# EE445L - Lab 02 Report

Kevin Gilbert
Gilberto Rodriguez

Professor Bard
Lab: Monday/Wednesday 5-6:15

February 7, 2014

## Ojective

- To develop software debugging techniques
  - Performance debugging (dynamic or real time)
  - Profiling (detection and visualization of program activity)
- To pass data using a FIFO queue,
- To learn how to use the oscilloscope and logic analyzer,
- To observe critical sections,
- Get an early start on Lab 3, by writing a line drawing function.

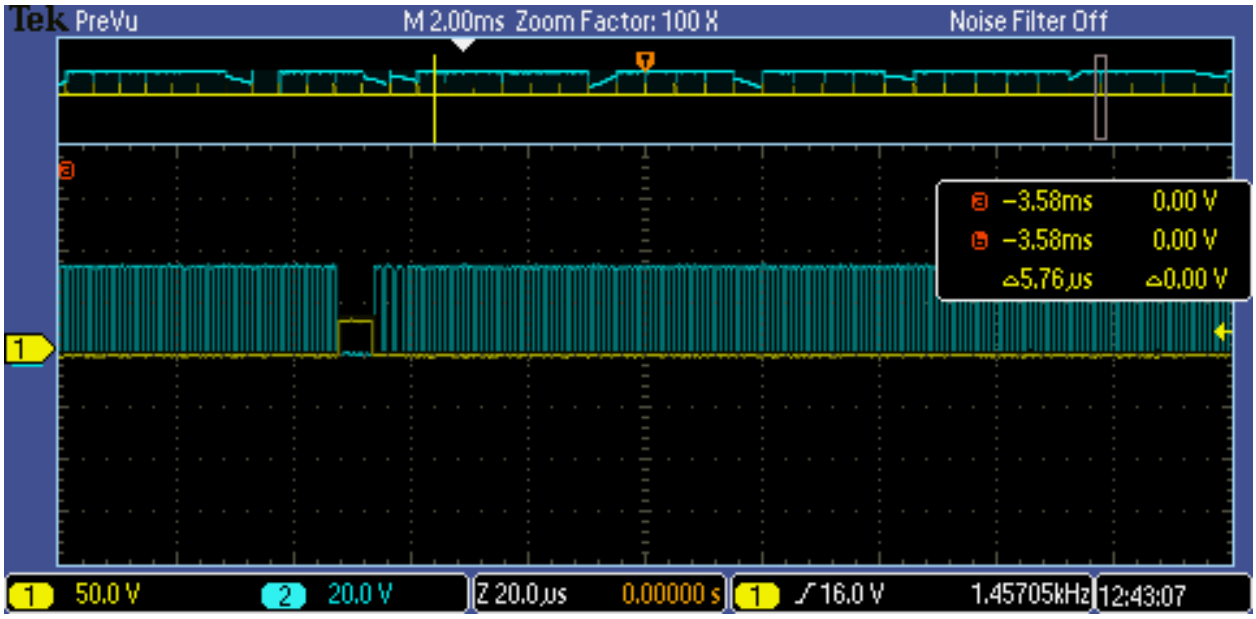
## Mesurement Data

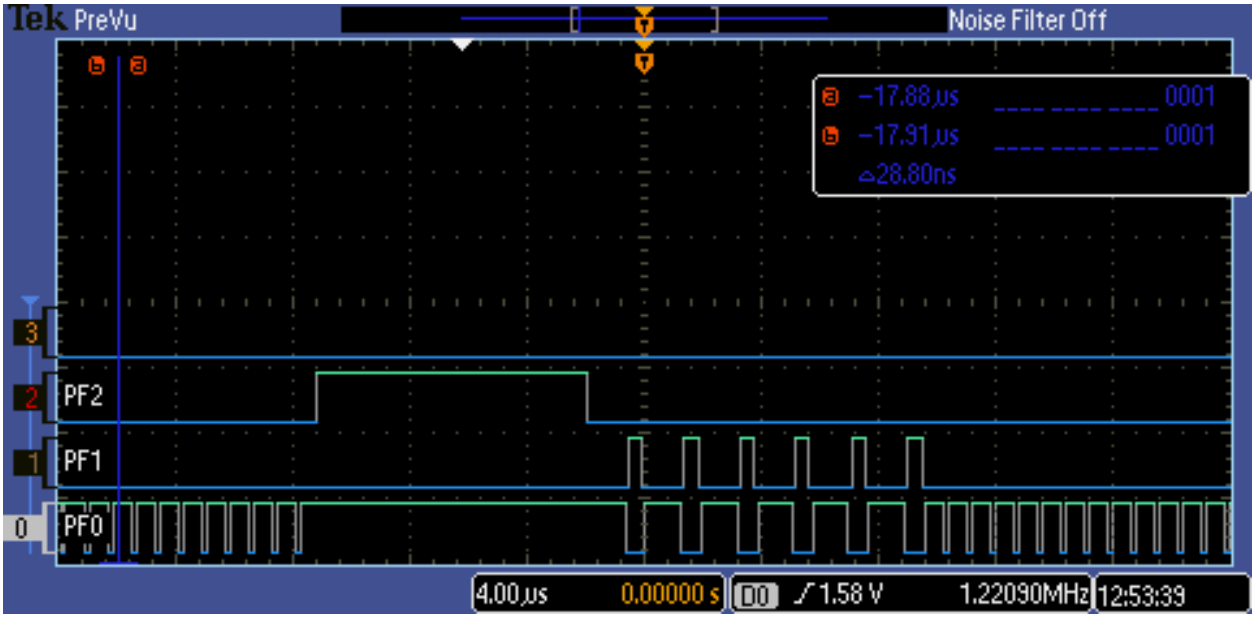
0x000009C4 4601 MOV r1,r0
0x000009C6 481D LDR r0,[pc,#116] ; @0x0A3C
0x000009C8 6800 LDR r0,[r0,#0x00]
0x000009CA 4A1B LDR r2,[pc,#108] ; @0x0A38
0x000009CC 6812 LDR r2,[r2,#0x00]
0x000009CE 4290 CMP r0,r2
