Project 2

ECE 561

Embedded System Design

OPTIMIZING I2C COMMUNICATION CODE
WITH AN RTOS (V1.1)

ADITYA SHIWAJI RASAM
200153631
(arasam@ncsu.edu)

Mode 1 - Blocking Code Implementation

1. Screenshot showing three debug bits over duration of a read message.

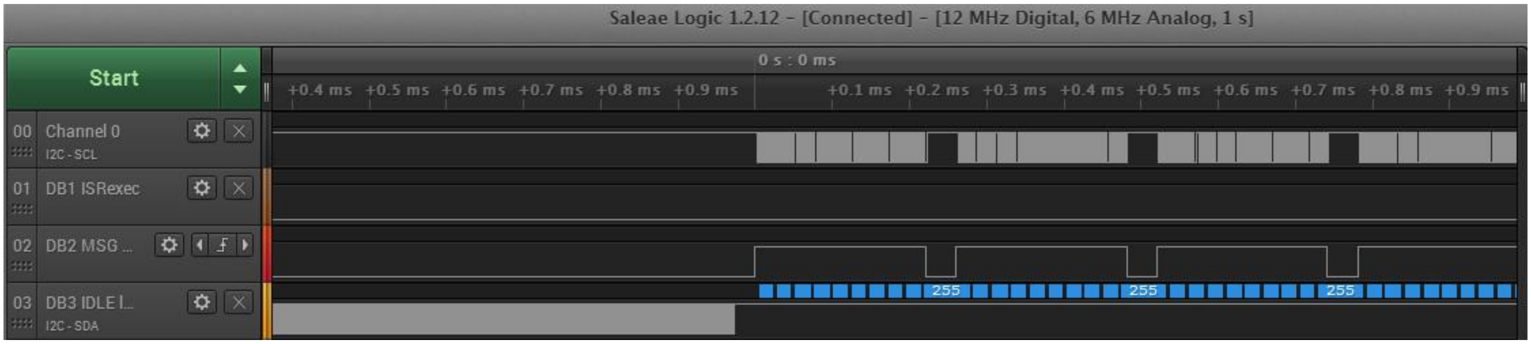


Figure 1: Three Bits ISR, I2C MSG, IDLE

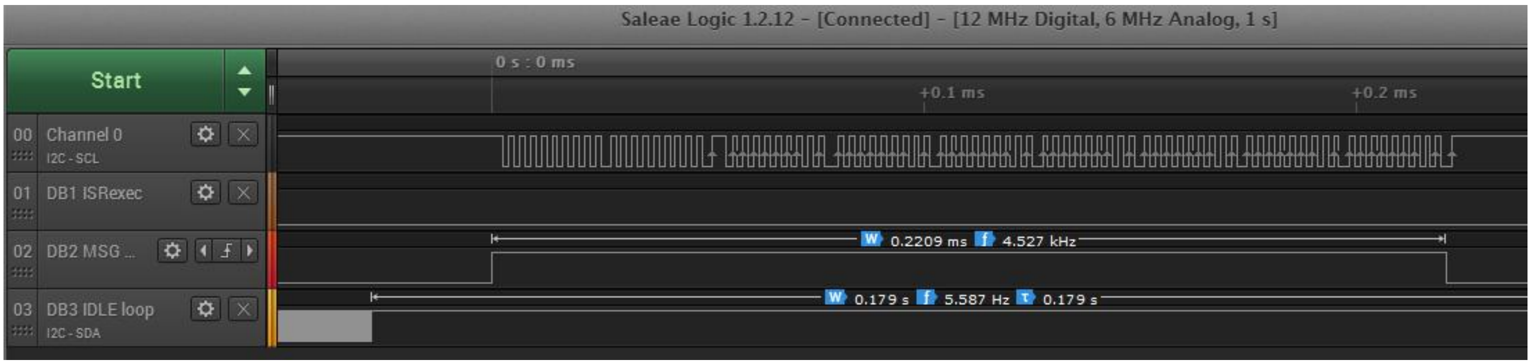
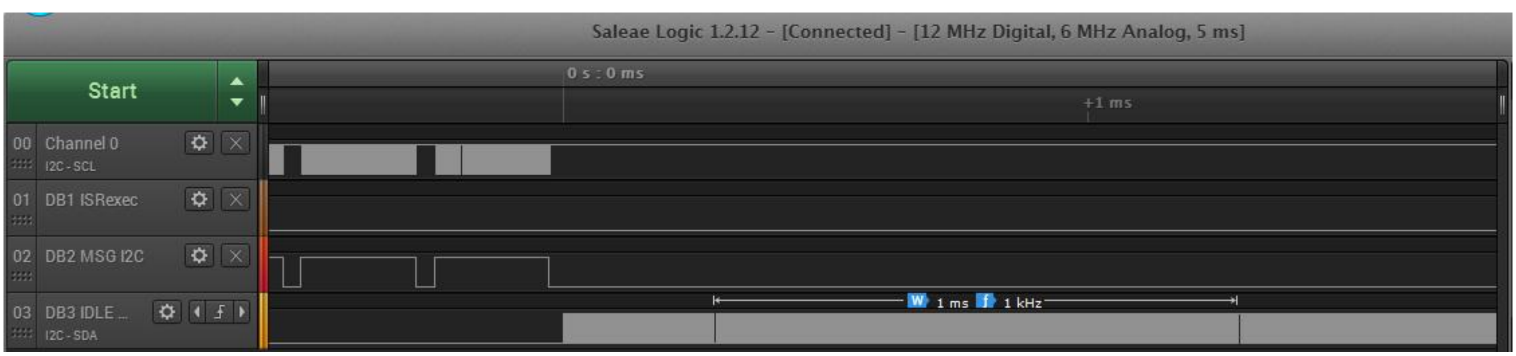


Figure 2:Zoomed in Three Bits ISR, I2C MSG, IDLE with I2C read time

2. How much time does it take to complete an I2C read transaction?
   a. From the figure its seen that the time taken to complete an I2C read transaction is ***220.9 usec***

3. How much idle time is available during a transaction?
   a. No idle time is available during transaction as program control is kept busy in busy wait function

4. How long does it take for the RTOS to execute its periodic timer tick?



It takes **1 msec** as seen from above signal diagram for RTOS to execute periodic tick.

Extra Credit: With time optimization level - 3

1. Screenshot showing three debug bits over duration of a read message.

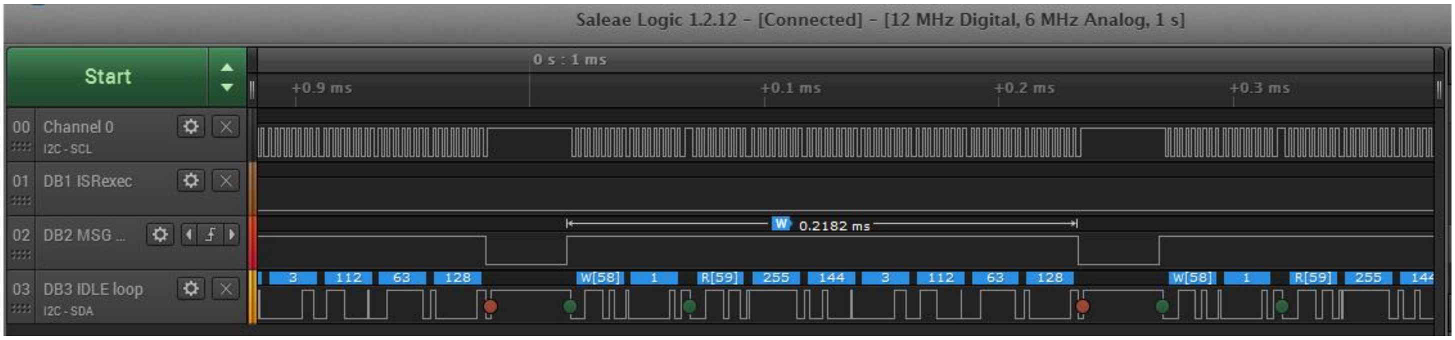


Figure 3:Zoomed in Three Bits ISR, I2C MSG, IDLE with I2C read time

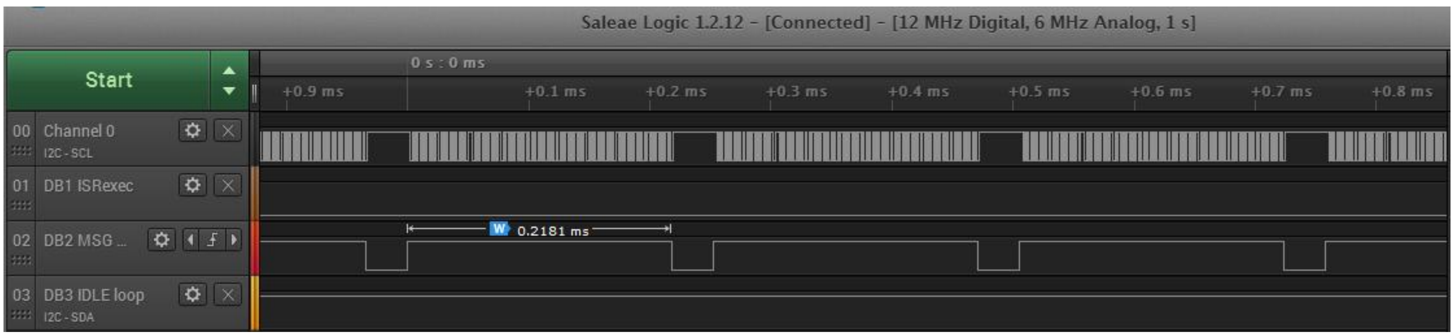


Figure 4: Three Bits ISR, I2C MSG, IDLE

2. How much time does it take to complete an I2C read transaction?
   a. From the figure its seen that the time taken to complete an I2C read transaction is ***218. Usec***
   b. ***Thus with optimization the time for transaction has reduced by 2 usec***

3. How much idle time is available during a transaction?
   c. No idle time is available during transaction as program control is kept busy in busy wait function

