# Microprocessor Systems Laboratory (ELCE333)

## Laboratory Experiment No.3

## MARKINGSHEET

**Student Names and ID No's:** **Amal AlQasimi 100036830**
**Shaikha AlBesher 100037064**

**Laboratory Section:** **Wednesday** **Experiment Date:** **February 11, 2015**

| **No.** | **Criteria** | **Description** | **Weight %** | **Mark** | **Comments** |
|---|---|---|---|---|---|
| 1 | Pre-lab | A mark will be allocated to each student that reflects his preparations for the lab. | 20 | | |
| 2 | Performance In the lab | A mark will be allocated to each student individually that reflects his performance in the lab | 10 | | |
| 3 | Results and Analysis | Documentation and analysis of the results for each task performed in the lab | 30 | | |
| 4 | Summary/ Conclusions | Conclusions for each task performed in the lab | 10 | | |
| 5 | Assignment Questions | Answers to assignment questions | 20 | | |
| 6 | Report Presentation | Overall presentation of the report including proper layout and clarity of figures, tables, and graphs. Correct use of English language. | 10 | | |
| | **Total** | | **100** | | |

ELCE 333: Microprocessor Systems Laboratory

# Lab Report- Experiment No.3

## Experiment Title: HCS12 Input and Output Ports

Completed by:

Amal Al Qasimi (100036830)

Shaikha Al Besher (100037064)

Lab instructors:

Mahmoud Khonji

Mohammed Al Zaabi

**Spring 2015**

## Table of Contents

## Summary

The following lab report will include the details of the third Microprocessor System's Laboratory lab session. It will first start off by stating the aims and objectives of the tasks assigned to us throughout the lab session. Following the aims and objectives, the report will discuss the purpose of each task separately with the results obtained upon the completion of the task. Based on the observations of the tasks completed, the analysis and interpretation section will include the explanation of why we obtained the results we did. Finally the report will end with a brief conclusion. The assignment questions given to us as part of the lab are answered after the lab repot.

## 1. Introduction

In our labs we have been using the M68HCS12 microcontroller. The M68HCS12 microcontroller has many components like a fully integrated suite of I/O capabilities. The I/O capabilities include parallel and serial I/O, analog input, and timer functions. The various I/O pins in the microcontroller share dual purpose functions and all of them are done using set of 1024 control registers. The control registers are initially located at memory locations $0000-$03FF. Other important components that we familiarized with are parallel ports, light-emitting diode, dip-switches, and push buttons. Parallel ports in the microcontroller operate in either output or input and are available as general-purpose I/O ports. In addition, each port has a specific usage and there are nearly nine parallel ports which are: Port A, B, E, AD, H, J, K, P and port T. This laboratory experiment focused specifically on the use of LEDs controlled by ports B, J, P, the Dip Switches, and finally the push buttons controlled by port H. The light emitting diodes that exist in the HCS12 board are actually the LEDs (eight light-emitting diodes) and 7-segment anodes that are both controlled by port B. The ports of the HCS12 board will only be turned on and used under some required conditions and those conditions will be shown in the tasks of this laboratory experiment.

## 2. Aims and Objectives:

The following are the aims and objectives of the laboratory experiment:

***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:***

1. Understand the microcontroller IO ports.
2. Configure the ports as inputs or output.
3. Read and write data from input and output ports.
4. To examine the DIP switches of PTH for I/O programming on Dragon12+ Board.
5. To do I/O bit programming in HCS12 Assembly language.
6. To create binary counter on Dragon12+ Board.

7. Download, run, and test code on a Dragon12+ Board.

## 3. Lab tasks

### *TASK-1: Deriving Output Lines*

In the first task we were responsible of copying the assembly code below and observing the changes on the microcontroller's LEDs by running it in "HCS12 Serial Monitor" mode. The code is a program that allows users to switch the LED's ON and OFF according to different states of the 8 bits in accumulator A. The 8 bits in Accumulator A can store values in Hexadecimal from 00 to FF which are equivalent to 0000 0000 to 1111 1111 in binary.

The assembly code used to perform the described operations is:

```
;**************************************************************
;* This stationery serves as the framework for a              *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the                *
;* Freescale CodeWarrior for the HC12 Program directory       *
;**************************************************************
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'

; export symbols
        XDEF Entry, _Startup, main
        ; we use export 'Entry' as symbol. This allows us to
        ; reference 'Entry' either in the linker .prm file
        ; or from C/C++ later on

        XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack

; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.
Counter     ds.w 1
FiboRes     ds.w 1


; code section
MyCode:     SECTION
main:
_Startup:
 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 Entry
```

***<u>Conclusion:</u>***

We observed that the binary zeros and ones matches with the ON and OFF 8 LEDs on the microcontroller board. Case: Acc A had value of ( 55 ) in hexadecimal, the 8 LEDs showed the ones in binary as ON LEDs while zeros are the OFF LEDs ( 0101 0101). Another case when : Acc A had the value of ( 0F ), the 8 LEDs showed the equivalent of its binary state ( 0000 1111).