0x000009D0 D101 BNE 0x000009D6
0x000009D2 2000 MOVS r0,#0x00
0x000009D4 4770 BX lr
0x000009D6 4818 LDR r0,[pc,#96] ; @0x0A38
0x000009D8 6800 LDR r0,[r0,#0x00]
0x000009DA 7800 LDRB r0,[r0,#0x00]
0x000009DC 7008 STRB r0,[r1,#0x00]
0x000009DE 4816 LDR r0,[pc,#88] ; @0x0A38
0x000009E0 6800 LDR r0,[r0,#0x00]
0x000009E2 1C40 ADDS r0,r0,#1
0x000009E4 4A14 LDR r2,[pc,#80] ; @0x0A38
0x000009E6 6010 STR r0,[r2,#0x00]
0x000009E8 4610 MOV r0,r2
0x000009EA 6802 LDR r2,[r0,#0x00]
0x000009EC 4811 LDR r0,[pc,#68] ; @0x0A34
0x000009EE 3020 ADDS r0,r0,#0x20
0x000009F0 4282 CMP r2,r0

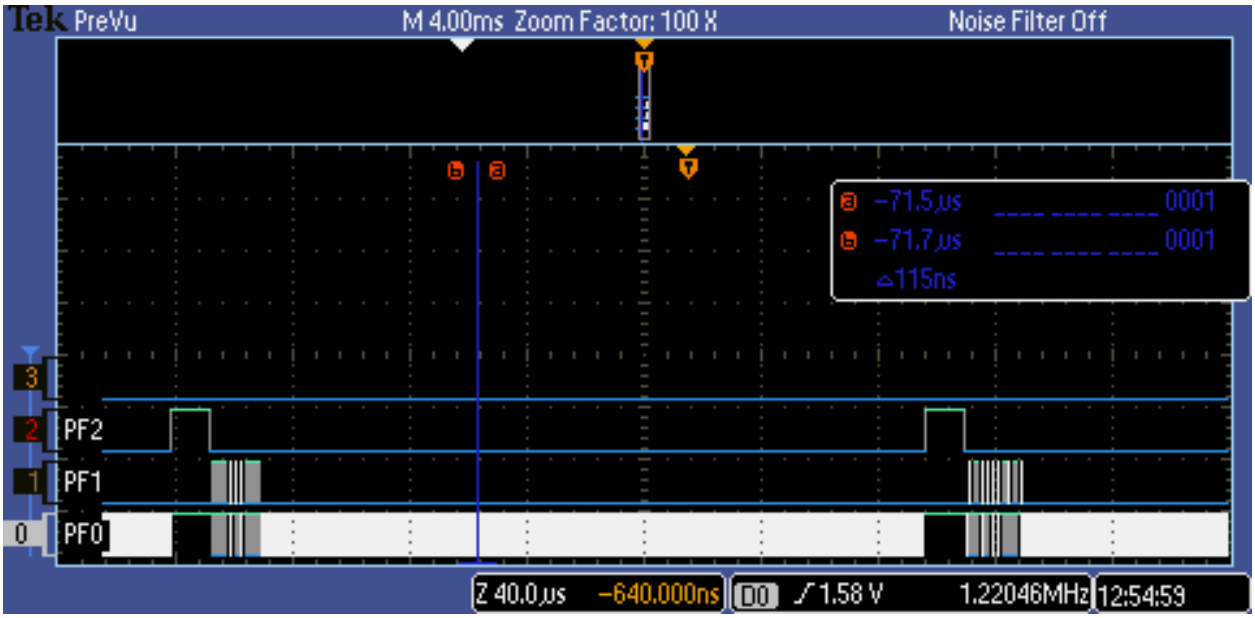
0x000009F2 D102 BNE 0x000009FA
0x000009F4 3820 SUBS r0,r0,#0x20
0x000009F6 4A10 LDR r2,[pc,#64] ; @0x0A38
0x000009F8 6010 STR r0,[r2,#0x00]
0x000009FA 2001 MOVS r0,#0x01
0x000009FC E7EA B 0x000009D4