Mode 2 - Thread Signaled by ISR per Byte Transferred

1. Screenshot showing three debug bits over duration of a read message.

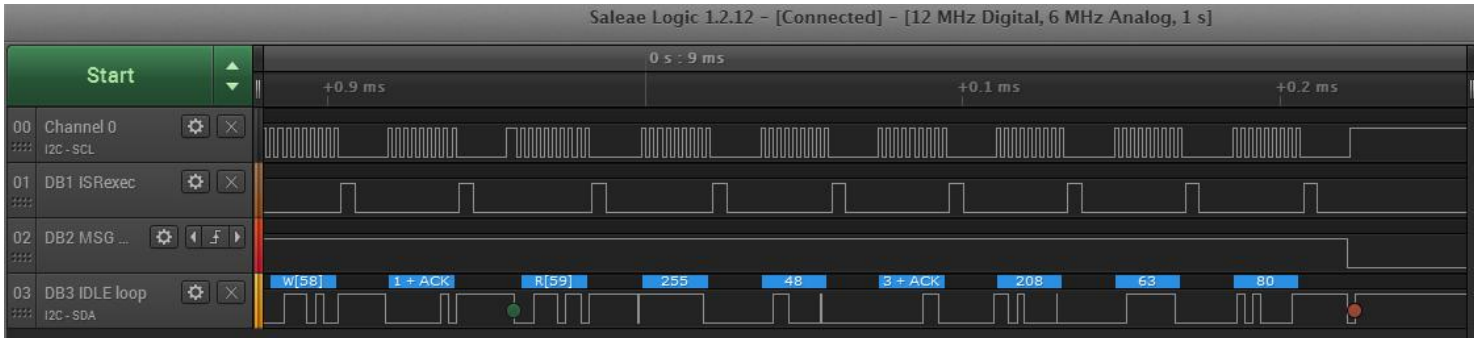


Figure 5: 3 bits ISR, I2C MSG with SCL & SDA I2C signals

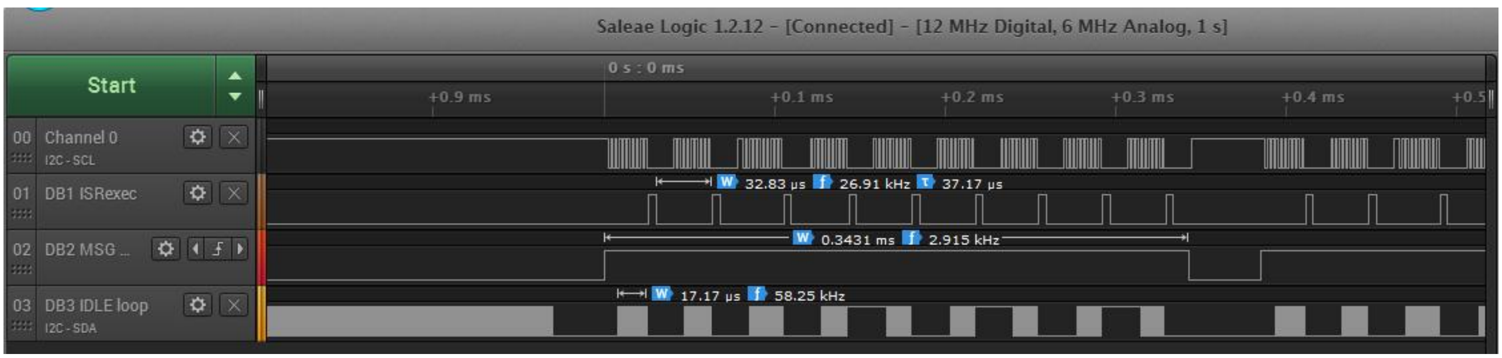


Figure 6:3 bits ISR, I2C MSG, Idle loop with I2C read transaction time

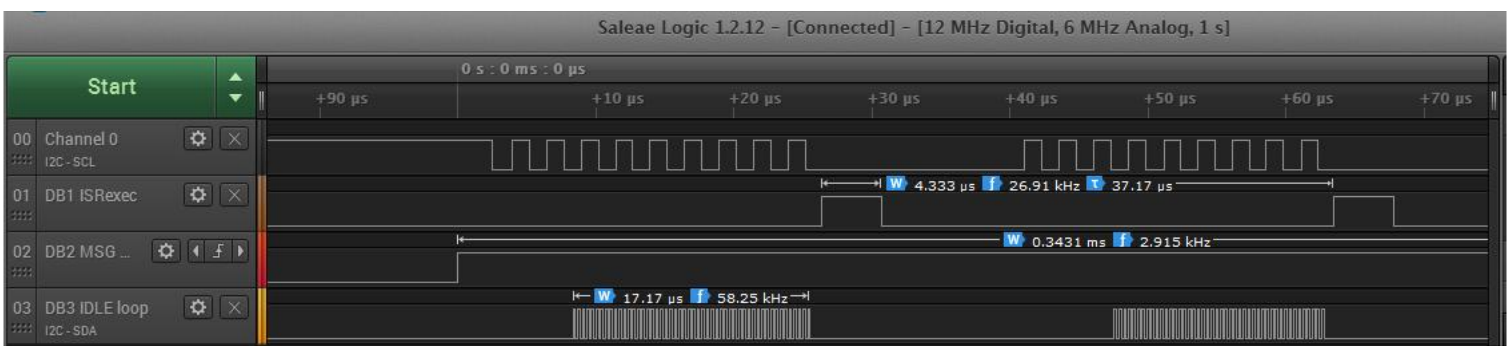


Figure 7: Zoomed in

2. How much time does it take to complete an I2C read transaction?
   a. From the figure it's seen that the time taken to complete an I2C read transaction is ***343 usec.***
   b. This is indicated by bit I2C MSG.

3. How much idle time is available during a transaction?-

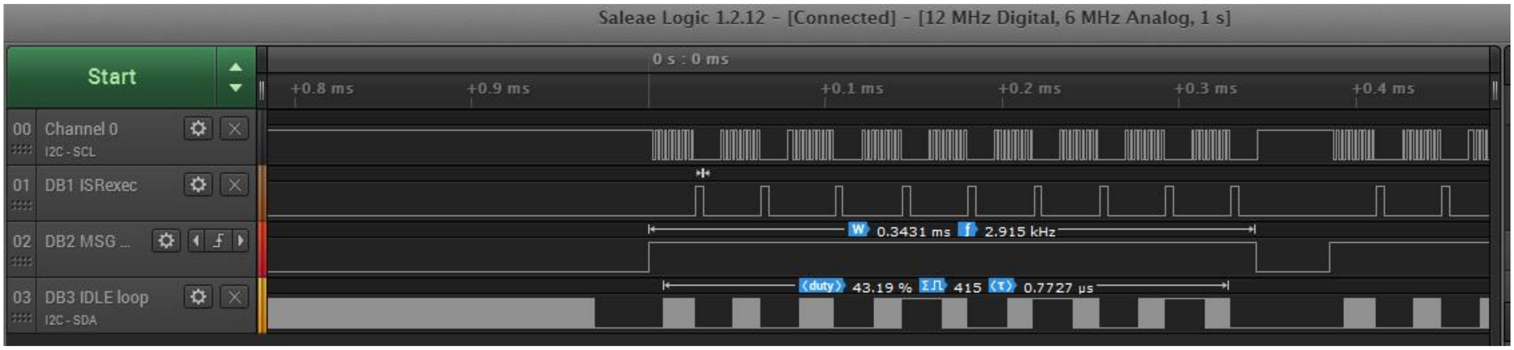


Figure 8: Idle Time measurement

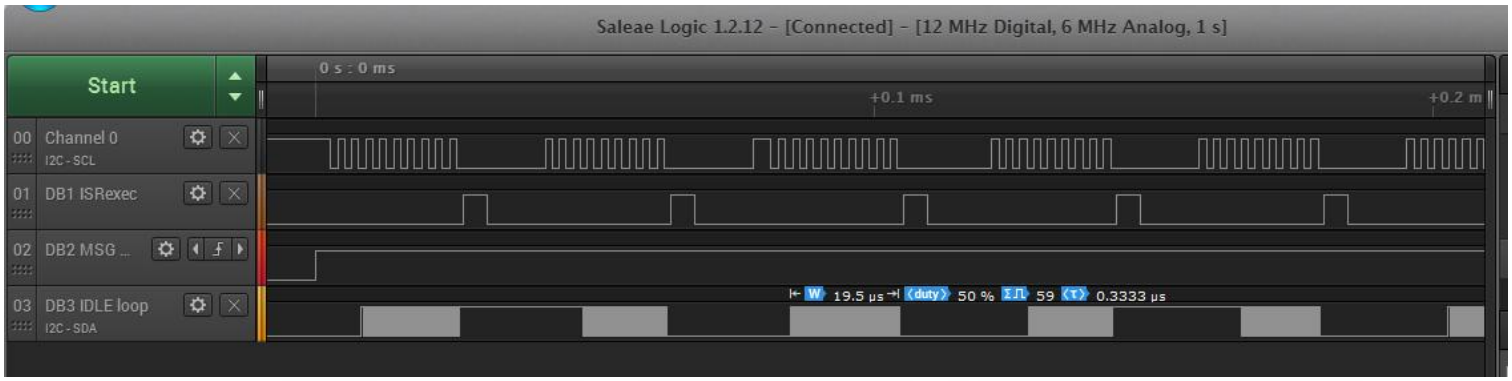


Figure 9: Zoomed in

a. **The Idle time can be calculated as follows**
   i. $Time = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$
   ii. The calculation is done in excel for each of the idle call that occurs within the entire I2C communication cycle for every communication command. Total time is sum of them.

Idle call for every byte transaction	Duty Cycle	#Pulses	Avg Period	Time (sec)	Units
1	0.5	52	3.33E-07	0.000008658	*sec*
2	0.5	46	3.33E-07	0.000007659	*sec*
3	0.51	59	3.33E-07	1.002E-05	*sec*
4	0.5	45	3.33E-07	7.4925E-06	*sec*
5	0.5	42	3.33E-07	0.000006993	*sec*
6	0.5	42	3.33E-07	0.000006993	*sec*
7	0.5	42	3.33E-07	0.000006993	*sec*
8	0.5	42	3.33E-07	0.000006993	*sec*
9	0.5	42	3.33E-07	0.000006993	*sec*
			Total Time	**6.87945E-05**	*sec*

Total idle time = 68.8usec

4. ISR Timing:
   a. How long does the ISR take to execute?
      i. $Time\ required = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$

Column1	Duty Cycle	#Pulses	Avg Period	Time (sec)	Time(usec)
T_ISR	0.1144	9	3.79E-05	3.89807E-05	38.980656

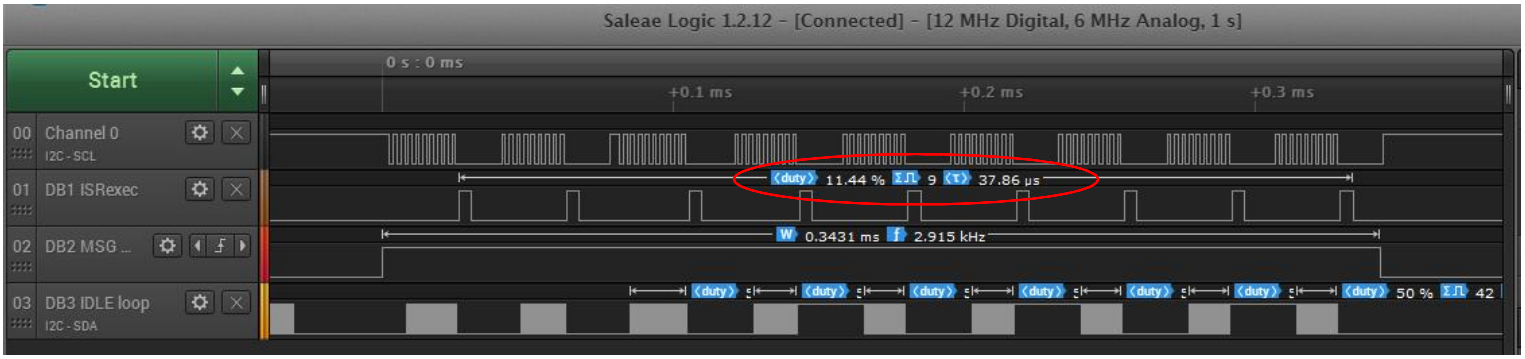


Figure 10: ISR Time Calculation

Also time required per call of ISR is **4.33 usec.**

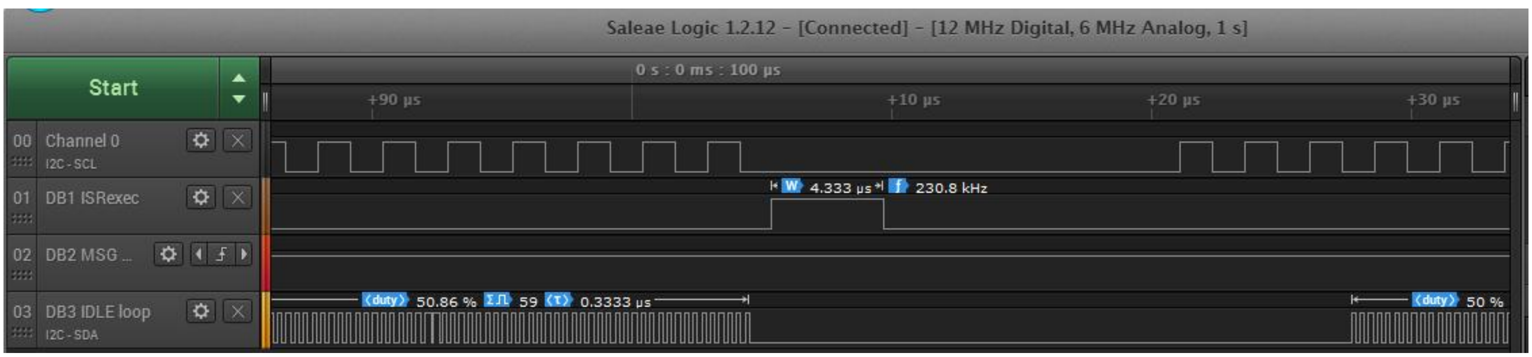


Figure 11: ISR Time per call

   b. How long is the delay between the idle loop running and the ISR running?
      i. As seen from the markers in the figure the delay is **0.833 usec**

