**Memorandum**

**To:** Dr. Sys Tems, CEO of “Sys Tems LLC”

**From:** Steve Tamayo, “Professional Staff Engineer”, Raymond Yung, “Professional Staff Engineer”, Duy Le, “Professional Staff Engineer”, Angela Albrecht, “Professional Staff Engineer”,

**CC:** Dr. Thomas Chmielewski, “Engineering Advisor”

**Date:** October 9th, 2019

**Subject:** Updated Memo: Switch Debounce Proposal

Dr. Sys Tems,

Our team has a proposal to increase efficiency and save costs at Sys Tems, LLC. We write to you to inform you of our solution in this memo.

Mechanical switches are prone to bouncing. This is when the state of the switch is adjusted and momentary electrical contact is lost, which sends several impulses before stabilizing and settling. In digital circuits this can prove to be inefficient as the switch simulates false cycling. Our team analyzed several company switches and developed software in order to test the integrity of our digital systems.

Proposed Solution:

There a couple ways to prevent switch bouncing and that involves a hardware solution or a software solution. Since we are on a time constraint, a hardware solution cannot be an option. Our solution is to implement a software approach which would save the company additional hardware expenses. To do so, we initially propose a circuit which will evaluate the number of switch bounces. By measuring the count of the switch bounce, we can conclude which of our switches is the best suited for our future company projects. Furthermore, we implement a debounce circuit to test for results. Since this solution is software-based, Sys Tems, LLC will save additional costs in the future. The solution is simple to implement and the steps needed are the following.

Debounce Circuit and Experimental Findings:

Figure 1 displays the state diagram for the bounce count software. In the following Figure 2, the graphic displays the simple circuit built to test the integrity of our digital systems. We stress once more, by measuring the count of the switch bounce, we can conclude which of our switches is the best suited for our future company projects. In addition we implement a switch debouncer. Both of those are implemented in one circuit in Figure 2. In the lower half of the bread board are two indicator LEDS which are responsible for the falling and rising parts of the bounce. The blue LED is responsible for H->L transition and the red LED is responsible for L->H transitions. When the first external interrupt primer button is pressed the program will be refined to start the bounce counting process for H to L. Then, the blue LED will light up, indicating a low to high bounce was found. Afterwards, the same process is used for the second external interrupt primer, which starts the bounce counting process for low to high. The red LED will light up for this portion. After each test, the code waits for the user to press the next primer button to start the next test. The user has the option to start a low to high or high to low test depending upon which external interrupt primer button is pressed first.

