to Disable the 7-segment display
  ;-------Switch on LEDs connected to PORTB based on the Acc A value
  LDAA PTH ; load the value of the switches into accumulator A
JSR DELAY; jump to subroutine DELAY
  STAA PORTB ;Store the value of Accumulator A into PORTB
  ;--------------------
  BRA Entry
DELAY ; Delay function
  PSHA ;Save the value of Reg A on Stack
  LDAA #10 ; load the immediate value 10 to accumulator A
  STAA R3 ; load the value of accumulator A into R3
L3 LDAA #100; load the immediate value 100 to accumulator A
    STAA R2; load the value of accumulator A into R2
L2 LDAA #240; load the immediate value 240 to accumulator A
  STAA R1; load the value of accumulator A into 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
```

Observations:

1) The delay introduced by the function was (1/24MHz) x 10 Clk x240x1000=10 msec. Overheads are excluded in this calculation.

2) There were 10 more Total of Instruction .Clk cycles introduced by No operation NOP and some condition operations. This introduced more delay. 10 msec *10 instruction clk cyc les

3) The values of R1,R2 and R3 can be increased up to 255 as a maximum value

4) To add more delays we can add more NOP instructions in the code

3. *Conclusion and Recommendations*

In the Laboratory experiment 3, we looked at four tasks on purpose that's to understand how the Dragon12 board works.Also, we learnt a new technique that's to make a use of ports of the micro controller harder such as we used a Dip switches as input whereas the LEDs light as output. Moreover, we understood how to receive an input form the Dragon12 board and store them in the registers. Not only that, also the delay implemented for the LEDs in this Laboratory experiment. Overall,we are able to achieve all the goals of this Laboratory experiment and without facing any problems .

4. 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 PORTB.

```
Entry:
  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
  STAA DDRH
  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

LOOP
    INCA
    BRA DELAY
    STAA PORTB
    BRA LOOP
```

A label is created in order to implement the binary counter. A binary counter can be constructed from J-K flip-flops by taking the output of one cell to the clock input of the next. This produces a binary number equal to the number of cycles of the input clock signal. first it increment A then goes through the delay branch and then store A to Port B. every time it goes through the loop A is incremented to know how many times the code ran.

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

```
Entry:
  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
  STAA DDRH
  LDAA #$01
  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

LP
    LSLA
   JSR  DELAY
    STAA PORTB
    BRA LP
```

In order to have a delay of 0.1, A is assigneed to the value 01. also in the same loop as before an instruction is added which shift A to the left (LSLA) in order to cause the 0.1 delay in between.

3) Re-examine the toggle program in assignment question 2 which flashes the LEDs of 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.

```
Entry:
  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
  STAA DDRH
  LDAA #$01
  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

LP
   LDAA PTH
   LDAB PTH
   JSR    DELAY
   STAA PORTB
   BRA LP
```

here the loop loads the accumulator A and B with the PTH switch.