0x00000A34 00FC LSLS r4,r7,#3
0x00000A36 2000 MOVS r0,#0x00
0x00000A38 0034 MOVS r4,r6
0x00000A3A 2000 MOVS r0,#0x00
0x00000A3C 0030 MOVS r0,r6
0x00000A3E 2000 MOVS r0,#0x00

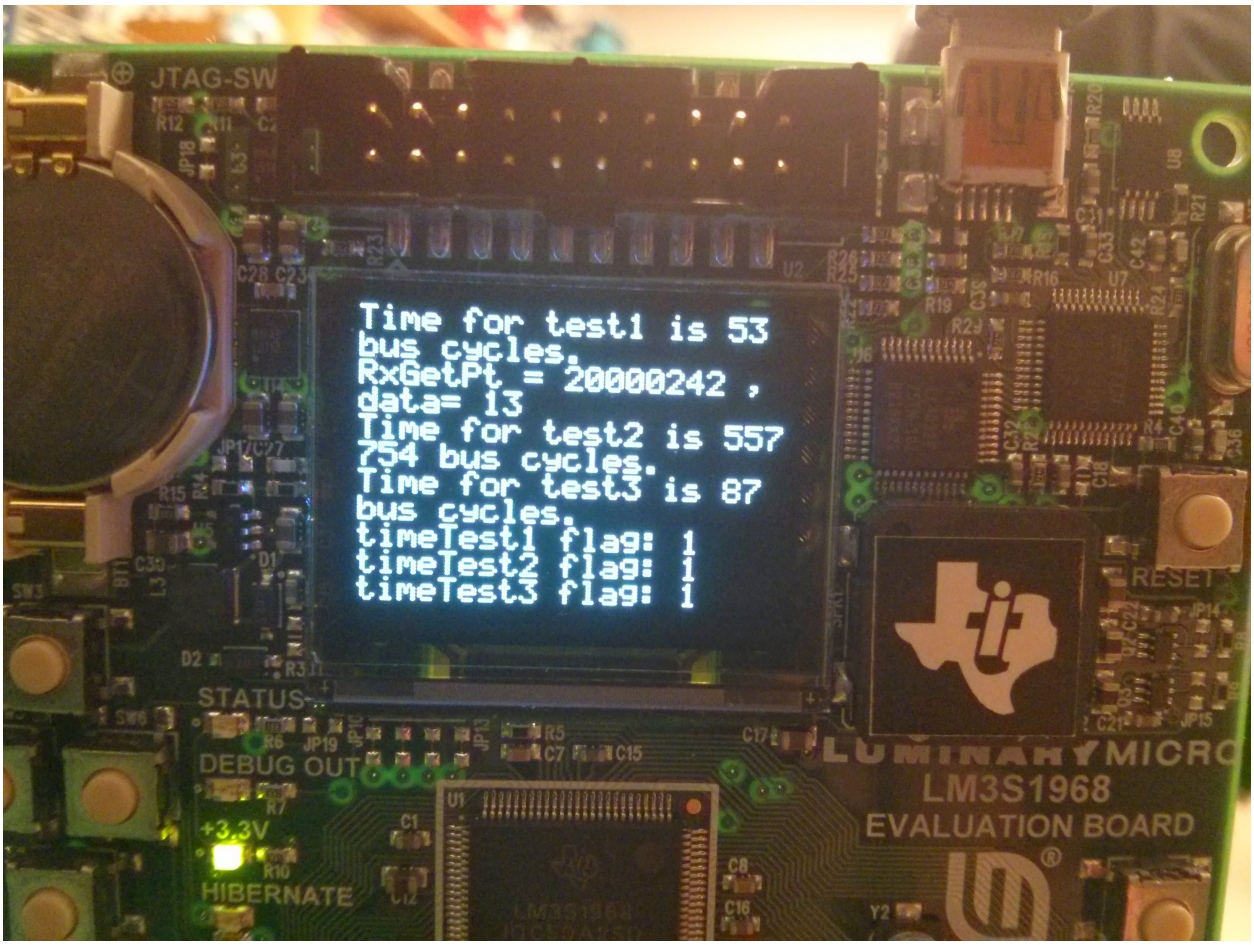
$$36cycles \times (1 \div 50MHz) = .72\mu s$$



