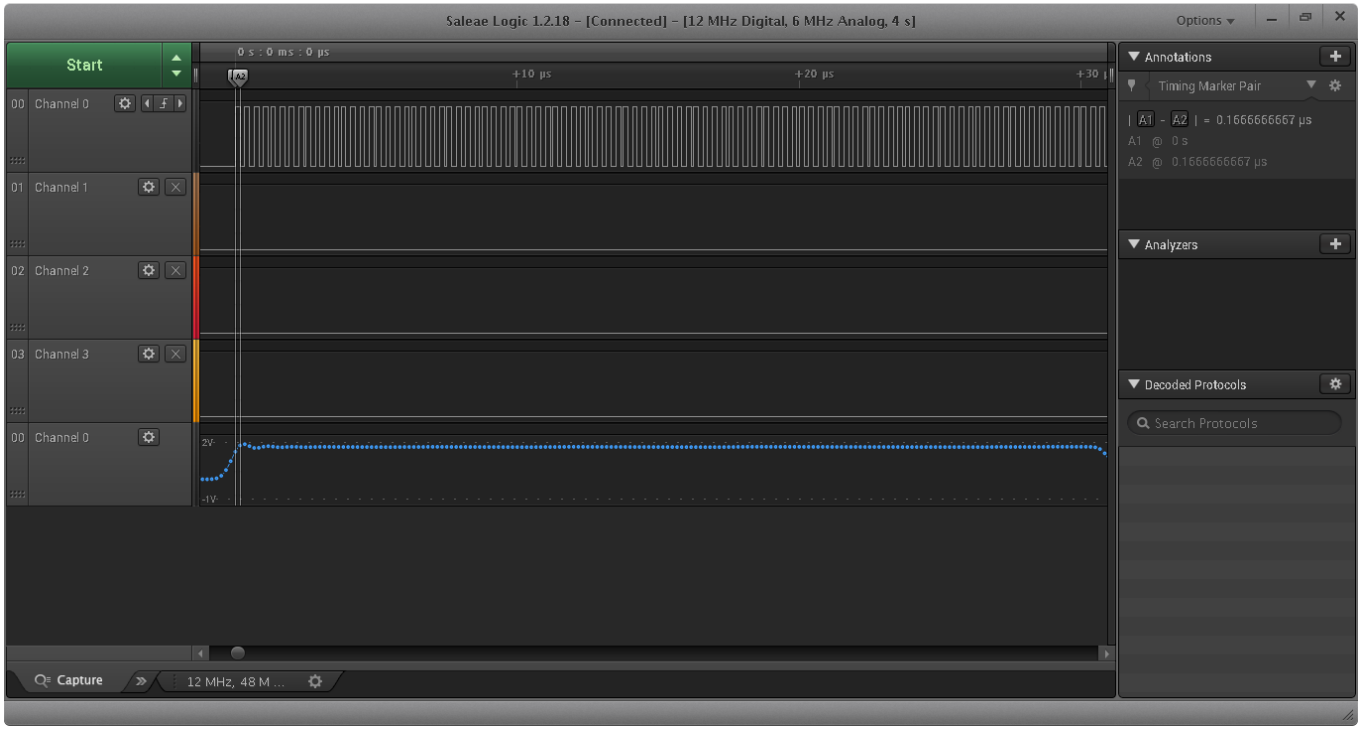
Project 2 Report

Date-11/07/2018
Name-Emil Prisquilas Peter
Unity Email Address- epeter@ncsu.edu

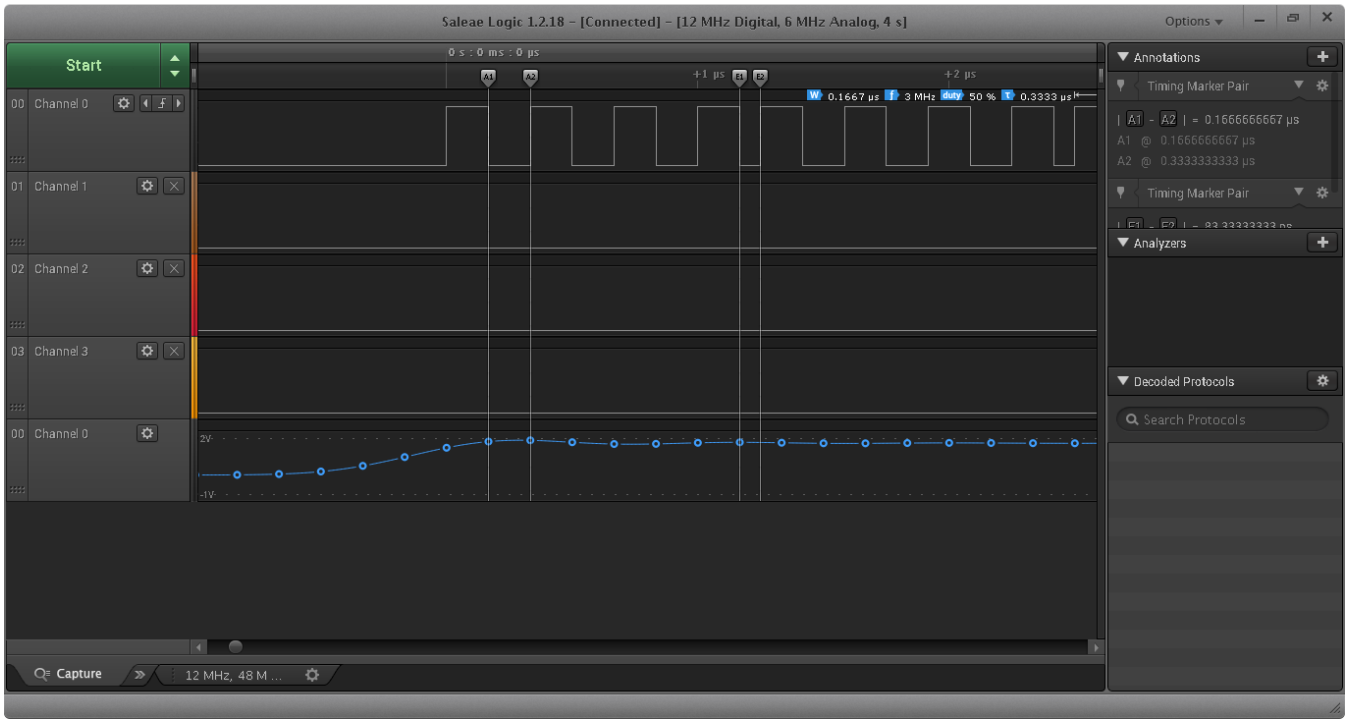
Part B

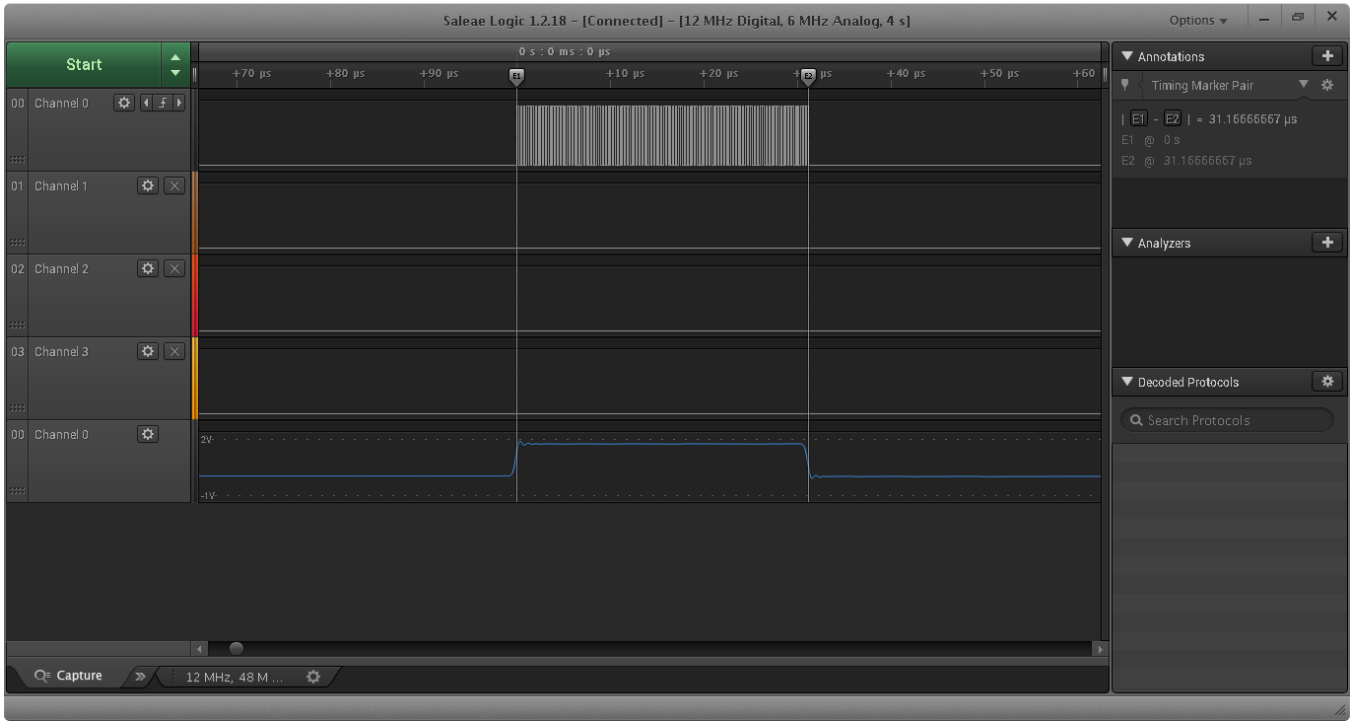
Port to CMSIS-RTOS2

5. **a. What is the "100% idle for one second" value for idle_counter?**
   3660310
6. **Idle loop analysis**
   a. **How long is an average idle loop iteration (based on 100 iterations)? Include a screenshot.**
   0.16667µs