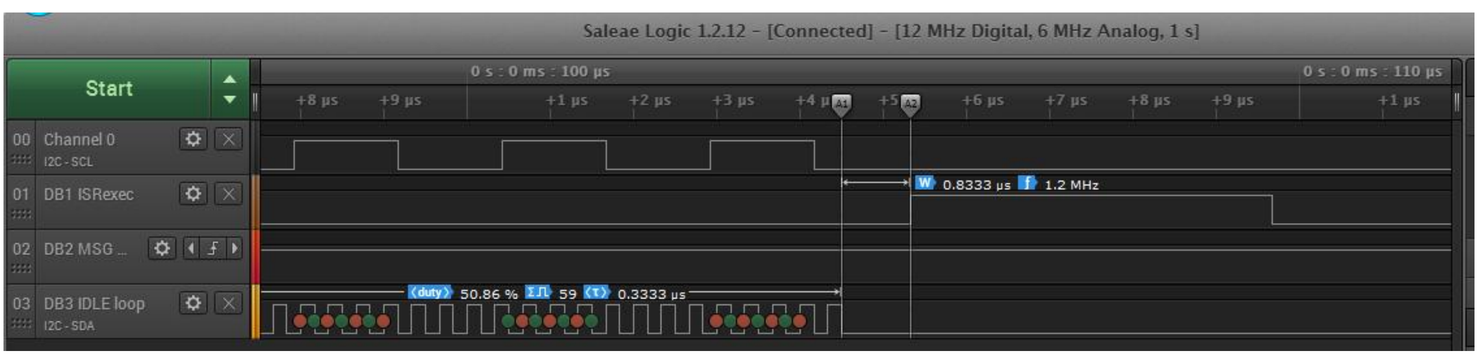


Figure 12 Time delay Idle loop running to ISR

   c. How long is the delay from the ISR completing to the idle loop resuming?
      i. As seen from the markers in the figure the delay is **18 usec**

Figure 13: Time delay from ISR run to IDLE

5. **OS Signal Timing:**
    a. How long does it take for the task code to resume executing after being signaled by the ISR? Explain your approach. Use one of the debug bits (or add a new one) to show this information.
        i. I have set the I2C MSG bit – 2 just below the osSignal Wait function in i2c read byte function after the first device address transaction.
        ii. Also the ISRexec bit resets immediately after setting the signal flag for Thread I2C
        iii. So the delay between the ISR exec bit and the I2C Msg bit is the required delay time.
            1. T_{delay} = **9.167 usec** as seen from diagram

Figure 14: Time delay between ISR signaling and task code execution

6. Idle Time Analysis
    a. Measure the idle debug signal to determine the percentage of idle time available during I2C message transmission.
        i. Avg period, Avg Duty Cycle & Positive – Total
        ii. Graph approach: From second & third section
            1. I2C msg transaction time = 0.343msec
            2. Idle exec time = 68.8usec
            3. Percentage of Idle time available = 0.0688/0.343 x 100 = 20%

    b. Use the idle_counter variable to measure this time and explain your approach. Does this time match your measurement of the idle debug signal? If not, why?
        i. Number of counter pulses = 671
        ii. Time = 671 x 0.221 = 148.291 usec

    c. What are the longest and shortest idle time segments?
        i. For ever idle call the average times calculated in table above
            1. Longest time segment is 10usec
            2. Shortest is 6.993usec

ii. Within every idle call as shown in graph below
    1. Shortest is 0.1667usec
    2. Longest is 0.25usec

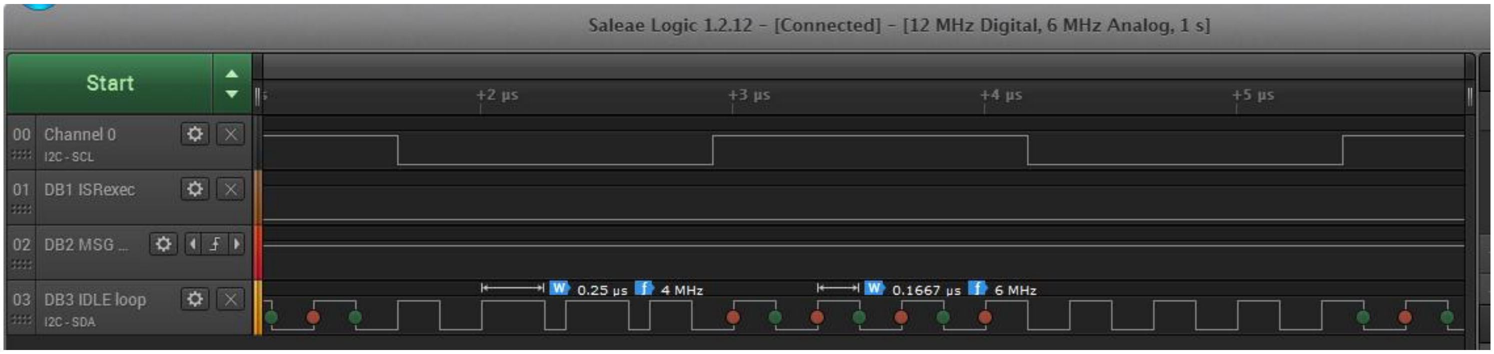


Figure 15: Longest & Shortest idle time segments within idle call during every byte transaction

7. How long does it take for the RTOS to execute its periodic timer tick?

It takes 1 msec as seen from above signal diagram for RTOS to execute periodic tick.

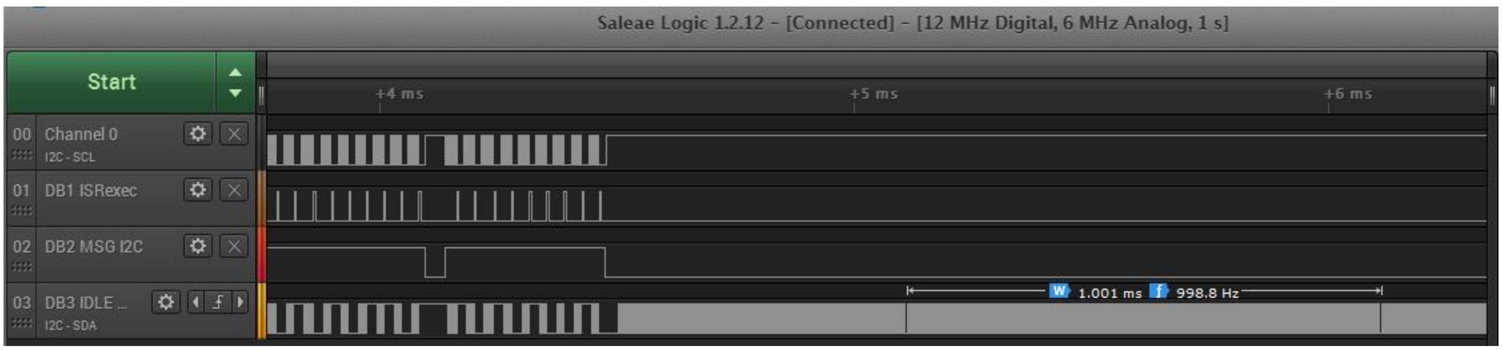


Figure 16: RTOS Timer Tick

Extra Credit: Mode-2 with time optimization level - 3

1. Screenshot showing three debug bits over duration of a read message.

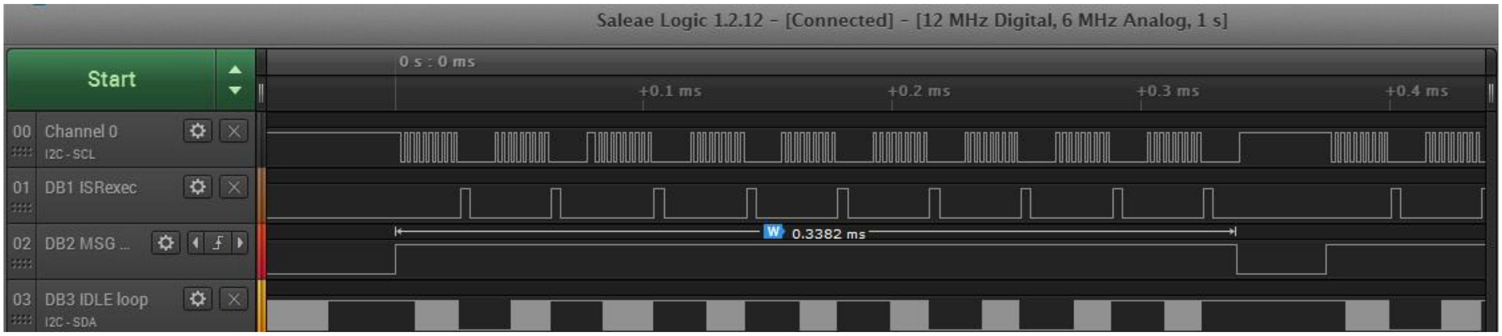


Figure 17: Three bits ISRexec, I2C MSG, IDLE

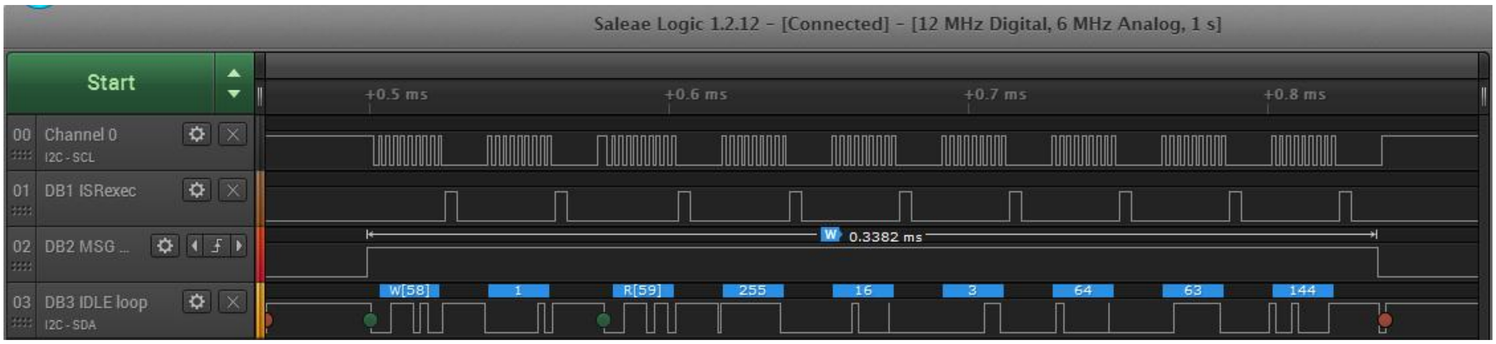


Figure 18: ISR exec, I2C MSG, SCL, SDA

2. How much time does it take to complete an I2C read transaction?
    a. From the figure it's seen that the time taken to complete an I2C read transaction is ***338 usec.***
    b. This is indicated by bit I2C MSG.
    c. Thus with Optimization the time has decreased by 5usec

3. How much idle time is available during a transaction?-
   a. **The Idle time can be calculated as follows**
      i. $Time = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$
      ii. The calculation is done in excel for each of the idle call that occurs within the entire I2C communication cycle for every communication command. Total time is sum of them.
      iii. The total idle time is **71.8 usec**.
      iv. Thus the time improvement is 3usec due to optimization for which system is in idle condition.

