Laboratory 3 I/O Configuration

Instructor: Dr. Lihong Zhang

1 Objectives

With the completion of this lab, you should be able to:

- Design and construct a circuit to connect external switches and LEDs to the STK600 board
- Configure I/O ports of AVR microcontrollers
- Read data from input devices and write data to output devices
- Write an assembly language program to manage arithmetic operations of two numbers

2 Introduction

The STK600 board has built-in LEDs and push-button switches. For this lab, we need to input binary numbers to microcontroller via its ports. As push-button switches can only temporarily be in logic 1 or 0 states, they are not good for inputting stable 8-bit numbers. For this reason, we will use 8-bit Dual-Inline-Package (DIP) slide-style switches (available in your kit) for 8-bit data input. The 8-bit DIP slide-style switches should be mounted on a bread-board.

In this lab we are going to multiply two 8-bit unsigned numbers input from DIP switches and display the result on LEDs. As the multiplication result of two 8-bit numbers might be a 16-bit number, the STK600 built-in 8-bit LEDs are not enough for this purpose. Therefore, we need additional 8-bit LEDs (i.e., an 8-bit LED array available in your kit) mounted on the bread-board to compose a 16-bit LED display. The STK600 build-in LEDs are used to display the high byte of the multiplication result, whereas the 8-bit LED array is used to display the low byte of the multiplication result.

3 Software

In this lab, the following interconnections between STK600 ports and LEDs/Switches are defined:
PORTA ⇔ 8-bit DIP slide-style switches
PORTB ⇔ 8-bit LED array
PORTC ⇔ STK600 built-in SWITCHES
PORTD ⇔ STK600 built-in LEDS

Write an AVR assembly language program that can input the first 8-bit number from PORTA (i.e., the 8-bit DIP slide-style switches). This 8-bit number should be displayed on PORTB (i.e., the 8-bit LED array) in the real-time manner (i.e., any change on the 8-bit DIP slide-style switches should appear on the 8-bit LED array right away). The first 8-bit number should be stored to one register of ATmega32 as soon as STK600 built-in SW0 (i.e., connected to PORTC.0) is pressed. Then the program will input from PORTA a second 8-bit number, which

should be displayed on PORTD (i.e., the STK600 built-in LEDS) in the real-time manner. The second 8-bit number should be stored in another register of ATmega32 as soon as STK600 built-in SW7 (i.e., connected to PORTC.7) is pressed. Once the SW7 is pressed, the multiplication result of the two input 8-bit numbers should be calculated and displayed on PORTD and PORTB as a 16-bit number. Here PORTD is for high byte and PORTB is for low byte of the multiplication result. Note that it is defined that lighted LED is logic 1 and extinguished LED is logic 0. For the 8-bit DIP slide-style switches, the relationship between switch position and logic level can be defined by yourself.

Question1: What are the appropriate configurations of DDRA, DDRB, DDRC, and DDRD (that is, what byte should you send to each of them)?

You have to be extra careful while configuring I/O ports. If a port is configured as output, and if an input device is connected to that port, this might result in catastrophe. On the other hand, if a port is configured as input, and if you connect an output device to it, there is no problem. That is the reason why AVR microcontrollers leave all the ports configured as input ports upon reset.

Question2: What two registers are used to hold multiplication result for the instruction of “MUL”? Which is for high byte and which is for low byte?

Question3: To output a register to the STK600 built-in LEDS, say, “OUT PORTD, R16”, we need to do a certain operation on R16 in order to meet the following requirement: “lighted LED is logic 1 and extinguished LED is logic 0”. Discuss what operation you want to use.

4 Hardware

Besides the SKT600 board, the external parts that you need to use are 8-bit DIP slide-style switches and an 8-bit LED array (HLCP-J1000) as well as 10K ohm (9X-1-103LF) and 470 ohm (9X-1-471LF) Single-Inline-Package (SIP) resistors. The data sheets of the components above are available at D2L.

Make sure that the Power Switch of the STK600 board is placed to OFF before you make any connections, or before you modify any connections.
Before entering the lab, you should have sketched a circuit diagram of this circuit. Show the connections needed for the DIP switches and the LEDs in your circuit diagram.

Before the placement and wiring of components, you need to use a multimeter to verify the interconnection of different holes on the bread-board. Similarly, the multimeter can be used to verify the correspondence of the pins/holes of two connectors on both ends of the 10-wire cables.

4.1 Circuit Diagram

Put DIP switches and LED array on the proper place of a bread-board. You may want to check the data sheet or use a multimeter to find anode and cathode of LEDs. Add SIP resistors to the bread-board on the proper place. Use your prepared circuit diagram for the interconnection.

4.2 Circuit Wiring

You should connect the DIP switches and LED array on the bread-board to the headers of PORTA and PORTB of the STK600, respectively, using the provided 10-wire cables (the ones with DIP connectors on one end). Make sure you connect them correctly. Bring the power supply (i.e., VTG, the target voltage) to the bread-board from the STK600 board using a 2-wire cable (with pins on one end).

Wire the interconnection based on your circuit diagram. Wire your circuit neatly – color-code your wires, and cut your wires to the right length. Make sure that all the ground points are connected together. Do not place the wires in a spaghetti-inspired manner, but flow the wires around ICs. This will allow for easy removal of the devices if they are working incorrectly, as well as testing of pins for correct logic levels. Get the circuit checked out by your partner before powering up.

Note that the TAs might refuse to assist you in debugging your circuit if you wired it messily.

Write and debug your assembly language program within AVR Studio. If it works as expected, you can download the compiled hex file to the target microcontroller (i.e., ATmega32). Check whether your program works with ATmega32. If not, break down the program and fix any problems. Eventually you need to demonstrate your functional work to a TA who will sign on your printed program sheet.

Question 4: In your lab report, describe your observations and comment on your debug procedures.

5 Submission

Before the announced lab report due time, submit the print-out of your file (.ASM, where appropriate) including your comments in the program. Also include your circuit diagram, answers to all questions in the lab and prelab, and test results.

Before leaving the lab, unwire your circuit and return your devices to the kit.