Khalifa University of Science, Technology and Research
Electronic Engineering Department

Microprocessor Systems Laboratory

ELCE333

Laboratory Report Experiment No.3

HCS12 INPUT AND OUTPUT PORTS

Lab Partners

Leena ElNeel	100037083
Muna Darweesh	100035522
Shamsah AlNabooda	100036984

Date Experiment Performed: 11/2/2015
Date Lab Report Submitted: 18/2/2015

Lab Instructors:

Mohammed Ali Saif Al Zaabi

Mahmoud Khonji

Spring 2015

Table of Contents

List of Figures and Tables

List of Figures

Summary

This report discusses two main microprocessor concepts. First, it aims to teach the students how to read and write data from input and output ports. In addition, this lab introduced the idea of delay and how it works in microcontrollers. This lab is made from four main tasks, the first deals with deriving output lines, the second focuses on reading input lines, the third is about writing read DIP switches to LEDs and finally implementing delay.

1. Introduction

Ports are widely used in microprocessor labs and they have several types and functions. Parallel ports are general input/output ports; HCS12 memory addressing modes are to be used when we want to read or write to such registers. The registers are found on the dragon12 plus USB board and are labeled such that register Port B is relative to PORTB label on the board. Moving on to the LEDs, port B gives a seven segment anode and when PTP=0x0F the segments are disabled. Figure 1 below shows how ports B, J and P are connected to the LEDs.

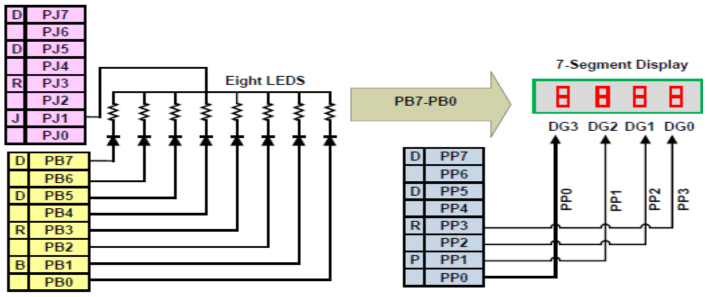


Figure 1: Figure showing how LEDs are connected

DIP switches are the red 8-DIP switches on the dragon board and are connected to PTH bits. DIP switches are active high inputs. In addition, there are four push buttons on the board which are active low inputs.

Aim:

To introduce the students to the read and write data from the input and output ports and how delays can be implemented using loops.

Objectives:

We should be able to:

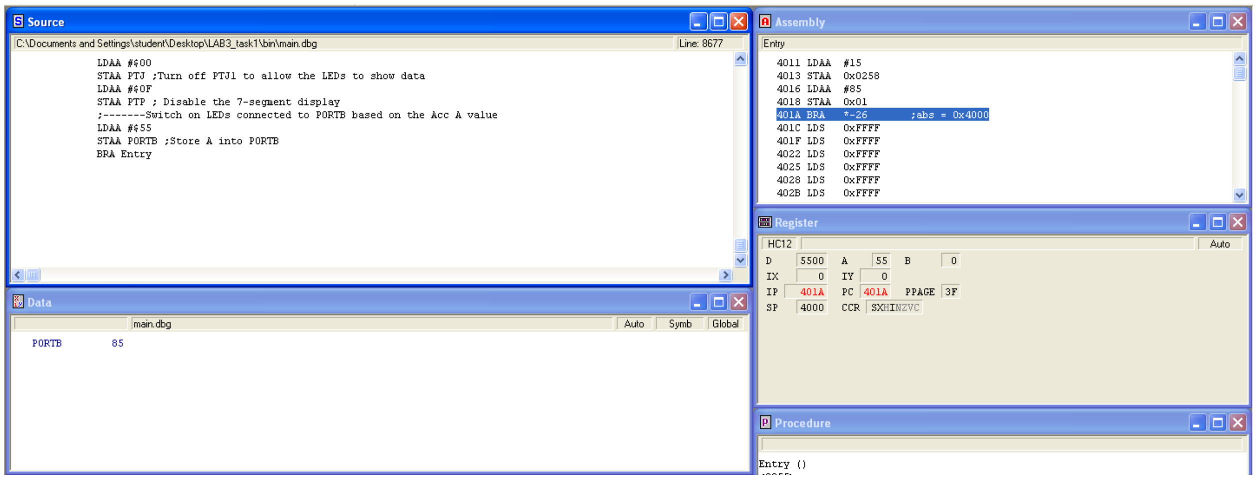
- Understand the microcontroller IO ports and configure the ports as inputs or output.
- Read and write data from input and output ports.
- To examine the DIP switches of PTH for I/O programming on Dragon12+ Board.
- To do I/O bit programming in HCS12 Assembly language.
- To create binary counter on Dragon12+ Board.
- Download, run, and test code on a Dragon12+ Board.