Figure 1. State Diagram for Bounce Count Software

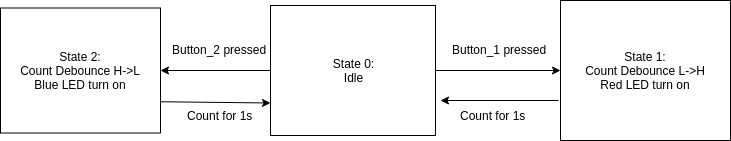


Figure 2. Labeled Switch Bounce Counter 

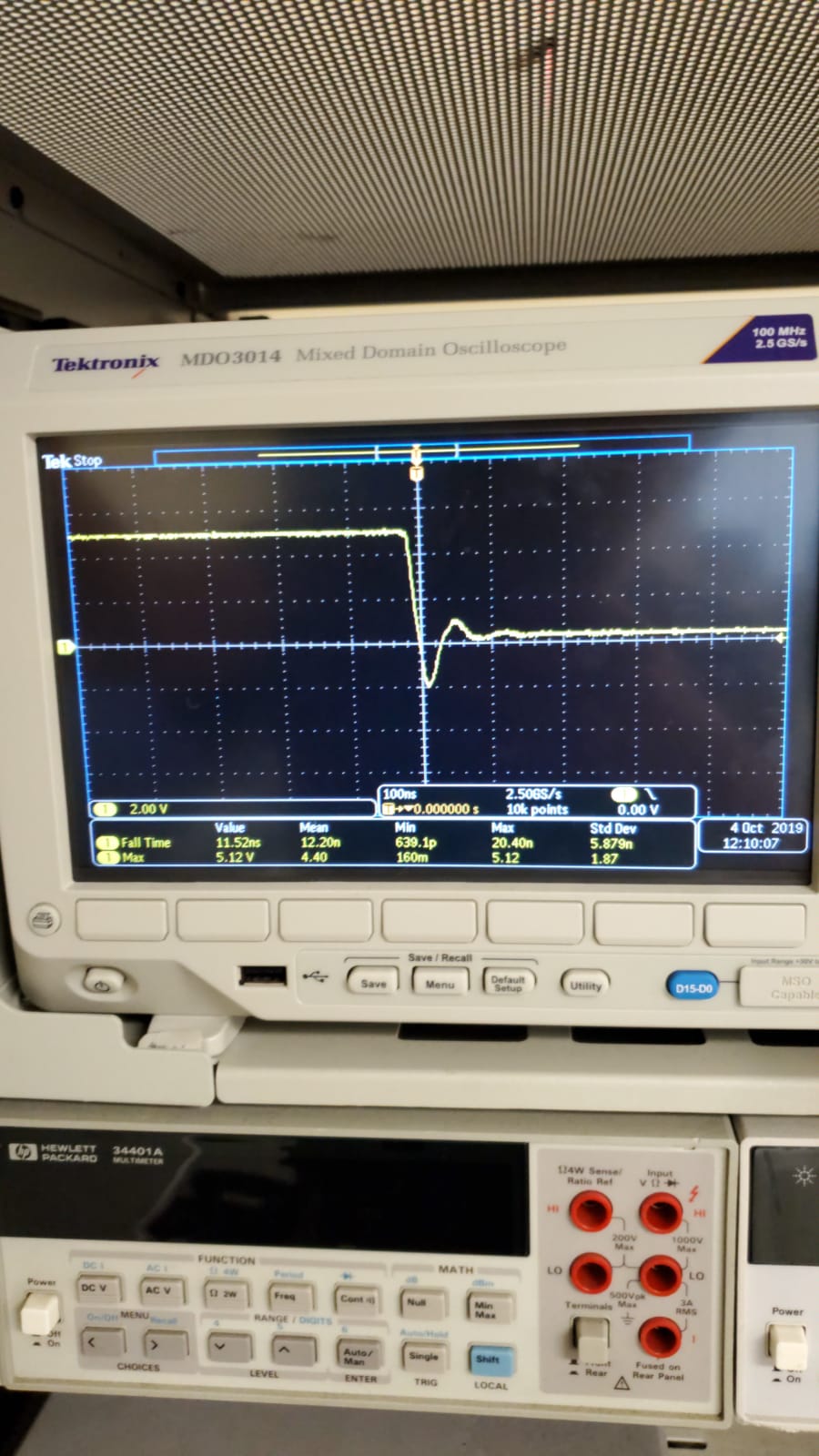
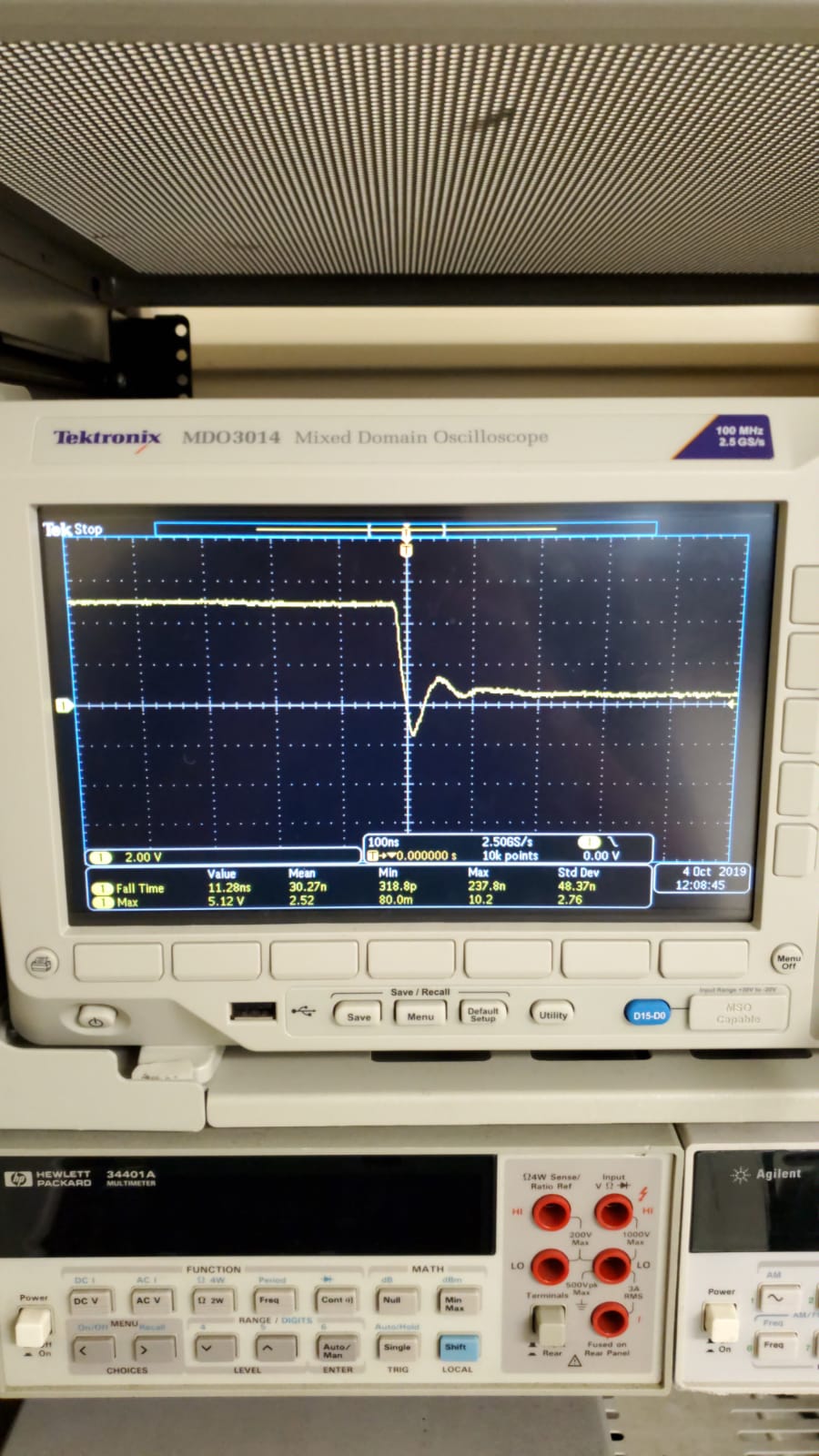
Figure 3. Labeled Switch Bounce Counter and Debounce Circuit Diagram



