**Lab 05 – Detecting signal edges and removing bounces**

**Due date**

Start this lab after your Lab 04 breadboard circuit has been scored **during** Week 05 (Feb. 06 – Feb. 09). Your circuit demonstration and upload to Blackboard of you question responses for this lab are due at the start of your Week 06 lab session (Feb. 13 – 16).

**In lab scoring and Questions to answer and upload**

[25 points in lab Week 06] Demonstration of working circuit [15 points]; answers to TA questions [10 points].

1. [5 points] How should the chip inputs be set so that the 74163 is in the mode to count, rather than one of its other modes: reset (clear), disabled (several types), and load 4-bit value into the counter?

We need to connect clear and load with +5V wire instead of ground wire so it won’t clear or load (signal 1).

Also, CET, CEP (enables) are connect to +5V (signal 1)

1. [10 points] Generate 16 Clock 1 signal rising edges (see below in instructions). Record in the Clock Signal Source table below the counter increment amount modulo 16 you see on LEDs D, C, B, A.

|  |  |
| --- | --- |
| **Clock Signal Source** | **Increment amounts modulo 16 observed via LEDs for 16 Set commands** |
| Clock 1 (bare wire switch) | Increments mod 16: 9,4,0,9,2,14,1,8,13,10,4,10,4,15,14,3 |
| Clock 2 (Q output of SR latch) | Increments mod 16: 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 |

1. [10 points] Now connect Clock 2 to the Clock input of the 74163 and reset and set the latch output 16 times. Record for each reset/set pair in the Clock Signal Source table below the increment amount modulo 16 that you see on the counter outputs.

1. [5 points] What changes if you connect Clock 3 to the 74163 and again operated the SPDT switch as before?

It still increments numbers in this pattern except now when red LED turns on, the increment happens.

1. [5 points] Is the 74163 count behavior the same whether you input 16 pairs of S’R’ or 16 pairs of R’S’?

Depends on the meaning of “count behavior”

But the idea is still the same:

if your input is Q, you expect increment when Q is signal 1.

If your input is Q’, you expect increment when Q’ is signal 1.

The incrementation of count behavior is still 1 for each rising edge.

But now R’ and S’ are swapped, which means Q and Q’ are swapped. So, if your input once was Q, now its Q’. If we connect it again to Q (our original clock) it still behaves the same. If we don’t, then the order of touching R1 and R2 will be flipped because now we are on the other clock.

1. [10 points] Why is skipping consecutive numbers in the output of the 74163 counter an indication of switch bounce?

Because when switch bounce occurs, switch makes and break contacts, voltages bounces high and low. So, we must skip consecutive numbers in the output and only see the final state. Thus, skipping consecutive numbers in the output of the 74163 counter is an indication of switch bounce.

1. [10 points] If the 74163 advances the count by 1 for a single Clock 1 input this means that the switch did not bounce that time. True or False? Explain your answer.

I think its false. Incrementation by 1 is just a case of many which happens that after bounces, the final state is a steady 1, same as how 74163 can advance count by other numbers.

1. [10 points] What is the mathematical expression for the number of bounces observed by the 74163 chip circuitry as contrasted with the number of bounces that your eyes are capable or observing by examining the 74163 output using LEDs?

Number of bounces 74163 observed = rand() + Number of bounces I observed

Rand() is random number expression.

Because I can’t really see bounces as 74163 can.

1. [10 points] How does the memory capability of the SR latch (NAND1 and NAND2) transform the bouncing SPDT switch input into a bounce-free latch output Q?

When bouncing happens, if pole and throw break contact, both S and R input 1, from the truth table, we can SR latch will sustain the current output (memory capability), if it makes contact, that is what we want. So, SR latch enables us to get the actual output that’s the same as desired output.

# Lab Introduction

The real world is messy. Real mechanical switches bounce. In this lab we will use the 74163 4-bit synchronous binary counter as a GDB-style debugging tool to display hardware voltage signals that cannot be displayed by an LED alone.

