Physics 3150, Laboratory 9
Clocked Digital Logic and D/A Conversion

March 28 and 30, 2016

By Ed Eyler and George Gibson, based in part on Hayes and Horowitz
Last revised March 24, 2016, by Ed Eyler

Notes:

(1) There will be only one more organized lab session, on April 4 and 6. The rest of the semester will be devoted to your final projects.

(2) If you think that your seven-segment display will be useful for your final project, it's OK to leave it wired on a breadboard after the lab ends.

Purpose:

1. To use a binary "decoder" chip to drive a seven-segment numeric display that can be used for future projects.
2. To study sequential logic by building counters, first with flip-flops, then with a dedicated counter chip. Your display can be used to show the changing count values.
3. To use a DAC at the counter output, generating a 10-step or 16-step analog ramp.

References:

Chapter 8 of Eggleston; Chapter 7 of Meyer; Data sheets on the "Resources" section of the course web page.

Equipment:

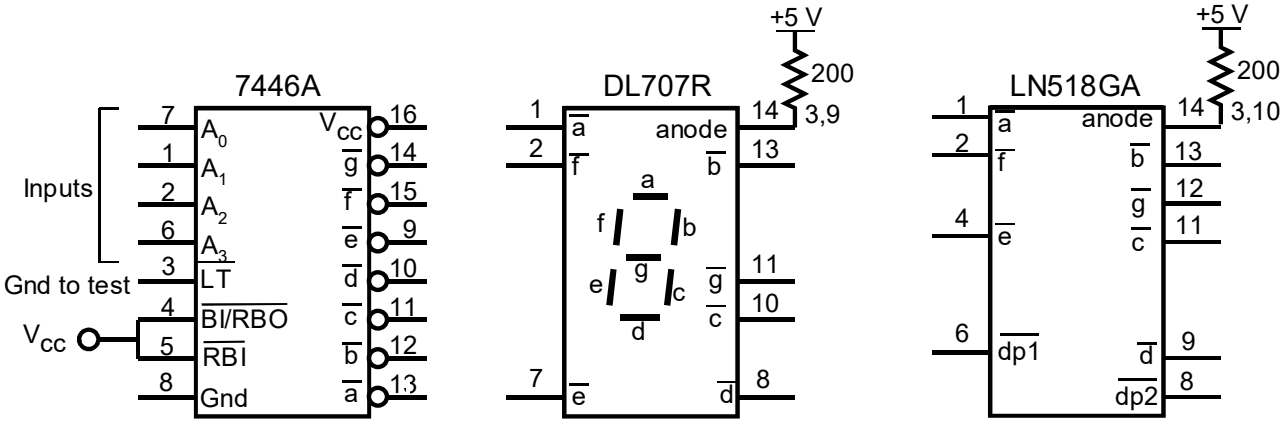
1. Digital oscilloscope.
2. Breadboard with logic switches, logic indicator LEDs, and 5V supplies.
3. 7446A or 74LS247 decoder/drivers.
4. DL707R or MAN71A or LN518GA seven-segment LED display.
5. 74LS192 or CD40192 or 74LS193 8-bit counters.
6. 74LS76 or 7476 flip-flops.
7. TLC7524 8-bit DACs and LF411 op amps.
8. Decoupling capacitors, 0.01 μF or 0.1 μF.

I. Driving a seven-segment display

A typical LED display chip consists of seven LED line segments plus a decimal point, all sharing a common anode. Normally the anode is connected to a +5V supply through a current-limiting resistor of about 200-300 ohms. Then when the pin corresponding to any of the line segments is pulled low, the segment is illuminated. (The brightness will obviously vary depending on the number of segments illuminated; for more uniform intensity, separate pullup resistors could be used for each segment.)

To display numbers in the way we're used to seeing them, we need an array of gates that determine which segments are to be lit. This would clearly be a lot of work if you built it up from NAND gates. But fortunately just such an array is provided for you in the 7446A or the very similar 74LS247, members of a broad class of chips described as decoders, which monitor two or more input lines and assert various combinations of output lines based on the input data and the internal logic map. (Incidentally, arbitrarily complex tasks of this type can be performed by programmable gate arrays, or alternatively by memory chips if they are programmed so the data at each address corresponds to the desired output for the combination of input lines selecting that particular address.)