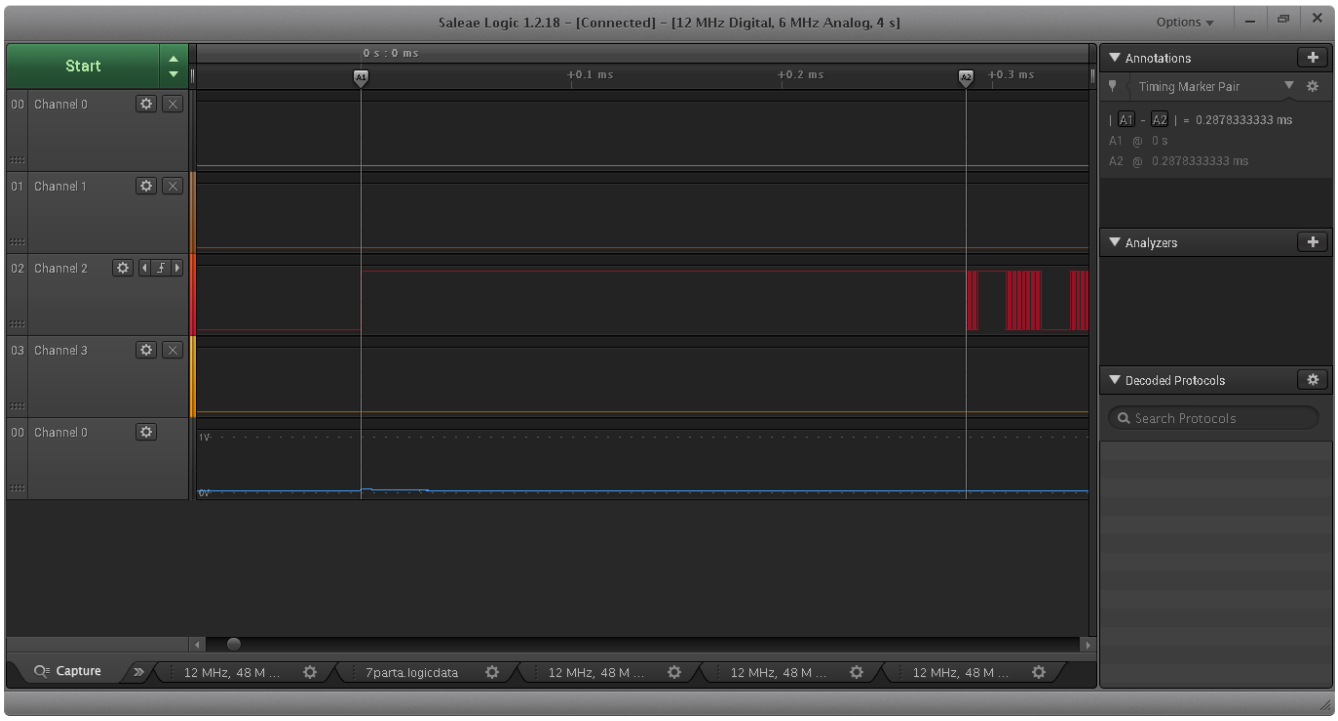


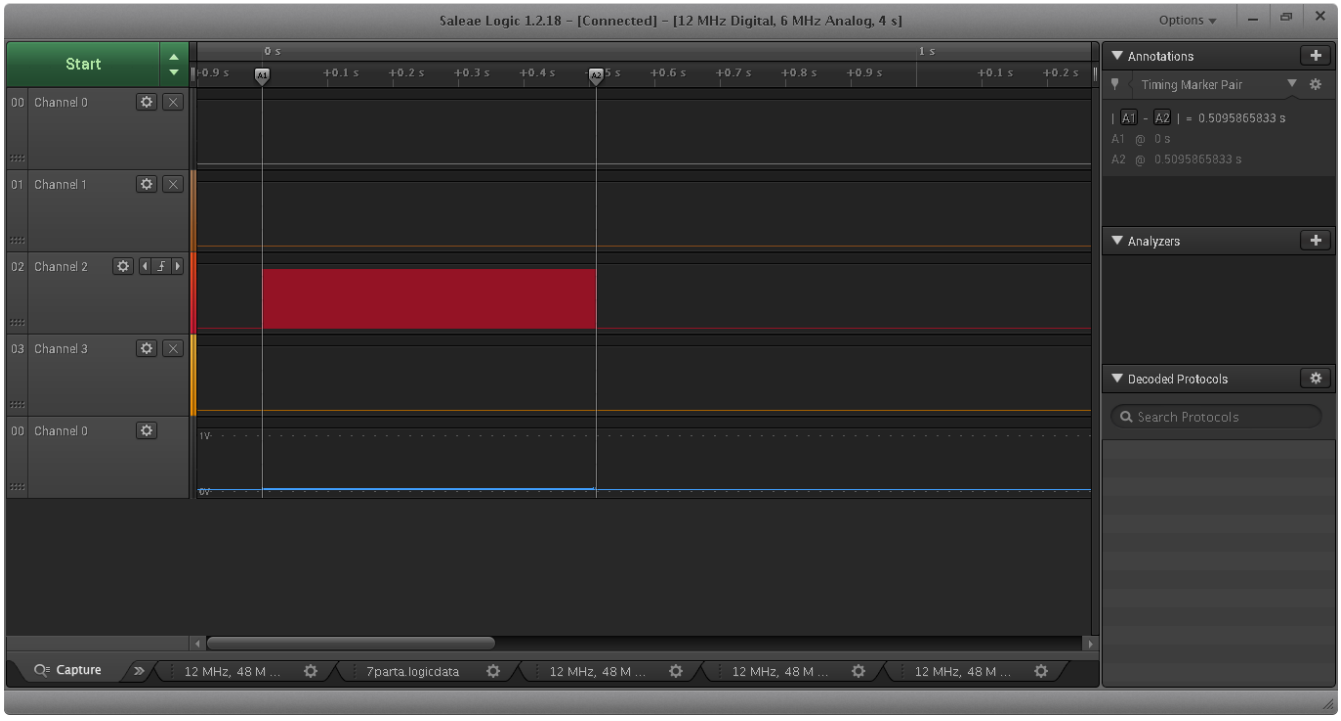
   b. **How often does the scheduler tick run, and how long does it take each time? Determine this using the gaps in the idle debug signal. What fraction of the processor's time does the timer tick use? Include a screenshot.**
   It runs 3 out of 4 times for every iteration. Its takes 40.22% of the processor's time.
   Total time taken for 100 iterations is 31.1667 µs. The scheduler tick runs for 0.16667 µs