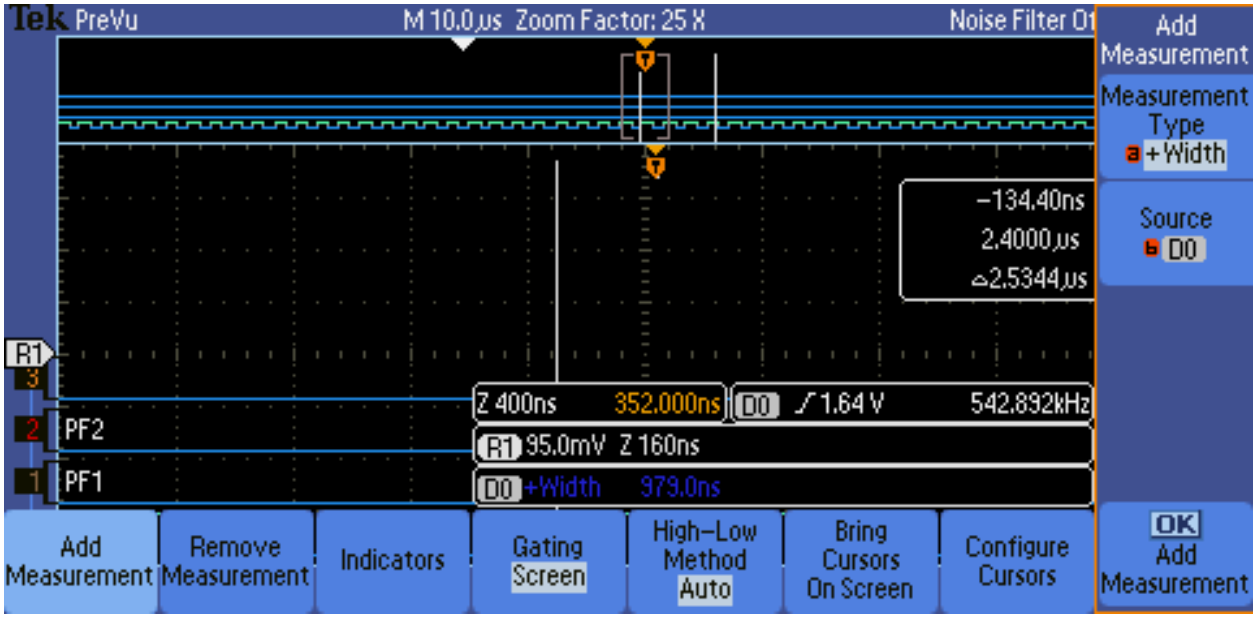


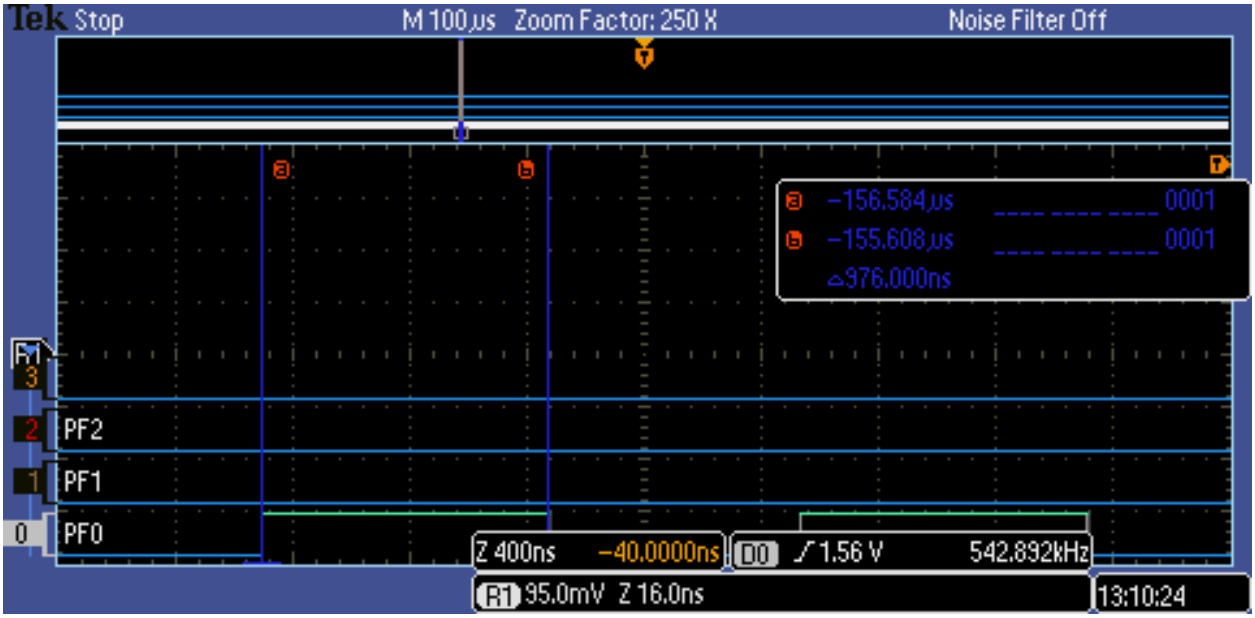



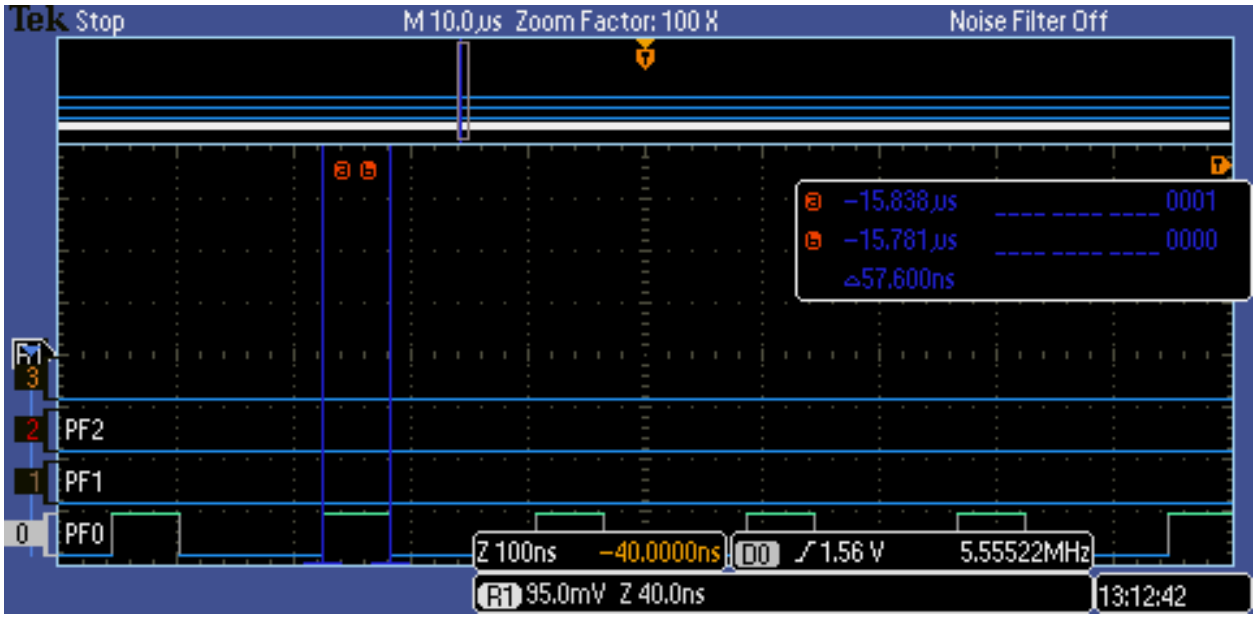

Part B and C: When the interrupt is triggered, the code in the foreground in put on pause. The interrput handler then takes care of it's assigned code. This way, RxFifo_Put is never executed with RxFifo_Get.


Time for test1 is 53
bus cycles,
RxGetPt = 20000242 ,
data= 13
Time for test2 is 557
754 bus cycles,
Time for test3 is 87
bus cycles,
timeTest1 flag: 1
timeTest2 flag: 1
timeTest3 flag: 1
JTAG-SW
RESET
STATUS
DEBUG OUT
+3.3V
HIBERNATE
LUMINARY MICRO
LM3S1968
EVALUATION BOARD

