TO: Professors J. Hung and V. Nelson
FROM: "The PokeSquad", Demetris Coleman and Justin Wahlers
SECTION: 003 – Tuesday 3:30 p.m.
DATE: September 23, 2016
SUBJECT: Labs 4 and 5 – Interrupt Handling and Interfacing Devices Using Parallel IO

The objectives of labs 4 and 5 are to learn how to design C programs for the STM32L1xx microcontroller to handle devices in interrupt-driven I/O mode and control a peripheral device, interfaced through parallel I/O ports. The device used was the Direction Pad in figure 1. For the hardware setup, the STM32L100 input pins with pullup resistors were connected to columns (pins 4 and 5 in figure 1) which were connected to AND gate inputs. The AND output was connected to PA1 to trigger the interrupt when the signal fell low. The rows (pins 1-3) were connected to output pins on the STML32L100.

Specifications for lab 4's program were to make two counters counting on the range 0-9 and display them on LEDs. The first counter was to increment every half second and the second was to change every second in a direction specified by two push-buttons (shown in Table 1). Lab 5's specification was to use display a continuously counting counter that increased from 0 to 9 once every second and an interrupt routine that identified the button being pressed on the D-pad and displayed its opcodes (shown in Table 2) on the counters display for 5 seconds before displaying the counter again without a break in its count sequence. Both programs can be seen at the end of the report (3 pages each).

Table 1: Lab 4 operation specifications

	PA0 Pushed	PA1 Pushed
Count	Increasing count	Increasing count
Count2	Decreasing count	Increasing Count
PA0 LED	Toggles	Nothing
PA1 LED	Nothing	Toggles

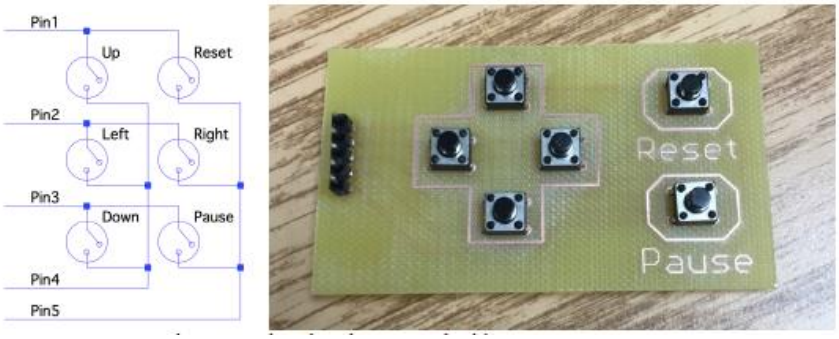

Figure 1: Directional Input Pad with two face buttons (right) and its circuit schematic (left)

Table 2: D-pad Button Opcode Values

D-pad Button	Opcode
Up	0x01
Left	0x02
Right	0x04
Down	0x08
Pause	0x10
Reset	0x20

Testing Procedure and Observations

In lab 4, the program was tested to see if it entered the interrupt service routine by placing breakpoints inside of it. Initially, the program never entered the ISR. This happened because the IRQ_Handler function was misspelled and the ISR did not clear pending flags and setup the IMR and PR for external interrupts before exiting. After the corrections were made, the period of each count was tested by measuring the LSB of each and found to be 1.000168 s for the first counter and 2.000234 s for the second as seen in figure 6. To verify the behavior of count 2 explained in table 1, the logic analyzer (figures 2 and 3) was used to watch the values of both counters. It and the oscilloscope (figures 4 and 5) were used to show the LEDs toggle when the push-buttons were pushed. In figure 5, there was a bit noise after the digital push-button was pressed that had no effect on the toggle. Figure 5 also had a higher signal level than figure 4, which was toggled by the push-button on-board the STM32L100.

In lab 5, many similar problems from lab 4 were faced because parts of an older version of lab 4 code were reused instead of the most current version. Also, the GPIOB clock was not set up which caused only one GPIOB pin to enable the pull-up resistors while the others did not. After making corrections, interrupt operation was verified by inspecting the LEDs and using the watch window to track the variable OPCODE for the debugger. The same tools were used with the addition of the variables row and column to verify that the correct opcode was being detected. Sometimes the buttons produced the correct opcode, but immediately snapped to an incorrect code. Other times it produced the wrong code all together. The problem seemed to be that the wrong row was sometimes detected. Adding short delays between setting a row high and checking to see if the column returned to high in our row scan algorithm fixed the problem. The logic analyzer (figure 7) was used to show the program worked to the specification explained in the previous section. However, the program returned to its former behavior before the end of lab. While it was working the display did not display anything for some opcodes because the it had insufficient bit space to display 2 of the opcodes (0x20, 0x10).

