Experiment 13 –Arduino Microcontroller II: Seven-Segment Display Controller with Serial Interface using a Shift Register

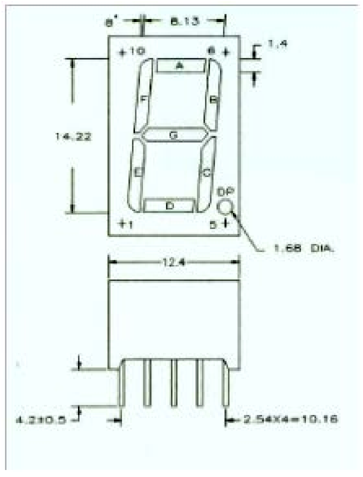
Physics 242 – Electronics

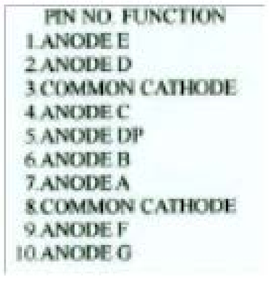
Introduction

We have seen that the Arduino microcontroller can be used to control other devices like the 7-segment digital display, but that project required most of the Arduino's available digital outputs. It would be very helpful if there were a way to economize on the use of the digital output pins—and fortunately this can be done conveniently using a shift register. A shift register chip accepts data serially, and this data is shifted onto its outputs as clock pulses are applied to the chip's CLK input. Using the 74LS164 shift register chip, we can use just two digital lines from the Arduino as the serial data input and the clock input for the shift register, and the shift register outputs can be used to light the segments of the display. Even better, if you need to display more digits, you don't need to use any more digital outputs of the Arduino—you can connect multiple shift registers together in a chain, with the last output of one used as the input to the next. So just two lines of the Arduino can be used to control a multi-digit output display.

Procedure

Your object in this lab is to use Arduino and a shift register to control a seven-segment digital display using just two digital outputs of the Arduino as described above. As in the previous lab, your Arduino should run connected to a computer with the Serial Monitor window open, so that you can send instructions from the computer to the Arduino. You should arrange is so that you can type "1" and the seven-segment display will light up "1", you can type "2" and the display will show "2", and so on.





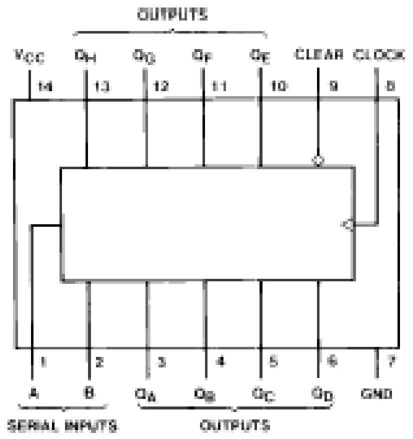
PIN NO	FUNCTION
1	ANODE E
2	ANODE D
3	COMMON CATHODE
4	ANODE C
5	ANODE DP
6	ANODE B
7	ANODE A
8	COMMON CATHODE
9	ANODE F
10	ANODE G

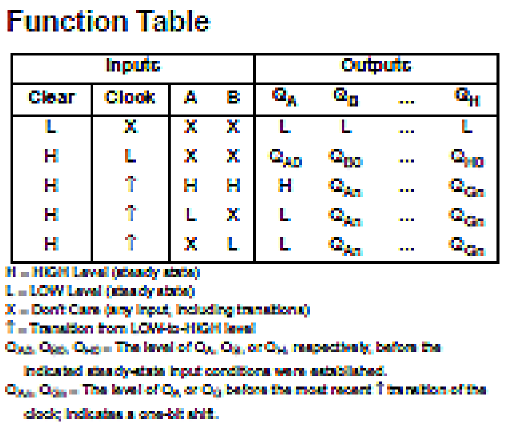
The pin connections for the seven-segment display are shown in the diagram above. It is a common-cathode display, which means that the negative side of the diode elements are connected together, and should be connected to ground. The positive side of each diode should be connected, through a 1 kΩ resistor, to the appropriate digital output of your Arduino board.

Be sure to use a resistor: if 5 V is applied directly across a diode, it will be destroyed by the large current that flows.

You can use any two of the digital outputs on the Arduino board, except pins 0 and 1, which are reserved for serial communications with the computer. Pins 0 and 1 should be left unconnected.

The pin connections for the 74LS164 shift register chip are shown below. The active-low CLEAR input on pin 9 should be connected HIGH. The chip provides two serial inputs A and B, NANDed together, which is more complicated than we need—you can connect one of the inputs HIGH and use the other for your serial data input from Arduino.





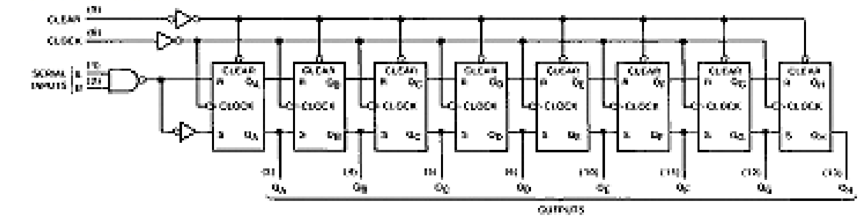
Function Table

Inputs				Outputs			
Clear	Clock	A	B	Q_A	Q_B	...	Q_H
L	X	X	X	L	L	...	L
H	L	X	X	Q_{A0}	Q_{B0}	...	Q_{H0}
H	↑	H	H	H	Q_{An}	...	Q_{Gn}
H	↑	L	X	L	Q_{An}	...	Q_{Gn}
H	↑	X	L	L	Q_{An}	...	Q_{Gn}

H = HIGH Level (steady state)
L = LOW Level (steady state)
X = Don't Care (any input, including transitions)
↑ = Transition from LOW-to-HIGH level
Q_{A0}, Q_{B0}, Q_{H0} = The level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = The level of Q_A or Q_G before the most recent ↑ transition of the clock; indicates a one-bit shift.

To understand how the shift register works internally, look at the logic diagram shown below. Serial data is applied to an R-S flip-flop connected so that the output of the first flip-flop follows its input at the next clock transition. The input of the second flip-flop is connected to the output of the first, and so on, so that the data is shifted from left to right along the chain of flip-flops as serial data comes in. Output Q_H can be used as the serial input of a second shift register if needed.

Logic Diagram



When your project is working, demonstrate it for the instructor. No lab report required!

Arduino Programming Hint: Check out the description of the shiftOut() function on the Arduino website (go to http://www.arduino.cc and look under the Reference tab). This function makes it easy to send data to the shift register one byte at a time.

Extra Credit Challenge:

Now that you know how to display multi-digit output from the Arduino, you can undertake a more interesting project…if you dare! The challenge is to build a two-digit computer-controlled voltmeter. Your voltmeter should respond to a user's command "v" to measure the voltage at analog input A0. The measured voltage (a value between 0 and 5 V) should be rounded to 2 digits (tenths of a volt) and reported back to the serial monitor and displayed on two 7-segment displays, using two shift registers. The decimal point should be lit, so voltages like "5.0" and "0.4" can be displayed appropriately. The voltmeter should also respond to the command "q" by blanking the displays. Be careful not to apply more than 5 V to the Arduino input during your testing (it's convenient to use the 0 V, 5 V and 3.3 V power outputs on the Arduino as voltages to test).