

Experiment 7

Digital Logic Devices and the 555 Timer

Purpose: In this experiment we address the concepts of digital electronics and look at the 555 timer, a device that uses digital devices and other electronic switching elements to generate pulses.

Equipment Required:

- **Voltmeter** (Rensselaer IOBoard)
- **Oscilloscope** (Rensselaer IOBoard)
- **Function Generator** (Rensselaer IOBoard)
- +5V Power Supply (Rensselaer IOBoard)
- 555 Timer IC

Helpful links for this experiment can be found on the links page for this course:

<http://hibp.ecse.rpi.edu/~connor/education/EILinks.html#Exp7>

Part A – Basic Logic Gates

Background

Digital logic gates: All digital logic gates are based on binary logic. Binary logic has two values, called TRUE and FALSE or LOGIC 1 and LOGIC 0 or ON and OFF or HIGH and LOW. The corresponding binary number can have two possible values, 1 and 0. Digital logic gates perform many common logic operations on binary signals, such as AND, OR, NOT, NAND, and NOR. The table in figure A-1 contains the common symbol for each type of gate, an expression for the function of the gate in Boolean algebra, and a truth table for the device. The truth table shows how the gate will behave for all possible combinations of digital inputs.



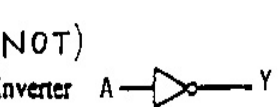
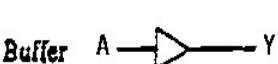
Name	Symbol	Algebraic function	Truth table															
AND		$Y=AB$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1
A	B	Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$Y=A+B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
(NOT) Inverter		$Y=\bar{A}$	<table border="1"> <thead> <tr> <th>A</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	Y	0	1	1	0									
A	Y																	
0	1																	
1	0																	
Buffer		$Y=A$	<table border="1"> <thead> <tr> <th>A</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </tbody> </table>	A	Y	0	0	1	1									
A	Y																	
0	0																	
1	1																	

Figure A-1

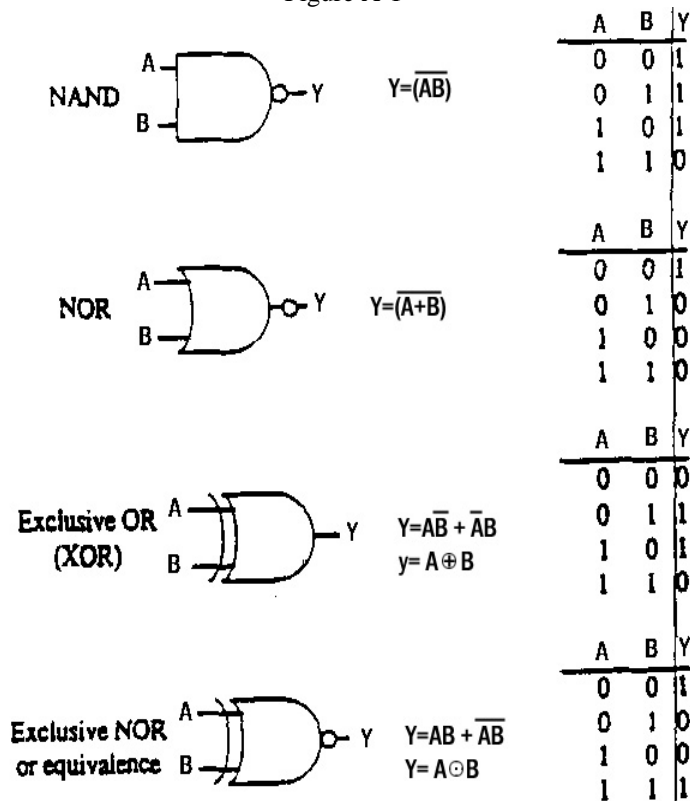


Figure A-1 (continued)

Digital logic chips: In TTL digital electronic circuits, the representation of binary numbers in terms of voltages is about 5 volts for LOGIC 1 and about 0 volts for LOGIC 0. About 5 volts usually means a voltage between 2 and 5 volts while about 0 volts means any voltage in the range 0 to 0.8 volts. The voltage levels when using TTL devices must always be in the ranges indicated, or the circuits will not function correctly. LOGIC 1 and LOGIC 0 are the only output levels one should see with logical devices. This is one characteristic that makes them differ from analog devices. They also switch very fast from one state to the other. Switching speeds are usually much faster than for analog devices, especially cheap devices like the 741 op amp.

A digital chip generally has 14 or 16 pins. It usually contains more than one of the same logic gate. (For example, a 14 pin chip will have six single-input gates or four 2-input gates.) By convention, the upper right pin on a digital chip is always connected to HIGH (+5 volts) and the bottom left pin to LOW (0 volts). On a 14 pin chip this corresponds to pin 7 (0V) and pin 14 (5V) and on a 16 pin chip, pin 8 (0V) and pin 16 (5V). These two connections provide reference values for the operations the chips perform. Generally circuit diagrams do not show these two reference connections. If you forget to connect these two pins, your circuit will not function correctly.

Timing diagrams: Timing diagrams are a special kind of transient output which is useful for viewing many binary signals. Since the voltage levels of binary signals can only be either high or low, knowing the exact voltage level is not as significant as knowing when the different signals transition from high to low (or low to high). A timing diagram is much easier to read when you need to compare many binary signals. Unlike a regular PSpice plot (where all signals are displayed with the same voltage and time axis) a timing diagram displays the signals on separate lines with the same time scale. A sample is shown in figure A-2.