## *TASK-2: Reading Input Lines*

<u>Part 1:</u> In contrast to what we did in task 1 in task two, the following program below was downloaded on the microcontroller and run on the "HCS12 Serial Monitor" mode. It then took the input binary value from the user as he/she changes the Dip Switches Positions. The output was observed to be hexadecimal values in Accumulator A.

```
;**************************************************************
;* This stationery serves as the framework for a            *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the              *
;* Freescale CodeWarrior for the HC12 Program directory       *
;**************************************************************
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'

; export symbols
        XDEF Entry, _Startup, main
        ; we use export 'Entry' as symbol. This allows us to
        ; reference 'Entry' either in the linker .prm file
        ; or from C/C++ later on

        XREF __SEG_END_SSTACK      ; symbol defined by the linker for the end of the stack
```

```
; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.
Counter     ds.w 1
FiboRes     ds.w 1


; code section
MyCode:     SECTION
main:
_Startup:
 ORG $4000 ;Flash ROM address for Dragon12+
Entry:
    LDAA #$0
    STAA DDRH
    LDAA PTH
    ANDA #PTH_PTH2 ;(or #%00000100 or #$04 – to check PH2)
    BRA Entr
```

***Conclusion:***

After changing the Dip Switches Positions and it verifies that we are getting the same values in Acc A.

Part 2: We were then asked to modify the program to read only the right four bits of PTH while setting the high four bits to zeros. This was accomplished by masking after loading PTH to Acc A.

***Conclusion:***

We observed a case where when Accumulator A which was ( **2** ) and NOT ( **F2** ), the input in Dip Switches was (11110010).

## *TASK-3: Reading Dip Switches and Writing Them to LEDs*

In this task, we were responsible for reading to states of the DIP switches and reflecting these states immediately and continuously to the LEDs. In order to do that, we combined the codes of task1 and task2 and resulted with the following code:

```
;**************************************************************
;* This stationery serves as the framework for a            *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the              *
;* Freescale CodeWarrior for the HC12 Program directory      *
;**************************************************************
; Include derivative-specific definitions
          INCLUDE 'mc9s12dg256.inc'

; export symbols
          XDEF Entry;, _Startup, main
          ; we use export 'Entry' as symbol. This allows us to
          ; reference 'Entry' either in the linker .prm file
          ; or from C/C++ later on



 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
 STAA PORTB ;Store A into PORTB

 BRA Entry
```

Firstly, we load $FF to register A which in turn store it in DDRB, DDRJ, and DDRP, so that we make these three ports serve as an output. Then, we load $00 into register A and stores it in PRJ1 to allow the LEDs show the data, after that we disable the 7-segment display. Finally, we set Port H as input by loading $00 to register A and storing it to DDRH, so that we can read the input of the DIP switches and show the output on the LEDs. The BRA instruction is used here to enable us to repeat the read and write processes continuously.

## *TASK-4: Nested If-Then-Else Statements*

In this task, we had to modify task 3 in a way that we introduced a delay between reading the switches and writing to the LEDs using the following delay subroutine:

```
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 Instruction 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
;-------------------
```

We call the delay subroutine just after reading the DIP switches, and before writing to the LEDs.

The full code is provided below:

```
;**************************************************************
;* This stationery serves as the framework for a             *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the                *
;* Freescale CodeWarrior for the HC12 Program directory       *
;**************************************************************
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'

; export symbols
        XDEF Entry;, _Startup, main
        ; we use export 'Entry' as symbol. This allows us to
        ; reference 'Entry' either in the linker .prm file
        ; or from C/C++ later on
```

```
 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
 JSR DELAY
 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:
   R1: EQU $1000
   R2: EQU $1001
   R3: EQU $1002

   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
;-------------------
```

## 4. Analysis and Interpretation

It was understood that this laboratory was different than the previous ones in a sense that it depended on the use of the ports of the microcontroller board.  It explained how to use PORTB to control the LEDs in the first task and PTJ should be turned off with the use of Accumulator A. Dip switches were also used as inputs in the second task and it was controlled using PTH. Both codes of taks1 and task2 were combined together in order to use the dip switches as inputs and the LEDs as outputs. The final task included a subroutine of a delay in order to use it in other codes and it showed how the delay was created between the LEDs.

