School of Advanced Technology

Lab 5: Multi-segment Displays

Lab Objectives:

1. Design a 7-segment display circuit with target voltage & current operating parameters.
2. Programmatically control individual elements in a 7-segment display.
3. Create a software loop to iterate through a set of digits appearing on the display.
4. Create a multiplexing loop, to alternately activate a pair of 7-segment displays.
5. Install a timer interrupt – an interrupt that will reset the clock at a specified time.

Required Equipment:

- Computer with Arduino IDE & Teensy extensions installed and working
- Teensy board and USB cable
- Resistors – suitable for the 7-segment displays
- 2 x 7-segment displays

References and Resources:

- Theory class circuit schematic.
- Course textbooks: "Beginning Arduino" and "Getting Started with Arduino", or other…
- Parts List file with part numbers, for referencing data sheets
- Data sheet for your 7-segment display
- Teensyduino IntervalTimer https://www.pjrc.com/teensy/td_timing_IntervalTimer.html

Task 1: Determine Operating Parameters for 7-Segment Display

1. It's always important to know the specifications of your equipment. Find the data sheet for the 7-segment display (use the parts list to identify the part number). A 7-segment display is a very common component. An online search using the part number as a keyword would lead to the vendor's specification datasheets. Identify the part in your electronics parts package. There are two of them.

2. Examine the data sheet. Identify the pinouts and note whether the device is common-***anode*** or common-***cathode***.

3. Identify the required operating voltage and current parameters for this device and compare the current requirements with what you know about the specifications (current limits) of the ARM processor that is on board the Teensyduino 3.2. Make an engineering decision regarding what current to design for. Use Ohm's law and other formulae as required when determining a suitable value for the current limiting resistor.

Task 2 (Demo #1): Basic set-up and functional verification of the 7-segment display

1. Install a 7-segment display with the five pin rows placed on opposite sides of the center gap of the protoboard. If uncertain about this step, look at the uploaded photo of a sample circuit, or ask your lab prof. Recall the layout of a protoboard. Don't accidently short any pins to each other.
2. The data sheet specifies that pins 3 and 8 (the common anode) are internally connected to each other. Only one of these pins is required to be connected to the Vcc (power) rail. Reasons why the two pins are internally connected include the convenience of connection options when bringing power to the display.
3. Signals from the teensy board are not required at this part of the design process – only the 3.3V rail and the GND rail are required.
4. Build a test probe. Test each of the segments of the 7-segment display and verify that each is working. The test probe is constructed by connecting a long wire in series with a 1 kΩ resistor. The other end of the 1 kΩ resistor is connected to a GND rail. The long wire is used to complete a circuit with single segments a-g of the display. Verify that each segment a-g lights up when you GND its respective pin.

Task 3: Complete the wiring of one 7-segment display

For this task you will continue to wire pin 3 (or 8) of the 7-segment display to the Vcc rail.

1. Connect each of the display's segments a-g, to a teensy pin, with a current limiting resistor installed in series. Be certain to install an appropriate size resistor as calculated in Task 1. Be certain that you know how to justify your calculations at "demo time". Use the sample schematics presented in both online resources and theory class as a reference.
2. You will issue digitalWrite(*pin, state*) commands (with appropriate delays). All teensy pins 0-23 are digital signal pins. The digital signal voltage level can be only one of two values: either a GND voltage level or a Vcc voltage level. The *state* argument specifies the voltage level, using either HIGH for a Vcc, or LOW for a GND. Sketch source code is written in a very "C-like" language, and *state* can also be specified as either binary 1 for Vcc or binary 0 for GND. Experiment to see if keywords *true* and *false* also work.
3. Write a small sketch that controls the outputs of the teensy. At this point you just want to verify wiring integrity of the circuit and correct pin number assignments in the sketch. Control each segment a-g of each 7-segment display. Use an On/Off pattern with a discernable delay.
4. Don't forget: the 7-segment display is a common ***anode*** device. Should the *state* argument be HIGH or LOW in order to illuminate a segment of the display?

Task 4: Single Display Numeric Characters & Counting (Demo #2)

For this task you will continue to wire pin 3 (or 8) of the 7-segment display to the Vcc rail.

1. Create a program which will cycle through and display all digits from 0 – 9. Issue digitalWrite(pin, state) commands to each of the segments a-g, in a way that illuminates any required segment to make any digit. Test and debug the program.
2. Demonstrate to the lab prof.

Task 5: Two Digit Decimal Counter (Demo #3)

For this task you will wire pin 3 (or 8) of each 7-segment display **to a teensy pin**.

You will use **multiplexing** to control a pair of 7-segment displays. You will use the minimum possible number of teensy connections (9 connections = 7 segments (a-g) + 2 common anodes).

1. Add a second display to the circuit from task 3. Extend the wiring controlling individual segments a-g, so that the second display is connected in parallel to the first. In other words, segment 'a' from the first display is connected to segment 'a' from the second display. Segment 'b' from the first display is connected to segment 'b' from the second display etc.. This is also called "daisy-chaining".
2. Remove the common anode pin connection from the Vcc rail. Re-connect the common anode from each display to a teensy pin, such that two separate connections are made, one for each anode to teensy pin connection. These two teensy pins are now referred to as ***driver*** pins.
3. Add two new pin definitions to your sketch for the new ***driver*** connections. These connections provide enable/disable functionality to each 7-segment display by issuing appropriate timed statements such as digitalWrite(tensDisplay, 1) and digitalWrite(tensDisplay, 0). These commands should be issued while simultaneously issuing digitalWrite(pin, state) commands to each of the segments a-g, in a way that illuminates any required segment to make any digit.
4. Modify your program to see the two digits count from 00 – 99, pausing at each value long enough so that it is distinguishable from the next. There should be no flickering of lights. You will need to rapidly alternate power between the two displays, fast enough such that the human eye doesn't detect the switching. This human eye "trickery" is known as "Persistence of Vision".

Task 6: Install a Timer Interrupt Demo #4)

1. Modify your code from task 4 by adding an interrupt. The interrupt will reset the count at a number to be determined by your lab instructor. Make your code *scalable*.
2. Demo to your lab prof.

Deliverables:

Complete the demo of: [2 marks each item]

i. Test probe 7-segment display (task 2)
ii. Single digit basic counter 0-9 (task 4)
iii. two digit decimal counter 00 – 99 (i.e. 10, 11, 12, 13….) (task 5)
iv. Implement an interrupt service routine (ISR) to react to a timer (task 6).