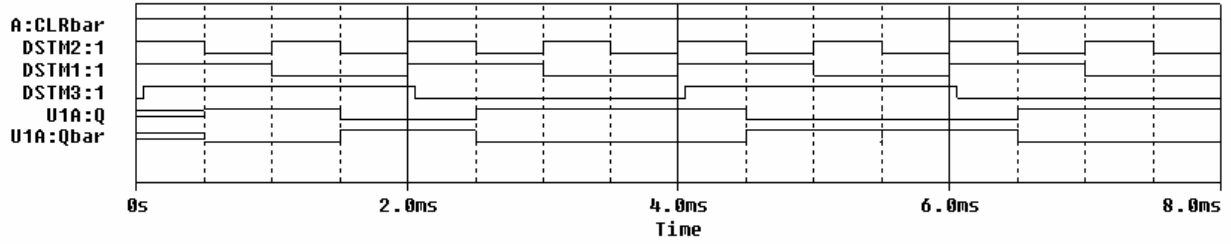


Figure A-2

Note that there are six signals shown on this plot. We can see where they are high and low and we can also see the relative time that each signal changes state.

Experiment

Truth Tables of Basic Logic Gates

We will now consider three basic logical elements: a two input NOR gate, a three input NAND gate and an INVERTER.

- Wire the circuits in figure A-3 on your protoboard. Do not forget to tie pin 14 to 5V and pin 7 to 0V on each chip. (Also note that the 74107 chip and the 7410 chip in your kit are not the same chip. Here we want the 7410.)

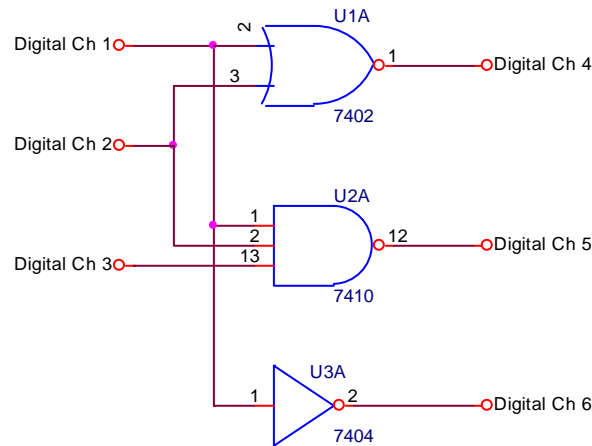


Figure A-3

Set Digital Ch 1, 2 and 3 to Output. Set Digital Ch 4, 5, and 6 to Input

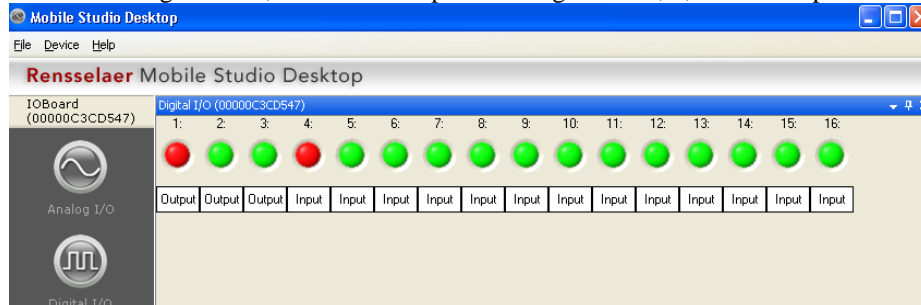


Figure A-4 Select the Digital I/O Instrument and set channels 1, 2 and 3 to output and channels 4,5, and 6 to input.

- Consider all possible combinations of inputs to generate a truth table for each device.

- The NOT gate has only one input. Therefore, we need only need to observe the output when the input is HIGH (5 volts) and LOW (0 volts).
 - First set channel 1 to high (green) and record the status of channel 6.
 - Then set channel 1 to low (red) and record the status of channel 6.
 - Does the gate invert the input?
 - Take a picture (screen capture) of one of the input/output trace combinations.
 - Draw a truth table for this gate on the output plot. Include this plot in your report.
- Now you will repeat this process for the other two gates.
 - The NOR gate has two inputs, so we must observe the output at pin 1 (Digital I/O channel 4) for all possible combinations of binary inputs at pins 2 and 3: (LOW, LOW), (LOW, HIGH), (HIGH, LOW) and (HIGH, HIGH).
 - Take a picture (screen capture) of one of the input/output trace combinations.
 - Record the truth table on the plot. Include this plot in your report.
 - The NAND gate has three inputs. How many combinations of HIGH and LOW are required to fully test this gate?
 - Record the input and output for this gate in a truth table on a sample screen capture, as well. Include this plot in your report.

Simulation of Basic Logic Gates

We will now wire the same three basic logical elements using PSpice.

- Create the following circuit (figure A-4) in PSpice.

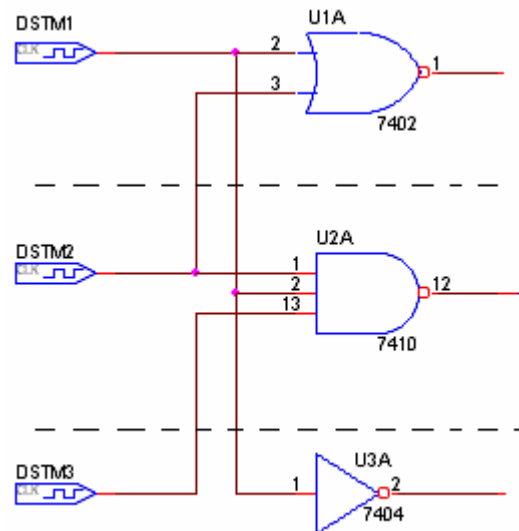


Figure A-4

- Wiring the circuit in PSpice is somewhat different than on the protoboard.
 - PSpice assumes that the +5 volt and 0 volt references have already been wired, so you do not need to make these connections.
 - We cannot simply move the wires to record all possibilities. Therefore, we use digital clocks with different pulse lengths to create the signals we need to test the gates.
 - We have removed the resistors connecting the gate outputs to ground. This tells PSpice to output timing diagrams by default.
- Now we need to set the clocks up to work with different pulse lengths.
 - Use DigClock in the SOURCE library
 - Use the default settings for DSTM1 (no delay, on time = 0.5us, off time = 0.5us).
 - For DSTM2, double the on and off times to 1us.
 - For DSTM3, double them again to 2us.
- Run a simulation
 - Simulate for 8us with a step size of 0.01us.
 - Display all the inputs from the clocks and the output of each of the three gates.
 - Produce a hardcopy of the timing diagram with the inputs and the outputs for all three gates.
 - Mark the output trace for each gate on the diagram. For each gate, generate the truth table for the device based on the outputs and inputs you observe on the timing diagram. Write them on the output plot.
 - Include this plot in your report.
 - Do your results agree with the truth tables you found using from the circuits you built?