Transac	Duty Cycle	#Pulses	Avg Period	Time (sec)	Time(usec)
1	0.5	83	2.09E-07	8.6652E-06	*8.6652*
2	0.5	73	2.09E-07	7.6212E-06	*7.6212*
3	0.51	94	2.09E-07	1.00099E-05	*10.009872*
4	0.5	73	2.09E-07	7.6212E-06	*7.6212*
5	0.5	70	2.09E-07	0.000007308	*7.308*
6	0.5	70	2.09E-07	0.000007308	*7.308*
7	0.6	69	2.09E-07	8.64432E-06	*8.64432*
8	0.6	70	2.09E-07	8.7696E-06	*8.7696*
9	0.4	70	2.09E-07	5.8464E-06	*5.8464*
			Total time	**7.17938E-05**	***71.793792***

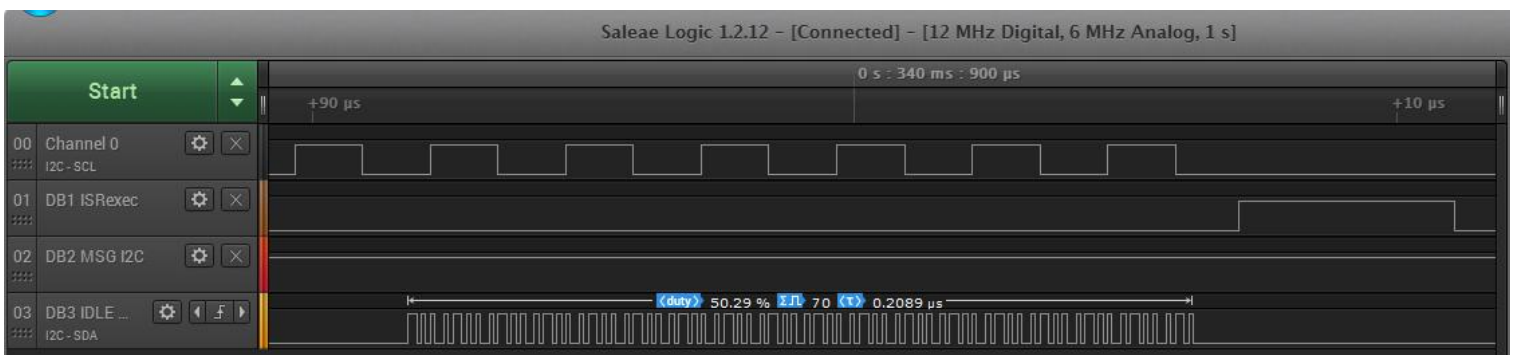
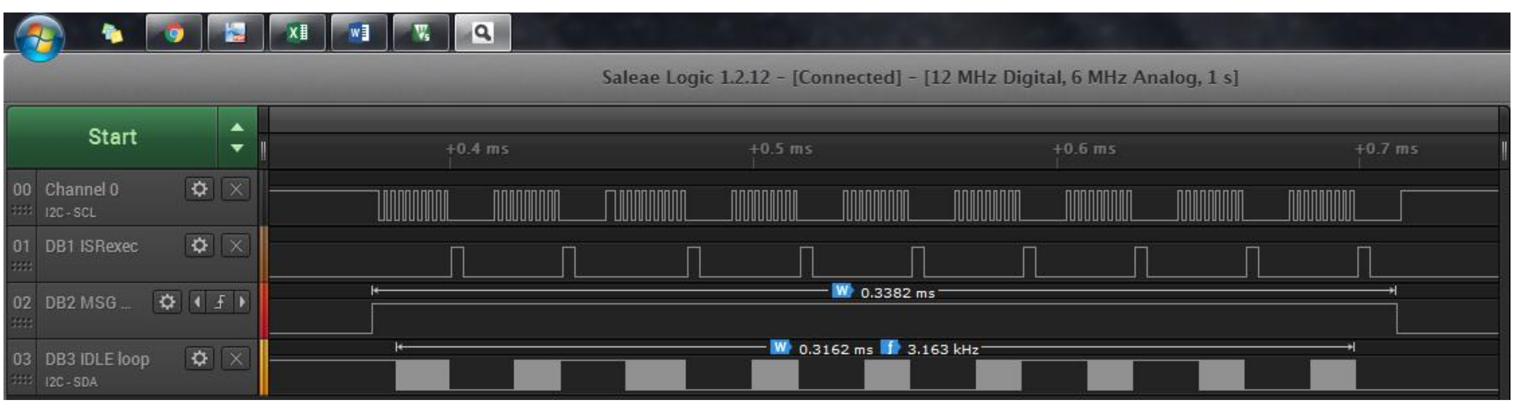


Figure 19: Idle sequence per I2C transaction



4. ISR Timing: How much idle time is available during a transaction?
   a. How long does the ISR take to execute?
      i. $Time\ required = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$

Column1	Avg Duty Cycle	#Pulses	Avg Period	Time(usec)
T_ISR	0.1037	9	3.73E-05	34.85

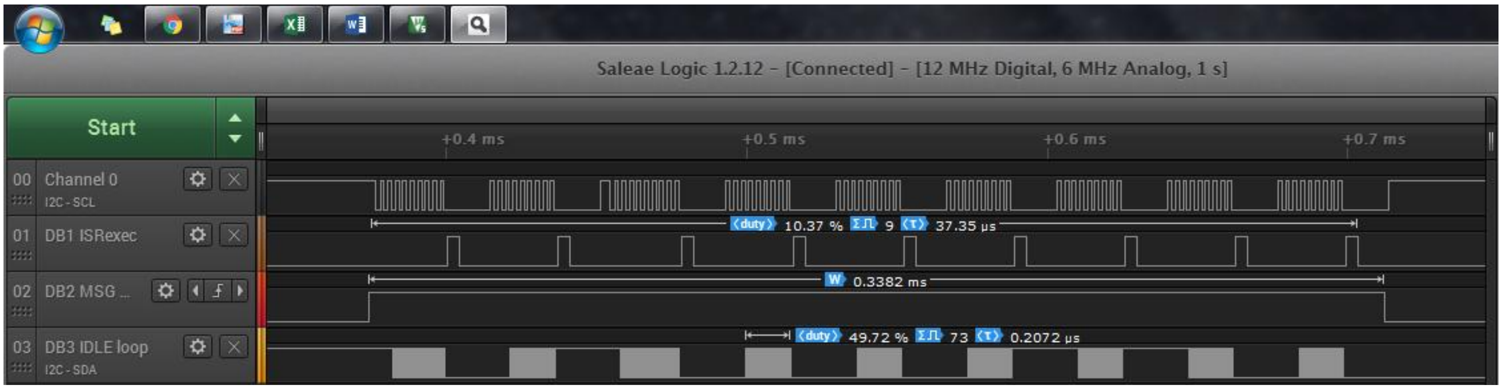


Figure 20: ISR Time calculation

Also time required per call of ISR is **3.833 usec.** Which shows an improvement of **0.5 usec** over last case where it was **4.33 usec.**

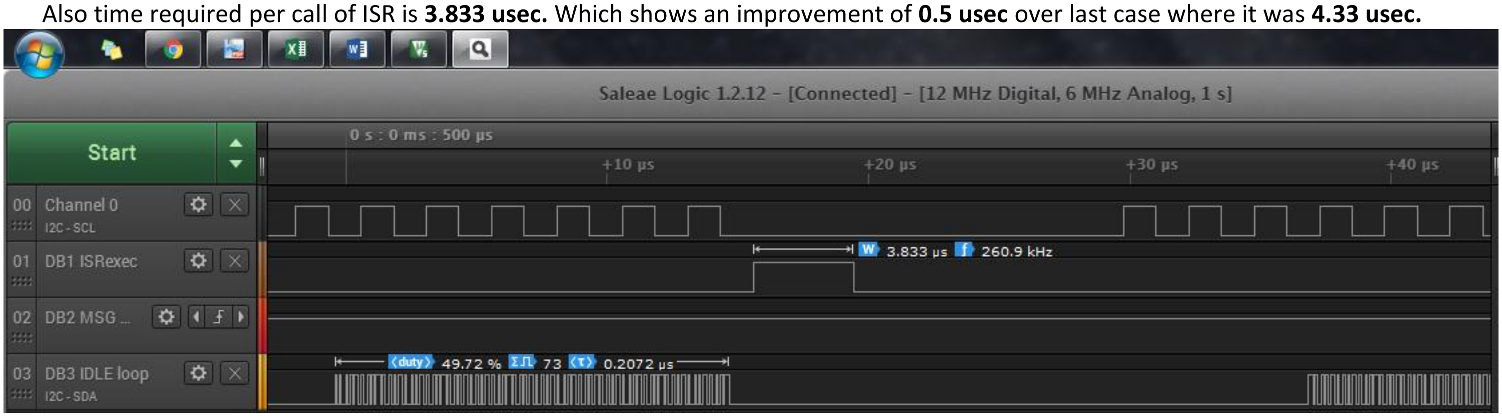


Figure 21: ISR per call

   b. How long is the delay between the idle loop running and the ISR running?
      i. As seen from the markers in the figure the delay is **0.916 usec**