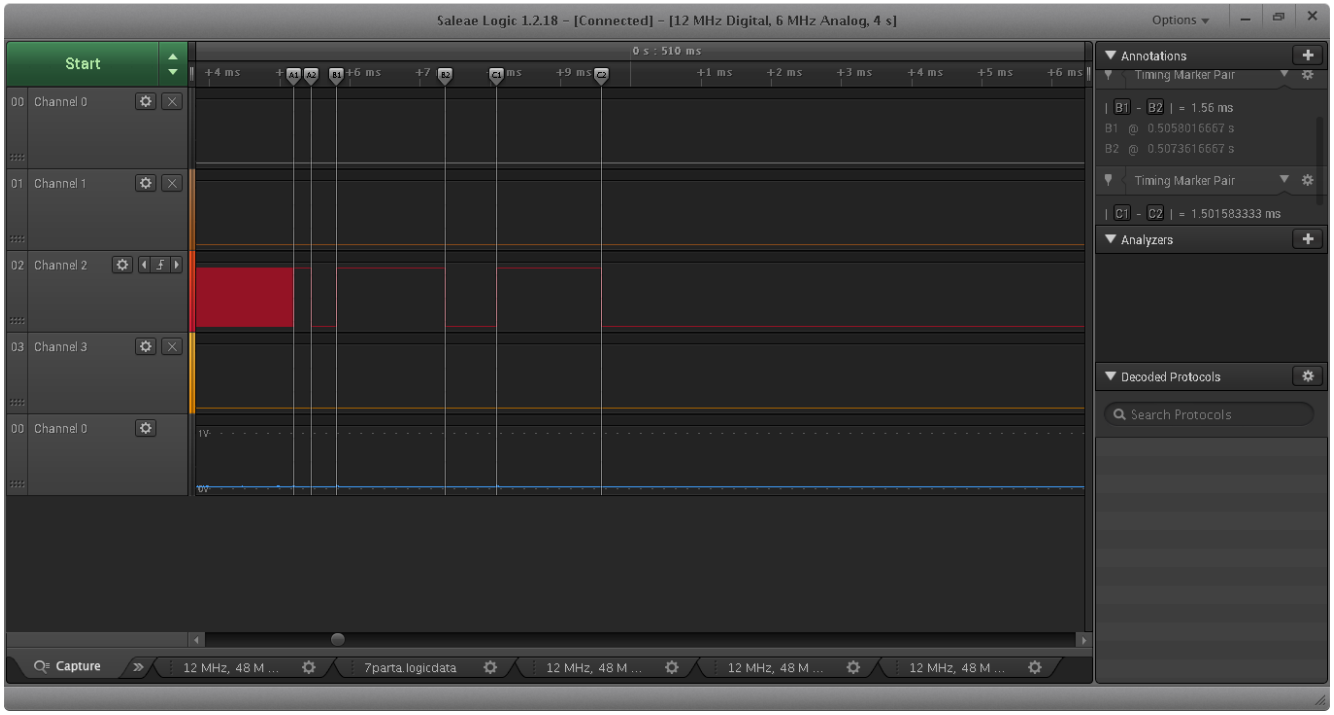

Saleae Logic 1.2.18 - [Connected] - [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
0 s : 0 ms : 0 µs
+1 µs
+2 µs
W 0.1667 µs
f 3 MHz
duty 50 %
T 0.3333 µs
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
00 Channel 0
2V
-1V
Annotations
Timing Marker Pair
A1 - A2 | = 0.1666666667 µs
A1 @ 0.1666666667 µs
A2 @ 0.3333333333 µs
Timing Marker Pair
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...


Saleae Logic 1.2.18 - [Connected] - [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
0 s : 0 ms : 0 µs
+70 µs
+80 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
+60
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
00 Channel 0
2V
-1V
Annotations
Timing Marker Pair
E1 - E2 | = 31.16666667 µs
E1 @ 0 s
E2 @ 31.16666667 µs
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...

7. **How long does each of the following functions take, and how long is each segment (computation, SPI comm. for polling, other SPI comm.)? The computation time is how long the function would take if the SD card and the SPI communication were infinitely fast (taking no time). Include a screenshot marked to indicate the busy waiting time.**
    a. **SD_Init**







A1-A2=0.304ms (Computation)

B1-B2=0.500s(Computation)

C1-C2=0.2826ms(Computation)

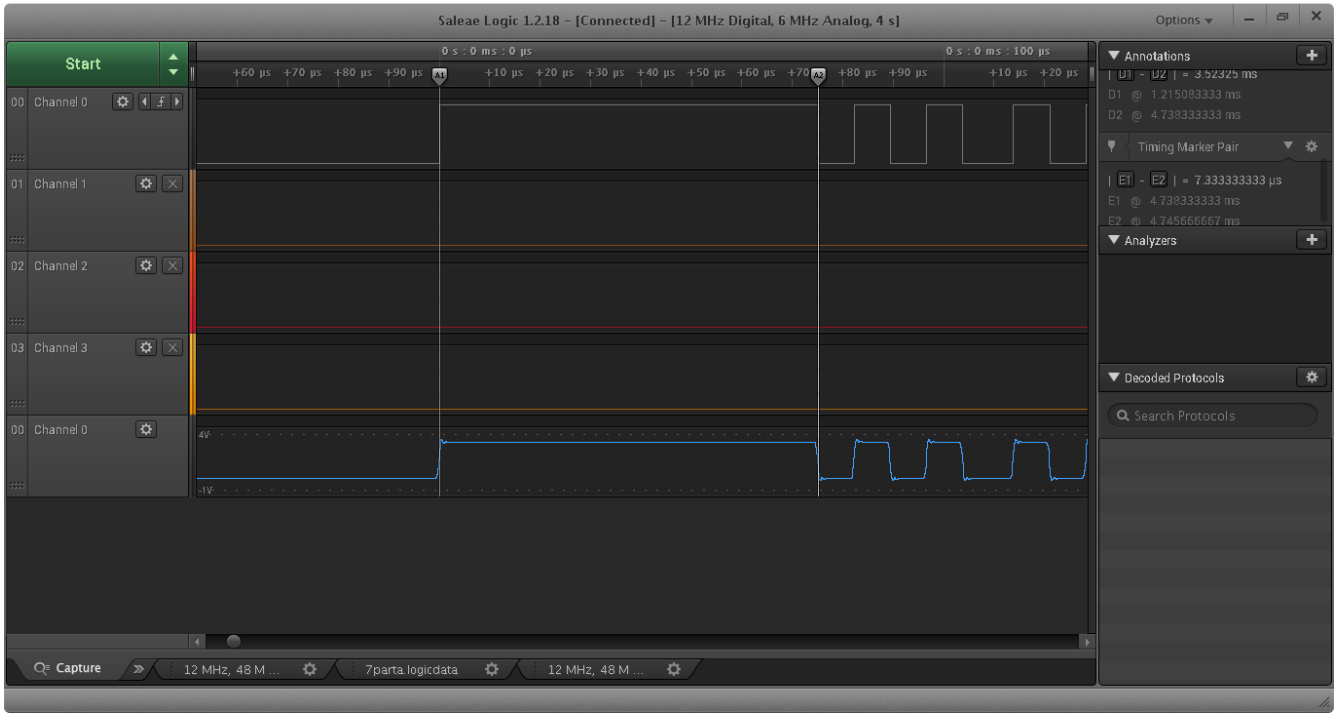
D1-D2=0.301ms (SPI Polling)

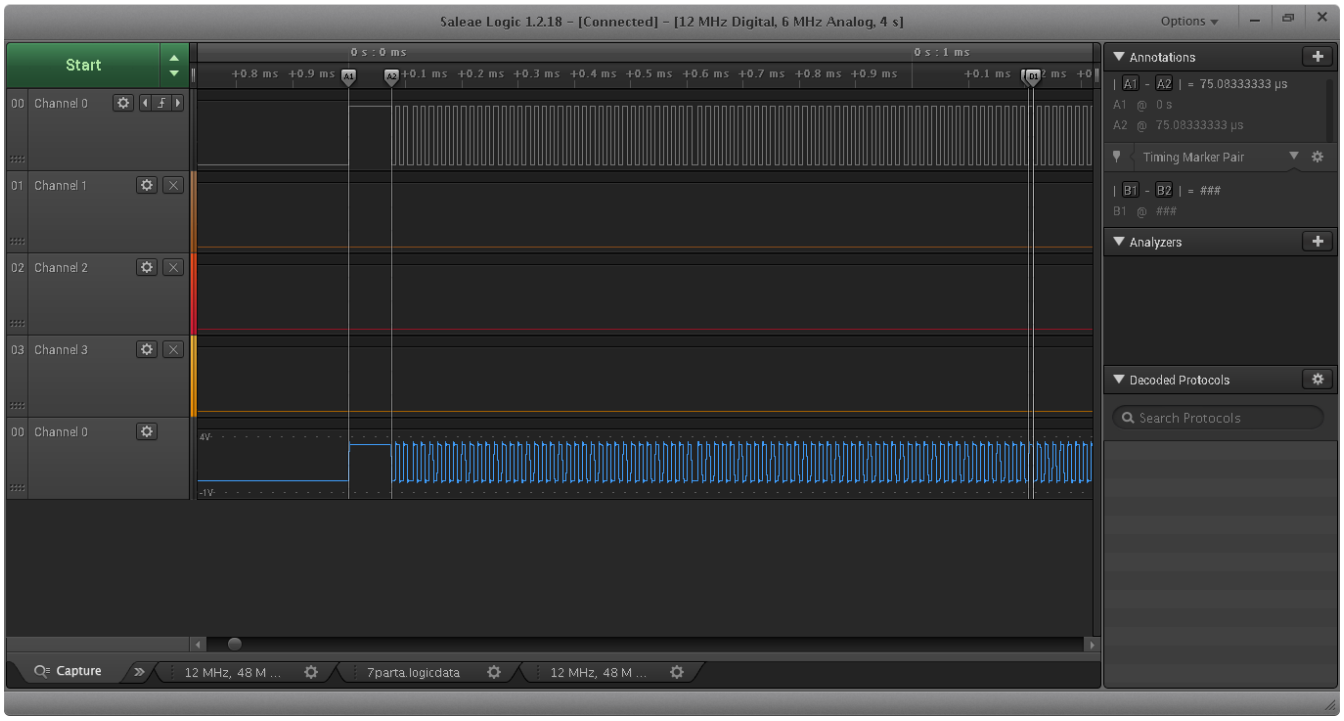
E1-E2=1.327ms(Other SPI communication)