Summary

Basic logic gates allow you to use electronic signals to perform operations on digital signals. They can also be combined to perform more complex operations, such as addition and subtraction. This makes them a basic building block of digital computers.

Part B – Flip Flops

Background

Flip Flops: It is possible using basic logic gates to build a circuit that remembers its present condition. These circuits are called flip flops. The PSpice symbol for a J-K flip flop is pictured in figure B-1. There are several different kinds of flip flops with slightly different characteristics. In this course we use the JK flip flop. JK flip flops, like other flip flops have four inputs, two outputs and the usual two power connections (V_{CC} and ground). The outputs are labeled Q and \bar{Q} (also called Qbar and NQ). They are complements of one another. Thus, when Q is LOW, Qbar is HIGH, etc. CLK is the digital clock. A flip flop only changes its output when the clock pulse at CLK goes from HIGH (5V) to LOW (0V). This is called the “falling edge” of the clock. The input \overline{CLR} , when LOW, will reset the outputs to a known state. It has the following truth table:

J	K	C	Q	\bar{Q}
0	0	p	no change	
0	1	p	0	1
1	0	p	1	0
1	1	p	toggle	

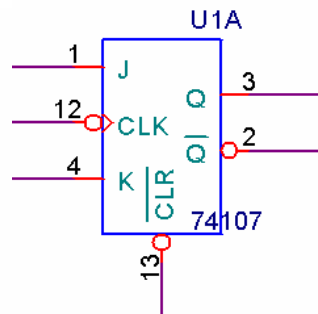


Figure B-1

Note that the flip flop is an edge-triggered device. This means that instead of changing state as soon as its inputs change, it “waits” until it receives a falling edge at the clock input (CLK). To decide how to set the output, the flip flop “looks” at the values at the inputs at J and K AND at the current value of the output. Based on these three values, it decides how to reset the output.

You may be familiar with clocks. They are used to coordinate the instructions performed by the CPU and other devices in a computer. When a computer has a clock speed of 1 Giga Hertz. It can handle 1×10^9 instructions per second. One for every clock cycle. The flip flop works on the same principal. With every clock cycle, it looks at its inputs and changes state accordingly.

The flip flop is a memory device. If both inputs are zero, its output will remain the same indefinitely. A bank of 4 flip flops can store a digital byte of memory in a computer. If you want to change the value of a single bit of that byte to zero, you can set the inputs to the corresponding flip flop to $J=0$ and $K=1$. On the next clock cycle, the output will change to zero. If you want to change a single bit of the byte to one, you can set the inputs of the corresponding flip flop to $J=1$ and $K=0$. On the next clock cycle, the output will change to 1. You can also toggle the value of the flip flop output by setting both inputs to 1.

Timing diagram for a flip flop: To illustrate the behavior of a JK flip flop, we wired the circuit in figure B-2 in PSpice and created the timing diagram shown in figure B-3.

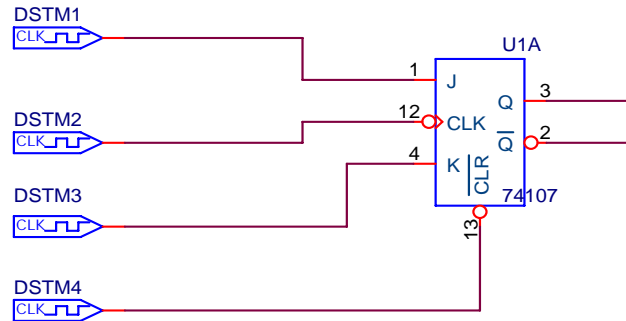


Figure B-2

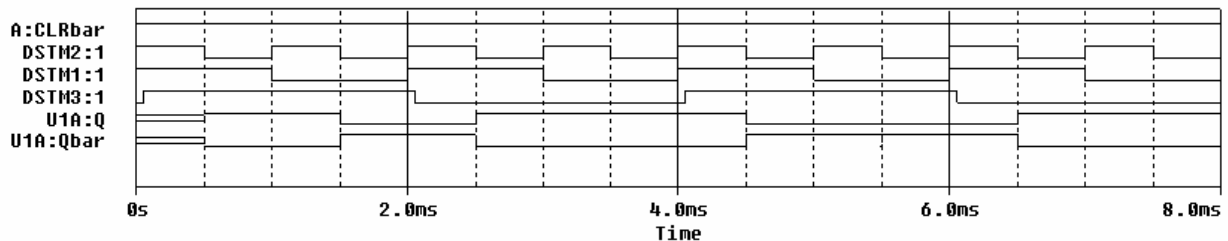


Figure B-3

DSTM2 is the clock. Each time the clock falls, look at the values of the inputs at J (DSTM1), K (DSTM3), and the output signal (U1A:Q). If the truth table is correct, what should the output value at U1A:Q be for each combination? Is the timing diagram correct? Please check the datasheet for the SN74107 flip flop located on the course web page for details about this device.

Noise and Digital Circuits: Any time you build a circuit, there will be noise. In an edge-triggered device, where timing is a factor, noise can cause many problems. A noise spike might be interpreted as the falling edge of the clock. If this happens, the flip flop will change state at the wrong time and possibly with the wrong inputs. To avoid this problem, it is essential to use a *bypass capacitor* in every timed digital circuit you build. A bypass capacitor is simply a capacitor placed between the source voltage and ground. What it does is filter out high frequency noise, so that any spikes that might be misinterpreted are filtered out. Figure B-4 shows a bypass capacitor.

