# Laboratory 7

# Input and Output

**Concepts:**

* Using the S12 general purpose I/O ports

**Objectives:**

* Use the S12 I/O ports connected to the Dragon12+ 7-segment LED displays and pushbuttons.

**Files Needed:**

* Lab07.zip and Lab07b.zip from Blackboard

**Assignment:**

The follow section walks though Ports B, H, and P on the Dragon12+ board.

1. Download Lab07.zip, unzip it, and open the project. Make sure the “HCS12 Serial Monitor is selected as the interface, then click on Debug to open the Real-Time Debugger. DO NOT RUN THE PROGRAM YET.
2. Change PTP ($0258) to $00 and DDRP ($025A) to $0F.
3. Change the value of DDRB ($0003) to $FF.
4. Change the value of PortB ($0001) to $FF.

**Question 1:** What do you see on the 7-segment displays?

1. Change the value of PortB to $6D.

**Question 2:** What do you see on the 7-segment displays?

1. Change the value of PortB back to $FF, then change the value of DDRB to $71.

**Question 3:** What do you see on the 7-segment displays, and why is this displayed if PortB is $FF?

1. Change PTP to $0E.

**Question 4:** What do you see on the 7-segment displays?

1. Change PTP to $0D.

**Question 5:** What do you see on the 7-segment displays?

1. Change PTP to $0B.

**Question 6:** What do you see on the 7-segment displays?

1. Change PTP to $07.

**Question 7:** What do you see on the 7-segment displays?

1. Change PTP to $0A.

**Question 8:** What do you see on the 7-segment displays?

1. Change DDRH ($0262) to $00, and make sure that all eight DIP switches are in the **upper** position. In the Memory window, right-click and select “refresh”.

**Question 9:** What value is in PTH ($0260)?

1. While holding down the pushbutton “SW5”, do another memory refresh.

**Question 10:** What value is in PTH?

1. Release the pushbutton SW5, slide the right-most DIP switch (switch 1) into the lower (i.e. “on”) position, and then do another memory refresh.

**Question 11:** What value is in PTH?

While switch 1 is still in the “on” position, press and hold pushbutton SW5, and do a memory refresh.

**Question 12:** What value is in PTH?

**Question 13:** Describe the behavior of bit 0 of PTH in terms of switch 1 and pushbutton SW5.

1. Experiment with the pushbuttons and the DIP switches to observe their effects on PTH.
2. Run the program and experiment with SW4 (the second pushbutton from the right).

This program uses several common concepts to interface with a mechanical switch. The first concept is debouncing. When a physical switch is pressed, the mechanical contacts act like a spring. The contacts literally “bounce” several times, thus making and breaking the electrical path, before coming to rest. This causes the electrical signal to toggle several times before reaching a steady value. This “bounce” may be incorrectly interpreted as several presses instead of one noisy one. To “debounce” the input signal, the supplied program generates a pause after seeing a signal transition. If the signal has the same new value after the delay, the transition is registered. Otherwise, it is treated like noise and ignored.

1. Back in the main program, change the #define value for DEBOUNCE to 1, recompile the program and run it.

**Question 14:** Is there a noticeable change in the behavior? If so, what?

The next program uses all four push buttons to control the different LED 7-segment displays. The approach used in the first program won’t quite work. It continuously polled the pushbutton to look for a change. With four digits, the S12 needs to repeatedly display each digit (called time division multiplexing), and the program can’t get stuck in a tight polling loop staring at one button, much less four. In the second program, the processor stores the value of a button each time it is examined. If the button was not pressed in the previous loop and it is pressed in the current loop, the processor has detected a new button press, and the corresponding digit will be updated.

1. Download Lab07b.zip.asm, unzip it, open the project, and run the program. Experiment with the pushbuttons to determine the program’s behavior. Notice that there is a “pause10usec(500);” function call after each 7-segment digit is displayed. One reason for this is to make the main loop consume enough time between iterations to debounce the buttons.
2. Change the four “pause10usec(500);” lines in the main program to “pause10usec(5);” and remake the program. Run it and experiment with the pushbuttons. You will notice “ghosting” of the LED 7-segment displays.

**Question 17:** Explain the cause of the LED ghosting effect. Be specific.

**Question 18:** Other than generating a long pause after updating the 7-segment displays, describe a second method to eliminate the effect observed above. Alter the program to implement this method and demonstrate it to the instructor.

**What to Demonstrate/Submit:**

* Answers to questions.
* Successful demonstration of the modification to remove ghosting.

Note: pause10usec is a subroutine written in assembly that delays for approximately 10 microseconds times the value passed as an input. Given the CodeWarrior parameter passing convention, this single input is by default the rightmost and is passed in a register. In this case, it uses D since an "int" is two bytes. This subroutine is written in assembly since we know exactly what the generated code is, and we can calculate an accurate timing. If we tried to write this in C, we would have a delay proportional to the input parameter, but we would not have any direct control over how much time was delayed for each count.