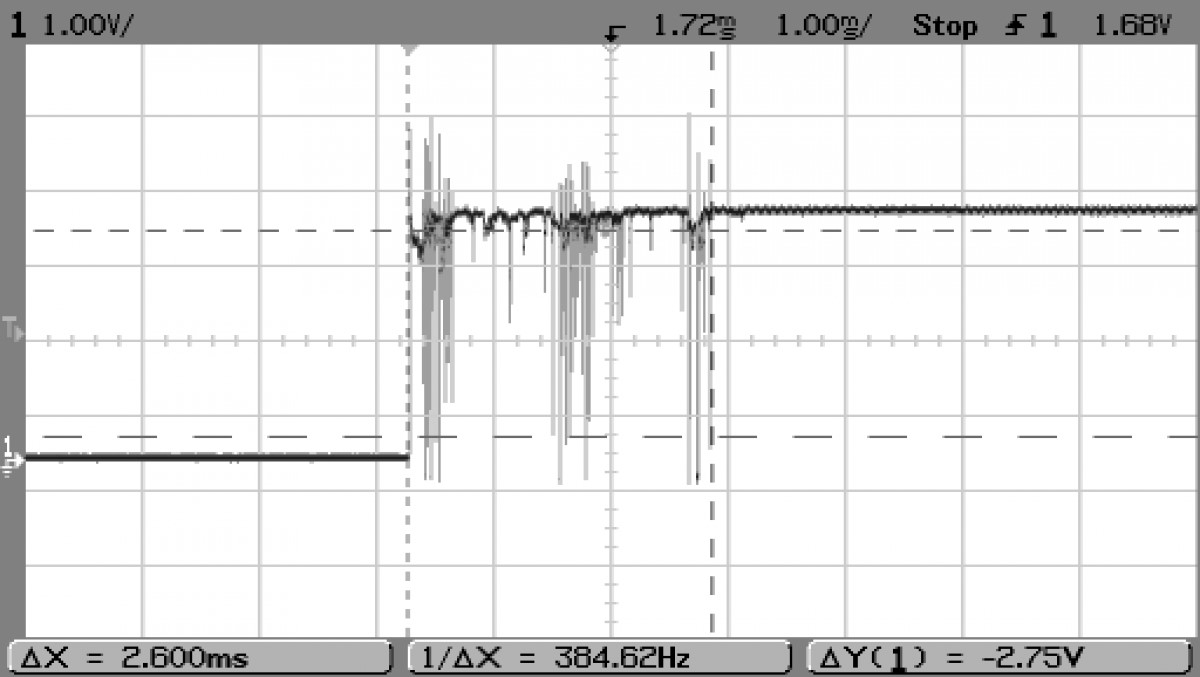
Our use to date of an LED as a printf( ) analog does not work when we want to display bouncy voltages from mechanical switches that are part of a voltage divider. This is because our human eyes do not respond fast enough to perceive the flickering output of the LED when a switch bounces. We only see the output when the voltage has stabilized, after the bouncing has ended. The 74163 rising-edge counting chip will give us “better eyes” to see more of what is happening in our computational circuit.

Then we will see how an SR latch can de-bounce (remove bounces from) the voltage divider signal even though the mechanical switch bounces.

# What is mechanical switch bounce

The electrical resistance between the contacts of a mechanical switch is often quite unstable for a few thousandths of a second as the contacts grind, or bounce, microscopically before settling to a stable physical contact. This causes the voltage delivered to the switch output to transition between switch-open and switch-closed levels any number of times.

An oscilloscope is an instrument that can plot voltage as a function of time. Below is an image from an oscilloscope plotting switch output voltage (vertical axis) versus time (horizontal axis) as the switch is closed. The horizontal axis shows time over an interval of 10 milliseconds. The switch bounces for 2.6 milliseconds at the center of the time interval, before completely making or breaking contact, we cannot tell which is the case because switches can bounce at both times. The jagged transitions of voltage tend to be random in number and duration each time the switch makes or breaks contact.