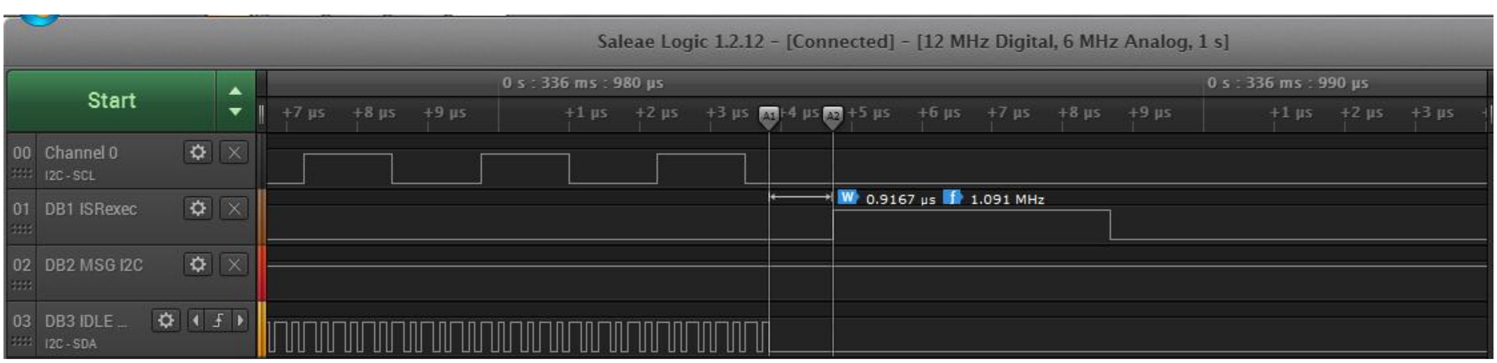


Figure 22: Time delay between IDLE to ISR

c. How long is the delay from the ISR completing to the idle loop resuming?

   i. As seen from the markers in the figure the delay is **16.6 usec**

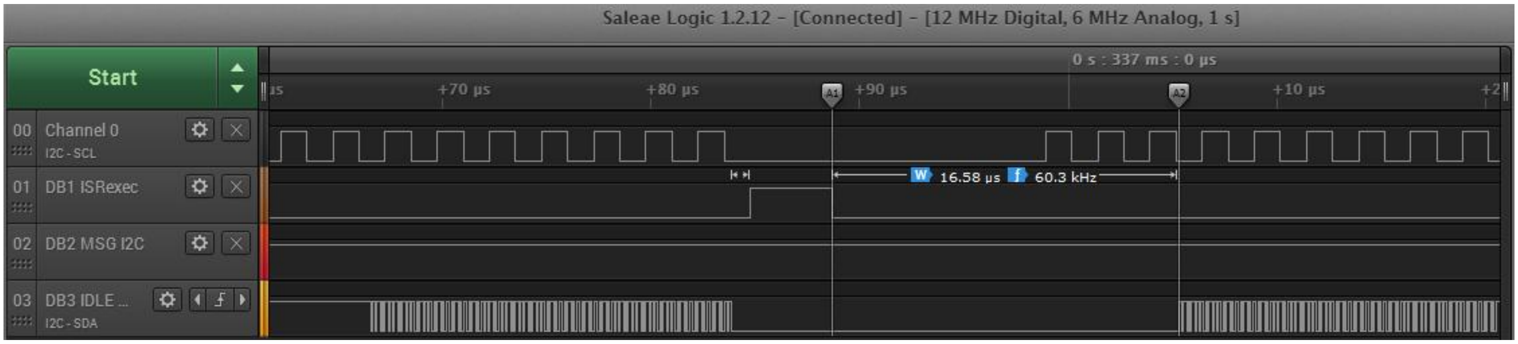


Figure 23: Time Delay ISR to IDLE loop

5. OS Signal Timing:

a. How long does it take for the task code to resume executing after being signaled by the ISR? Explain your approach. Use one of the debug bits (or add a new one) to show this information.

   i. I have set the I2C MSG bit – 2 just below the osSignal Wait function in i2c read byte function after the first device address transaction.

   ii. Also the ISRexec bit resets immediately after setting the signal flag for Thread I2C

   iii. So the delay between the ISR exec bit and the I2C Msg bit is the required delay time.

      1. T_{delay} = **9 usec** as seen from diagram

      2. Thus with optimization the time decreased from 9.167 usec to 9 usec.

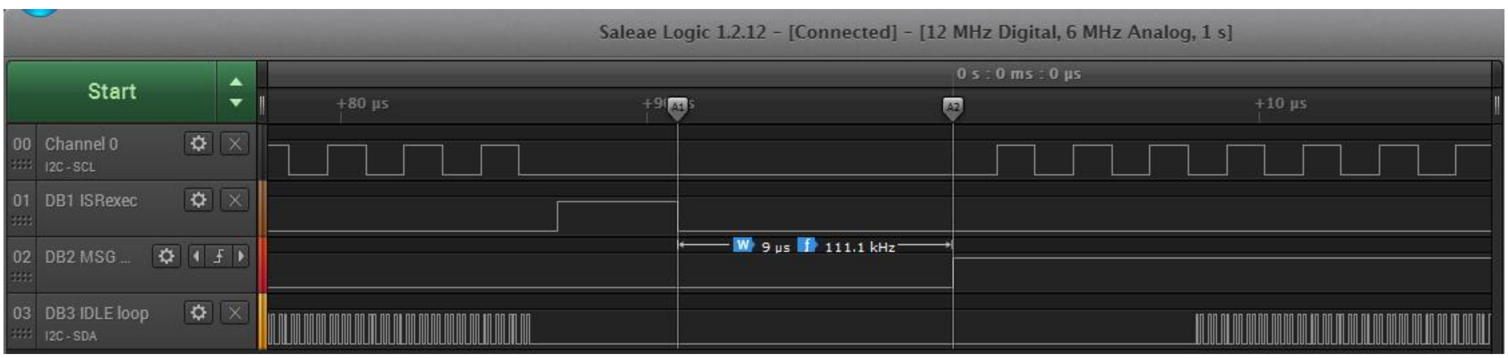


Figure 24: Time Delay between ISR signaling Task code resumption

6. Idle Time Analysis
    a. Measure the idle debug signal to determine the percentage of idle time available during I2C message transmission.
        v. Avg period, Avg Duty Cycle & Positive – Total
        vi. Graph approach: From second & third section
            1. I2C msg transaction time = 0.338msec
            2. Idle exec time = 71.8usec
            3. Percentage of Idle time available = 0.0718/0.338 x 100 = 21.24%

    b. Use the idle_counter variable to measure this time and explain your approach. Does this time match your measurement of the idle debug signal? If not, why?
        vii. Time = Number of counter pulses x 0.221

    c. What are the longest and shortest idle time segments?
        viii. For ever idle call the average times calculated in table above
            1. Longest time segment is 10usec(no change)
            2. Shortest is 5.8 usec (as compared to 6.993usec).
        ix. Within every idle call as shown in graph below
            1. Shortest is 0.0833usec (previous of 0.1667usec)
            2. Longest is 0.1667usec (previous of 0.25usec)

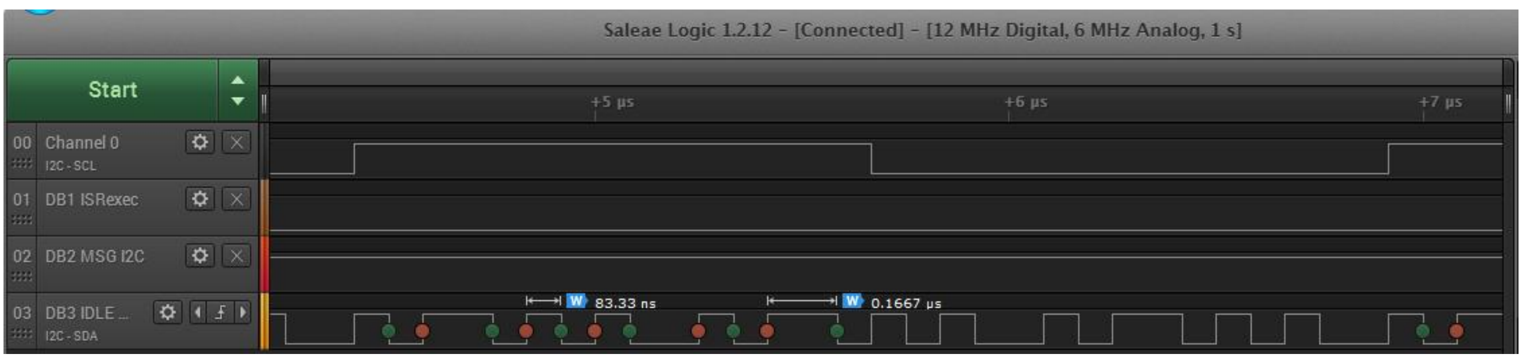


Figure 25: Longest & Shortest idle time per transaction

7. How long does it take for the RTOS to execute its periodic timer tick?

It takes 1 msec for RTOS to execute periodic tick which is same as indicated by figure 17 in non-optimized mode.

Mode 3 - Thread Signaled by ISR per I^{2}C Transaction

1. Screenshot showing three debug bits over duration of a read message.

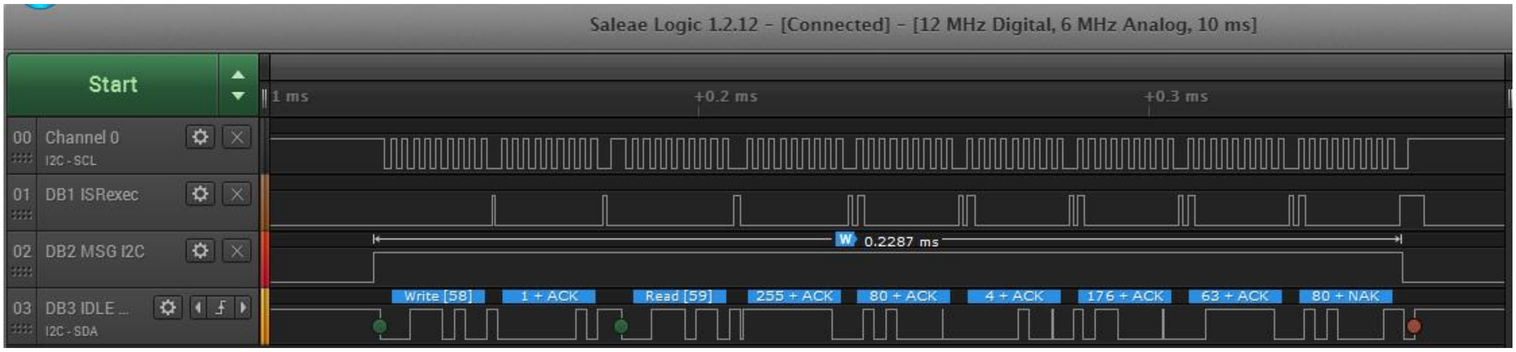


Figure 26: ISR exec, I2C MSG, SDA, SCL

For read data part I have created two FSM states. DataCheck that creates a dummy read and then Dataread that updates the data array. So two calls are made to ISR and hence two pulses for ISR exec.

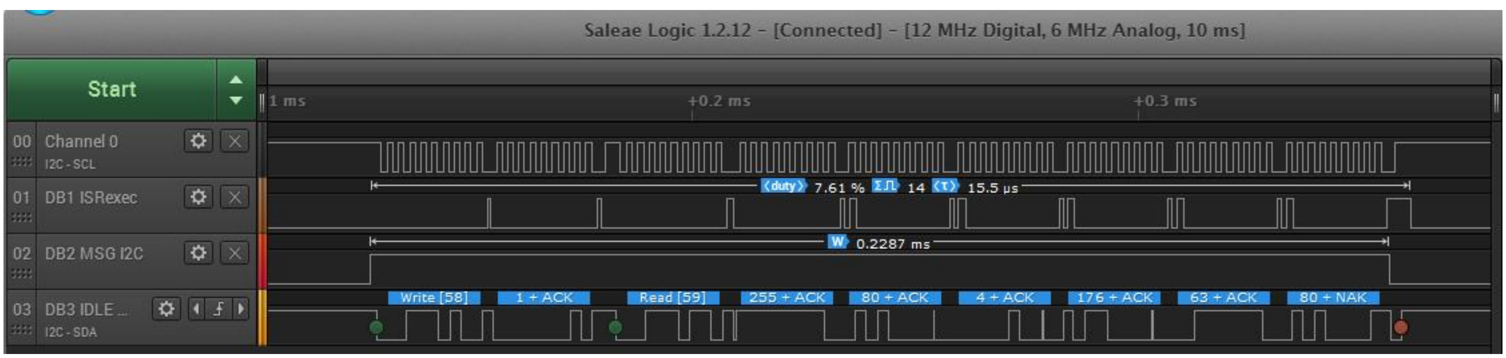


Figure 27: ISR EXEC, I2C MSG, IDLE LOOP

2. How much time does it take to complete an I2C read transaction?
   a. From the figure it's seen that the time taken to complete an I2C read transaction is ***229 usec.***

   b. This is indicated by bit I2C MSG.