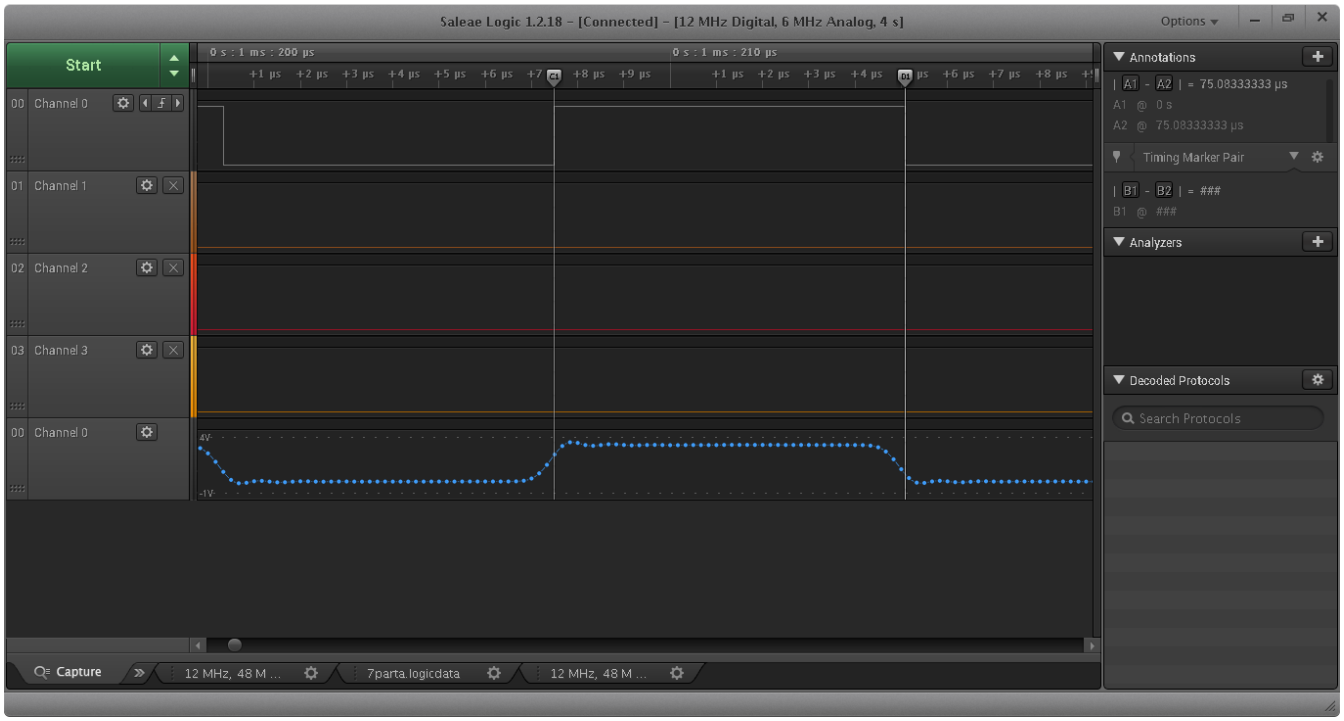
F1-F2=0.601ms(SPI Polling)

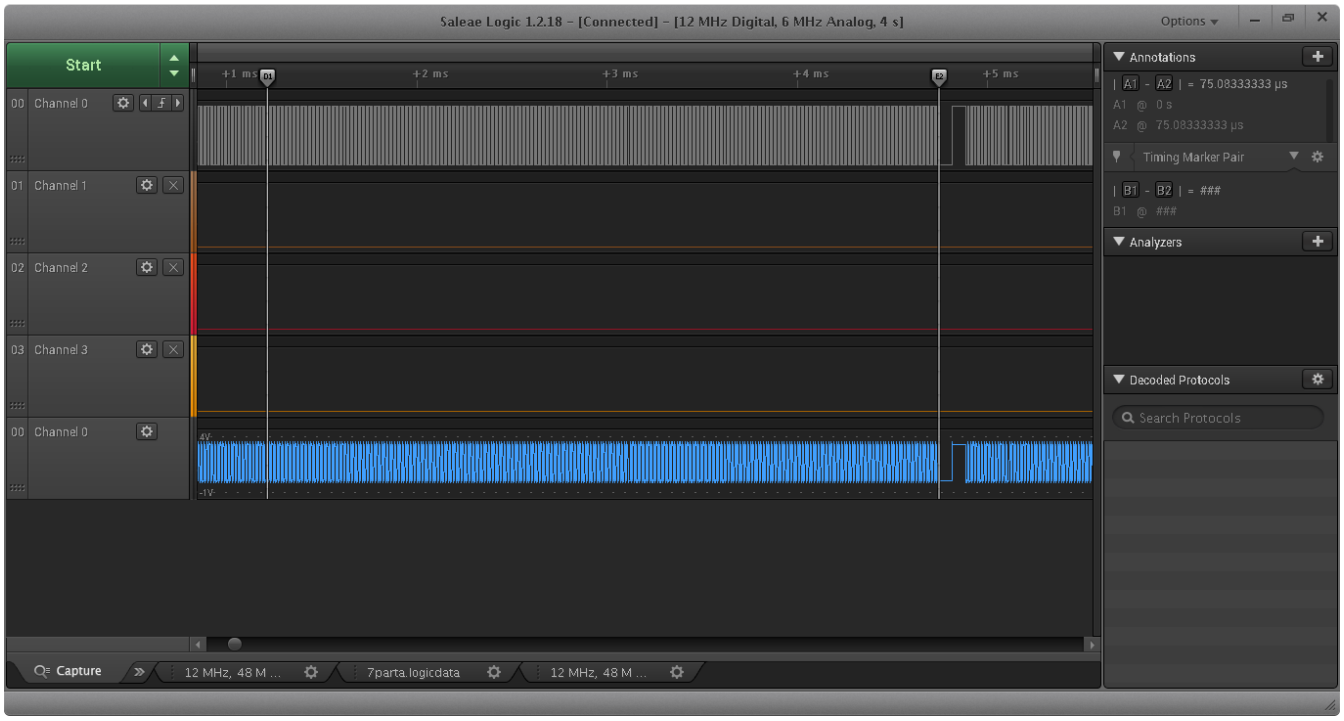
G1-G2=1.3375ms(Other SPI communication)

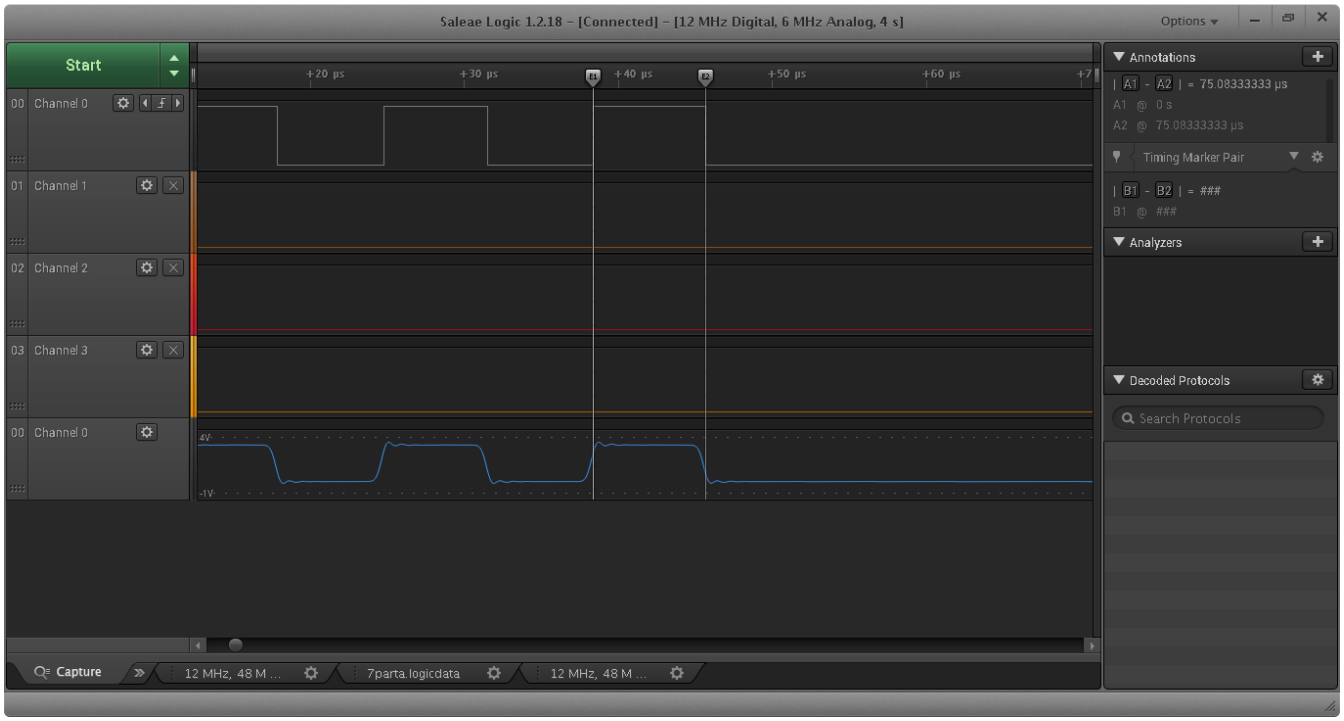
b. SD_Read




Saleae Logic 1.2.18 – [Connected] – [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
0 s : 0 ms
0 s : 1 ms
Channel 0
Channel 1
Channel 2
Channel 3
Channel 0
Annotations
Timing Marker Pair
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...
7parta.logicdata
12 MHz, 48 M ...