[Image credit: This image taken from post #12 of 22 at

<http://www.mytractorforum.com/24-gravely/259067-new-life-816-gravely-w-electronic-ignition.html> by MTF Member # 51208 per fair use for educational purposes.]

If this switch output was powering an LED, that LED would flash on and off in time with the voltage variations because the response time of an LED to a change in voltage difference between its two leads is quite fast. Human eyes, however, would not notice the flickering because it has too high a frequency, just as some sounds are ultrasonic in frequency and cannot be heard by humans but can be heard by bats. All we can see is the final LED state after the switch stops bouncing between closed and open. Crucially though, if this “bouncy” signal is input to a digital logic gate that computes its output in nanoseconds or even just microseconds, that gate would re-compute its output logic level with every input signal transition. Because the number and timing of these bounce transitions is random, it is unlikely that we, as designers want the circuit to pay attention to the bounces.

While clever mechanical design can reduce switch bounce, it is not feasible to build a mechanical switch that never exhibits bounce. Our goal for this lab is to investigate bouncing and a way to eliminate it from logic input signals.

The 74163 chip <http://www.futurlec.com/74HC/74HC163.shtml> in your lab kit is a presettable, synchronous, rising-edge-triggered 4-bit binary counter with synchronous reset circuit. These characteristics mean that the 74163 will output a count, modulo 16, of the number of rising edges (transitions from low voltage to high voltage) appearing in the signal sent to its Clock input. You will sample various signal nodes on the breadboard for rising edges.

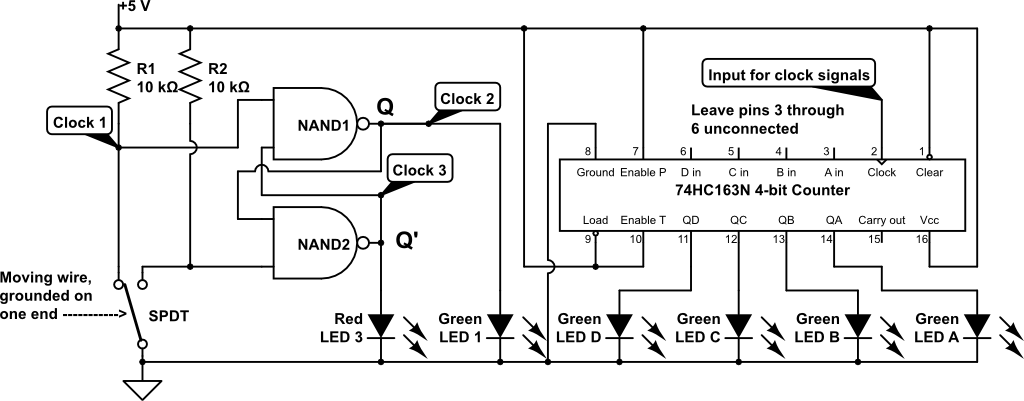
The 74163 can count rising edge events at a rate of up to 40 million per second, more than fast enough to “see” mechanical switch contacts bounce. Each rising edge of a signal delivered to the Clock input of the 74163 will increment the count by 1. The 74163 counts modulo 16, so when the count is 15 (when QD, QC, QB, and QA equal 1111), the next increment takes the count to 0 (QD, QC, QB, QA equal 0000).

A pin diagram for the 74HC163 is available at <http://www.futurlec.com/74HC/74HC163.shtml>.