2. Design and Results

Task 1: Deriving Output Lines

In the first task, we are asked to run a code that will control the state of the LEDs. We used the HCS12 serial monitor mode and added PORTB to the data window. We checked that it holds the write value for port B; changing the value of accumulator A changes the state of the LEDs. The code we ran is shown below:

```
            ORG $4000 ;Flash ROM address for Dragon12+
    Entry:
            LDAA #$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
            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
            BRA Entrv
```



```
C:\Documents and Settings\student\Desktop\LAB3_task1\bin\main.dbg    Line: 8677

            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
            BRA Entry
```

```
main.dbg

  PORTB        85
```

```
Entry

  4011 LDAA   #15
  4013 STAA   0x0258
  4016 LDAA   #85
  4018 STAA   0x01
  401A BRA    *-26        ;abs = 0x4000
  401C LDS    0xFFFF
  401F LDS    0xFFFF
  4022 LDS    0xFFFF
  4025 LDS    0xFFFF
  4028 LDS    0xFFFF
  402B LDS    0xFFFF
```

```
HC12                                   Auto
D    5500   A     55   B        0
IX      0   IY     0
IP   401A   PC  401A   PPAGE   3F
SP   4000   CCR SXHINZVC
```

```
Entry ()
```

Figure 2: Results for task 1

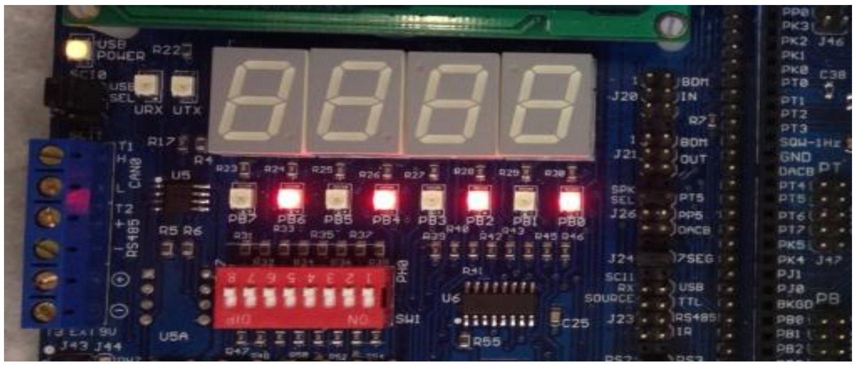
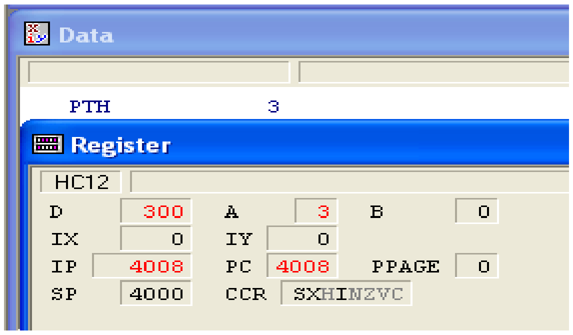