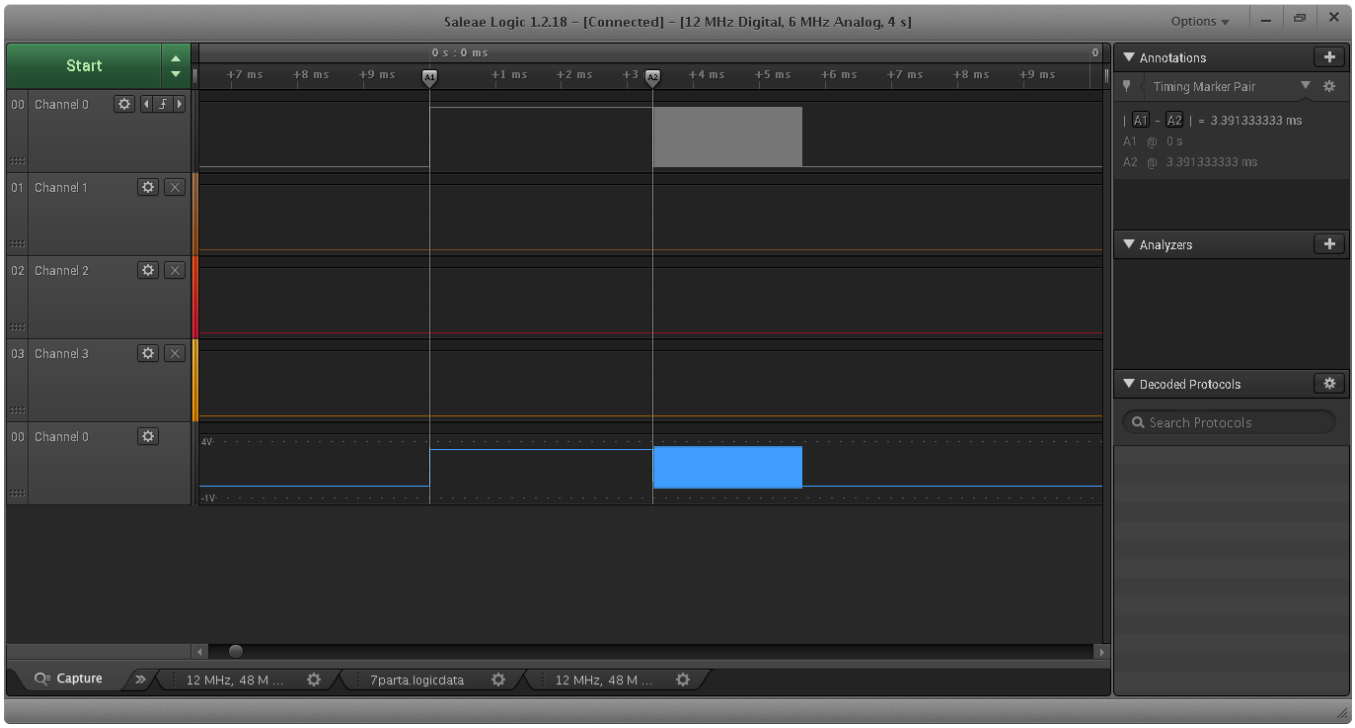

Saleae Logic 1.2.18 – [Connected] – [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
0 s : 1 ms : 200 µs
0 s : 1 ms : 210 µs
Channel 0
Channel 1
Channel 2
Channel 3
Channel 0
Annotations
Timing Marker Pair
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...
7parta.logicdata
12 MHz, 48 M ...