## 5. Conclusions and Recommendations

This laboratory encompassed four different tasks each of them determines a specific skill to achieve in order to understand how the Dragon12 board works. This laboratory experiment was unique in showing the use of various ports of the microcontroller hardware. It showed the basic techniques of the way of using the Dip switches as inputs and the LEDs lights as output. Delays were also implemented in this task which is an important thing to achieve. For the delay it had a precise code to add on each program in order to see the effect on the output. Each of the four tasks included a particular code in order to notice the benefit of the usage of each port on the board.

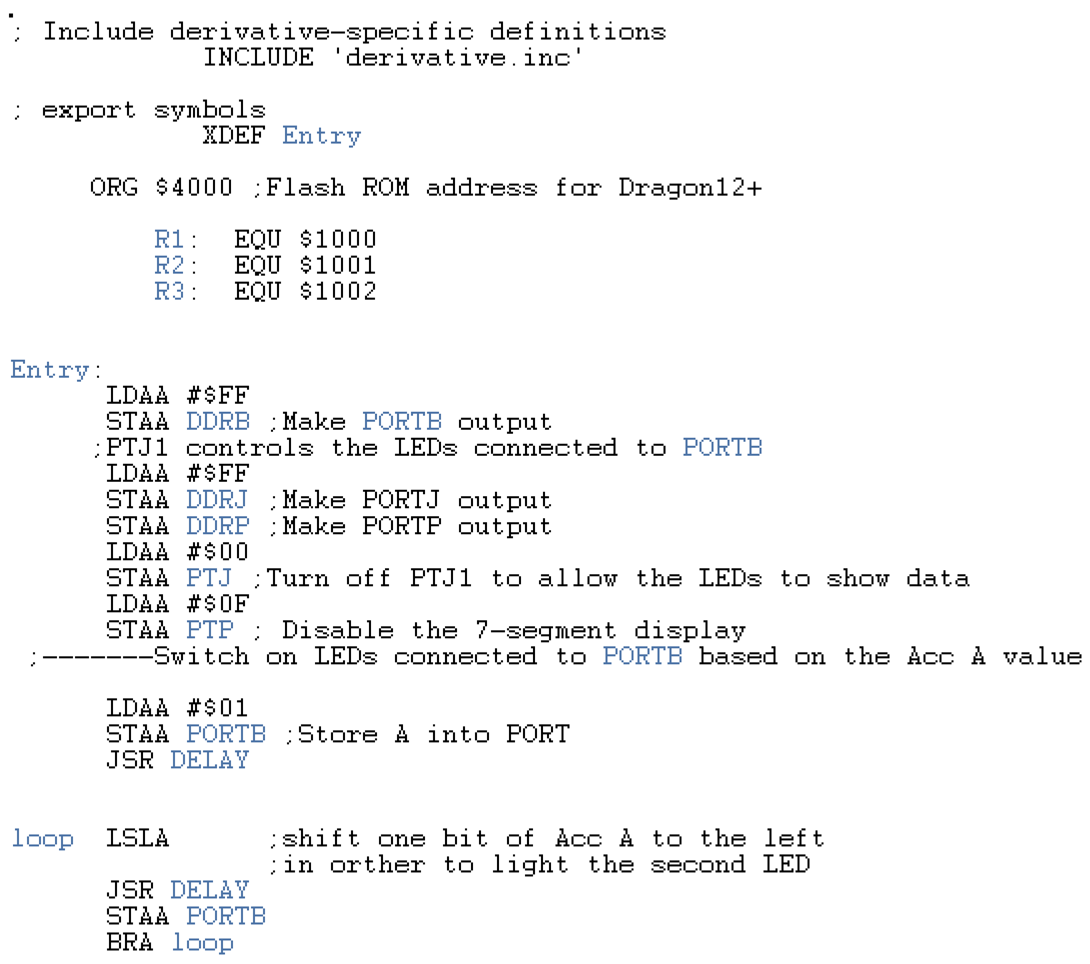
## 6. Assignment Questions

**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.**

```
; Include derivative-specific definitions
            INCLUDE 'derivative.inc'

; export symbols
       XDEF Entry

      ORG $4000 ;Flash ROM address for Dragon12+

         R1:  EQU $1000
         R2:  EQU $1001
         R3:  EQU $1002
Entry:
      LDAA #$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
      BRA DELAY
      STAA PORTB ;Store A into PORT

loop  INCA
      BRA DELAY
      STAA PORTB
      BRA loop

DELAY
     PSHA ;Save Reg A on Stack
     LDAA #200 ;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
```

