Mission

This project is due by 1:59pm on April 22, 2008. You are expected to begin wiring (NOT designing!) your project 3 at 2:00pm on that day. You have been contracted to build a Launch Sequencer (LS) for the Space Shuttle. The purpose of the LS is to control the final sequence of events starting 15 seconds prior to launch. The LS must have the following:

1) An **ENABLE display LED** (LED 0). This light is normally on; if a problem develops, this light should turn off.
2) A **time display**, using two 7-segment displays. When the circuit is first powered on, the display must indicate 15 seconds. The time should hold at 15s until a NASA engineer begins the countdown using the leftmost momentary (blue) switch.
3) A **momentary switch** (as mentioned earlier in item 2), which begins the countdown process from 15s. The display should use blanking as appropriate. Subsequent toggling of this switch should not affect the circuit.
4) A **manual ENABLE/ABORT switch** (switch 0). Normally, this switch is low. If the switch is changed to ABORT (high) at any time between 15s and 3s, the countdown is aborted. Whenever the launch is ABORTED, the timer must stop, and the enable light must go off. Once the launch is aborted, it cannot be restarted, even if the switch is subsequently returned to ENABLE. The stopped timer must continue to display the time at which the launch was aborted.
5) When t = 14.0 seconds, retract the movable gantry (activate **LED 1** to display this event).
6) When t = 10.0 seconds, disconnect the Liquid Oxygen (LOX) and Liquid Hydrogen (LOH) fill lines (activate **LED 2** to display this event).
7) **Switches 1, 2, and 3** will each represent the status of one of the three main engines. If the engine is fine, this switch will be low. If engine #1 (for example) has a malfunction, switch #1 will be set to high. These switches will be set prior to powering the circuit.
8) When t = 7.0 seconds, fire the 3 main shuttle engines (activate **LEDs 3, 4 and 5**). Of course, the LED for any malfunctioning engine should not light.
9) When t = 3.0 seconds, check all 3 engines to see if they have correctly fired. If one or more engines are off, then AUTO-ABORT, and shut down *all* the engines. All engine lights should go off, even for operational engines, and the timer should stop and hold at 3 seconds. The ENABLE light should obviously also go off.
10) If all three engines are lit at t = 3.0 seconds, then lock out the abort switch for the remainder of the countdown. From this time forward, the ABORT/ENABLE LED (#0) should *blink* at a rate of 4 Hz.
11) When t = 2.0 seconds, ignite the solid rocket boosters (**activate LED 6**).
12) When t = 0.0 seconds, launch the shuttle (**activate LED 7**). The timer must stop and hold at 0 seconds.

Input Summary:	Momentary Switch 0: begin launch
	Switch #0: Manual Enable/Abort
	Switches #1 through #3: set an engine malfunction
Output Summary:	2 digit 7-segment time display with blanking.
	LED 0: Enable light (sometimes blinking)
	LED 1: Gantry removal indicator.
	LED 2: Fuel lines removal indicator.
	LED 3, 4, and 5: Engine Ignition indicator.
	LED 6: Solid Rocket Booster Ignition indicator.
	LED 7: Launch Complete indicator.

Grading

An optional checklist is provided below for your use. Your grade will be partially based on neatness. This includes using wires of the appropriate lengths and colors. Although you may assign colors of your choice to various other tasks or concepts, all Vcc wires must be red, and all ground wires must be black.

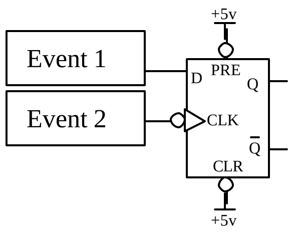
You should design and build your circuit in an incremental/modular fashion. Don't try to get everything figured out all at once. If you don't get everything done, you can still get a B or a C on this project based on what you *do* get done.

You are permitted to use up to eight I/O pins of the BASIC Stamp for this project. However, you are not required to use the BASIC stamp at all.

Discussion of New Equipment

D Flips Flops

We have already used JK Flip Flops to "remember" the current status of a numerical result (like the number of jets in a queue). However, the fundamental unit of memory is really the D Flip Flop, rather than the JKFF. Whenever you want to remember whether some event has already happened or not, you'll want a DFF. For example, use DFF to keep track of whether the timer has started yet, and use a DFF to keep track of whether the manual abort has ever been toggled. You'll probably need 5 to 8 DFFs for this project.