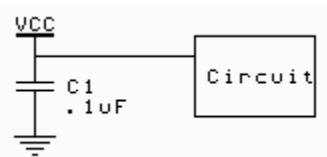


Figure B-4

To understand how the bypass capacitor works, we only need to recall that a capacitor looks like an open circuit at low frequencies and a short at high frequencies. Digital signals consist of two DC values, 0V and 5V. A DC voltage has zero frequency. Therefore the bypass capacitor will look like an open circuit and all of the DC signal will pass into the circuit. A noise spike is a very sudden high frequency event. At high frequencies, the capacitor looks like a short. Therefore, the high frequency event will pass through the capacitor to ground and not continue on to the circuit.

When a signal changes from low to high (or high to low), the temporary noise of the transition may cause the device to make the wrong output decision. Since the values of the inputs, as they are changing, is indeterminate, the output that a flip flop will generate if the inputs are changing is unknown. In PSpice, this unknown state is depicted by a double red line. If you look at the timing diagram above you will see that both Q and Qbar start out in an unknown state until the first falling edge of the clock. Also notice that the transitions for J and K take place well before the falling edge of the clock. We deliberately set up the

timing of DSTM1 and DSTM3 so that they do not change the state of the input at the same time the clock transitions from high to low. If we had, PSpice would continue to display the double red line, signifying that it cannot properly set the output because it isn't sure what the inputs are supposed to be.

Experiment

The JK Flip Flop

In this part of the experiment, we will look at the behavior of a flip flop on your protoboard.

- Set up the 74107 JK flip flop on the protoboard.
 - Provide 0V at pin 7 of the chip and +5V to pin 14.
 - Place a 0.1uF by-pass capacitor between 0 and 5V.
 - Use three Digital I/O channels to create the J, K, and Clock signals. These should be set as outputs
 - Use 2 Digital I/O channels to read the Q and Qbar signals. These are set as inputs.
 - Set the CLR to zero to be sure that the FLIP FLOP begins in a known state. THEN attach it to +5V to enable the function of the chip. (Or use yet another Digital I/O channel as an output and cycle it low and then high.)
 - Run all possible combinations of logic levels for J and K, see below. *Each time you change J or K, you need to cycle the Digital I/O that is the Clock signal. This means to make the changes in J and/or K and then switch the clock from low to high and then back to low. By monitoring the Q output, you should be able to complete the table below. It may take a little thought.*
 - Fill in the following truth table for this device.

Q (before pulse)	J	K	Q (after pulse)	Qbar (after pulse)
0	0	1		
1	0	1		
0	1	0		
1	1	0		
0	0	0		
1	0	0		
0	1	1		
1	1	1		

- Take a screen capture of the Digital I/O status for the case when J and K equal to 1, Label each Digital I/O channel used with J, K, Clock, Q and Qbar. Copy the truth table above for the flip flop onto a print out of the Digital I/O status screen capture. Include this plot in your report.

Summary

The JK flip flop is an edge-triggered device that uses an external clock to coordinate state changes with other clocked devices in a circuit. It is capable of holding its output stable, changing its output directly to high or low, or toggling the output to the opposite of its current value. It can be used as a memory device or as a building block to create more complex devices, such as counters.

Part C – Counters

Background

Binary Counters: JK flip flops can be connected in a counter configuration as shown in figure C-1. The output of each flip flop is used as the input clock to the next flip flop. On all flip flops, J and K are tied high. This means that they will toggle on each pulse of their clocks. The first flip flop toggles every clock cycle. The second flip flop toggles every time the output from the first flip flop changes. This causes it to toggle at a rate equal to half of the clock rate. By the same reasoning, the third flip flop toggles at a rate $\frac{1}{4}$ of the clock rate.

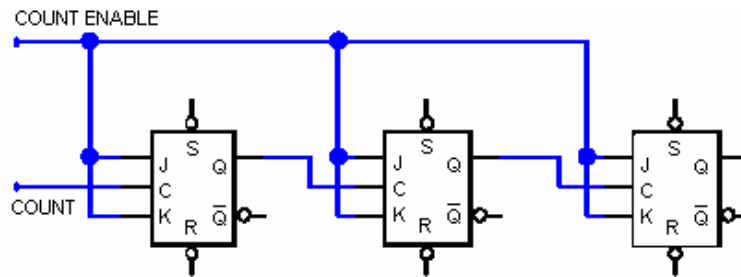


Figure C-1

If we attach a bit of a binary number, (b_0, b_1, b_2), to the output of each flip flop (b_0 to the first, b_1 to the second, and b_2 to the last). We get a pattern out which corresponds to the binary counting shown in figure C-2 below:

Decimal	Binary	Hex	Octal	Decimal	Binary	Hex	Octal
00	00000	00	00	08	01000	08	10
01	00001	01	01	09	01001	09	11
02	00010	02	02	10	01010	0A	12
03	00011	03	03	11	01011	0B	13
04	00100	04	04	12	01100	0C	14
05	00101	05	05	13	01101	0D	15
06	00110	06	06	14	01110	0E	16
07	00111	07	07	15	01111	0F	17
				16	10000	10	20

Figure C-2

The lowest order bit of the binary numbers toggles from 0 to 1: (0-1-0-1-0-1-...). The second order bit also toggles between 1 and 0, but at half the rate: (0-0-1-1-0-0-1-1-...). The third order bit toggles also, but at half the rate of the previous bit: (0-0-0-0-1-1-1-1-0-0-...). The pattern in the table is the same as the pattern created by the cascaded set of flip flops. Thus, the counter is counting.

It is not necessary for us to configure several flip flops to create a counting circuit, because this is already done in many kinds of chips. The SN74393, for example, has two sets of four JK flip flops connected as binary counters.

Light Emitting Diodes: LEDs (light emitting diodes) are very useful when dealing with digital circuits. Because they have two states, ON and OFF, you can hook them to a binary signal, and use them directly to observe whether or not the signal is HIGH or LOW. Although there are conventions for the polarity of diodes, many manufacturers do not seem to follow them. The best way to determine the polarity of a diode is to place it directly between +5V and ground. If it lights, use it in that direction. If it doesn't light, try

switching the polarity. If it lights that way, then use that polarity. If it lights in neither polarity, throw it away. It is burnt out.