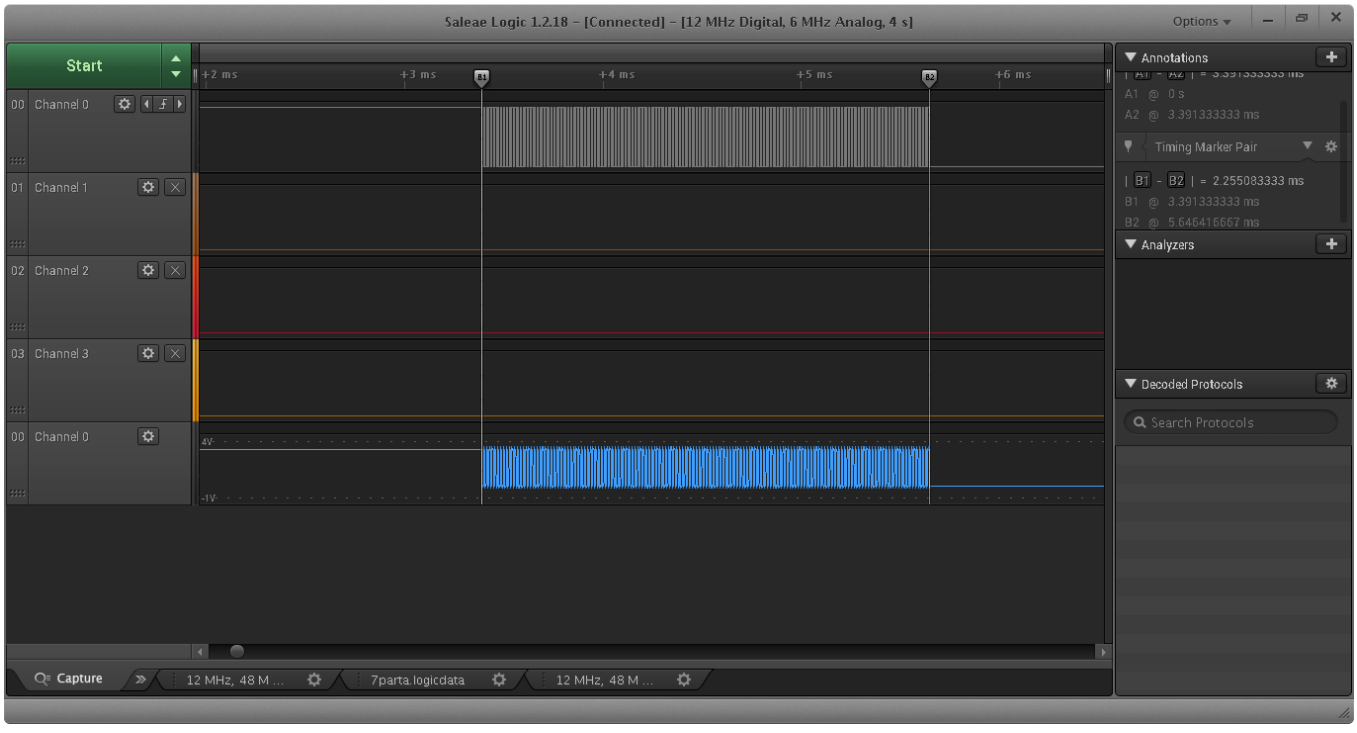


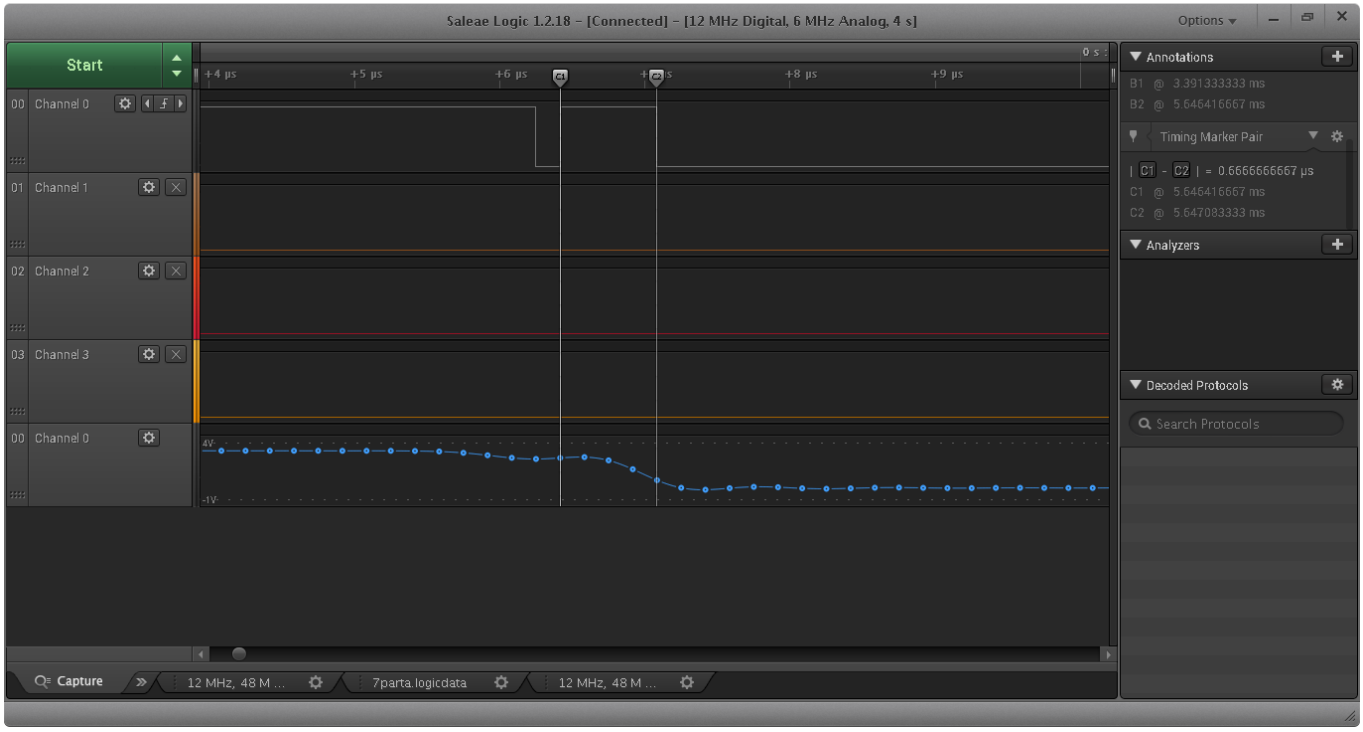


A1-A2=75.083µs(Other SPI communication)
B1-B2=1.1324ms(SPI Polling)
C1-C2=7.5833 µs(Computation)
D1-D2=3.52325ms(SPI Polling)
E1-E2=7.33 µs(Computation)

c. SD_Write


Saleae Logic 1.2.18 – [Connected] – [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
0 s : 0 ms
+7 ms
+8 ms
+9 ms
+1 ms
+2 ms
+3 ms
+4 ms
+5 ms
+6 ms
+7 ms
+8 ms
+9 ms
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
00 Channel 0
4V
-1V
Annotations
Timing Marker Pair
A1 - A2 = 3.391333333 ms
A1 @ 0 s
A2 @ 3.391333333 ms
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...
7parta.logicdata
12 MHz, 48 M ...


Saleae Logic 1.2.18 – [Connected] – [12 MHz Digital, 6 MHz Analog, 4 s]
Options
Start
+2 ms
+3 ms
+4 ms
+5 ms
+6 ms
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
00 Channel 0
4V
-1V
Annotations
A1 @ 0 s
A2 @ 3.391333333 ms
Timing Marker Pair
B1 - B2 = 2.255083333 ms
B1 @ 3.391333333 ms
B2 @ 5.646416667 ms
Analyzers
Decoded Protocols
Search Protocols
Capture
12 MHz, 48 M ...
7parta.logicdata
12 MHz, 48 M ...



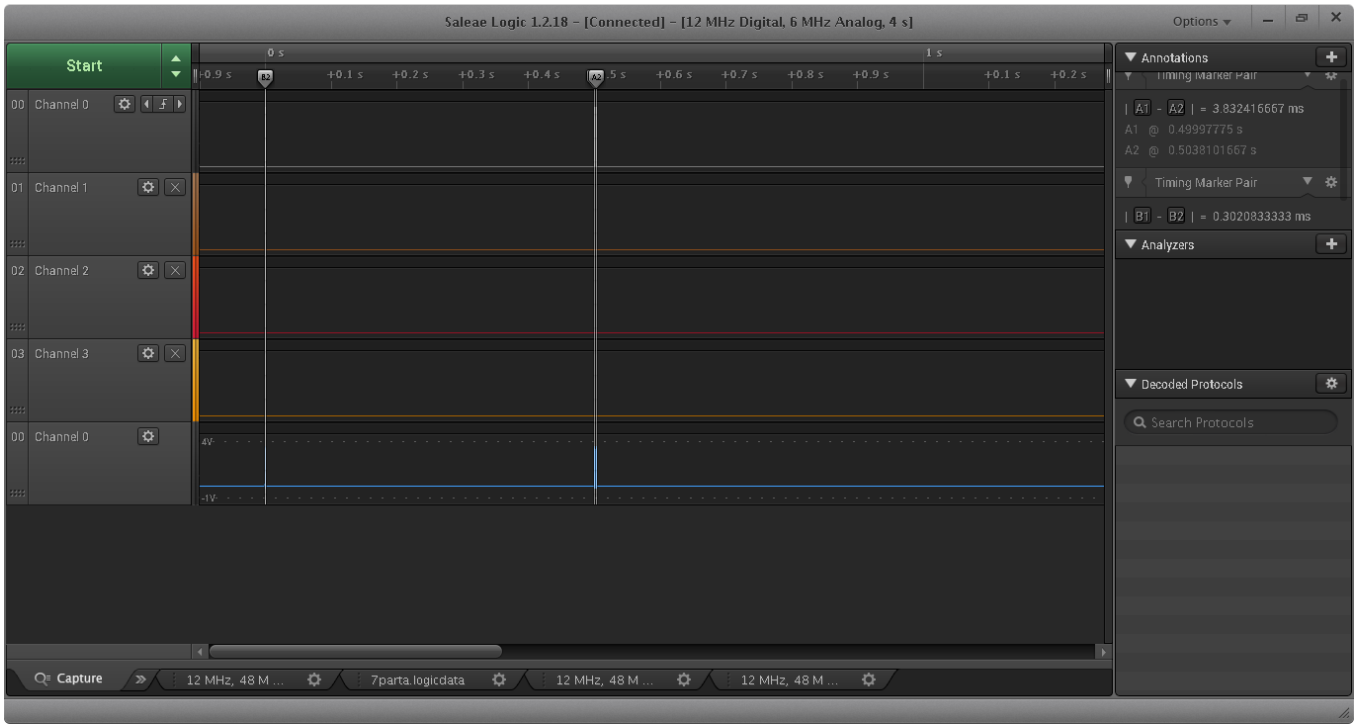
A1-A2=3.391ms(Other SPI communication)
B1-B2=2.255ms(SPI Polling)
C1_c2=0.667 µs(Computation)

SD_Init

10. Analysis

a. How long does SD_Init take to execute? How long is each segment now? Include a screenshot.

4.1375ms



Except for the Timer delay section,rest of the timings remain same.

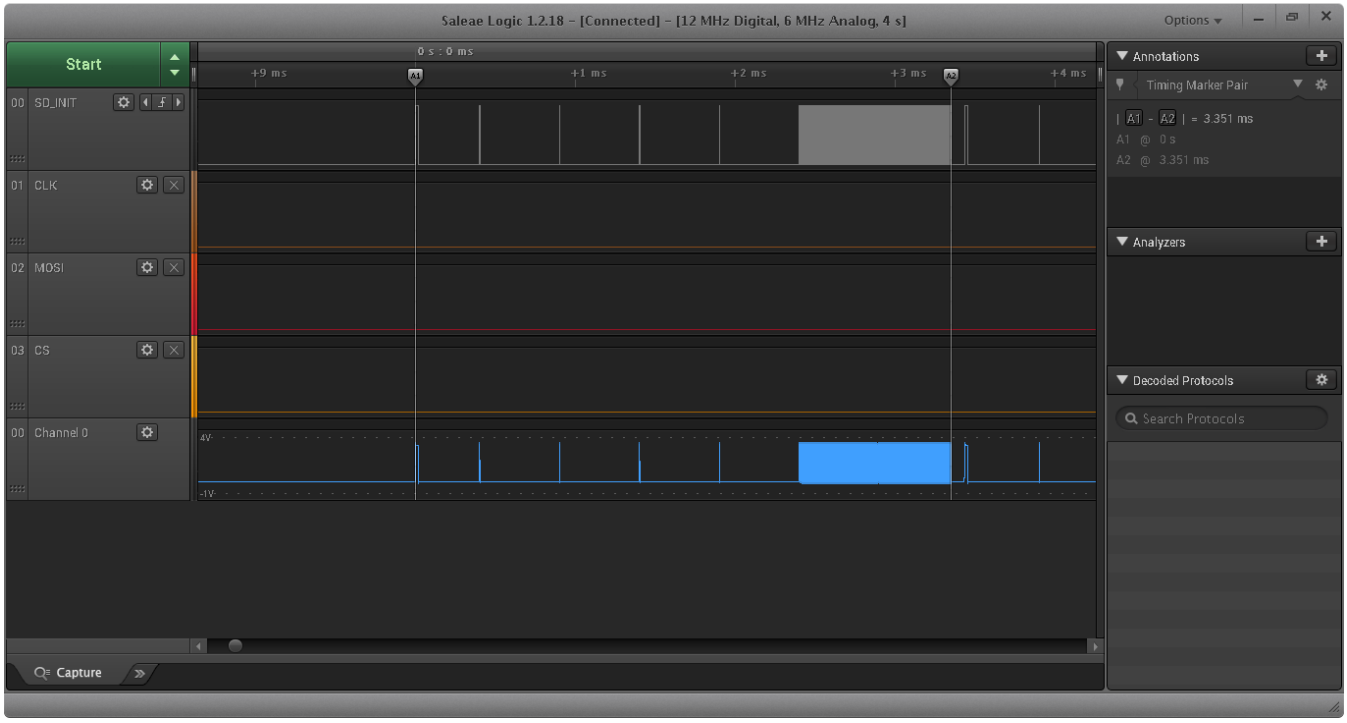
b. **How much idle time is available now, based on idle_counter?**
0.3048s

