LAB 3

By Christopher Tait (s3899374)

Introduction

This lab focuses on using the OUSB board to interface with an external keypad

Results & Discussion

Functionality of template code:

- Initialize the data direction registers, so that PORTC can be used for input and PORTB for output
- Initializes the stack pointer
- Displays the value of the temp register
- Loops forever

The .equ in the template creates a variable that can be substituted in for an actual numeric value

Part 1: DIP switches

In the initialization part of the code, DDRB and DDRC are setup so that the board can output to the LEDs on PORTB and input from the DIP switches on PORTC. The DDR bits are set high for output and low for input, so DDRB = 255 and DDRC = 0. Then in ReadSW, the value of PINC is read to the temp register. Finally the value of the temp resister is displayed of the LEDs. This is then looped, excluding the initialization.

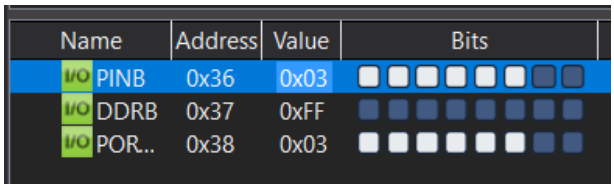

Figure 1: Output after a testing value of 0x03 was inputted

The above figure shows the output in the simulator with a manual value of 0x03 was set.

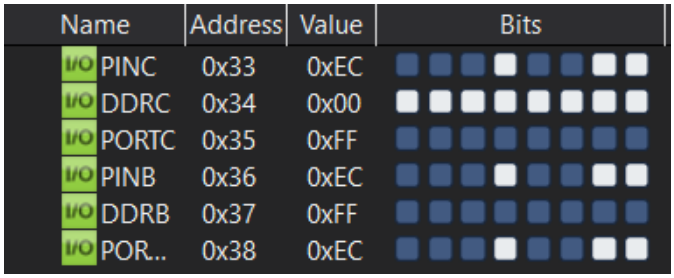

Figure 2: Output of the simulator

As shown in the above picture, the value of PORTB is the same as the value of PINC, after the code was run.

Part 2: Keypad in assembly

In my code, the keypad is scanned until a key press is detected, then the key press is converted to a hexadecimal index and saved. Then the code waits for the key to be released. This is all done one more time, then the value to display is read from a table, using the index as an offset in that table.

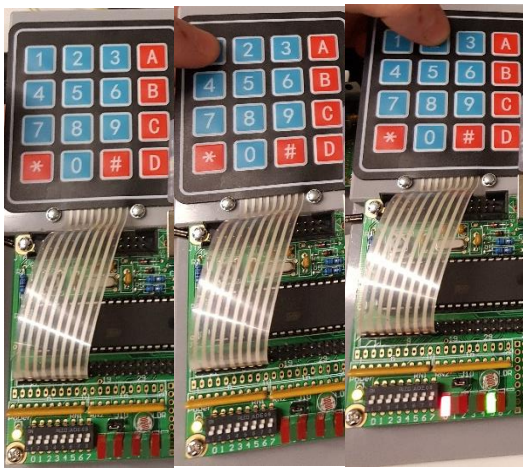

Figure 3: The lights and keypad before a key is pressed, after the 1 is pressed and after the 2 is then pressed

As seen in the above pictures, the system works as intended and shows what buttons were pressed after two are pressed.

Part 3: Keypad in C

The C version of the project is able to display after each key press by shifting the value of PORTB left, then combining in the new value. The C version uses a pointer to a table of values containing what the LEDs should display when each different key is pressed. For the non-number keys, a value of 255 is used as they are not meaningful in this context. The compiled c program (in release mode) goes up to address 0xCD, which means that it has 205 instructions.

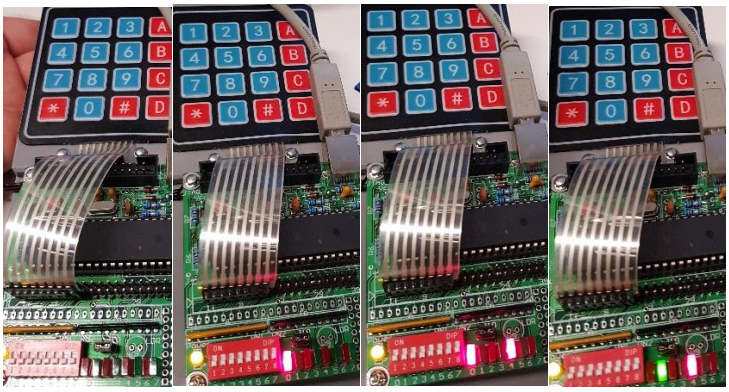

Figure 4: The keypad and LEDs before the buttons were pressed, and after 1, 1, 2 was entered

The above four pictures show how the LEDs react to buttons being pressed, and it shows how the value entered shifts over when a new value is entered.

Conclusion

In conclusion the keypad was interfaced with successfully using the ATMEGA32A's IO ports. The method of key scanning to detect the row and column of the keypad was implemented successfully. The application written in C was contrasted with the assembly code, it was much easier to develop and performed the same.