Lab 1: I/O ports – controlling input and output

Goals:

The main goal of this lab is to become familiar with data ports, simple electrical components (resistors and LEDs), and using the breadboard.

Laboratory Report Submission Details

A laboratory report briefly describing what was done in the lab, including any relevant observations, and giving the responses to the questions in the laboratory description.

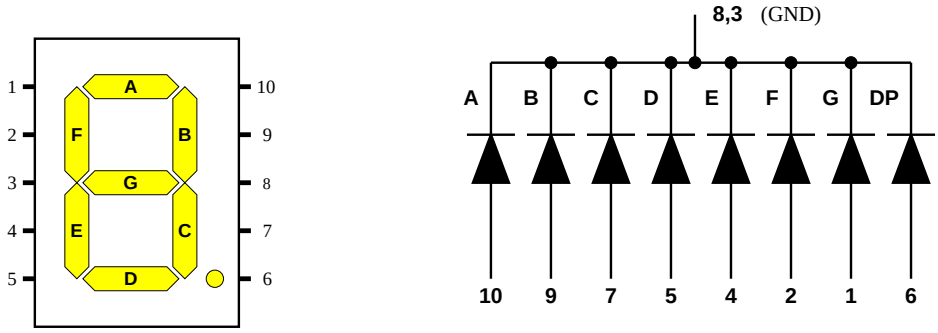
Any code written should also be provided, and demonstrated to the lab instructor before it is submitted.

The lab report is expected at the end of the lab period.

The 7 segment display

We have used discrete LEDs as output displays for binary numbers. There is a standard display element for decimal (or hexadecimal) numbers using 7 horizontal and vertical LED segments, called a *7 segment display*.

The following diagram shows the placement of the segments, and their usual labels.



An animated graphic showing which segments are lit for each of the hexadecimal numbers can be seen at

`http://en.wikipedia.org/wiki/Seven-segment_display`

Each segment is an individual LED, and (for the type we will be using – the em common cathode configuration, is lit when a positive voltage is applied to the input pin.

In order to limit the current through the LED, a resistor is placed in series with the LED. We will use a resistor array, containing 8 220 ohm resistors.

Write a C program which uses the switches, connected to port D, as input, and displays the number of the switch pressed. The switches are numbered 0 to 7, and the display should be blank when no switch is pressed. Use port C as the output port.

In the C language, the character data type is 8 bits, so it is common to represent the bit patterns required as elements of type `char`. We could also use the exact-width integer types `int_8t` or `uint_8t`. These are defined in `stdint.h` in the C standard library.

A sample `Makefile` to compile a C program and burn it to the flash memory of the ATmega1284P can be found in the support files for this lab.

Provide the source code of your implementation as part of your report. Demonstrate the working program to the instructor.

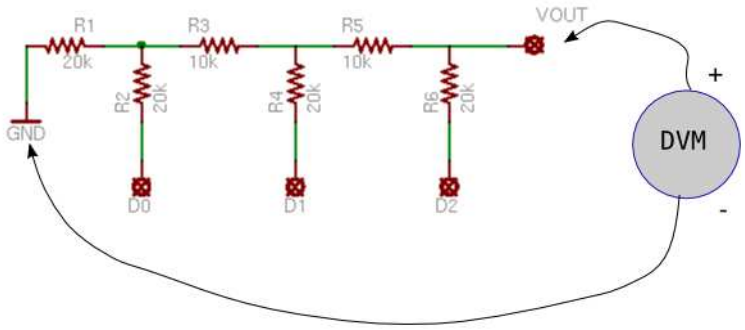
Optional:

Using a second 7-segment display and resistor array, display two digits on the pair of displays. This is normally accomplished by connecting the each of the grounds of the 7-segment displays to an output port, and selecting the particular port to display the value. If the program switches from one of the digits to the other fast enough, each display can appear to show a separate value continuously. This is called *time multiplexing*, ans is a common way to implement a multi-digit display.

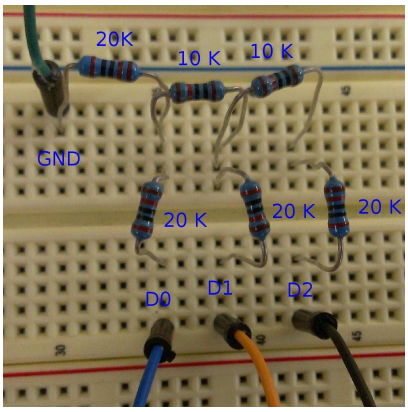
Write a program to exhibit this; e.g., a two-digit counter with a 1 second delay.

R-R2 Ladder and Digital to Analog Converter (DAC)

Using 10K and 20K resistors construct the following circuit on the breadboard:



Following is a picture of the circuit to be constructed:



- Connect the GND connection to one of the STK500's GND pins.

- Connect all possible combinations of GND and V_{target} to D0, D1, and D2 and measure Vout with a DVM. Record you measurements in tabular form in your lab report.

  Is there a pattern to the measured voltages? What do you think would be the effect of adding another stage (resistor pair) to the R-2R ladder?

AVR control of the DAC

Add another stage to the R-2R ladder. Configure bits 0, 1, 2, and 3 of PORTC for output. Connect PORTC, bits 0, 1, 2, and 3, to D0, D1, D2 and D3 of the R-R2 circuit. Write a C program to successively write the numbers 0 to 15 to PORTC, in an infinite loop.

Connect an oscilloscope probe to Vout, and observe the output. (This waveform is called a ramp, or sawtooth, waveform.) Relate the observed output to the voltage measurements tabulated earlier. Comment on your observations.

Provide the source code for your implementation. Demonstrate the working program to the instructor.

What changes would be required to display a triangular (rising and falling at the same rate) waveform?

Finding the switching point of an AVR input pin

Configure bit 7 of PORTD for input. Connect Vout to bit 7 of PORTD. Configure bit 7 of PORTB for output, and connect bit 7 of PORTB to LED7.

Modify the program from the previous step to sample bit7 of PORTD, and to turn LED7 on and reset the counter to 0) if bit7 of PORTD is 1. LED7 should be off if bit7 of PORTD is 0.

Write the maximum count value to PORTB[0-3], and display this value in binary with the on board LEDs. (A 10 pin cable can be used to connect PORTB to the LEDs.)

Record the binary number output on the LEDs. Comment on its value, with reference to the table generated earlier.

Provide the source code for your implementation. Demonstrate the working program to the instructor.