CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

EXPERIMENT 2
NUMBER SYSTEMS AND WAVEFORMS

Text: Mano and Ciletti, *Digital Design, 5th Edition*, Sec. 1-1 through 1-4 (for additional information on counters, refer to Sec. 6-3).

Commented [NWP1]: Figure out which section talks about number systems. Remove reference to counter

Required chips: **7493** ripple counter.
Oscilloscope: For information on the oscilloscopes in the lab, refer to "USING THE AGILENT DSO1012A OSCILLOSCOPE" at the end of this experiment. (There is also a manual, "Oscilloscope DSO1012A", on the lab support website, http://www.calstatela.edu/ecst/ece/labs, however, you probably won't need it.)

2.1 Here is a quick review of number systems:

- Humans use decimal which is base 10 because we have 10 fingers.
- Digital logic systems, such as computers and microcontrollers, use binary which is base 2 because they are built using transistors. We saw in Experiment 1 that a transistor is an electrical switch that is on or off and thus has 2 states. A binary digit is known as a "bit" and 8 bits are called a "byte".
- Hexadecimal (base 16) and Octal (base 8) are shorthand for binary. Both are powers of two. 16 = 2^4 and 8 = 2^3. Digits in a binary number (bits) can be grouped together to convert to either hexadecimal or binary. Groupings must start to the left of the binary point (to the left of the one's place or the 2^0 place). To convert to hexadecimal, group by 4 bits and convert each grouping into the corresponding hex digit (where 10 through 15 are represented using letters A through F). To convert to octal, group by 3 bits and convert each grouping into the corresponding octal digit.

Binary (Base 2) Number Example:

$$\mathbf{(10100101)_2} = (1)*2^7 + (0)*2^6 + (1)*2^5 + (0)*2^4 + (0)*2^3 + (1)*2^2 + (0)*2^1 + (1)*2^0$$

Binary to Decimal (Base 10) Conversion Example:

$$(10100101)_2 \rightarrow (\overset{128}{1}\ \overset{64}{0}\ \overset{32}{1}\ \overset{16}{0}\ \overset{8}{0}\ \overset{4}{1}\ \overset{2}{0}\ \overset{1}{1})_2 \rightarrow 128 + 32 + 4 + 1 = \mathbf{165_{10}} = (1)*10^2 + (6)*10^2 + (5)*10^0$$

Binary to Hexadecimal (Base 16) Conversion Example (A=10, B=11, C=12, D=13, E=14, F=15):

$$(10100101)_2 \rightarrow (\underline{\overset{8}{1}\overset{4}{0}\overset{2}{1}\overset{1}{0}}\ \underline{\overset{8}{0}\overset{4}{1}\overset{2}{0}\overset{1}{1}})_2 \rightarrow \mathbf{A5_{16}} = (10)*16^1 + (5)*16^0$$

Binary to Octal (Base 8) Conversion Example:

$$(10100101)_2 \rightarrow (\underline{\overset{4\,2\,1}{010}}\ \underline{\overset{4\,2\,1}{100}}\ \underline{\overset{4\,2\,1}{101}})_2 \rightarrow \mathbf{245_8} = (2)*8^2 + (4)*8^1 + (5)*8^0$$

Complete the math in the above examples to verify that $(10100101)_{2}$, 165_{10}, $A5_{16}$, and 245_8 are the same number just represented in a different base.

Question: Which base, hex or octal, is more commonly used today as a shorthand representation for binary numbers and why? To answer, consider the following: 1) which base, hex or octal, will require fewer digits to represent a binary number? And 2) given that a byte (B) is comprised of 8 bits (b), which base, hex or octal, can be more easily used to represent an 8-bit (1-byte) binary number? Write the answer, including the explanation, in your lab journal.

- Table 2.1 shows the decimal numbers from 0 to 31 and the binary, octal, and hexadecimal equivalents.

Table 2.1. Numbers 0 to 31 in decimal (base 10) and the corresponding equivalents in binary (base 2), octal (base 8) and hexadecimal (base 16).