LED's are like light bulbs. They burn out after a while. In order to prolong the life (and brightness) of an LED, it is a good idea to wire it in series with a small resistor (330 ohms is about right). If the LED is not bright enough, you can replace the resistor with a smaller one.

Experiment

Simulation of a cascaded binary counter

In this part of the experiment, we will use PSpice to create a simulation of a two counters cascaded together. This will allow us to count to numbers greater than 15.

- Simulate the circuit in figure C-3 in PSpice

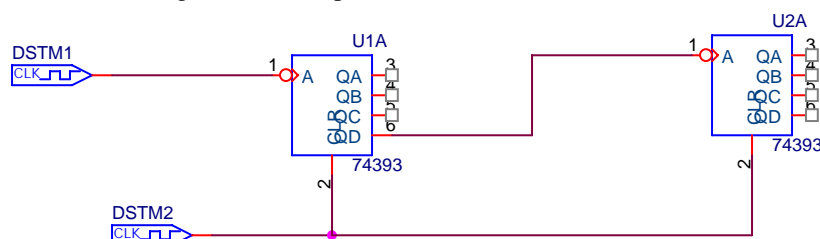


Figure C-3

- DSTM1 is the actual clock for the counter, DigClock in SOURCE library with OFFTIME = ONTIME= 0.5us.
 - DSTM2 (also DigClock) is set up so that it first clears the counters, lets them count, and then clears them again. (OFFTIME=25us, ONTIME=2us)
 - *If the PSpice models for counters or flip flops are not cleared initially, they will not give any data.*
 - By connecting the two counters together as shown, the sequence of numbers 2QD, 2QC, 2QB, 2QA, 1QD, 1QC, 1QB, 1QA (IN THAT ORDER) form the binary number. Since the counter is set up to count clock pulses, it will count up from 0 (at reset) to the number of pulses sequentially.
- Run the simulation
 - Generate the output for this circuit for time 0 to 30us using 1us increments.
 - You should display the reset pulse, the input clock at pin 1 of U1A, and all eight of the counter outputs (QA,QB,QC,QD for both counters).
 - Using the timing diagram, verify that the counters are actually counting. You can use the cursor to easily get your binary number.
 - What is the highest number it counts to before it resets? At what time does it reset? Express this number both as a binary number and a decimal number.
 - Write it on the timing diagram for the circuit. How many pulses will the clock have to cycle through between the time it is reset and when it hits its maximum value of 11111111?
 - Include the timing diagram output in your report.

Build a counter circuit

In this part of the experiment, we will build a counter circuit on the protoboard.

- Build the circuit in figure C-4 on your protoboard.

Use MAX473
same pins as uA471

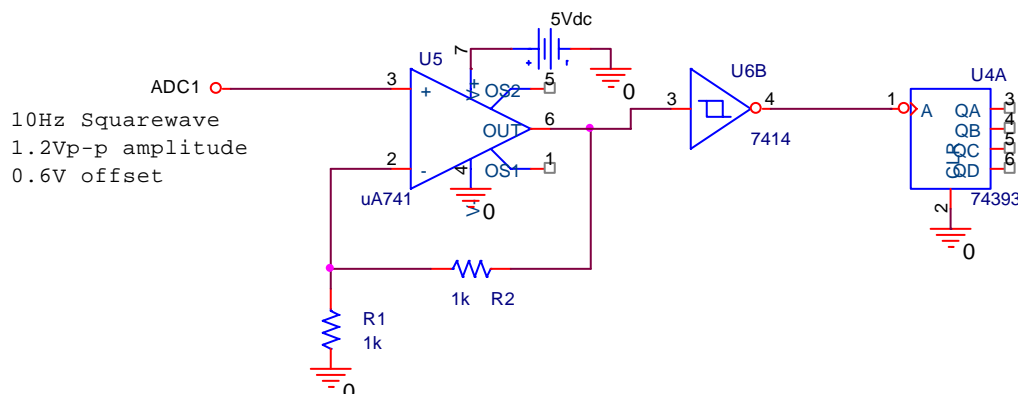


Figure C-4 ADC1, Function Generator 1, has a maximum output of 1.25V. The op-amp amplifies this by a factor of 2. The Schmitt trigger inverter cleans up the signal. Don't forget that the 74LS14 and the 74LS393 both need power and ground.

- Use the function generator for the clock. Set it for a square wave with a frequency of 10Hz. We are using a fairly low frequency so that we will be able to see the switching with the LEDs. *Use the offset feature to shift the square wave up such that it cycles between 0V and 1.2V.* Use a Maxim MAX473 op-amp to boost the signal by a factor of 2. This chip works with a single +5V power supply and can drive the output to very near the power supply voltages. Be sure that you use the 'scope to check the operation of the function generator.
- Tie pin 7 to ground and pin 14 to 5V on both the 74LS14 and the 74LS393 logic chips.
- Place a 0.1uF by-pass capacitor between +5 and ground.
- Check your LED's by connecting one with a 220 Ohm resistor between +5V and ground to see which polarity causes it to light. Place it in the circuit with the correct polarity.
- If you cannot find 220 ohm resistors, use ones close to 220 ohms.
- Set the CLR to +5V to be sure that the counter begins counting at zero. THEN attach it to 0V to enable the chip to function. Note that this is the opposite of the flip flop. The circle at the input to the clear tells you whether the CLR signal must be high or low. (Circle→Hold high for function. No Circle→ Hold low for function. If you tie this pin incorrectly, the chip will continuously reset itself and it will not work.
- Once you have the circuit working, observe the output.
 - Are they changing at the expected rates? Change the function generator frequency if it helps you determine the frequencies.
 - Place channel 1 of the scope on the clock signal and set the time controls so that it displays 50 clock cycles. Do not change the time scale as you take your pictures. All four should show 50 clock cycles for comparison purposes.
 - You will need to take four pictures using the Mobile Studio. Include them in your report.
 - Channel 1 = clock and Channel 2 = QA
 - Channel 1 = clock and Channel 2 = QB
 - Channel 1 = clock and Channel 2 = QC
 - Channel 1 = clock and Channel 2 = QD
 - What do you observe about the rates of the different signals with respect to the clock? Can you tell which output corresponds to which bit in the binary number?
 - **Keep this circuit for part D.**