Results

For lab 4, the counters work to the specifications in table 1. The operation is demonstrated in figures 2 and 3 and the time period of the counters can be seen in figure 6.

For lab 5, the counter continuously counts from 0 to 9 and when a D-pad button is pressed, the interrupt routine displays an opcode on the counter's display for 5 seconds before displaying the counter again without a break in its count sequence. The operation can be seen in figure 7.

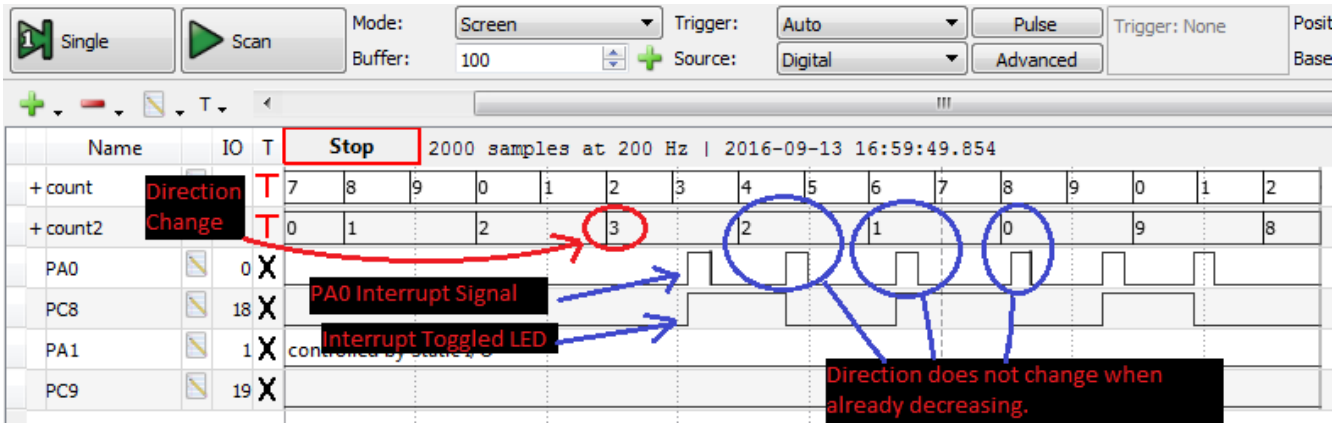

Figure 2: Logic analyzer showing count2 increasing to decreasing and led toggle.