Decimal	**Binary**	**Octal**	**Hexadecimal**	**Decimal**	**Binary**	**Octal**	**Hexadecimal**
0	00000	00	00	16	10000	20	10
1	00001	01	01	17	10001	21	11
2	00010	02	02	18	10010	22	12
3	00011	03	03	19	10011	23	13
4	00100	04	04	20	10100	24	14
5	00101	05	05	21	10101	25	15
6	00110	06	06	22	10110	26	16
7	00111	07	07	23	10111	27	17
8	01000	10	08	24	11000	20	18
9	01001	11	09	25	11001	21	19
10	01010	12	0A	26	11010	22	1A
11	01011	13	0B	27	11011	23	1B
12	01100	14	0C	28	11100	24	1C
13	01101	15	0D	29	11101	25	1D
14	01110	16	0E	30	11110	26	1E
15	01111	17	0F	31	11111	27	1F

2.2* (* = lab work) Connect the 7493 as a 4-bit counter as a "modulo-16" counter which counts from 0 to 15 (F in hex) and returns to 0 on the 16th count. Also bring QD,QC,QB,QA to the LED bargraph. Ground the Ro(1) reset pin (pin 2) to allow the 7493 to count. (Note, that Ro(1) and Ro(2) are connected internally so that if either reset input is grounded, the circuit will continually count. So, you could have grounded pin 3 or both pins 2 and 3 and it would continually count.) Next, connect the normally high pulser output (push button switch) to the counter's clock input at pin 14. Figure 2.1 shows the *wiring* diagram for this circuit. As you make connections, it is a good idea to put a check mark next to the wire in order to know what part of the circuit you have completed and to ensure that you do not forget any wires.

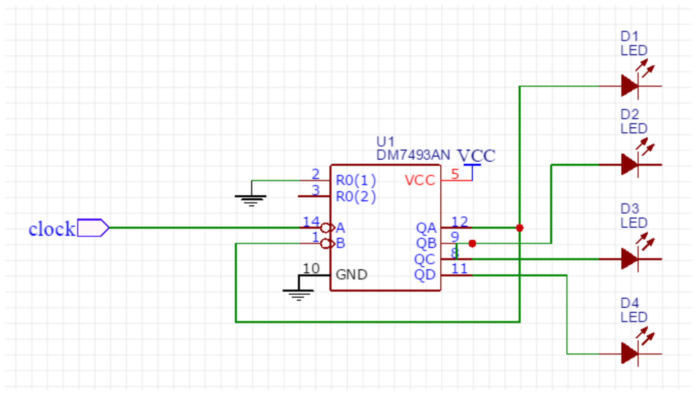

Figure 2.1. Wiring diagram for a Mod-16 counter.

Figure 2.2 illustrates how to place the 7493 chip on the board. It should straddle the center line of the breadboard section (between columns e and f). Notice that the holes on the breadboard along one row are connected (e.g., columns a through e along one row are connected and columns f through j along one row are connected). That allows you to connect various wires to the same pin on a chip (or to connect two wires together if there is not a pin from a chip at that row). Notice the wiring in this circuit. Where possible the wires do not cross one another and lay flat close to the board. Avoid putting wires over the chip as it will be difficult to remove the chip without having to rewire the circuit in the situation arises where there is a problem with the chip. Taking the time to wire circuits neatly will save you a lot of time in troubleshooting your circuits (finding mistakes also known as “bugs”).

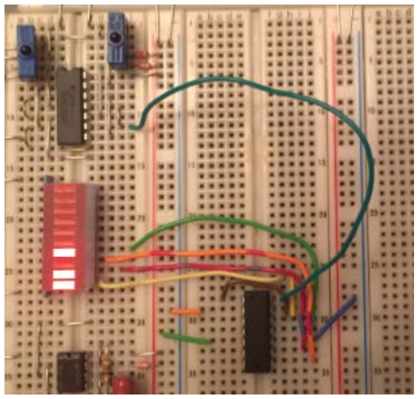

Figure 2.2. Mod-16 counter circuit with the clock input connected to the normally high pulser output and the counter output connected to the LED bargraph. Count showing on LED bargraph is 1011_2 (11_{10} and B_{16}). Note subscripts are used to indicate with base is being used X_2 is a base 2 (binary) number, X_{10} is base 10 (decimnal) number, and X_{16} is base 16 (hexadecimal) number.