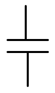
Consider the following circuit. This is a one-time use circuit. Suppose that the DFF has somehow been initialized "low" (see next section), and that both "events" are active high. So at the start, both event 1 and event 2 are low. The output can go high only if event 1 is triggered before event 2. If this happens, then the output will go high and stay high, regardless of whether the "event" inputs change later.

Analog Presets

For this project, we want to build a circuit that has a timer start out at some preset value ("15"). You may have observed that when the power goes out, your VCR's and digital clocks almost always seem to "boot up" at 12:00. On the other hand, our 192 counters and our Flip-Flops seem to boot up with almost random values. How can we ensure that the 74192, or any of our other memory devices (including JK Flip Flops and D Flip Flops) start out properly initialized, without requiring the user to push a button to get it going?

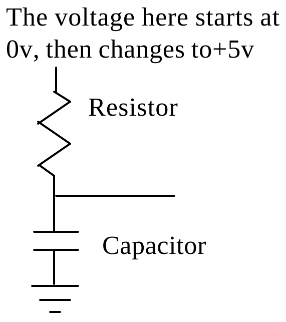
The 74192 chip has a "LOAD" input; it is active low. What we want is to create an input to this "Load" that is Low for a very short time when the circuit is first powered up, but which is High forever after that. During this short time period, we'll use the LOAD to input the starting value (15), and after that time period, we'll ignore the LOAD.

We need a device with a behavior that is explicitly and controllably time-varying. Analog devices are our only real options; we will use a circuit combining a capacitor with a resistor. A capacitor is a little analog device having two wires coming out of it. Inside the capacitor are two small, metal plates; each of the plates is connected to one of the wires, but not to the other plate. The symbol for the capacitor is the two parallel flat plates:



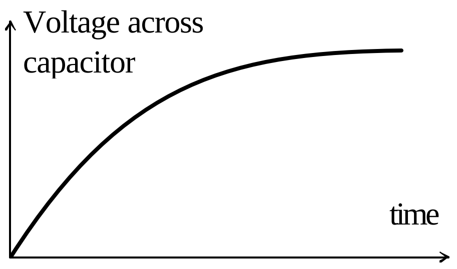
capacitor

Let's look at the following circuit. The top end of the resistor is connected to Vcc, which actually starts out at 0 volts when the box isn't yet powered, but quickly rises to +5v when the circuit is turned on.

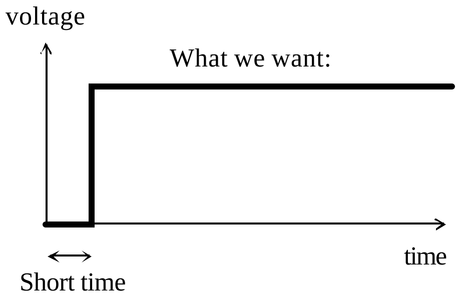


While this circuit is unpowered, the voltage on the wire between the capacitor and the resistor is at zero volts; *everything* is at zero volts.

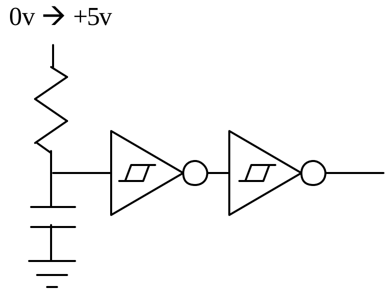
Then, the power is turned on. The capacitor acts like a slowly charging battery. How fast it charges is a function both of the size of the resistor and of the capacitor. Basically, if you increase either *R* or *C*, then it takes longer for the capacitor to charge up.



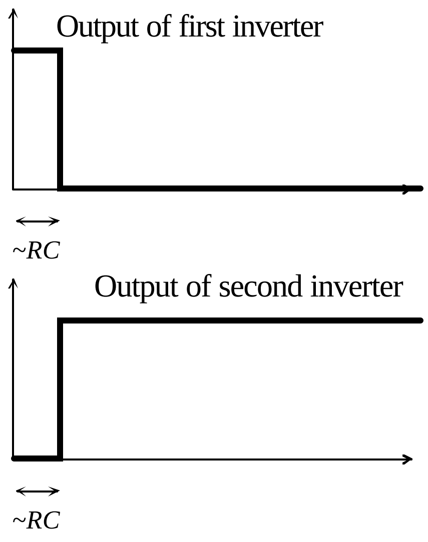
The capacitor is kind of like a bucket of voltage.The voltage difference between the two ends of the capacitor is directly related to how "full" the capacitor is. The capacitor starts out empty (the voltage on each side of the capacitor is zero volts), and then it fills up over some time period. The resistor keeps it from filling *too* quickly. Eventually, the capacitor becomes "full" (i.e., the difference in voltage between the two ends of the capacitor becomes 5 volts). Once it is full, the voltage stays there forever (unless you drain the capacitor by some other method, such as touching the top wire to ground). So, the timing diagram for the middle wire looks like this.