Figure 3: Results for task 1 on board

Task2: Reading Input Lines

In the second task, we are asked to run a code that will control the state of the LEDs by reading input lines. We used the HCS12 serial monitor mode and added PTH to the data window. We changed the DIP switch positions to check that it holds the right value for accumulator A and PTH in the data window. We modified the given code in order to enable it to read bit 4 of PTH by the masking technique. The code we ran is shown in the next page:

```
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'
; export symbols
        XDEF Entry
        ; symbol defined by the linker for the end of the stack
        ORG $4000 ;Flash ROM address for Dragon12+
Entry:
        LDAA #$0
        STAA DDRH
        LDAA PTH
        ANDA #%00010000 ;(or #%00000100 or #$04 – to check PH4)
        BRA Entry
```



```
Data
  PTH                3
Register
HC12
D      300    A      3    B      0
IX       0    IY     0
IP    4008    PC  4008    PPAGE  0
SP    4000    CCR SXHINZVC
```

Figure 4: Results for task 2

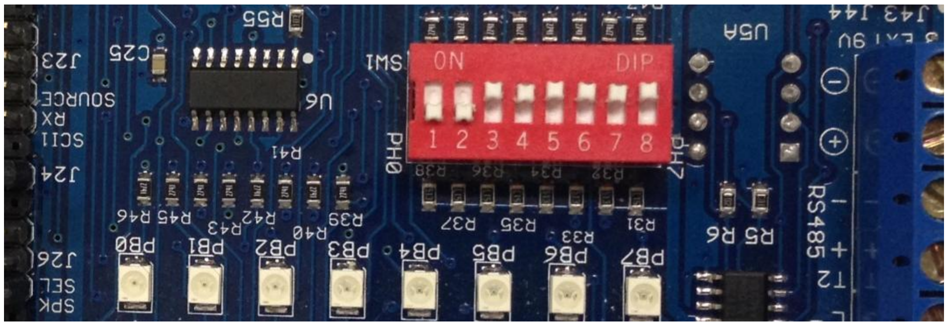


Figure 5: DIP switch position on board

Task 3: Reading DIP switches and writing them to the LED

In this task we were required to write an assembly language code to take inputs from DIP switch and reflect them directly into LEDs. In order to do that, codes from task 1 (deriving output line) and task 2 (reading input lines) had to be combined. The new code firsts start with the three basic steps needed to switch LEDs connected to port B on according to the value in acc. A, those steps are: 1) define direction inputs and outputs using DDRx, 2)define port B (LEDs) as an output, 3) assign PTJ1 to zero to allow the LEDs to show data.

After that, since the initial status for the LEDs should be off, acc. A was loaded with #$00. Also port H was assigned to be an input. The role of acc. A here was an intermediate that read the value form port H and feed it back to port B. The code in next page shows the full program for this task.

```
           ORG $4000 ;
Entry:
           LDAA #$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
          LDAA #$0F
          ;-------Switch on LEDs connected to PORTB based on the Acc A
value
          LDAA #$00
          STAA DDRH
          LDAA PTH
          STAA PORTB ;Store A into PORTB
          BRA Entry
```