Part D – The 555 Timer

Background

The 555-Timer: The 555-timer is a chip that allows us to create a variety of useful digital and analog signals. Much like the op-amp, it can be used to perform different functions depending upon what circuit you place it into. The 555-timer can be used to generate digital pulses. When it is wired as a “one-shot” (also called mono-stable mode), it generates a single, clean, digital pulse at the output, when it experiences a (possibly noisy) pulse at the input. This is useful when de-bouncing a mechanical switch. In this experiment, we are concerned with the 555-timer when it is wired in astable mode. This is also called an astable multivibrator. In this mode, the 555-timer circuit creates a stream of regular pulses. The wiring diagram for the 555-timer in astable mode is shown in figure D-1.

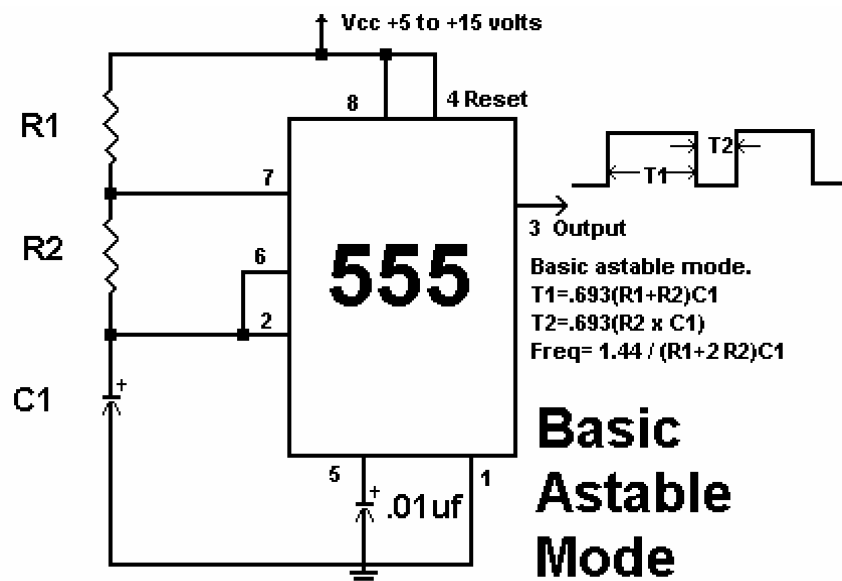


Figure D-1

Inside the 555-Timer: In order to understand how the 555-timer can create this regular stream of pulses, we need to look inside and see how it functions. As you can see in figure D-2, the inside of the device contains many of the components we have studied in experiments 6 and 7.

By selecting just the right values for the resistors and capacitors in this circuit, we can make the voltage at pin 3 (the OUTPUT) go from zero to V_{CC} at whatever rate we desire. We can also control the percentage of time that the output will be on relative to the length of an entire cycle. The equations that govern this behavior are:

$$T_{ON} = 0.693(R1 + R2)C1 \quad T_{OFF} = 0.693(R2)C1 \quad f = \frac{1.44}{(R1 + 2R2)C1}$$

Pulse width modulation One of the most important things we can use 555 timers for is to control and drive a large variety of systems with pulse width modulation. Please read over the links on motor control and flow valve control on the course links page. The power of pulse width modulation comes from its simplicity. Rather than controlling the flow of some liquid by carefully opening a valve part way, you can alternately open and close the valve fully in such a manner that the average open time produces the same effect as a partially open valve. In effect, the rate of flow is controlled by the duty cycle of the controlling voltage. It is much easier to fully open or close a valve than to precisely open it part way. One can also apply power to a motor in this manner to control the speed of rotation. The key goal of this modulation process is to achieve a desired average value for some process. The range of possibilities is shown in figure D-6 where A has a high duty cycle (fast) and C a low duty cycle (slow).

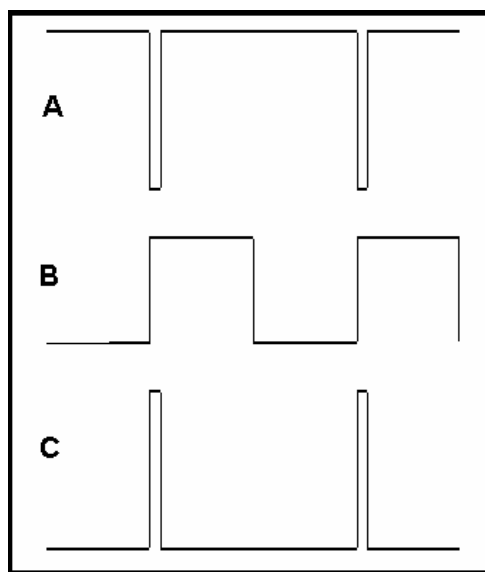


Figure D-6

Experiment

Simulation of a 555-timer circuit.

In this part of the experiment, we will use PSpice to demonstrate the operation of the 555-timer chip in astable mode.