3. How much idle time is available during a transaction?-
    a. **The Idle time can be calculated as follows**

Idle segment	Time(usec)
1	17.08
2	22.25
3	26.67
4	22.08
5	19.42
6	19.42
7	19.33
8	19.17
9	19.42
Total	184.84

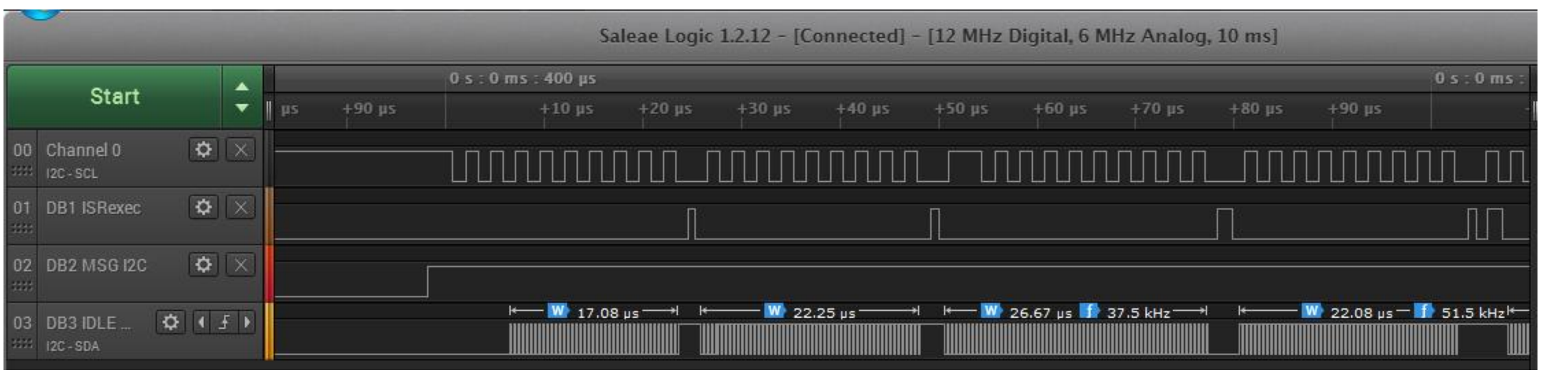
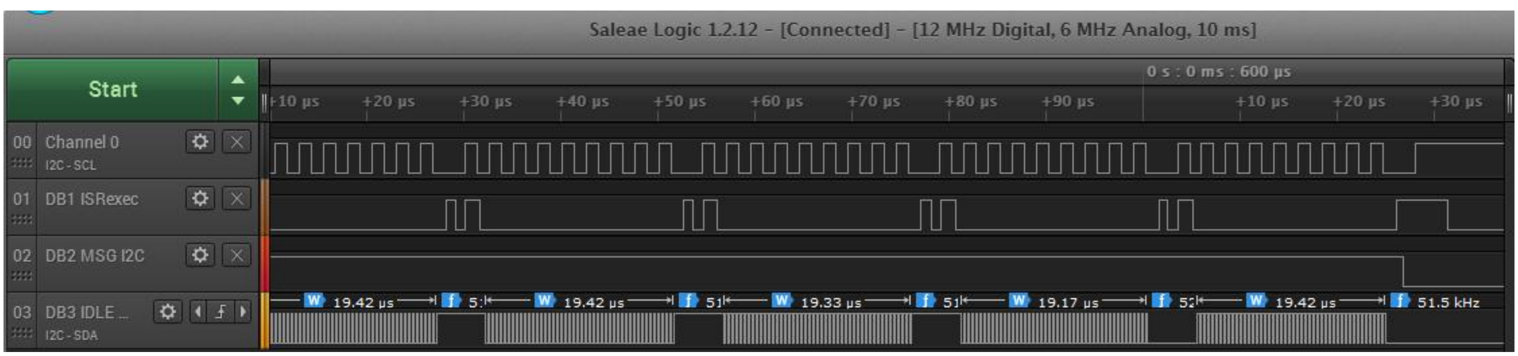


Figure 28: Idle call per byte



4. ISR Timing:
    a. How long does the ISR take to execute?
        a. $Time = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$
        b. Time = 0.075 x 14 x 15.5usec = 16.275 usec

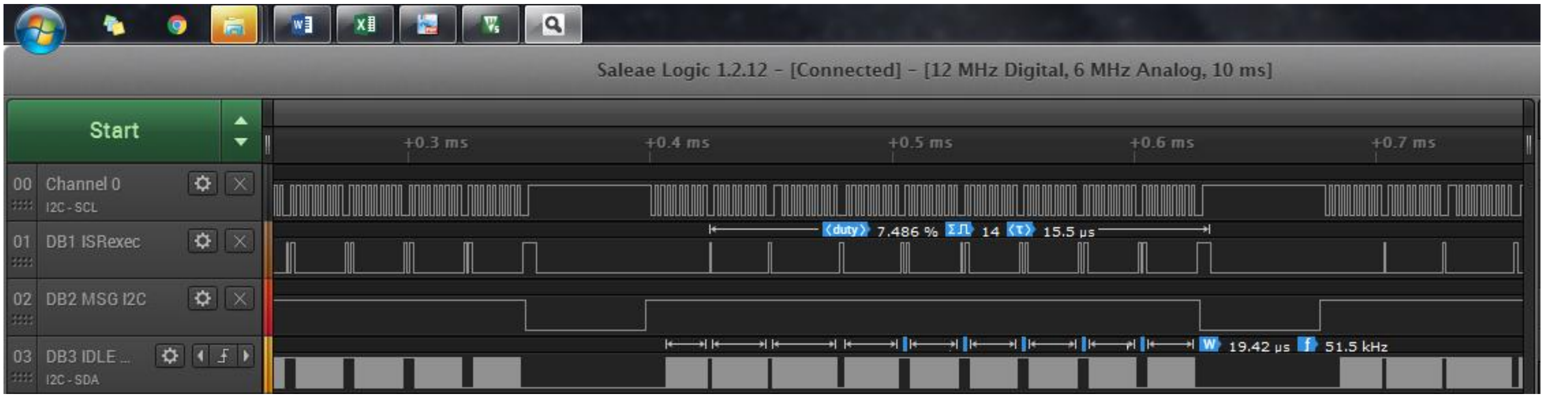


Figure 29: ISR Execution time

b. How long is the delay between the idle loop running and the ISR running?
    ii. As seen from the markers in the figure the delay is **1.083 usec**

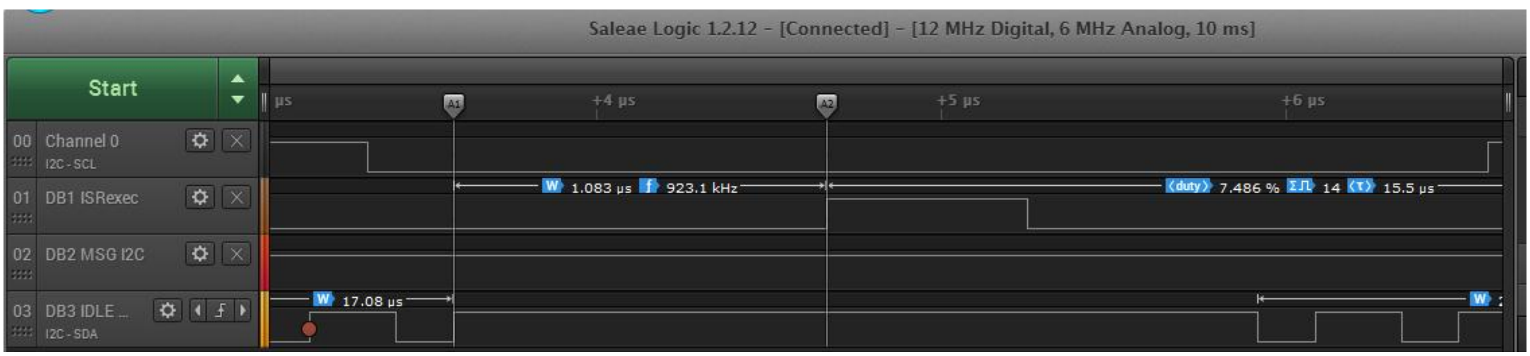


Figure 30 Idle loop to ISR time delay

c. How long is the delay from the ISR completing to the idle loop resuming?
    i. As seen from the markers in the figure the delay is **0.667 usec.**

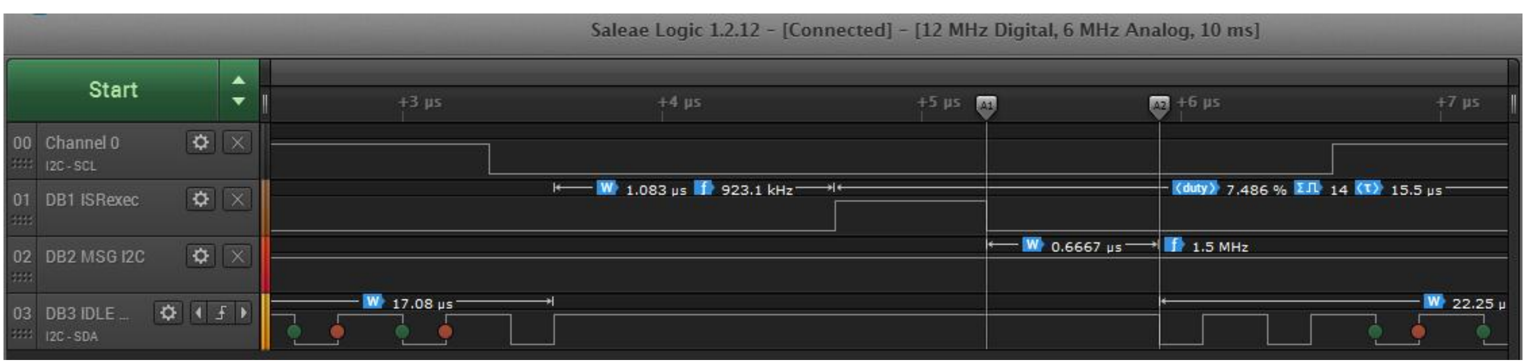


Figure 31: ISR to IDLE Loop time delay

5. OS Signal Timing:
    a. How long does it take for the task code to resume executing after being signaled by the ISR? Explain your approach. Use one of the debug bits (or add a new one) to show this information.
        i. I have set the I2C MSG bit – 2 just below the osSignal Wait function in i2c read byte function.
        ii. Also the ISRexec bit resets immediately after setting the signal flag for Thread I2C
        iii. So the delay between the ISR exec bit and the I2C Msg bit is the required delay time.
            1. T_{delay} = **9 usec** as seen from diagram