Note: Since you have never wired a circuit before, Figure 2.2 is provided to help you understand how to connect the circuit. However, as you are connecting the wires to the chip, try to follow the wiring diagram in Figure 2.1 rather than the circuit diagram in Figure 2.2. For the remainder of the experiments you will only be given, or will design you own, wiring diagrams.

Verify that the counter cycles through the count 0000 to 1111. Note, depending on the state of the counter when the chip powers up, the starting value will likely not be 0000, but eventually it will roll over from 1111 to 0000 and you will be able to observe a complete count cycle.

Now connect the timer output to the counter's clock input at pin 14 (as shown in Figure 2.3). Run the clock at its slowest rate so you can see the LEDs flash on and off. Again, the LEDs should go through a full 16-count sequence:

0000 → 0001 → 0010 →….→ 1110 → 1111 → 0000 → .... etc..

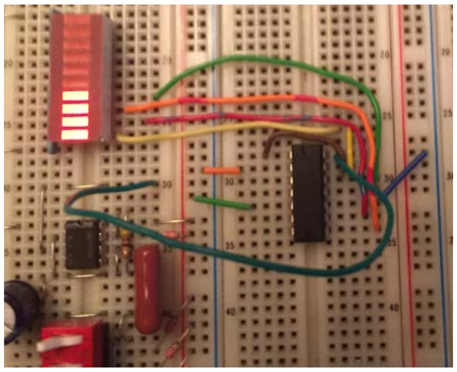

Figure 2.3. Mod-16 counter circuit with the clock input connected to clock generator output (row 30 on the breadboard) counter output connected to the LED bargraph. Count showing on LED bargraph is 1111_2 (15_{10} and F_{16}).

2.3* Once you are satisfied that the counter is running correctly, switch the clock into its high range. The counter voltages should now be changing too fast to see their patterns using LEDs. Instead, you will observe them using an oscilloscope. (Please see “Controlling the Oscilloscope Display” at the end of this experiment for information on using the Agilent DS01012A oscilloscope.)

Figure 2.4 shows the timing relationships between the counter outputs. The count values are shown at the bottom of the figure; each one represents the binary pattern in the corresponding vertical slice read from the bottom up; i.e. from QD up to QA. For example, count 7 corresponds to QD, QC, QB, QA = 0 1 1 1.

In this section, you will display the waveforms of Figure 2.4 two at a time on an oscilloscope.

a) Bring QD to channel 1 and QC to channel 2.
b) Set vertical sensitivity (Volts/Div) for QD and QC to 2 V/Div. Each waveform should swing approximately between 0 and 4 volts.
c) Set up the scope to trigger on the ***negative edge*** (or slope) of channel 1 (i.e. QD).
d) Adjust the scope-trace speed (Time/Div) so you see QD and QC displayed as in Figure 1.
e) Position the display with the Horizontal control so the point where QD goes low is on a vertical grid line near the left side of the screen (as in Figure 2, below).
f) Adjust Vertical positon control for QD and QC so their 0 V indicators lie on horizontal grid lines.

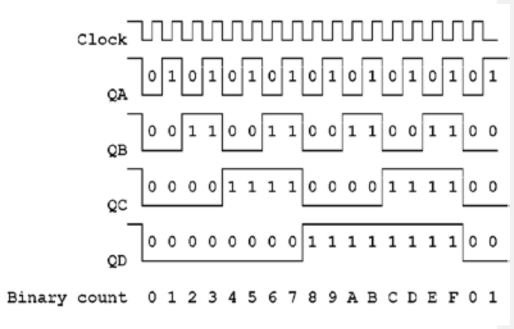

Figure 2.4. Expected waveforms for a mod-16 counter.

Important: in order to make it easy to read the binary count at any point in the display, do the following: adjust the scope's Time/Div setting until 16 counts--the time for one cycle of QD-- take up 8 divisions on the screen. Then each division will represent 2 counts. Thus QA will toggle each half-division, QB each division, QC each two division, and QD each four.

Verify that the timing relationship between QD and QC is as in Figure 2.4.

Repeat with QD and QB on the screen, then with QD and QA, and finally with QD and the clock. Don't change any scope setting while doing this--**QD should appear the same each time**. Show each display to your instructor (once you have verified that it works properly, you should be able to demonstrate it to the instructor quickly by moving the wire connected to channel 2 from QC to QB to QA).