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| POWER | As usual, Pin 16 and Pin 8 correspond to Voltage Supply (Vcc = +5 volts) and Ground (Zero reference, or GND). |
| CLOCK INPUT | Pin 2 is the clock input pin, the signal for which to count rising edges. |
| DATA INPUT | Pin 3 to Pin 6 are data input pins corresponding to D0 to D3. |
| COUNT ENABLE | Connect Pin 7 to Vcc to enable the counting function. |
| FLIP-FLOP OUTPUT | Pin 14 to PIN 11 are the output pins. They are each the output of a flip-flop, which is a latch circuit with certain improvements that make it easier to incorporate into larger circuits. |
| COUNT ENABLE CARRY INPUT | Connect Pin 10 to Vcc to enable the carry input. |
| PARALLEL ENABLE INPUT | Connect Pin 9 to Vcc to disable the ability to load the counter with an unsigned integer count value. |

# Preparation

Build this circuit on your breadboard.



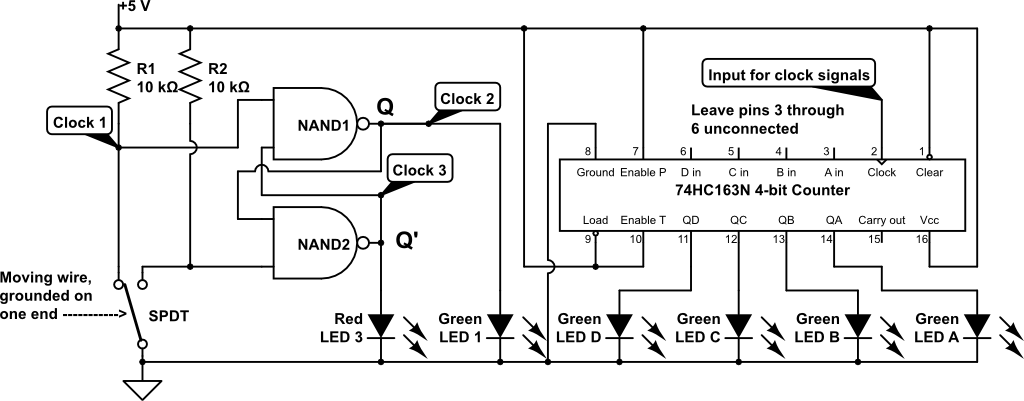
Place the 74163 chip so that output pins 11, 12, 13, and 14 and Green LEDs D, C, B, and A will conveniently appear in the left to right order of most significant bit to least significant bit as you view your circuit board.

The SPDT switch is implemented using a grounded wire with an unconnected free end that you can move to touch the bare leads of R1 and R2. Be sure to separate in space the bare leads of R1 and R2 by more than the diameter of the grounded wire.

You will need a long wire to connect pin 2, Clock, of the 74163 chip to the circuit nodes labeled Clock 1, Clock 2, and Clock 3 at various times and you conduct your experiment.

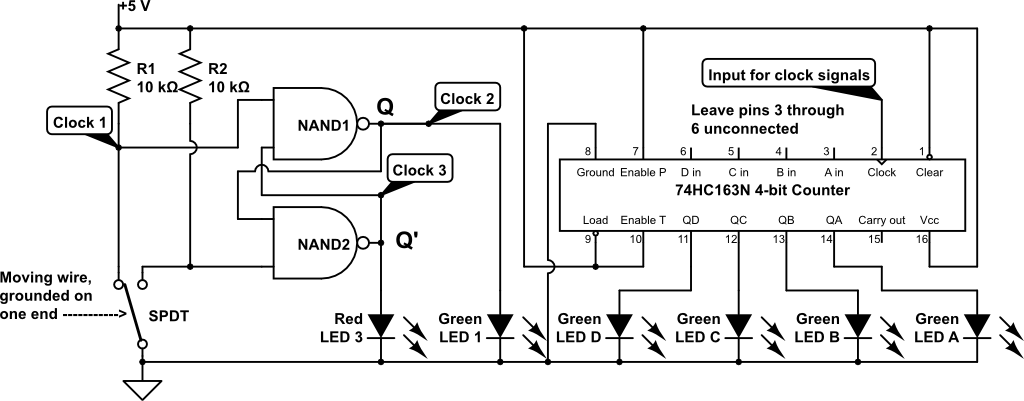
## Single-pole single-throw (SPST) switch made with bare wire: Clock 1