c. **Do these values differ from the previous implementation? Why or why not?**
Yes,the values differ from the previous implementation since the CPU spends less within the SD_Init thread as the thread is blocked during the 500ms delay duration.

SD_Read

12. Analysis

a. **How long does the SD_Read function take to complete? How long is each segment now? Include a screenshot.**
3.2294ms



b. **How much idle time is available now, based on idle_counter?**
0.001017

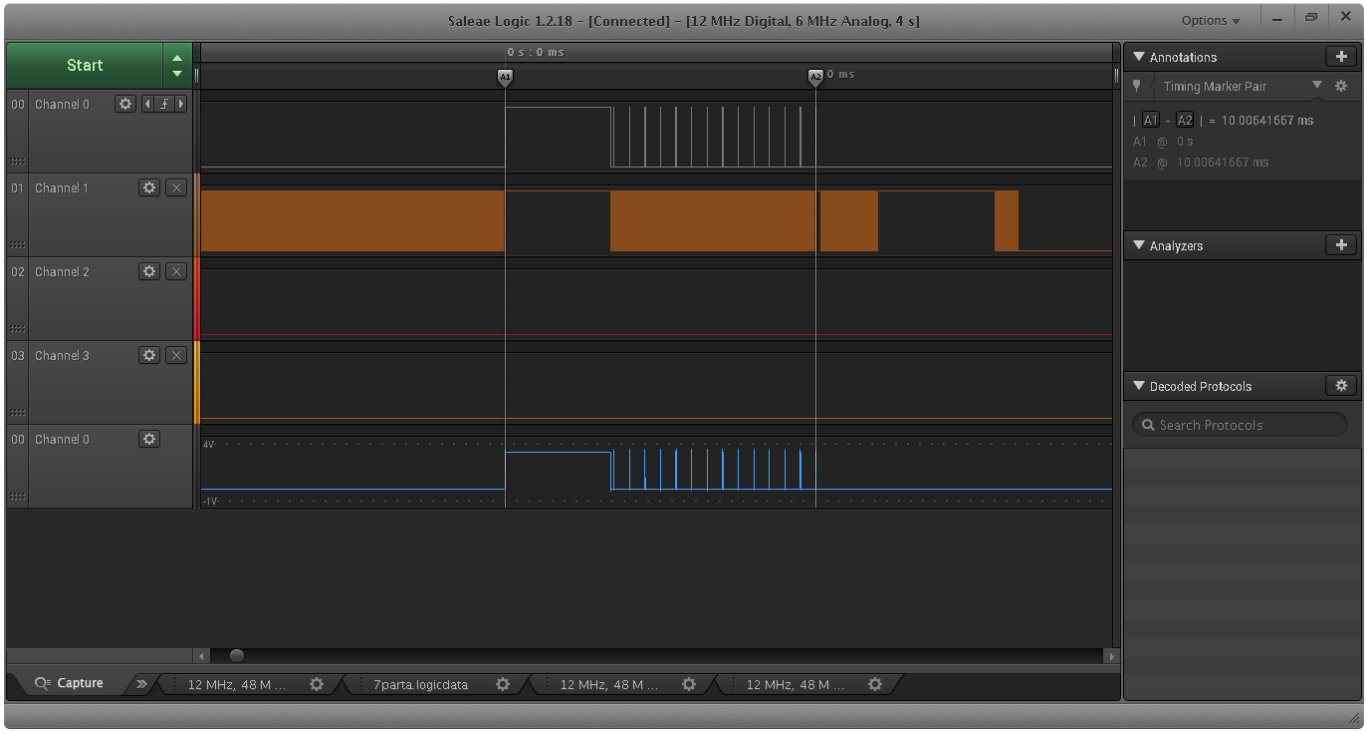
c. **Do the times differ from the previous implementation? Why or why not?**
Yes, the times differ from the previous implementation since the polling frequency has decreased due to the introduction of osDelay function in the thread. During the osDelay the thread is blocked and the idle thread is run.

SD_Write

13. Change this code to use an osDelay() call to reduce the polling frequency to once per timer tick. Include a screenshot from the logic analyzer.

14. Analysis

a. **How long does the SD_Read function take to complete? How long is each segment now? Include a screenshot.**
3.387ms



b. How much idle time is available now, based on idle_counter?

0.001813s

c. Do the times differ from the previous implementation? Why or why not?

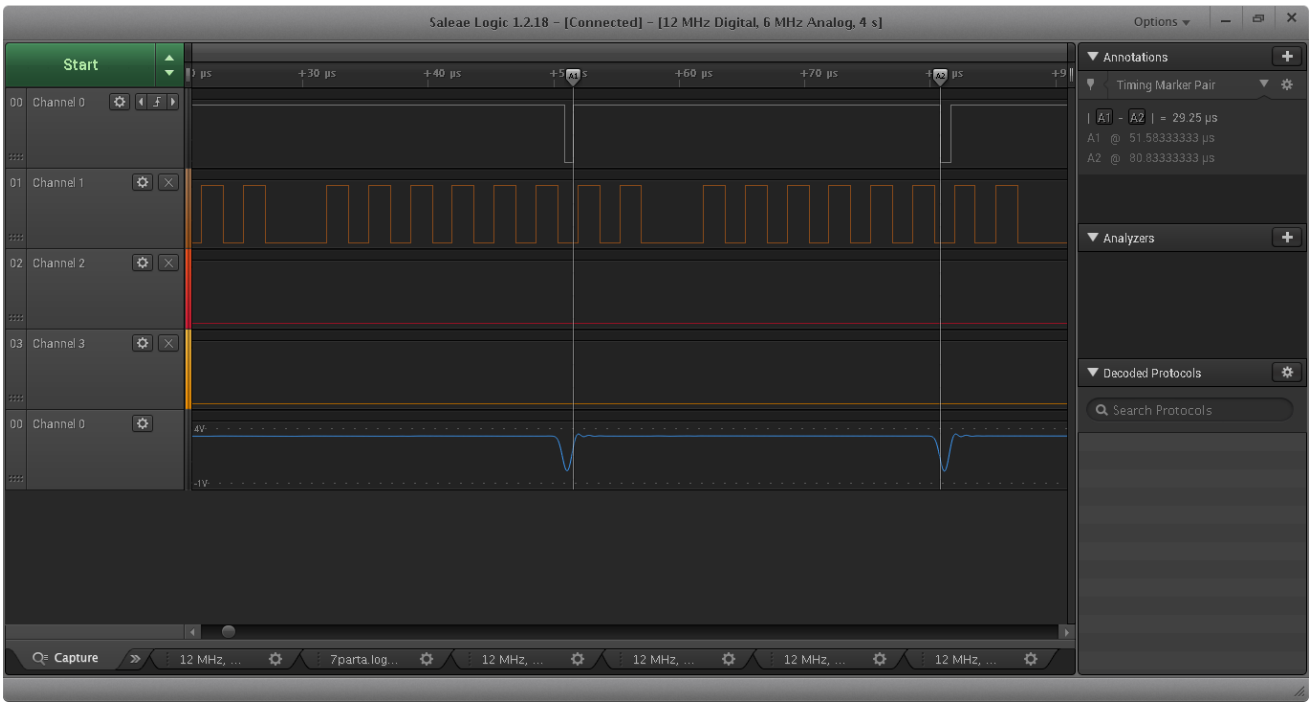
Yes, the times differ from the previous implementation since the polling frequency has decreased due to the introduction of osDelay function in the thread. During the osDelay the thread is blocked and the idle thread is run.