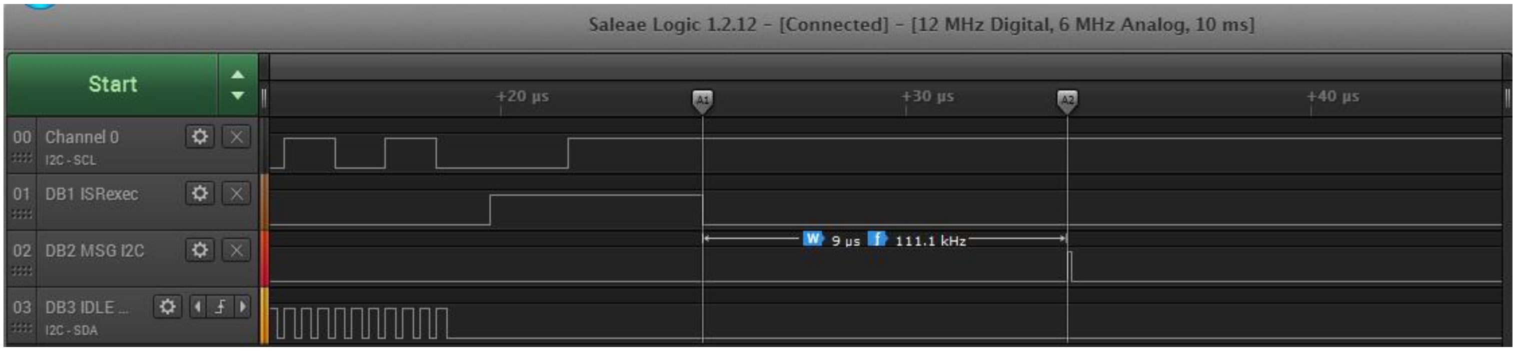


Figure 32:Time delay between ISR signaling & Task code execution

6. Idle Time Analysis
    a. Measure the idle debug signal to determine the percentage of idle time available during I2C message transmission.
        i. Graph approach: From second & third section
            3. I2C msg transaction time = 229 usec
            4. Idle exec time = 184 usec
            5. Percentage of Idle time available = 184/229 x 100 = 80%

    b. Use the idle_counter variable to measure this time and explain your approach. Does this time match your measurement of the idle debug signal? If not, why?
        i. Number of pulses = 900
        ii. Time = 900 x 0.221
        iii. 198.9 usec

    c. What are the longest and shortest idle time segments?
        i. For every idle call per byte the average times calculated in table above
            6. Longest time segment is 26.67 usec.
            7. Shortest time segment is 17 usec.

Figure 33Total Idle call per I2C transaction

Figure 34 Segments for Idle call per byte transaction

7. How long does it take for the RTOS to execute its periodic timer tick?
    a. The periodic Timer takes 1 msec which is same as Mode 1 & 2.

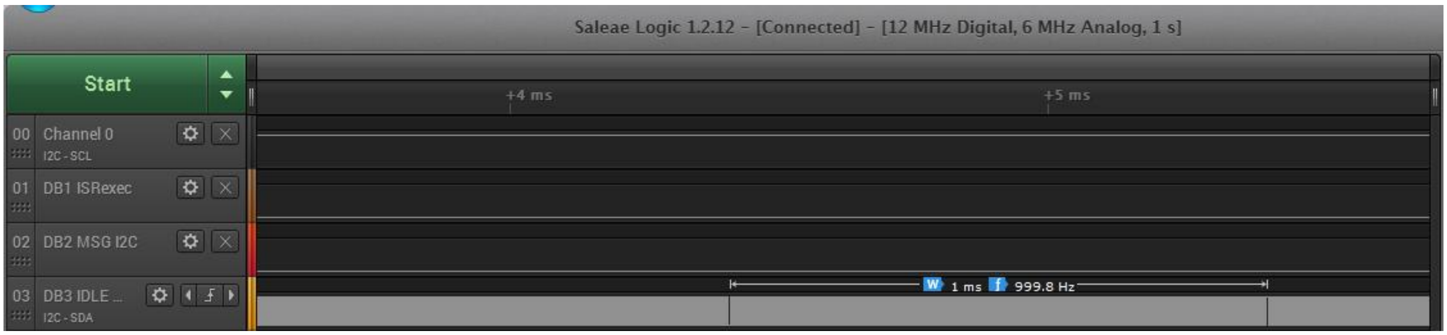


Figure 35 RTOS Timer Tick

Extra credit: Mode 3 with Time optimization Level-3.

1. Screenshot showing three debug bits over duration of a read message.

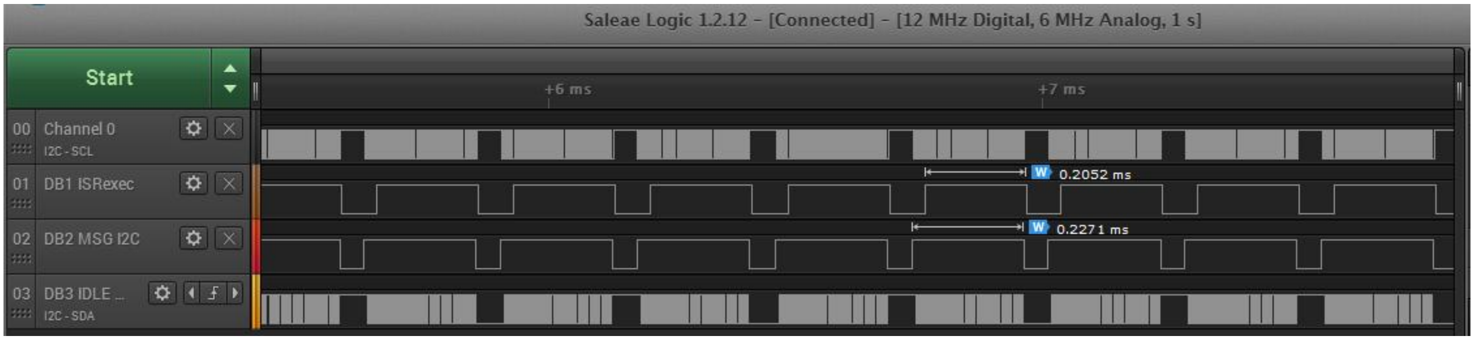


Figure 36: ISR exec. I2C MSG, IDLE LOOP

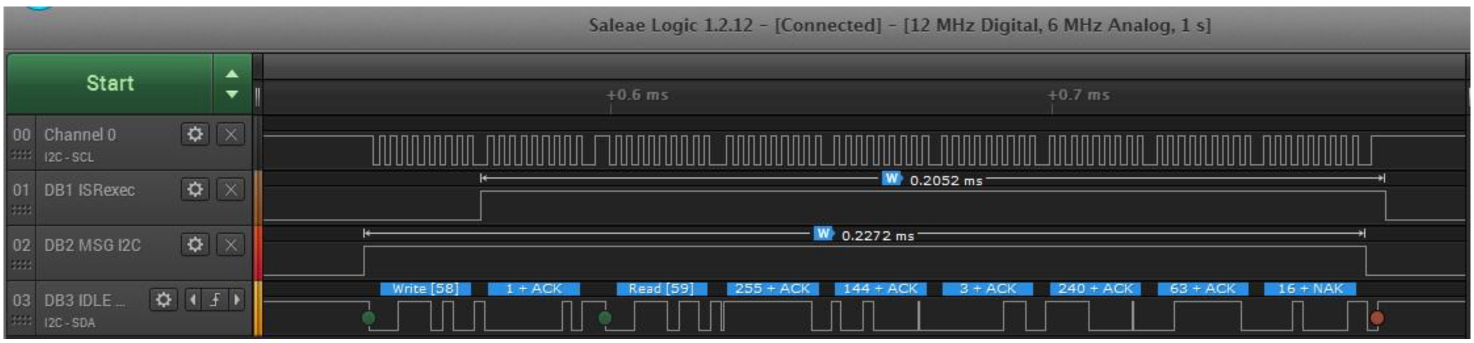


Figure 37 SDA & SCL

2. How much time does it take to complete an I2C read transaction?
   a. From the figure it's seen that the time taken to complete an I2C read transaction is ***227 usec*** *(previous of 231 usec)****.***
   b. This is indicated by bit I2C MSG.
   c. Thus with Optimization the time has decreased by 4 usec

3. How much idle time is available during a transaction?
   a. The Idle time can be calculated as follows
      i. $Time = Avg\ duty\ Cycle * Avg\ Time\ period * Number\ of\ positive\ pulses$
      ii. The calculation is done in excel for each of the idle call that occurs within the entire I2C communication cycle for every communication command. Total time is sum of them.
      iii. The total idle time is 99.34 usec (improving from **90 usec without** optimization**)**.

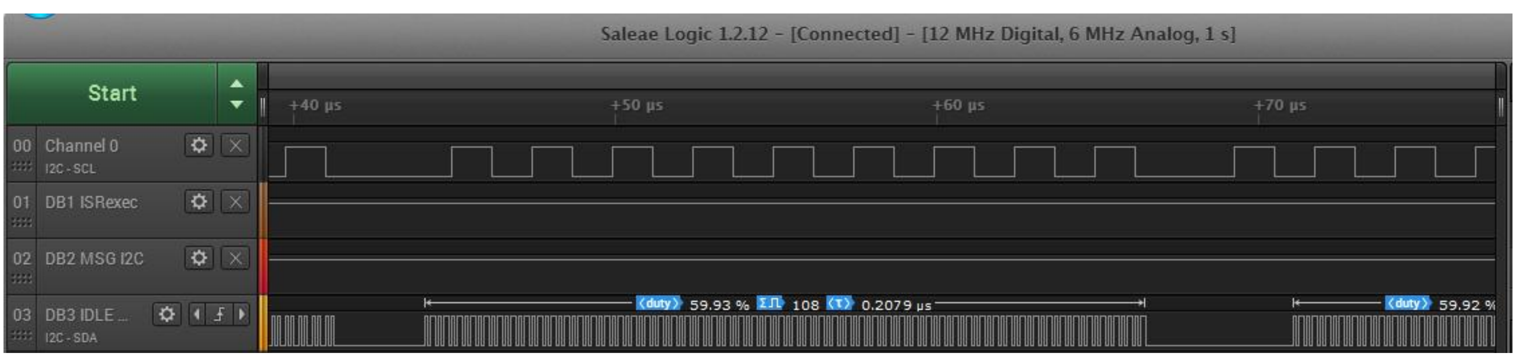


Figure 38: Idle call per byte transaction

Idle call per byte of transaction	Duty Cycle	#Pulses	Avg Period	Time (sec)	Time(usec)
1	0.6	83	2.08E-07	1.03733E-05	*10.37334*
2	0.6	108	2.08E-07	1.34978E-05	*13.49784*
3	0.4	129	2.08E-07	1.07483E-05	*10.74828*
4	0.6	108	2.08E-07	1.34978E-05	*13.49784*
5	0.6	95	2.08E-07	1.18731E-05	*11.8731*
6	0.6	94	2.08E-07	1.17481E-05	*11.74812*
7	0.4	95	2.08E-07	7.9154E-06	*7.9154*
8	0.4	95	2.08E-07	7.9154E-06	*7.9154*
9	0.6	95	2.08E-07	1.18731E-05	*11.8731*
				9.94424E-05	***99.44242***