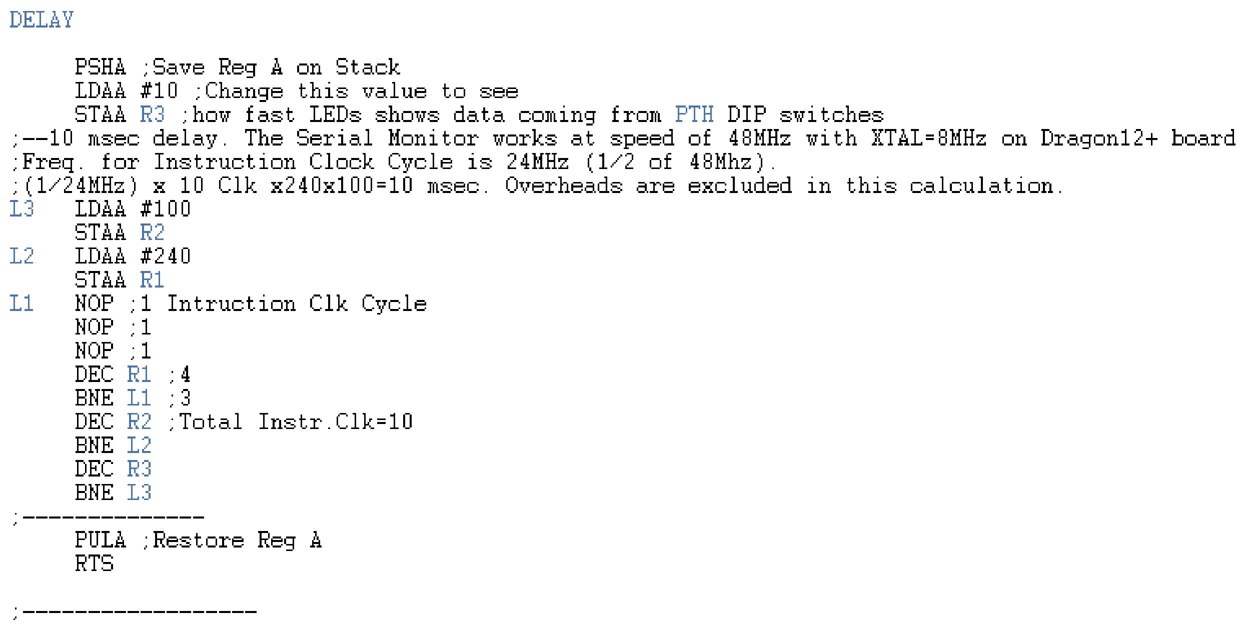
**2) Create a program that flash all the LED's with a delay of 0.1 sec in between.**

To write the code for this question we need to use the code of task4 in addition we need to modify it. We used the LSL command in order for us to move to the next LED as 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.



```
; Include derivative-specific definitions
            INCLUDE 'derivative.inc'

; export symbols
            XDEF Entry

      ORG $4000 ;Flash ROM address for Dragon12+

         R1:  EQU $1000
         R2:  EQU $1001
         R3:  EQU $1002


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
      JSR DELAY


loop  LSLA      ;shift one bit of Acc A to the left
                ;in orther to light the second LED
      JSR DELAY
      STAA PORTB
      BRA loop
```