SPI_RW

18. SPI Processing Analysis. Consider the operations occurring when executing SPI_RW() once (assuming SPI_Freq_Low() has been called). Mark these events and times on a screenshot.

a. How long does SPI_RW() take to execute (DBG_SPI_RW rising edge to falling edge)?

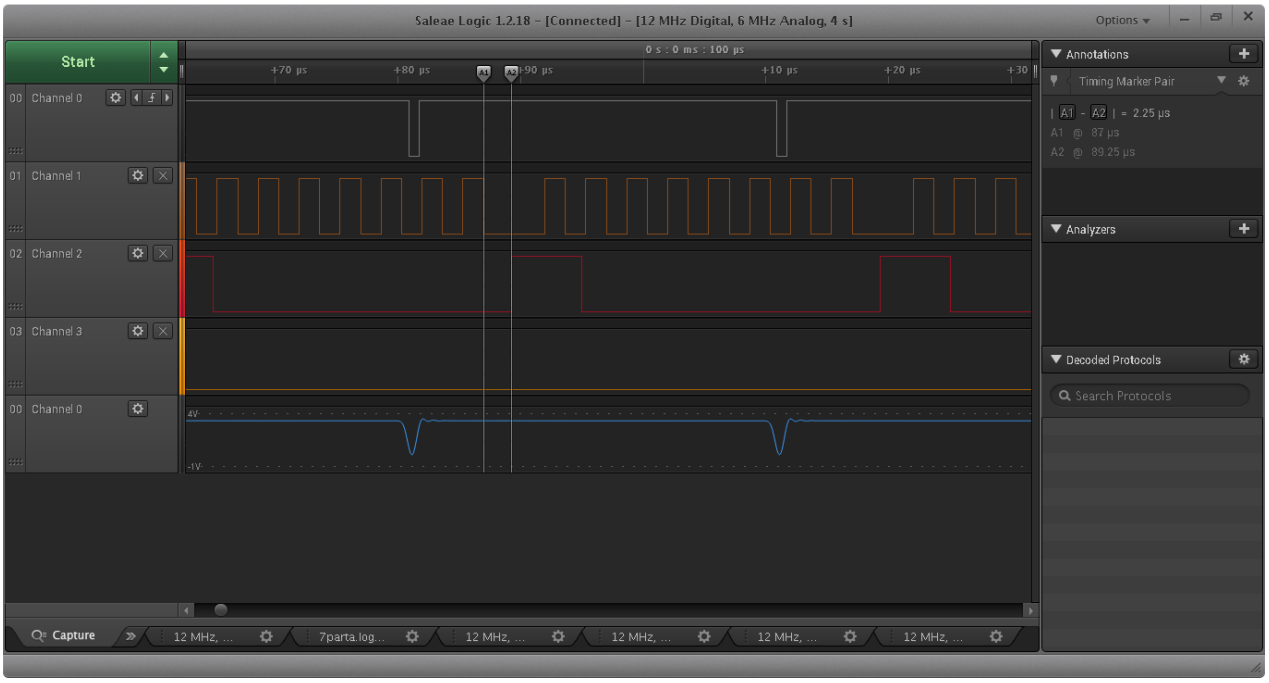
29.25µs.



b. **How much of this time is used for SPI communications (observe SPI Clock)?**
24.334 µs.

c. **How much idle time is available during an execution of SPI_RW()?**

d. **What is the delay from SPI communication completion to the start of SPI1_IRQHandler()? How much of this is the Cortex-M0+ CPU's delay in responding to an interrupt (see ESF chapter 4 and KL25 Reference Manual)?**
2.25 µs.
The CPU takes 15 cycles to compute instructions during hardware exceptions .Therefore, a major part of CPU's delay goes in responding to hardware interrupts.



e. **How long does SPI1_IRQHandler() take to execute?**
5.75 µs.

f. **What is the delay from the SPI1_IRQHandler() completing to SPI_RW() completing? How much of this is the Cortex-M0+ CPU's delay in responding to an interrupt (see ESF chapter 4 and KL25 Reference Manual)? What else causes this delay?**
15.83 µs.
The CPU takes 15 cycles to compute instructions during hardware exceptions .Therefore, a major part of CPU's delay goes in responding to hardware interrupts.

19. **ECE-560 Only: Alternatives**

a. **Why are we using a message queue instead of simply inserting osDelay(1) calls in the SPI_RW() blocking loops?**
Message queues are used since they don't eat much of CPU time compared to osDelay(1).In case of os Delay(1) the CPU performs polling for every tick, while the CPU performs computation in the thread only when the message is placed in the queue.

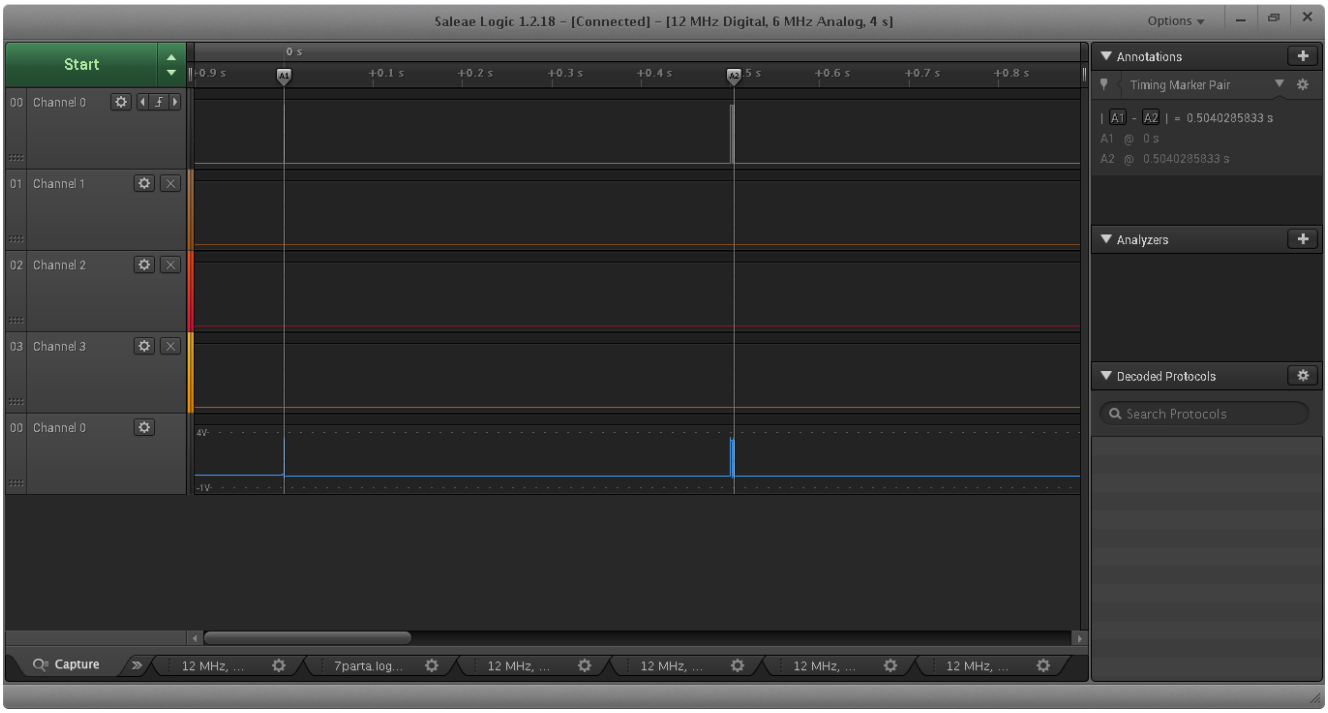
b. **Why didn't we change the first blocking loop in SPI_RW?**
The transmit buffer gets empty quickly due to the high SPI bit rate.Hence we didn't change the first blocking loop.

c. **Estimate how long it would take for SPI_RW() to execute if we raised the SPI bit rate to 1 MHz.**
24.25 µs.

20. **Thread Analysis**

    a. **How long does SD_Init take to execute? How long is each segment now? Include a screenshot.**

    4.32ms



    b. **How much idle time is available now, based on idle_counter?**

    0.3049s

    c. **Do these values differ from the previous implementation? Why or why not?**

    Yes the values have increased compared to the initial value since hardware exception handling has been added to the delay.

ECE 560-Only: Eliminating the use of the SPI Timer

21. **Find all of the loops with time-outs in the functions (SD_Read, SD_Write, SD_Init) and explain what each loop does and what the timeout period is.**

    SD_Init-

    The first loop sends command to the SD card and waits for the response to be true.The timeout is 500ms.

    The second loop sends ACMD41 command to put the SD card in the High speed mode.The timeout is 1s.

    The third loop sends CMD command to the SD card. The timeout is 250ms.

    SD_Read-

    The loop waits for the data packets to arrive. The timeout is 100ms.

    SD_Write

    The loop waits for the completion of data programming.The timeout is 250ms.