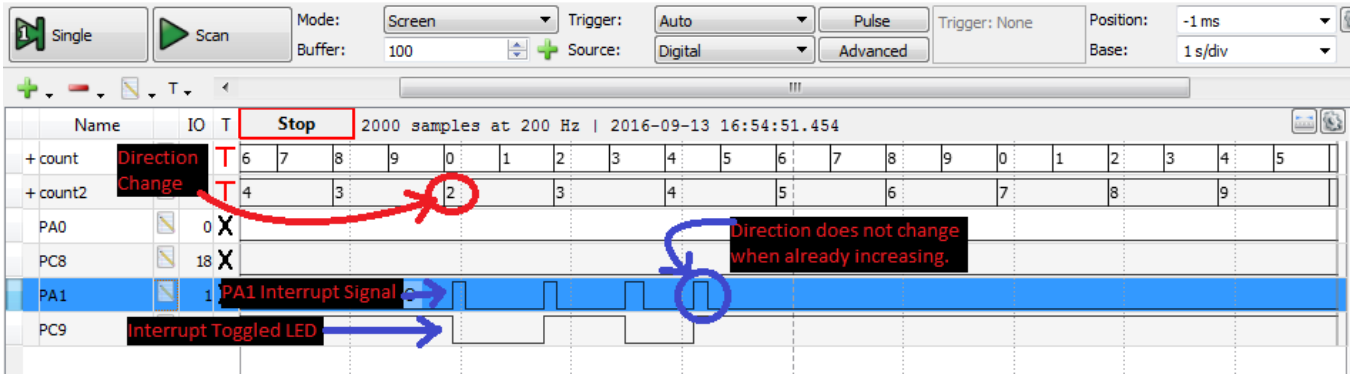

Figure 3: Logic analyzer showing led toggle and count2 decreasing to increasing

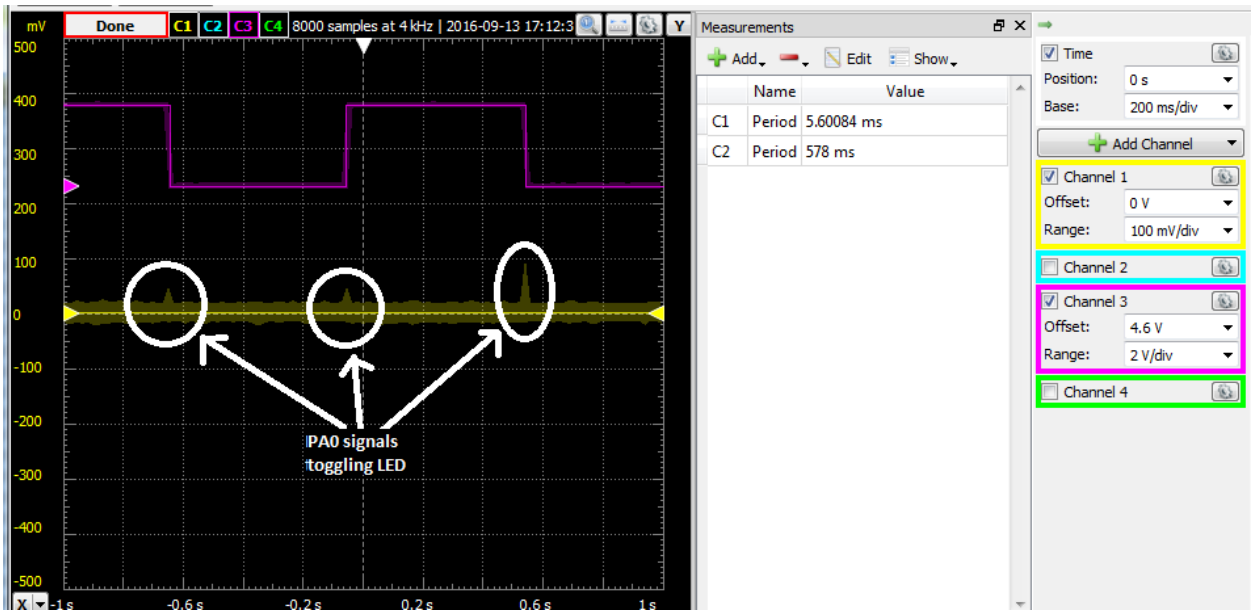

Figure 4: PA0 and PC8 LED toggling on oscilloscope

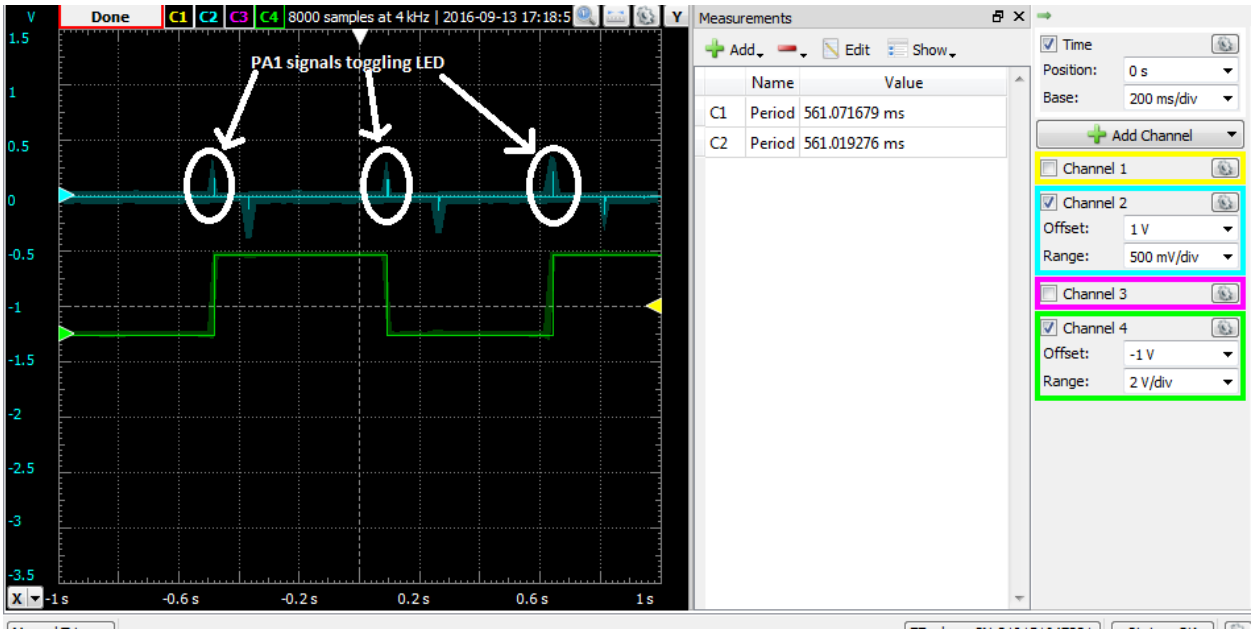

Figure 5: PA1 (blue) and PC9 (orange) LED toggling on oscilloscope.