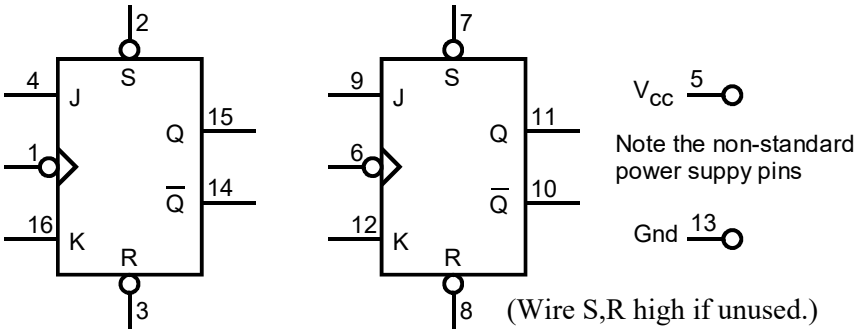
The outputs of the 7446A can sink enough current in their low state to directly drive the display, which is why it's called a decoder-driver chip. In one corner of your breadboard, wire up your display to the driver as shown, then test it by asserting the active-low test input on pin three. All seven segments should then illuminate to form the number '8'. If you like, play with the inputs by grounding them or letting them float high to display various other numbers.



Decoder/driver and 7-segment displays. The MAN71A is pin-compatible with DL707R if its extra pins are ignored.

II. Basic counter using flip-flops

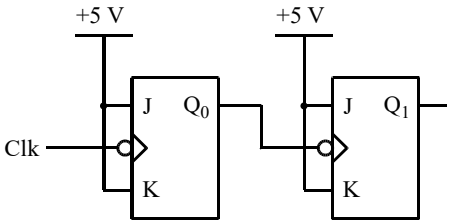
The J-K flip-flop, which is not very widely used any more, offers a lot of versatility with its several operating modes. In this lab you need to know only two of them: when J and K are both high, the output will change state, or "toggle," with every clock pulse. By contrast, when they are both low, the output will remain unchanged when the clock input is pulsed. We will use the 74LS76 (or 7476), which has an unusual pinout, as shown in the figure. Note that these are 16-pin packages. The pins are still numbered counterclockwise around the chip.



74LS76A flip-flop

There are two kinds of counters, ripple and synchronous. In both cases, each flip-flop counter stage divides by 2, so the outputs constitute a binary number counting the number of clock pulses. We will build a counter with just two stages, so we can only count from zero to three. By simpy adding more stages, you can count higher.

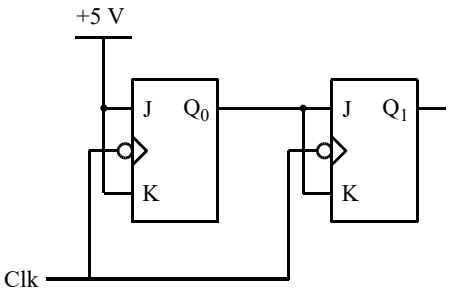
A. Ripple counter



In a ripple counter, the clock pulse for any flip-flop is derived from the previous stage, and so the clock pulse "ripples" through the circuit and the outputs do not change simultaneously. *Caution:* Counters are very susceptible to noise. Be sure to use bypass capacitors from the power supply pin to ground, and use a clean layout for the clock signals. Also, wire all four of the S and R inputs high, so that they will not cause erratic resets due to noise pickup.

1) Build the circuit and clock it with a square wave input. Examine the outputs with a scope or by connecting them to your display driver.
2) Trigger your oscilloscope with the clock and see if you can detect the delay in the transition of Q_1.

B. Synchronous counter



In the synchronous counter, each flip-flop is clocked with the input clock, and each of the outputs is immediately updated, as needed, depending on the state of the previous stage.

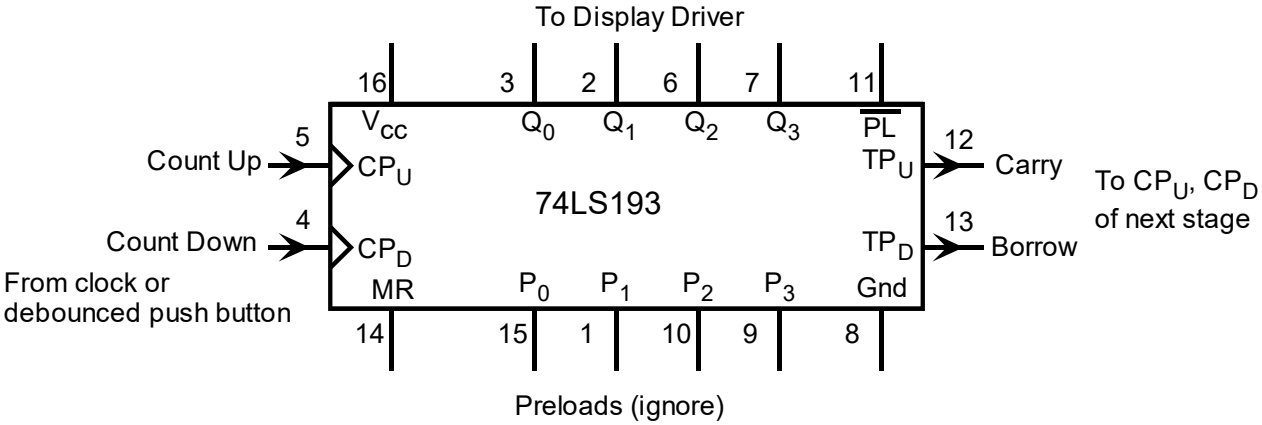
Build a two-stage counter as indicated, and try to verify that there is no ripple delay.