2.4 As mentioned, the 4-bit binary counter of 2.2 is a "modulo-16" (Mod-16 or hexadecimal) counter since it counts from 0 to 15 (F in hex) and returns to 0 on the 16th count. In general, a counter that counts up to a maximum value of N-1 and resets to 0 on the Nth count, can be called a modulo-N (or “mod-N”) counter. The reset inputs, Ro(1) and Ro(2), of the counter can be used to reset the counter. If both reset inputs are high (logic 1) the counter will reset. Note that if a reset input is not connected to anything, internally it will be pulled high.

In a 4-bit mod-N counter with N < 16, the count does not roll over to 0000 naturally (as it does with N = 16); instead, it ends prematurely after only N counts. With the asynchronous 7493, the count must enter state N for just a few nanoseconds in order to be cleared. (As we’ll see in later experiments, with a synchronous counter, the count never enters state N, but goes directly from state N-1 to 0000.)

In state N, certain Q outputs will be 1's. You can combine these in such a way that they make both reset inputs Ro(1) and Ro(2) go high, and thereby trigger an immediate reset just after state N is entered.

For N=10_{10} (a decimal counter), simply connect QD to Ro(1) and QB to Ro(2). In state 10_{10} (**1**0**1**0_2), these both go high. The internal logic responds by clearing the 7493 (1010 → 0000). You don't see it enter state 10 since reset is almost instantaneous. Thus, the counter will seem to count from 0 to 9 (10 states) and then start counting up from 0 again.

2.5* Design a mod-N counter. Your instructor should give you a value for N from the set **5, 6, 8, 9, 12**. If not, you should choose your own value from this set. Draw the wiring diagram for your circuit in your lab manual (similar to Figure 2.1).

N =

As you connect your circuit, *be sure to remove any grounding wires you connected to Ro(1) and/or Ro(2) in section 2.1 above. If you don't, your circuit will never reset to 0 when the count reaches N.*

Test your design with a pulser or slow clock and observe the count on the LED bar graph.

Next, bring the counter outputs to the scope as in Section 2.3. Use a fast clock and again **trigger on the falling edge of QD**. This time, record QD-QC, QD-QB, and QD-QA from the scope display.

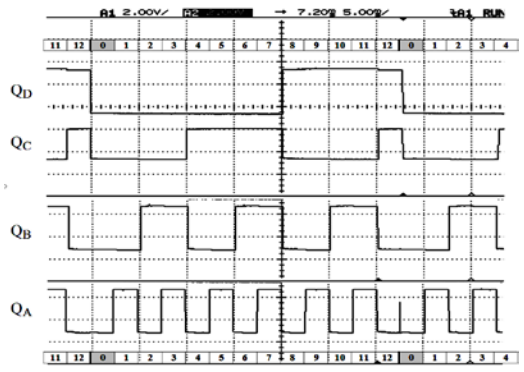

Figure 2.5. Expected waveforms for a mod-13 counter.

As an example: the diagram shown in Figure 2.5 is for N = 13. The display starts at the left shortly before QD goes low to trigger a new cycle of 13. The count proceeds from 0000 to 1100 (12) for a total of 13 full count periods. It then enters 1101 (13) just long enough to reset itself to 0000 again. Notice that, at the end of count 12, QA shows a short spike as it rises and then immediately falls, having reset all the Q's, including itself, to 0. This spike is much too narrow to be seen at the scope settings you will be using since it only lasts about 50 nanoseconds. But it is there (it must be, or what else would reset the counter?).

After you have designed and tested your Mod-N counter, make sure that the following are recorded in your lab manual:

- clearly **describe the connections** you made to reset the counter

- show the wiring diagram with the correct connections
- draw the waveforms you observe on the oscilloscope. As done in Figures 2.4 and 2.5, stack the waveforms for QA, QB, QC, and QD (even though you can only observe two at a time), and
- answer all questions.

Question: For your Mod-N counter, what is the value of N? Given that value of N, what base is your counter implementing?