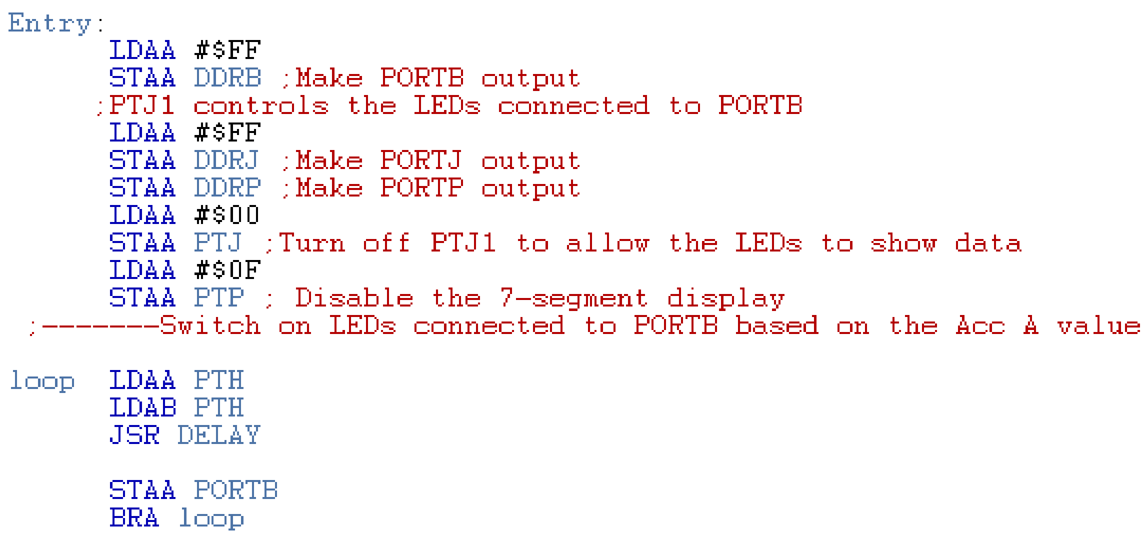
In the DELAY subroutine we modify the second line with LDAA #10 to make the delay faster and it is almost by 0.1 sec.



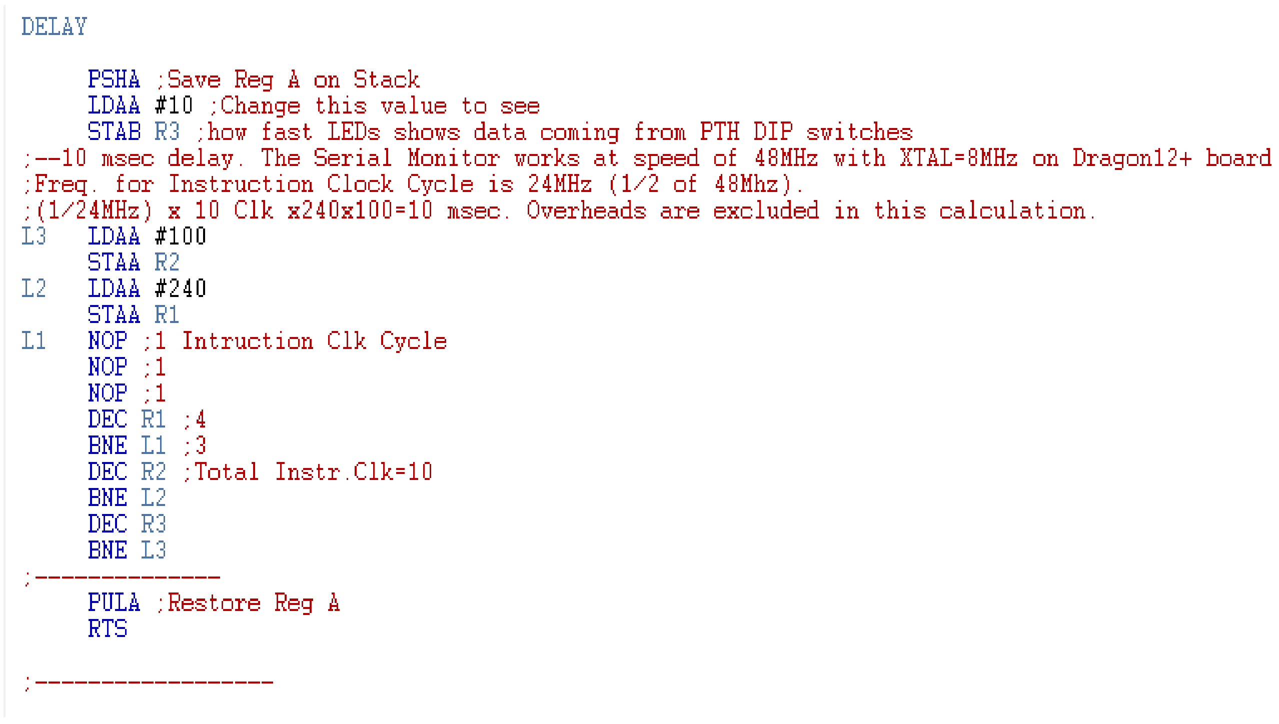
```
DELAY

     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) 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.**

The code of questionn2 is modified as follows:



```
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

loop    LDAA PTH
        LDAB PTH
        JSR DELAY

        STAA PORTB
        BRA loop
```



```
DELAY

     PSHA ;Save Reg A on Stack
     LDAA #10 ;Change this value to see
     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

;-------------------
```

## References

1. CourseTextBook
2. http://www.evbplus.com/
3. MC9S12DP256User'sManual