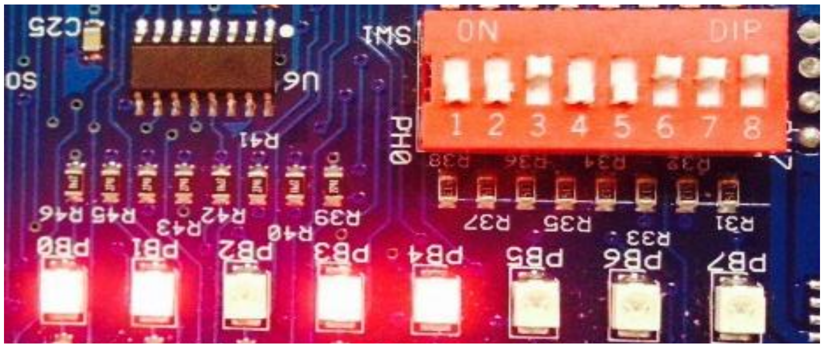


Figure 6: Reading input from DIP and reflect their output on LEDs

Task 4: Using Subroutine to Implement Delay

In this task, it is required to modify the program that is used in task 3 by having delays. Subroutines are for implementing delays. There are three values which are R_1, R_2 and R_3 will affect the delay based on a formula the delay is computed. The formula is as the following:

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

The Serial Monitor works at speed of 48MHz with XTAL=8MHz on Dragon12+ board and R_3 shows how fast LEDs shows data coming from PTH DIP switches. The instruction jump to subroutine (JSR) is used to implement the delay between the LEDs. The instruction contains the effective address of the subroutine to which the program must branch. The instruction follows the branch is called the return address and it is pushed into the stack.

The flowchart for the delay subroutine is the following.

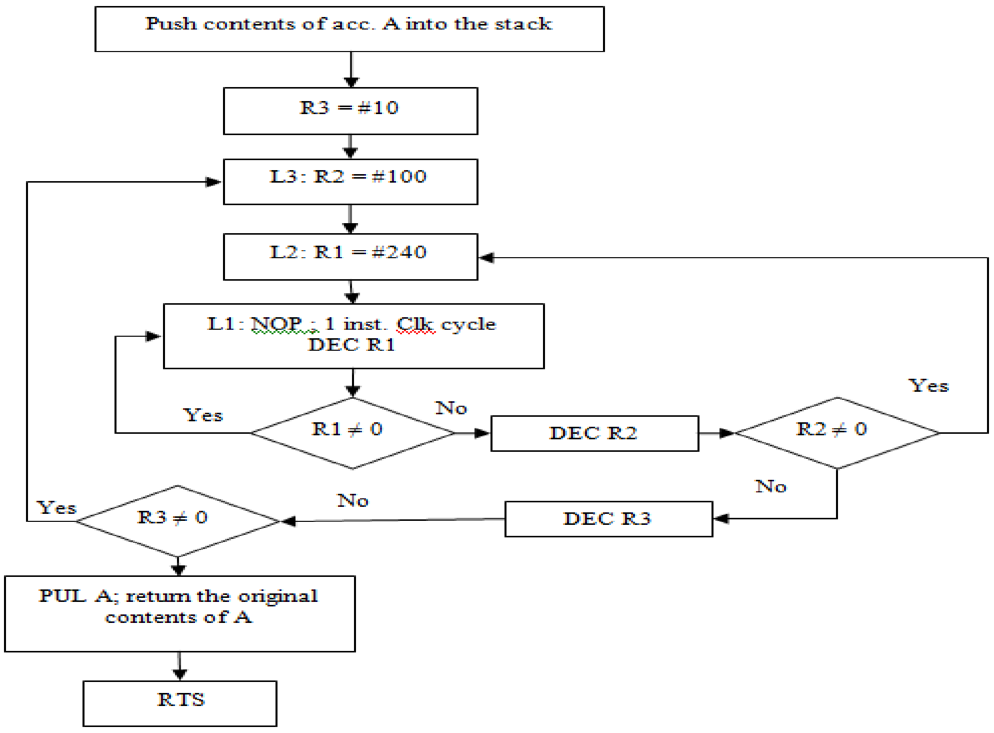

Push contents of acc. A into the stack
R3 = #10
L3: R2 = #100
L2: R1 = #240
L1: NOP ; 1 inst. Clk cycle
DEC R1
R1 ≠ 0
Yes
No
DEC R2
R2 ≠ 0
Yes
No
DEC R3
R3 ≠ 0
Yes
No
PUL A; return the original contents of A
RTS