Part D

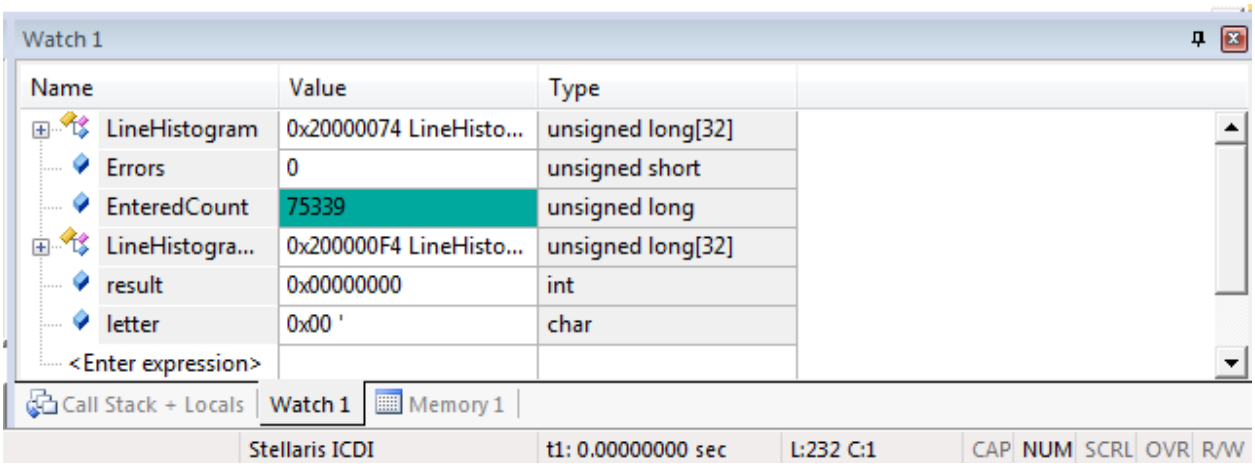

Tek PreVu
M 10.0µs Zoom Factor: 25 X
Noise Filter Of
Add Measurement
Measurement Type
a +Width
Source
b D0
−134.40ns
2.4000µs
Δ2.5344µs
R1
PF2
PF1
Z 400ns 352.000ns
D0 ⎍ 1.64 V 542.892kHz
R1 95.0mV Z 160ns
D0 +Width 979.0ns
Add Measurement
Remove Measurement
Indicators
Gating Screen
High−Low Method Auto
Bring Cursors On Screen
Configure Cursors
OK Add Measurement