Going more in depth – The 7493 Ripple Counter (Optional Reading):
The 7493 consists of 4 flip-flops with J-K inputs unconnected. In a TTL chip, unconnected inputs behave as logic-1, so J=K=1. As a result, each flip-flop is configured as a "T" and will toggle every time it is clocked; i.e. when its clock voltage falls (the 7493 has a negative edge triggered clock input). Refer to the diagrams of the 7493 on the first and last pages of the last section of the manual, "Description of IC'S".

QB,QC,QD are chained together to form a 3-bit counter; the output of each is connected internally to the clock of the next (QB goes to QC's clock, etc.). For a 4-bit counter, QA must likewise go to QB's clock, but *externally* (pin 12 to pin 1). In the resulting counter, **QD is the most significant bit (msb)**; i.e. the 4-bit counter output is [QD QC QB QA]. Clock pulses are input at pin 14.

The reset inputs Ro(1) and Ro(2) of the counter are connected internally to an AND gate. When both reset inputs are high, the AND's output goes high and resets the counter. Resetting is asynchronous; it is does not wait for the clock and occurs almost immediately. (Note: if the reset inputs are left unconnected, they are considered to be high. Therefore the counter is continuously reset to 0000. So ground at least one of them.)

USING THE AGILENT DSO1012A OSCILLOSCOPE

Here are some general comments. The detailed instructions follow.

A) Triggering: you always want to trigger on the waveform of the most significant bit of the counter outputs. Let's call these XYZ. Then bring X to a scope input channel, say CH1, and select that as the source channel for the trigger. (The other variable will go to CH2.)

You also want to trigger on the *falling edge* of X since when X $\rightarrow$ 0, it marks the start of a new period for both waveforms.

Finally, the trigger level needs to be somewhere in the middle of X's waveform. Either adjust the trigger-level line up or down with the Level knob or just push the knob to set the line halfway up X's waveform.

B) Preparing the screen image for transfer to a Word document: first press the Display button. Your options appear at the right of the screen.

- Increase the waveforms' intensity to 100% to make them more visible.
- Increase grid brightness to 100% for the same reason.
- Invert the screen colors so the waveforms and grid divisions appear dark against a white background.

C) To save screen image to your flash-drive, see instructions below.

To set up triggering:

1. Press ***Menu*** button below Trigger Level knob.
2. In the menu at the right of the screen, press ***Mode***.
3. Select ***Edge*** by repeatedly pressing ***Mode*** or by turning the "entry" knob. ↻
4. Next press ***Source*** and choose the channel you want to do the triggering, ***CH1*** or ***CH2***.
5. Finally, press ***Slope*** and choose ***Falling Edge***.

To adjust the triggering level:

1. Turn the ***Trigger Level*** knob so that the horizontal line lies somewhere within the triggering waveform. Or just push the knob to set the level in the middle of that waveform.

To set up the screen image:

1. Press ***Display .***
2. In the menu at the right of the screen, press ***Intensity*** and turn the entry knob ↻until 100%.
3. Then, go down the menu to ***Gridbright*** and, again, turn the entry knob until 100%.
4. Finally, press ***Screen*** to switch from Normal to ***Inverted***. The screen should now show waveforms against a white background with gridlines clearly visible.

To adjust trigger position:

1. Turn the ***Horizontal*** right-hand knob to move the trigger indicator to the first vertical grid line from the left edge of the screen. This will be the starting point of your display.

To save screen image to flash drive:

1. Insert flash drive.
2. Press ***Stop*** button. (By displaying only one sweep you eliminate noise fluctuations that may occur when display is running.)
3. Press ***Save/Recall*** button*.*
4. Press ***Storage*** button at right of screen.
5. Select ***24-Bitmap*** by repeatedly pressing ***Storage*** or turning the entry knob (image will be stored in this format).
6. Press the ***External*** button and then press ***New File***.
7. Type out a filename:

- Toggle between filename and keyboard fields by pressing ⇧⇩.
- In the keyboard field, select a character for your filename by turning entry knob.
- Press the knob after each character is chosen. It should appear in the filename field.
- You can enter characters in upper or lower case by toggling the lower right-hand key A/a, but it's probably not worth the bother.
- In the filename field, use the entry knob to select a character you wish to delete. Then press the X button. Or delete all characters to the left of some starting point in the name by repeatedly pressing X.

8. Press the ***Save*** button. The file should be saved on your flash drive with the extension ".bmp".