Figure 7: Flow chart for task 4

The inputs are feed in DIP switches inputs and the output on the LEDs. By executing the code on the board it was clear that a delay was created whenever we set one of the dip switches on as output the LED will take time be ON and then OFF. For example, according to our code, the delay is computed using the equation above, where

$$\text{Delay} = \frac{1}{24MHz} \times R_1 \times R_2 \times R_3 = \frac{1}{24MHz} \times 240 \times 100 \times 10 = 0.01$$

In this case the delay between the LEDs is not noticeable. To make this delay more obvious, we can change the value R_3 . For instance to make a delay of 1 sec , for example the value of R_3 is changed to 1000 or change any other R values so the delay is 1sec.

The code of the modified program with the required subroutine, actually, with JSR command is as the following.

```
; export symbols
          XDEF Entry
          XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack
R1          EQU $1000
R2          EQU $1001
R3          EQU $1002
          ORG $4000 ;Flash ROM address for Dragon12+
Entry:
          LDAA #$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
          LDAA #$0F
          ;STAA PTP ; Disable the 7-segment display
          ;-------Switch on LEDs connected to PORTB based on the Acc A value
          LDAA #$0
          STAA DDRH
          LDAA PTH
          JSR DELAY
          STAA PORTB ;Store A into PORTB
          BRA Entry
DELAY
          PSHA ;Save Reg A on Stack
          LDAA #10 ;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. 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.

In order to do that we added the following code to task 4 code:

```
             LDAA #$00 ; make the LEDs initially all off
             STAA PORTB ; Store A into PORT

loop        INCA      ; increment A by one for the binary counter
            BRA DELAY  ; delay before displaying it on LEDs
            STAA PORTB
            BRA loop
```

The code above shows that the LEDs were initially all off as acc. A value that is been fed to port B is #$00. In the loop, the value of acc. A was incremented before loading it to port B (LEDs) at each iteration. A delay between those two stages so the output can be observed by human eye. This was used to set the initial values to the ports and as an intermediate to read inputs and feed it back to the output.

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

We used the LSLA command in order for us to move to the next LED. Each led represents a multiple of 2 number from 1- 128. A loop was required as well to keep the movement between the LEDs each time we multiply A with 2.

```
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
      LDAA #$0F
      STAA PTP ; Disable the 7-segment display
 ;-------Switch on LEDs connected to PORTB based on the Acc A
value
      LDAA #$01
      STAA PORTB ;Store A into PORT
      BRA DELAY
loop  LSLA
      STAA PORTB
      BRA DELAY
```

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.

```
          ORG $4000 ;
  Entry:
           LDAA #$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
          LDAA #$0F
          ;-------Switch on LEDs connected to PORTB based on the Acc A value
          LDAA #$00
          STAA DDRH
          LDAA PTH
          BRA Entry

DELAY
          PSHA ;Save Reg A on Stack
          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
```

4. Conclusion

This third laboratory is different than the previous two in which there were more hands-on work than coding. In this lab we got to work on the Dragon 12 board and its various Ports, we were also able by the end of the lab to distinguish between the various ports and their usage. However, ports B and H had the biggest share in implementations in which they were used during this lab tasks in the I/O processes. The first task shows how port B controls the LEDs to show the output whereas the second task shows how port H that controls the DIP switches is used to feed an input to the program. The third task was a combination of the first two tasks in which a system was developed for reading an input from DIP switches and feed it back to the LEDs as an output. In all those three tasks acc. A was used to set the initial values to the ports and as an intermediate to read inputs and feed it back to the output. The forth task discusses the problem of delay in which it sometimes can be essential in some applications and for human eye observation. A certain code was given to be implemented and the amount of the delay depends on the values of R1, R2 and R3 that is used in the following equation to calculate delay in seconds:

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

And by changing the value of R3 with the rest remaining constant, various durations of delay were observed.

By the end of this lab we are able to say that we have achieved the aims and objective and have the ability to use ports of MCU to design and implement various goals systems.