In Table 1 below, our initial experimental data using the circuit above for Switch #15 is displayed. Our team conducted 40 separate tests, equal parts for the high to low process (20) and the low to high process (20). We noted that if the value received was < 2 there was no bounce detected by the software. The number of bounces is the value - 1. The average number of bounces for H to L was 1.85 while the average number of bounces for L to H is 1.65.

Table 1. Switch #15 Bounce Metrics

|  |  |  |
| --- | --- | --- |
| Test Number: | H to L: | L to H: |
| 1 | 1 | 2 |
| 2 | 1 | 2 |
| 3 | 3 | 6 |
| 4 | 7 | 2 |
| 5 | 2 | 6 |
| 6 | 6 | 1 |
| 7 | 2 | 5 |
| 8 | 2 | 1 |
| 9 | 9 | 5 |
| 10 | 3 | 3 |
| 11 | 1 | 4 |
| 12 | 8 | 1 |
| 13 | 2 | 1 |
| 14 | 1 | 2 |
| 15 | 2 | 2 |
| 16 | 2 | 1 |
| 17 | 2 | 3 |
| 18 | 1 | 4 |
| 19 | 1 | 1 |
| 20 | 1 | 1 |

Figure 4. Oscilloscope Output of blue LED (H to L) Figure 5. Oscilloscope Output of red LED (L to H)

In the figures above (4,5) we display the oscilloscope outputs for the bounce metric for Switch #15.

Proposed Testing Procedure for Switches

The procedure for testing switches is as shown in steps below.

1. Hook up the intended switch to the breadboard as shown in Figure 2.
2. Open up the Arduino IDE and upload the bounce metric code given in the Appendix.
3. Open the Serial Monitor to see instructions and results
4. To test number of bounces, press one of the external interrupt buttons
   1. To Test High-Low, prime the external interrupt button 2.
      1. The Blue LED should light up when the code is ready to count H-L.
      2. After you press the switch, the number of bounces will appear.
      3. Total count of bounces will be result -1. (A count of 1 H-L means there were no bounces and the change from H-L was stable.)
   2. To Test Low-High, prime the external interrupt button 1.
      1. The Red LED should light up when the code is ready to count L-H.
      2. After you release press the switch, the number of bounces will appear.
      3. Total count of bounces will be result -1. (A count of 1 L-H means there were no bounces and the change from L-H was stable.)

After testing each switch twenty times, note the switches with lowest average amount of bounces. The ones below the bounce threshold of 5 are the best suited for future company projects. For our test case, Switch #15 had an average of 1.85 for H to L and 1.65 for L to H. Switch #15 is an ideal example of what the bounce metric should be.