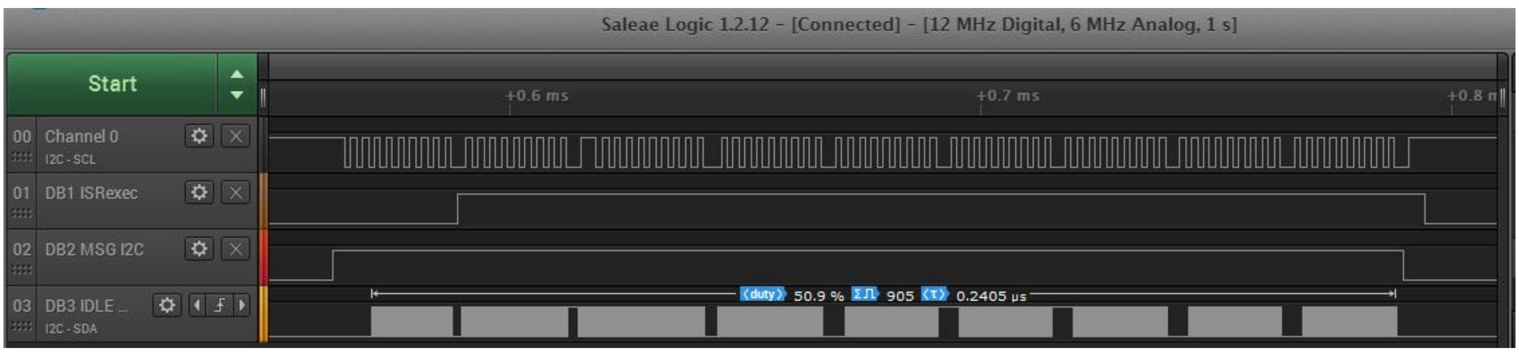


Figure 39 Idle call for entire I2C communication

4. ISR Timing:
    a. How long does the ISR take to execute?
        a. (From figure 35) Time taken to execute an ISR is 0.205 msec i.e **205usec**.
        b. Without optimization it was **210usec**.
        c. Thus an improvement of **5** usec.
    b. How long is the delay between the idle loop running and the ISR running?
        a. As seen from the markers in the figure the delay is **1.167 usec**
        b. Without optimization it was **1.33 usec**

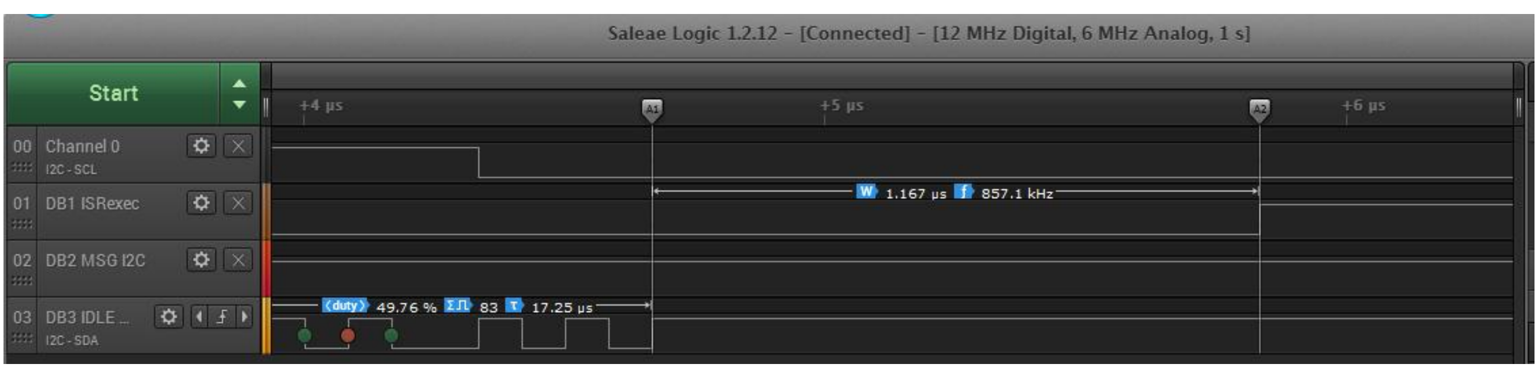
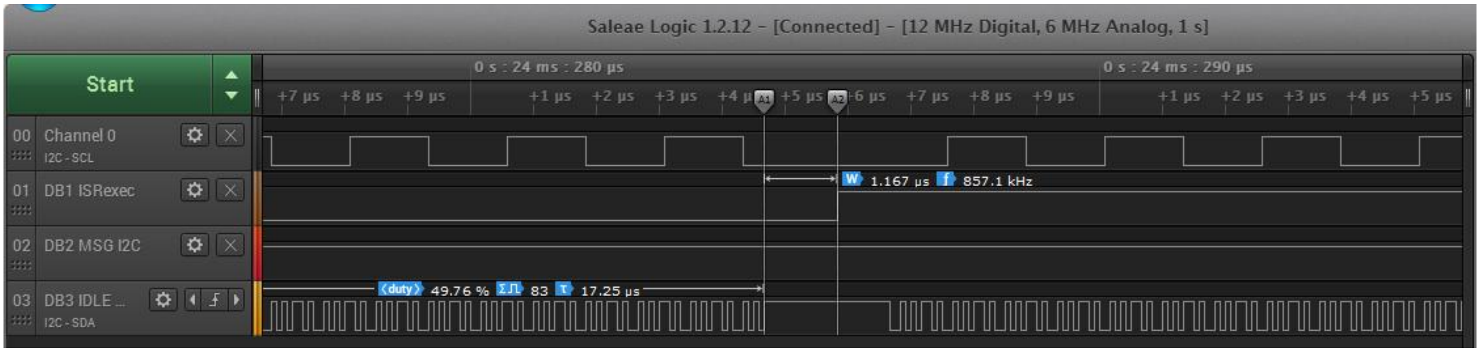


Figure 40 Idle to ISR time delay



c. How long is the delay from the ISR completing to the idle loop resuming?
   i. As seen from the markers in the figure the delay is **52.42 usec**
   ii. Prior to optimization it was **59.17 usec.**

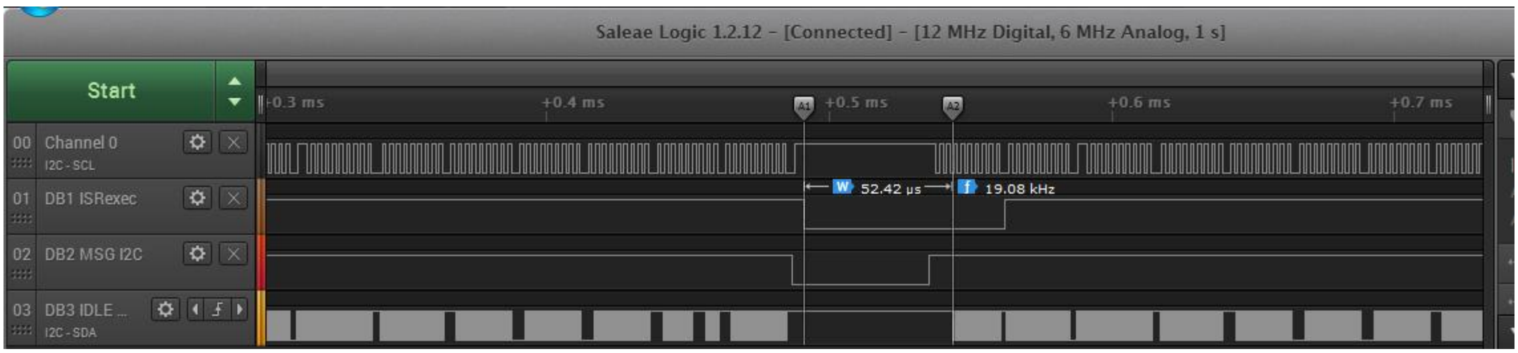


Figure 41 ISR to Idle delay

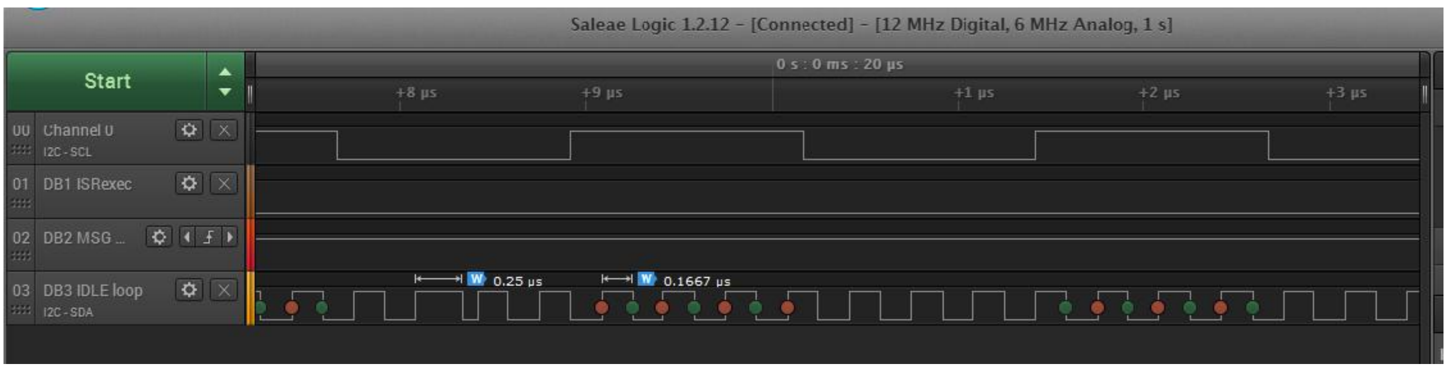
5. OS Signal Timing:
   a. How long does it take for the task code to resume executing after being signaled by the ISR? Explain your approach. Use one of the debug bits (or add a new one) to show this information.
      i. I have set the I2C MSG bit – 2 just below the osSignal Wait function in i2c read byte function.
      ii. Also the ISRexec bit resets immediately after setting the signal flag for Thread I2C
      iii. So the delay between the ISR exec bit and the I2C Msg bit is the required delay time.
         1. T_{delay} = **9 usec** as seen from diagram
         2. Prior to optimization it was 9.08usec

Figure 42: Time delay between ISR signaling & Task code execution

6. Idle Time Analysis
    b. Measure the idle debug signal to determine the percentage of idle time available during I2C message transmission.
        ii. Avg period, Avg Duty Cycle & Positive – Total
        iii. Graph approach: From second & third section
            8. I2C msg transaction time = 227 usec
            9. Idle exec time = 99.34 usec
            10. Percentage of Idle time available = 99.34/227 x 100 = 43.8%

    c. Use the idle_counter variable to measure this time and explain your approach. Does this time match your measurement of the idle debug signal? If not, why?
        i. **Time = Number of counter pulses x 3cycles x Tclock**

    d. What are the longest and shortest idle time segments?
        ii. For every idle call per byte the average times calculated in table above
            11. Longest time segment is 13.5 usec.
            12. Shortest is 7.9 usec.
        iii. Within every idle call as shown in graph below
            13. Shortest is 0.083u usec (prior to optimization it was 0.1667usec).
            14. Longest is 0.1667 usec ((prior to optimization it was 0.25usec).



8. How long does it take for the RTOS to execute its periodic timer tick?
    a. The periodic Timer takes 1 msec which is same as for non-optimized mode as shown in figure 35.