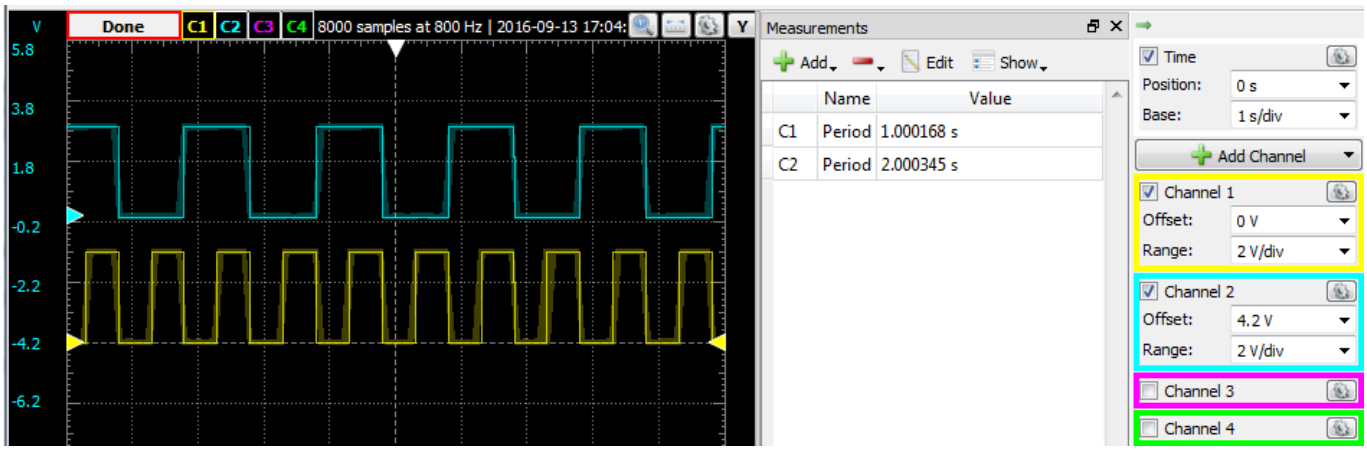

Figure 6: Period of count (yellow) and count2 (blue) on oscilloscope can be seen on left

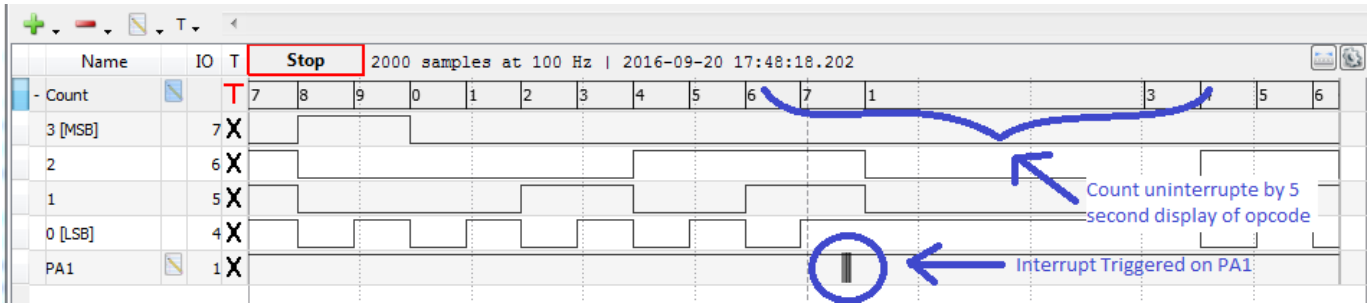

Figure 7: Counter display interrupted by Opcode value for 5 seconds

Conclusions

The past two weeks we have learned how to set up interrupts on the STM32L100 micro-controller and how to interface with a passive directional pad built with switches. I suspect the problem with the D-pad button detector has to do with our delays inside the ISR being too short for the signal to reach the register before being read because the behavior is similar to how it acted before adding the delays.