EE 308 – LAB 4

Parallel Ports and Subroutines

Introduction

In this week's lab you will write an assembly-language program to display various patterns on the LEDs of your breadboard. You will use the HC12's Port A as an output port to display the LED patterns, and Port B as an input port to decide which pattern to display.

Ports A and B are the easiest HC12 parallel ports to understand and use. For this week's lab, you will create programs to write to Port A, and read from Port B. You will use information from switches connected to Port B to control the pattern you output on Port A. You will test your programs by connecting Port A to 8 LEDs, and vary the Port A output by changing the switch settings connected to Port B.

Pre-Lab

Write a program to set up Port A as an 8-bit output port, and to implement (i) a binary down counter, (ii) a rotating bit, (iii) a flasher, and (iv) a turn signal on Port A. Samples from the sequences that you should generate are shown in Fig. 1. You will use eight LEDs to see the Port A output. Include an appropriate delay between changing the LED pattern so that you can easily and comfortably see them flash. Use a subroutine to implement the delay. Also, set up Port B as an 8-bit input port, and use Port B bits 4 and 1 to control which of the Port A functions are performed as shown in Fig. 2. When you switch between functions, the new function should start up where it ended when it was last activated, so set aside variables to save the states of the various patterns.

Write the program before coming to lab. Be sure to write the program using structured, easy-to-read code. Be mindful of the delay requirement before changing the display or reading the Port B pins to determine if you should switch to a new function.

The Lab

1. Run your program on the `ZAP` simulator. (Note that you will want to reduce the delay considerably before running on the simulator). You can simulate different input values on Port B by changing the value in Address `$0001`. If you have difficulty getting your program to work, start by trying to implement one function only — say, the down counter. Once this works, start working on your next functions.
2. Set a breakpoint at the first line of your delay subroutine. When the breakpoint is reached, check the value of the stack pointer, and the data on the stack. Make sure you understand what these mean.
3. Wire up Port A to 8 LEDs, and Port B bits 4 and 1 to a connection that can be switched between 5 volts and ground. Use the DIP switches on your breadboard to connect to the Port B pins.

   Begin by wiring up the Port B pins to switches, and make sure you can read the state of the switches. You can do this using D-Bug 12 to display the contents of Address `$0001` for several values of the switch settings. After you are sure you can read the state of the switches on Port B, try running your program on the HC12.

   When you get your program to work, have your lab instructor or TA verify the program operation.

1. A binary down counter:

```
· · · · · · · ·
● ● ● ● ● ● ● ●
● ● ● ● ● ● ● ·
● ● ● ● ● ● · ●
● ● ● ● ● ● · ·
```

Continue counting down.

2. A rotating bit:

```
● · · · · · · ·
· ● · · · · · ·
· · ● · · · · ·
· · · ● · · · ·
· · · · ● · · ·
· · · · · ● · ·
· · · · · · ● ·
· · · · · · · ●
```

Repeat the above sequence.

3. A flasher:

```
· ● · ● · ● · ●
● · ● · ● · ● ·
```

Repeat the above sequence.

4. A Ford Thunderbird style turn signal that alternates between right and left:

```
· · · · · · · ·
· · · ● · · · ·
· · ● ● · · · ·
· ● ● ● · · · ·
● ● ● ● · · · ·
· · · · · · · ·
· · · · ● · · ·
· · · · ● ● · ·
· · · · ● ● ● ·
· · · · ● ● ● ●
```

Repeat the above sequence.

Figure 1: Samples of the functions to be performed using Port A as an output. “●” is LED on and “·” is LED off.

PB4	PB1	Port A Function
0	0	Up counter
0	1	Rotating bit
1	0	Flasher
1	1	Turn Signal

Figure 2: Port B inputs to control the Port A functions.

4. Set a breakpoint at the first line of your delay subroutine. When the breakpoint is reached, check the value of the stack pointer, and the data on the stack. These should match what you found in Part 2 of the lab.

5. Put your program in the EEPROM at address `0x0D00`. Note that you will want the array which stores the turn signal patterns into the EEPROM (so the array will not disappear when you turn off power). To do this include any tables of the patterns in the CODE section of your program. You will want variables which will change as the program is executed to be placed in RAM, say at location `$0900`, as usual.

   Verify that your program runs correctly without reloading after cycling the power on your HC12.