Question:

1) Explain how each counter works.

III. Up-down counter

Caution: These counters are particularly susceptible to noise pulses. Be sure to follow the advice given in Section II.A above. The 74LS192 and 74LS193 are very similar general-purpose four-bit counters, varying only in that the '192 counts in BCD, so the next output after 9 is 0, not 0A hexadecimal. The CD40192 is a newer CMOS variant of the 74LS192, with similar properties when operated with a 5 V power supply. The counters have preload inputs (which we won't use), so that a specific initial value can be loaded when needed. It also has carry and borrow outputs, so that an arbitrarily large counter can be built by cascading these chips, connecting carry to the Count Up (CP_U) input of the next stage and borrow to the Count Down (CP_D) input.



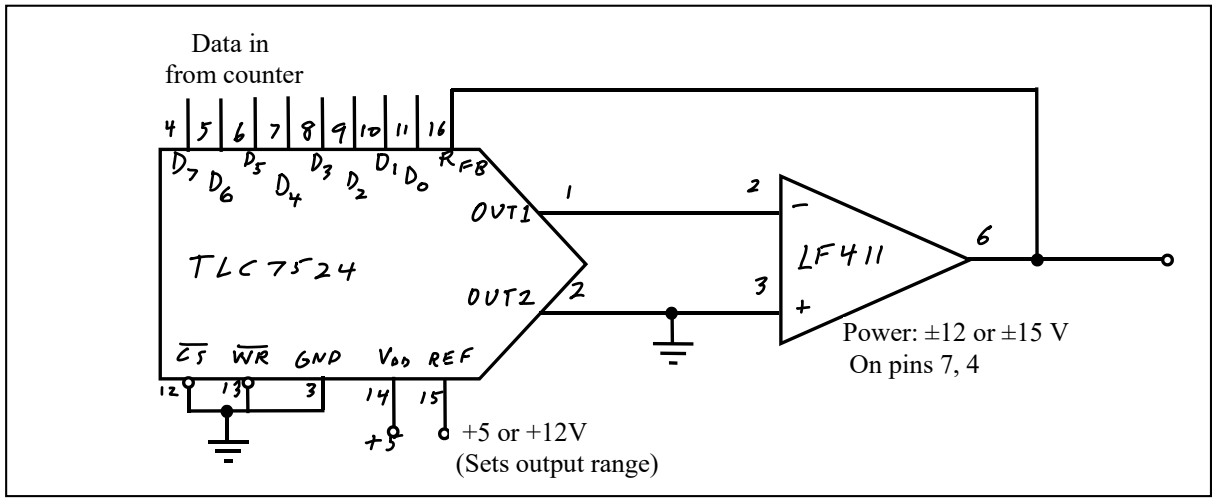
Wire up your counter using either chip, being sure to set Master Reset (MR) low so the counter is enabled. You should also set the Load input ($\overline{PL}$) high, together with whichever clock input (CP_U or CP_D) you're not using at the moment. In practice they will float high for the LS versions of the chips, but not for the CMOS version, and even for the LS version there will be problems with noise-induced changes of state if these inputs are left disconnected.

Using your seven-segment display driver, display the outputs Q_0-Q_3 while counting up and down using a slow clock. If you use a 'LS193, what happens to the display when you count through the states corresponding to digits larger than nine? Determine whether the MR input is synchronous or asynchronous by asserting it between clock cycles. Why do you think this choice was made by the chip designers?

IV. Digital to analog converter

Convert the counter you just built to a low-resolution ramp generator by using a digital-to-analog converter chip. The TLC7524 is a traditional current-switched DAC. It also uses a buffered digital input with latches, to facilitate computer bus interfacing, but we will make the latches transparent by activating the 'chip enable' and 'write' pins simultaneously.

This particular DAC has 8 bits of resolution, so unless you have time to wire another counter chip, you should just use the four most significant bits. The DAC switches a resistor ladder based on its inputs, so with the addition of a reference voltage on pin 15, it acts as a current source. It can also be used "backwards" as a stand-alone voltage source, as shown in Fig. 1 of its specification sheet (see the Resources section of the course web page). However, when a voltage source is needed, the DAC is most often used to drive an op amp as shown below.



Once your DAC is connected, watch it step through the counter output values, then set the clock faster to make a sixteen-level ramp. At very high clock speeds, you will probably see a large "glitch" from the DAC every time it switches. What do you think causes this?