- Wire the circuit in figure D-7 in PSpice

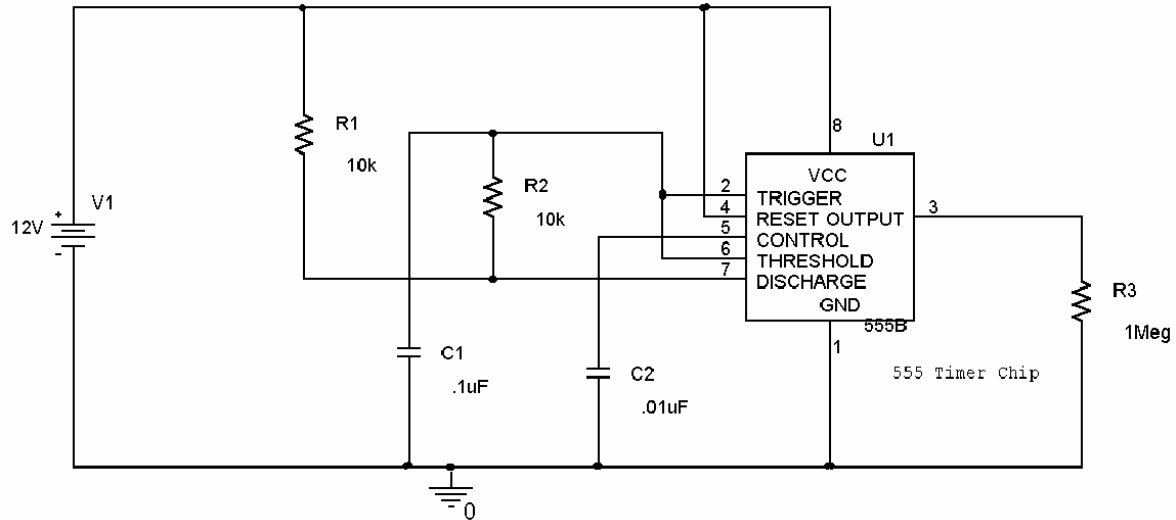


Figure D-7

- Run the simulation
 - Perform a transient analysis in increments of 2us up to 5ms.
 - Plot the threshold/trigger, discharge and output voltages. The trigger voltage is pin 2 and the threshold voltage is pin 6. (They are tied together.) The discharge voltage is pin 7, and the output is taken at pin 3.
 - Print your plot and include it in your report.
 - Verify that the timer output changes according to the rules listed for the 555-timer in astable mode. Use the plot to find the time period that the output is ON and the time period that the output is OFF. *Note: Do not use the first cycle of pulses produced by the timer circuit. It takes one cycle to settle in to its steady-state condition.* One of these times should be equal to $0.693(R1+R2)C1$ while the other should be equal to $0.693(R2)C1$. Which is which? What is the total period of this output?
 - Which of the three signals on your plot corresponds to the charging and discharging of the capacitor, C1? To what voltage does it charge each time? To what voltage does it discharge? What is the rate of charge? Is the rate of discharge the same?

- Determine the average voltage of the signal.
 - Change the end time for the transient analysis to 60ms.
 - Display only the output voltage (pin 3) on your plot.
 - Rerun the simulation.
 - Add a trace of the average of the output using the average function, AVG(). This should add a trace of the time average of the output voltage. The average at any instant of time is the average of the voltage from time = 0 to the time of interest. Thus, you should see that the average asymptotically approaches a particular value.
 - Print this plot and write the approximate voltage the average is approaching on the plot.
 - Include this plot in your report.

- Find a larger and smaller average voltage for your circuit
 - Find an expression for the duty cycle of an astable 555-timer circuit using the equations given. Consider what relative values of R1 and R2 would produce the highest duty cycle and what relative values of R1 and R2 would produce the smallest duty cycle.
 - Now, using any combination for 3kΩ, 10kΩ or 30kΩ resistors for R1 and R2, find the combination of two resistors which results in the largest average voltage and the combination of two resistors which results in the smallest average voltage.
 - Print your plot for each of these two cases, write the values for R1 and R2 you used on each plot. Include these two plots in your report.

- Verify in each case that the pulses produced by the multi-vibrator circuit obey the design rules. If the simulation does not work, the design rules are probably violated.

Build the 555-timer circuit on your protoboard

In this part of the experiment, we will build the astable multi-vibrator and use it as the clock for your timer circuit.

- Wire the astable multi-vibrator shown in figure D-8 on your protoboard.

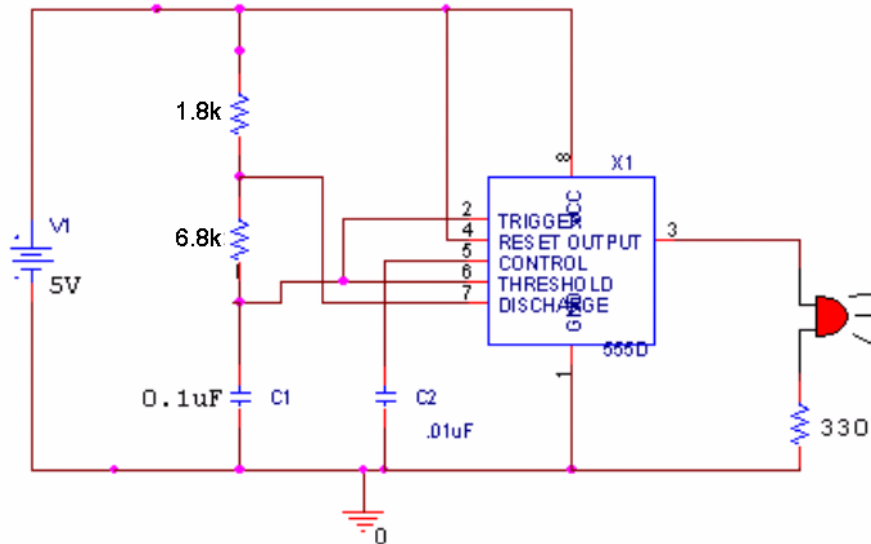


Figure D-8

- Record your results
 - You will not be able to see the LED flash because the period of your circuit is too fast.
 - Take a Mobile Studio picture of your output.
 - Print this plot and include it in your report.
 - What is the period of your output signal? What are the off-time and on-time? Use the equations to calculate what these values should be. How do they compare?
- Slow down the pulses so that you can observe them with the LED.
 - Keeping the resistors R1 and R2 the same, determine a new value for C1 such that the period of the timer will be around 1 second.
 - Replace C1 in your circuit and observe the LED. Does it flash once a second?
 - What is your on-time and off-time now? How are these related to the on- and off- times of the original circuit? Why does this relationship hold?
- Use the 555-timer circuit as the clock for your counter.
 - Remove the function generator from pin 1 of the counter.
 - Connect the output at pin 3 of the 555-timer circuit to pin 1 of your counter circuit.
 - Does the counter count the 555-timer pulses?
 - Can you see the lower order bits change now?