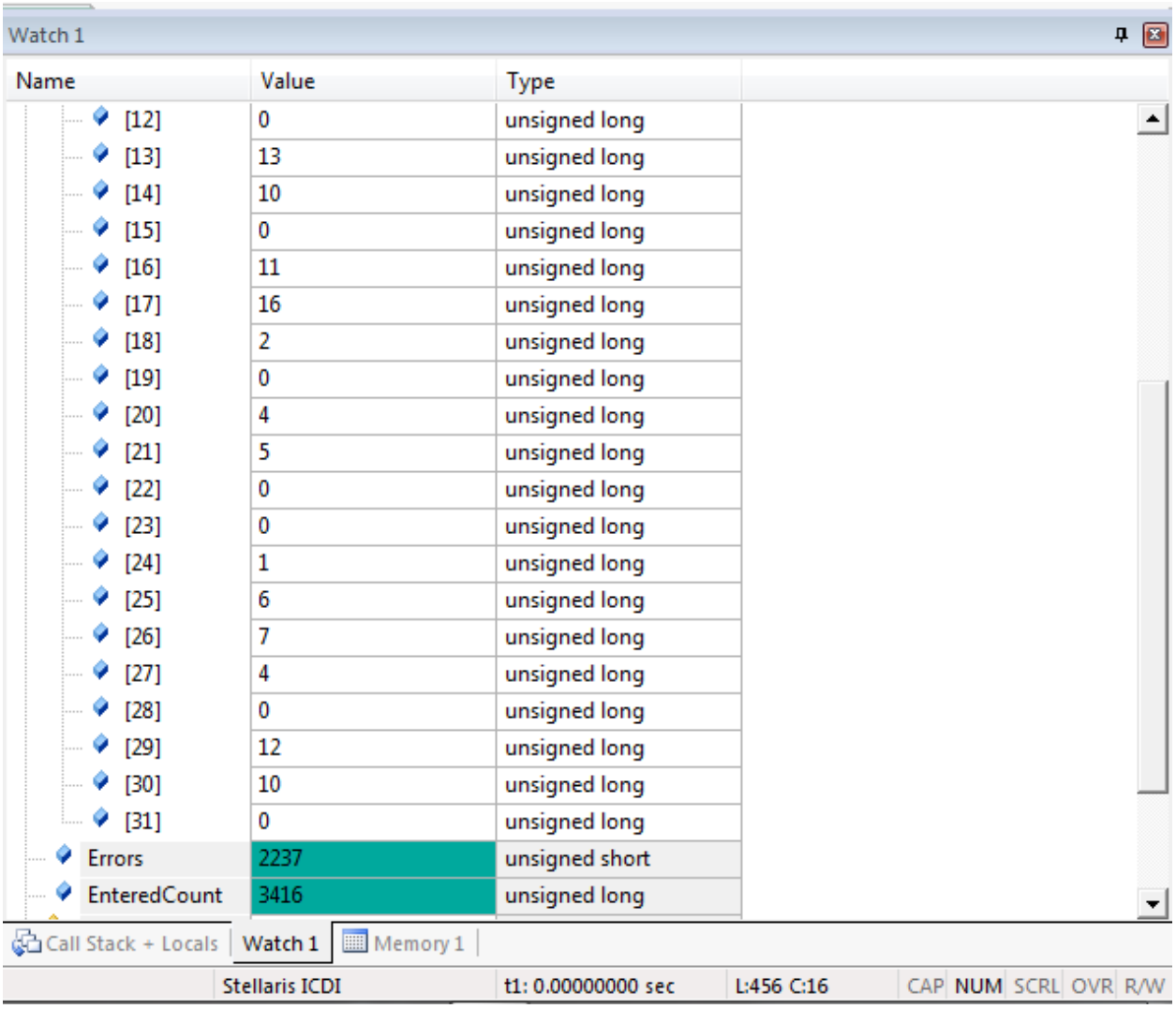




Part E: Refer to Problem 1

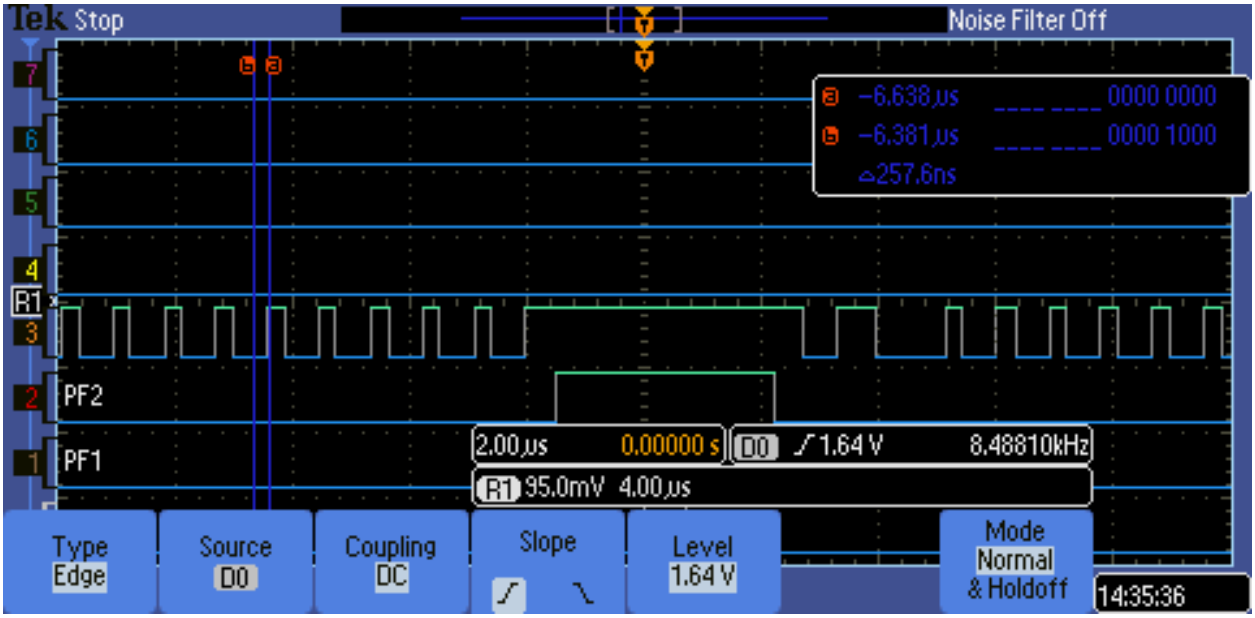
$$976ns - 57.6ns = 918.4ns$$

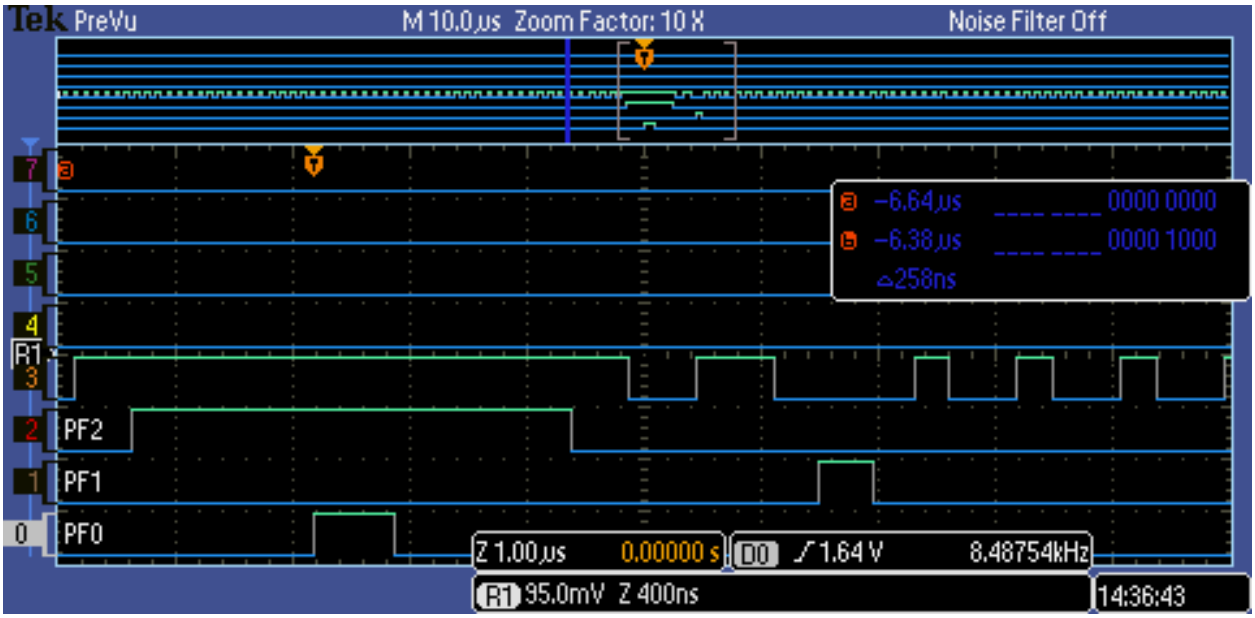

Watch 1
Name
Value
Type
LineHistogram
0x20000074 LineHisto...
unsigned long[32]
Errors
0
unsigned short
EnteredCount
75339
unsigned long
LineHistogra...
0x200000F4 LineHisto...
unsigned long[32]
result
0x00000000
int
letter
0x00 '
char
<Enter expression>
Call Stack + Locals
Watch 1
Memory 1
Stellaris ICDI
t1: 0.00000000 sec
L:232 C:1
CAP NUM SCRL OVR R/W

Part F


Watch 1
Name
Value
Type
[12]
0
unsigned long
[13]
13
unsigned long
[14]
10
unsigned long
[15]
0
unsigned long
[16]
11
unsigned long
[17]
16
unsigned long
[18]
2
unsigned long
[19]
0
unsigned long
[20]
4
unsigned long
[21]
5
unsigned long
[22]
0
unsigned long
[23]
0
unsigned long
[24]
1
unsigned long
[25]
6
unsigned long
[26]
7
unsigned long
[27]
4
unsigned long
[28]
0
unsigned long
[29]
12
unsigned long
[30]
10
unsigned long
[31]
0
unsigned long
Errors
2237
unsigned short
EnteredCount
3416
unsigned long
Call Stack + Locals
Watch 1
Memory 1
Stellaris ICDI
t1: 0.00000000 sec
L:456 C:16
CAP NUM SCRL OVR R/W

Part G




Part H:

$$PF0 \rightarrow TxFifo\_Put$$
$$PF1 \rightarrow Main\ Loop(checks\ for\ errors)$$
$$PF2 \rightarrow Timer\ Interrupt$$
$$PF3 \rightarrow TxFifo\_Get$$

## Analysis and Discussion

**Problem 1.** You measured the execution speed of RxFifo_Get three ways. Did you get the same result? If not, explain.

*We did not get the same results. Timing method one of reading the dissambly and counting instructions was the most time consuming and least efficienent; due to piping and profiling techniques on the board, execution times may differ. The next two timing techniques involved measuring with the systick timer, and by setting GPIO pins high/low and measuring with an oscilliscope. The first method with systick counted 53 cycles of execution at 20ns, providing us with a 1020ns execution time. The second method of