Unfortunately, this is not a "digital" signal, so it's unclear how a chip will react if this is the input. Let's try to connect this little analog circuit to some digital items. The gates shown here are called "Schmitt Trigger" inverters (7414); they give an especially clean output even if the analog input is ugly. They are basically inverters that perform well even when they have messy inputs, like the one shown above. In particular, when the input voltage is between 1 and 3 volts, these chips don't get confused like most chips. Instead, they just hold on to whatever value they had before the input entered the "forbidden" range.



The input to the first inverter starts out LOW, but over time, as the capacitor fills up, it changes to HIGH (as seen in the curved plot above). That means that the output of the first inverter starts out HIGH, and changes to LOW. But, what we wanted was a smooth signal that starts out LOW, and then changes to HIGH forever. So, we need a second inverter.



The voltage output from each of the inverters looks as shown here. The purpose of the inverters was to make the output distinctly digital (high or low), instead of the slowly changing analog curve seen above.

The time shown is *approximately* equal to *R* times *C*, where *R* is the resistance in Ohms, and *C* is the capacitance in Farads. Ohms times Farads equals seconds. You want to choose your values such that the product *RC* is more than 10 ms, but less than 200 ms (more or less). I have tried this in the lab with $R = 10\ \text{k}\Omega$, and $C = 10\ \mu\text{F}$, resulting in $RC = 0.1$ seconds, and it works fine.

This circuit is very convenient, because it essentially turns itself off within 0.1 s of startup. The output of this second inverter can be used to PRESET or LOAD your active-low flip-flops and 74192's. The output of the first inverter could do the same thing for functions that are active high.

Startup

- ☐ LED 0 (“enable”) initializes on.
- ☐ LED’s 1 through 7 initialize off.
- ☐ Numeric display initializes to “15”.
- ☐ Numeric display holds at “15” until the leftmost momentary switch (“begin”) is activated.
- ☐ Numeric display counts down at ~ 1 Hz after “begin”.
- ☐ Subsequent toggling of the “begin” switch has no effect.

Counting Down

- ☐ LED 1 lights when t = 14; LED 1 remains lit for $t \leq 14$.
- ☐ LED 2 lights when t = 10; LED 2 remains lit for $t \leq 10$.
- ☐ Numeric display has blanking of the MSD for $t \leq 9$.
- ☐ If engines are functional (switches 2 through 4 are low), LED’s 3 through 5 light when t = 7; they remains lit for $t \leq 7$.
  - ☐ If engines are not functional, then LED’s 3 through 5 (as appropriate) do not light.
- ☐ When t = 3, and if one or more engines are malfunctioning, then the countdown stops.
  - ☐ If countdown stops, then the display holds at t = 3.
  - ☐ If countdown stops, then LED 0 turns off.
  - ☐ If countdown stops, then LED’s 3 through 5 are all turned off (if they were on).
  - ☐ If countdown stops, then LED’s 1, 2, 6, and 7 are not affected.
- ☐ If $t \leq 3$, then LED 0 blinks at ~ 4 Hz.
- ☐ LED 6 lights when t = 2; LED 6 remains lit for $t \leq 2$.
- ☐ LED 7 lights and remains lit when t = 0.
- ☐ When t = 0, the timer display holds at t = 0 (the LSD is *not* blanked).

Manual Aborting

- ☐ Switch #0 (“abort”) causes the countdown display to hold, if $3 < t \leq 15$.
  - ☐ If the countdown is aborted, LED 0 turns off.
  - ☐ Setting “abort” has no effect on LEDs 1, 2, 6, or 7.
  - ☐ If “abort” is set, LEDs 3, 4, and 5 are turned off.
  - ☐ Unsetting “abort” has no effect.
- ☐ Switch #0 does nothing to the timer or to LED 0 if $t \leq 3$.

Wiring

- ☐ Wires are neat.
- ☐ Wires are color coded by concept (power = red and ground = black at a minimum).

Extra Credit

- ☐ Display includes one decimal place (10 Hz); triggering events still happen at correct times.