Conclusion:

Mechanical switches are prone to bouncing. Our team analyzed several company switches and developed software in order to test the integrity of our digital systems. The solution to implement a software approach arose from would save the company additional hardware expenses. We outlined a protocol above in this memorandum to use as the testing procedure of future switches. Any switch below the maximum threshold of 5 bounces is suited for the company standards. Any switch above 5 should be repurposed.

We propose in our next weekly meeting to review our results with you. In addition we will conduct a live demo and will be sending you video footage of our findings. If you have any questions feel free to contact us directly.

We look forward to speaking with you soon,

*Sys Tems, LLC Engineering Department.*

References:

Khatri, Pankaj. “What Is Switch Bouncing and How to Prevent It Using Debounce Circuit.” *What Is Switch Bouncing and How to Prevent It Using Switch Debounce Circuit*, 6 June 2019, circuitdigest.com/electronic-circuits/what-is-switch-bouncing-and-how-to-prevent-it-using-debounce-circuit.

“Arduino Project - Switch Bounce.” *Subsystems*, [www.subsystems.us/arduino-project---switch-bounce.html](http://www.subsystems.us/arduino-project---switch-bounce.html).

Appendix:

#Contents of Arduino Sketch File

// Switch Bounce Counter

//

#define btnSTARTFall 2 // Pin to Ready Bounce Counter

#define btnSTARTRise 7 // Pin to Ready Bounce Counter

#define swHIT 3 // External Interrupt 1

#define swHIT18 18 // External Interrupt 2

#define ledTriggerBlue 4 //Led Indicator 1

#define ledTriggerRed 5 //Led Indicator 2

volatile int bounceCountFall = 0;

volatile int bounceCountRise = 0;

int breakWhileCount = 1000;

bool testingFall = true;

void setup() {

// setup the switches and LED

pinMode(btnSTARTFall, INPUT\_PULLUP);

pinMode(btnSTARTRise, INPUT\_PULLUP);

pinMode(swHIT, INPUT);

pinMode(swHIT18, INPUT);

pinMode(ledTriggerBlue, OUTPUT);

pinMode(ledTriggerRed, OUTPUT);

// If you don't use pin 3, you will need to change the 1 below to the new interrupt number

attachInterrupt(1, bounceFall, FALLING);

attachInterrupt(5, bounceRise, RISING);

// By using FALLING, we capture every time the pin transitions from HIGH to LOW

// Turn off the LED

digitalWrite(ledTriggerBlue, HIGH);

digitalWrite(ledTriggerRed, HIGH);

// Start the Serial port

Serial.begin(9600);

Serial.println("Arduino Switch Debounce");

Serial.println();

}

// All the interrupt routine needs to do is increment bounceCount

// By keeping this routine small we maximize the chances of catching every bounce

void bounceFall() {

bounceCountFall++;

}

void bounceRise() {

bounceCountRise++;

}

void loop() {

Serial.println("Press START button when ready");

Serial.println("If Blue LED lights, the pressing test is ready.");

Serial.println("If Red LED lights, the releasing test is ready.");

// Wait for one of the START buttons

while (digitalRead(btnSTARTFall) && digitalRead(btnSTARTRise)) {}

testingFall = true;

// Then wait for it to be released

delay(10);

if(!digitalRead(btnSTARTFall) && digitalRead(btnSTARTRise)){

testingFall = true;

//Serial.println("Test1");

while(!digitalRead(btnSTARTFall)){}

} else if (digitalRead(btnSTARTFall) && !digitalRead(btnSTARTRise)){

testingFall = false;

//Serial.println("Test2");

while(!digitalRead(btnSTARTRise)){}

}

delay(1000);

// Start the testing

bounceCountFall = 0;

bounceCountRise = 0;

// Light the corresponding LED

if (testingFall){

digitalWrite(ledTriggerBlue, LOW);

Serial.println("Ready for testing Fall...");

// Wait for the switch to close

while (bounceCountFall == 0) {}

} else{

digitalWrite(ledTriggerRed, LOW);

Serial.println("Ready for testing Rise...");

// Wait for the switch to open

while (bounceCountRise == 0) {}

}

// If you are here, the switch was thrown

// Wait a second to collect the bounces

delay(1000);

// Output the results

digitalWrite(ledTriggerBlue, HIGH);

digitalWrite(ledTriggerRed, HIGH);

Serial.print("The switch bounced ");

if (testingFall){

Serial.print(bounceCountFall);

} else {

Serial.print(bounceCountRise);

}

Serial.println(" times.");

Serial.println();

}

//EndCode