using the GPIO pins provided a pulse width of 976ns. We then measured the time it took to set a pin high and low, which came out to 57.6ns. Therefore the GPIO method provided us with a run time of 918.4ns. The overhead of measuring the systick register added the slight drift in measurement times. The difference was about 100ns, or 5 bus cycles. This would match up with the overhead of reading a register and masking the data.*

**Problem 2.** Which method of measuring execution speed would you use if you expected the execution speed to vary a lot (e.g., ranging from 0.5 to 20ms), and wanted to determine the minimum, maximum and average speed? Why?

*Measuring varying pulsewidths on a logic analyser can be difficult. It also requires being around a logic analyser. It would be much simpler to dump execution times into an array using systick measurements, and then run averages on that data.*

**Problem 3.** Which method of measuring execution speed would you use if you expected the execution speed to be very large (e.g., 20 seconds)? Why?

*For very large values, you should use an oscilloscope since systick will overflow and reset.*

**Problem 4.** Define "minimally intrusive".

*Debugging techniques that have minumum overhead. In other words, the software should run nearly the same as if the debugging tools were not present, in real time. Printf and using breakpoints are examples of intrusive debugging.*

**Problem 5.** List the two necessary components collected during a "profile".

*Profilers collects time history of program execution. Two necessary components are the location and time that a thread executes. Data involved in the execution may also be logged.*

**Problem 6.** What is the critical section in the bad FIFO? How would you remove the critical section?

*BadFifo does not use a volatile put pointer for one, the data pointer may be corrupted if interrupted during its execution. A start and end critical section can also be placed around sections handling data.*