Report and Conclusions

The following should be included in your written report. Everything should be clearly labeled and easy to find. Partial credit will be deducted for poor labeling or unclear presentation.

Part A (12 points)

Include the following plots:

- 1) Mobile Studio plot of single trace from NOR gate with truth table (2 pt)
- 2) Mobile Studio plot of single trace from NAND gate with truth table (2 pt)
- 3) Mobile Studio plot of single trace from NOT gate with truth table (2 pt)
- 4) PSpice timing diagram for the three gates in the circuit with output traces marked (2 pt)

Answer the following questions:

- 1) What are the actual voltage values you observed as HIGH and LOW states in the hardware realization of the circuit? (2 pt)
- 2) How do the truth tables generated using the actual chips correspond to the truth tables you generated using your PSpice output? (1 pt)
- 3) If you had a gate with four inputs, how many cases would you have to consider to create the truth table? (1 pt)

Part B (10 points)

Include the following plots:

- 1) Mobile Studio plot of single trace from flip flop with truth table. (2 pt)

Answer the following questions:

- 1) Flip flops are called memory devices. Why do you think this is true? (2 pt)
- 2) Show that the flip flop is giving the correct output at the clock cycles (A, B, C and D) indicated on the timing diagram in figure S-1. DSTM2 is the clock signal, DSTM1 is J, and DSTM3 is K. Show how the truth table you found for the actual flip flop is consistent with the timing diagram at those four points. (6 pt)

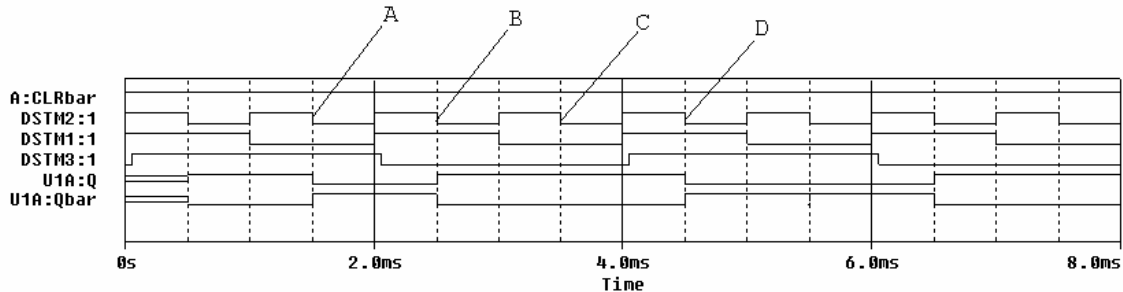


Figure S-1

Part C (14 points)

Include the following plots:

- 1) PSpice timing diagram of counters (2 pt)
- 2) Mobile Studio plot of clock and QA (2 pt)
- 3) Mobile Studio plot of clock and QB (2 pt)
- 4) Mobile Studio plot of clock and QC (2 pt)
- 5) Mobile Studio plot of clock and QD (2 pt)

Answer the following questions:

- 1) For the counter circuit configuration just studied, what is the highest number it counts to in the time shown on your output? Express as both a binary and decimal number. (2 pt)
- 2) How many pulses will the clock have to cycle through after it resets before the counter hits its maximum value? (1 pt)

3) Which output of the timer (QA, QB, QC, QD) correspond to the bits in the binary number (b3 b2 b1 b0):
 $N = b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$. (1 pt)

Part D (36 points)

Include following plots:

1. PSpice plot for the astable circuit with R1=10k and R2=10k. (2 pt)
2. PSpice plots for the average voltage of the astable circuit with R1=R2=10k. (1 pt)
3. PSpice plot for the average voltage of the astable circuit with highest duty cycle. (1 pt)
4. PSpice plot for the average voltage of the astable circuit with lowest duty cycle. (1 pt)
5. Mobile Studio plot of output from 555-timer circuit when R1=1.8K, R2=6.8K and C1=0.1μF (4 pt)

Answer following questions:

1. What are the on-time, the off-time, and the period of the signal in plot 1)? What are the calculated values for these? Are they consistent? (4 pt)
2. What are the maximum and minimum values for the voltage across the capacitor C1 (at pins 2 and 6)? (Ignore the voltage at times before it reaches steady state.) Why do these values make sense? (3 pt)
3. Calculate the value of τ (the decay constant) that controls the rate at which the capacitor C1 charges. Calculate the value of τ (the decay constant) that controls the rate at which the capacitor C1 discharges. (4 pt)
4. What is the minimum duty cycle that can be obtained from the astable multivibrator we modeled using PSpice? Can you show this mathematically using $\text{Duty Cycle} = T_1/(T_1+T_2)$? (3 pt)
5. What was the average voltage for your original circuit? What were the minimum and maximum average voltages when you considered different combinations of R1 and R2? (3 pt)
6. What are the on-time, the off-time, and the period of the signal in plot 5)? What are the calculated values for these? Are they consistent? (4 pt)
7. How did you find the value for C1 that gave the circuit you built a one second period? What value did you find? (2 pt)
8. What are the calculated on-time, off-time and period values for your circuit with the new capacitor? How do these relate to the initial values? Why? (4 pt)

Organization and responsibilities (8 points)

1. Discuss mistakes and problems (6 pts)
2. List member responsibilities (2 pts)

Total: 80 points for write up + 20 for attendance = 100 points

**Attendance: 3 classes (20 points) 2 classes (10 points) 1 class (0 points) out of 20 possible points
 Minus 5 points for each late.
 No attendance at all = No grade for experiment.**