In the schematic below, a wire is used for the moving pole of the switch that can touch either of the two throws that are connected to R1 and R2. The bare wire leads of R1 and R2 serve directly as the two switch throws and moving the bare end of the wire to ground back and forth between these two lead comprises the moving pole. A crucial part of the mechanical construction of this switch that you must ensure is that the pole (moving wire) cannot physically touch both throw wires at the same time. Because of this switch geometry, the switch always passes through a not-connected state when moving from touching one throw wire to touching the other throw wire.



Connect the counter input to Clock 1. Now answer Question 2. You should see a wide variety of incremental count amounts by the 74163: touching a bare wire with another bare wire is a very bouncy mechanical switch design.

## Single-pole double-throw (SPDT) switch providing S’ and R’ inputs to an S’R’ latch for electronic de-bouncing: Clocks 2 and 3



NAND1 and NAND2 are connected to form an S’R’ latch. The schematic also shows that R1 and R2 hold each of the S’ and R’ inputs to NAND1 and NAND2 to a logic 1 value, except when the SPDT switch pole makes contact with one of the two throws.

We can also see that by its mechanical design, which separates the two throws in space by more than the diameter of the moving pole wire, that the single pole cannot contact both throws at the same time. The mechanical design of this switch ensures that, in moving the pole from one throw to the other, there will be a time during which the pole makes no contact with either pole.

Here is the characteristic table for an SR latch. This latch design sustains (holds) the current Q output value as long as both inputs S’ and R’ remain at logic 1. However, when S’ becomes 0 the latch will set the Q output to 1 within two gate delays and will reset Q to 0 in two gate delays after R’ becomes 0.

|  |  |  |  |
| --- | --- | --- | --- |
| S’(t) | R’(t) | Q(t) | **Q(t+Δ); comment** |
| 0 | 0 | 0 | X; Disallowed S’R’ input combination |
| 0 | 0 | 1 | X; Disallowed S’R’ input combination |
| 0 | 1 | 0 | 1; Set; make output Q equal to 1 |
| 0 | 1 | 1 | 1; Set; make output Q equal to 1 |
| 1 | 0 | 0 | 0; Reset; make output Q equal to 0 |
| 1 | 0 | 1 | 0; Reset; make output Q equal to 0 |
| 1 | 1 | 0 | Q(t); Hold or sustain the current output |
| 1 | 1 | 1 | Q(t); Hold or sustain the current output |

Mechanical switch bounce happens on a time scale of milliseconds, while NAND gates react to changes on their inputs on a time scale of tens of nanoseconds, which is about 100,000 times faster. Imagine that we touch the switch pole wire to the S’ input, grounding it (logic 0), and then one-thousandth of a second afterwards the wire bounces mechanically to not touching the S’ input, changing the S’ logic value back to 1. Long before the bounce returns S’ to 1, NAND1 will have responded to S’=0 with Q(t+Δ)=1. Then, in nanoseconds, the feedback path of Q to the input of NAND2 along with the fact that R’=1 will have caused NAND2 output Q’(t+Δ)=0, which is fed back to NAND1. There, at the inputs to NAND1, regardless of whether S’=0 or S’=1, the latch is stable with Q(t+Δ)=1. The NAND1-NAND2 latch circuit “catches” the first touch of the pole wire that makes S’=0 and ignores all pole wire bouncing on the S’ contact afterwards.

Touching the SPDT pole wire to R2 makes R’=0, changing Q to 0 and Q’ to 1 causing Red LED3 to turn on.

**Upload your answers for all questions in PDF format to Blackboard before the start of your lab session next week.**