Project -1

ECE561

Embedded Systems Design

OPTIMIZING I2C COMMUNICATION CODE (V1.0)

Aditya Rasam
Unity ID: arasam
02/03/2017

Blocking Code Implementation

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

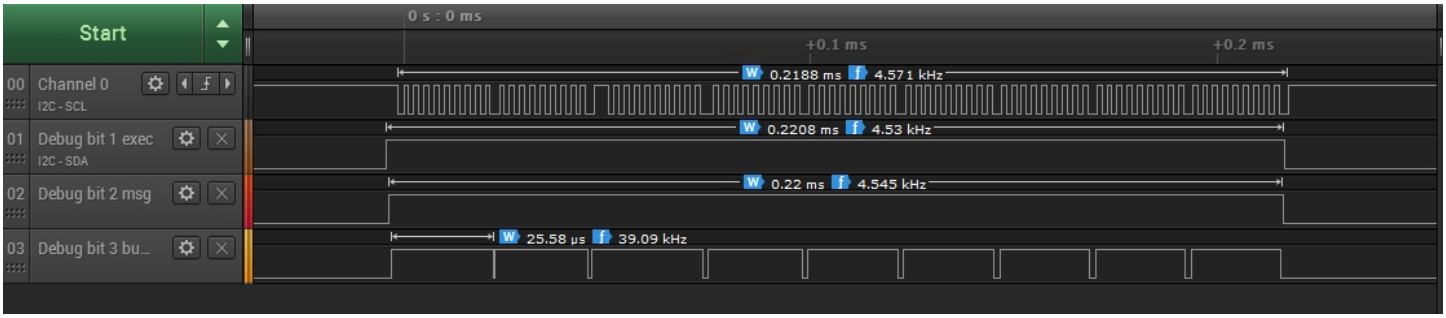


Figure 1: Clock, Bit 1-exec, Bit 2-message, Bit 3-busy wait

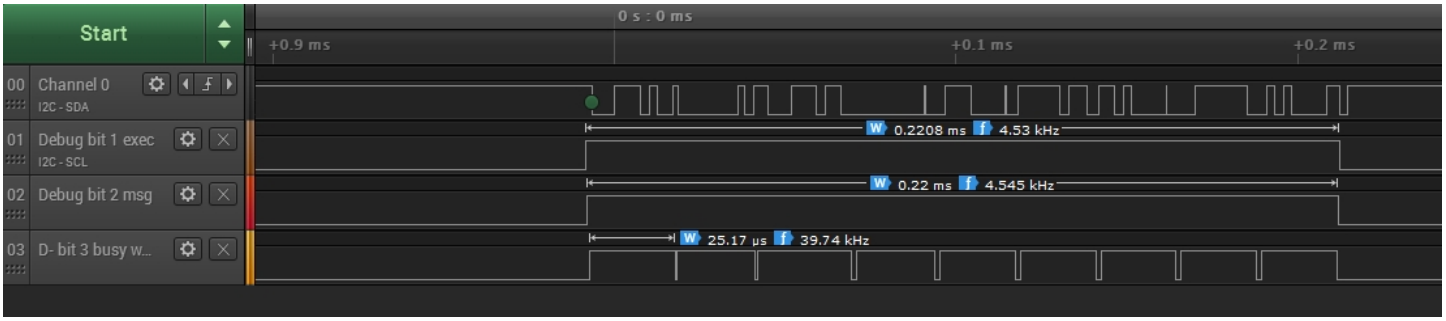


Figure 2: SDA, Bit 1-exec, Bit 2-message, Bit 3-busy wait

2. How much processor time is used to execute the i2c_read_bytes_fsm function to read the accelerations? What is the duration of the message on the bus?

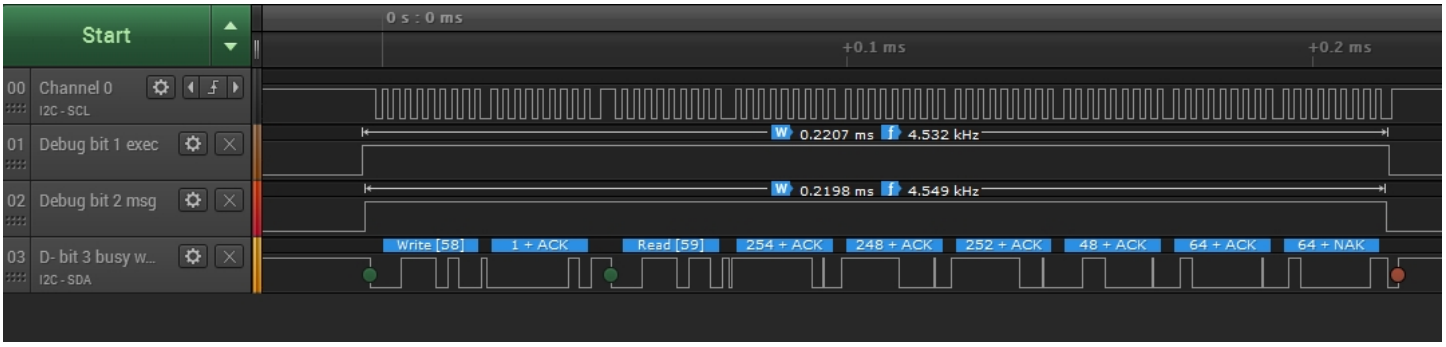


Figure3: FSM – Processor time – Busy Waiting

For Busy wait state

From the signal diagram it is seen that

1. The processor time to execute the read byte function is 0.2208msec.
2. The duration of message on bus is 0.2198msec.

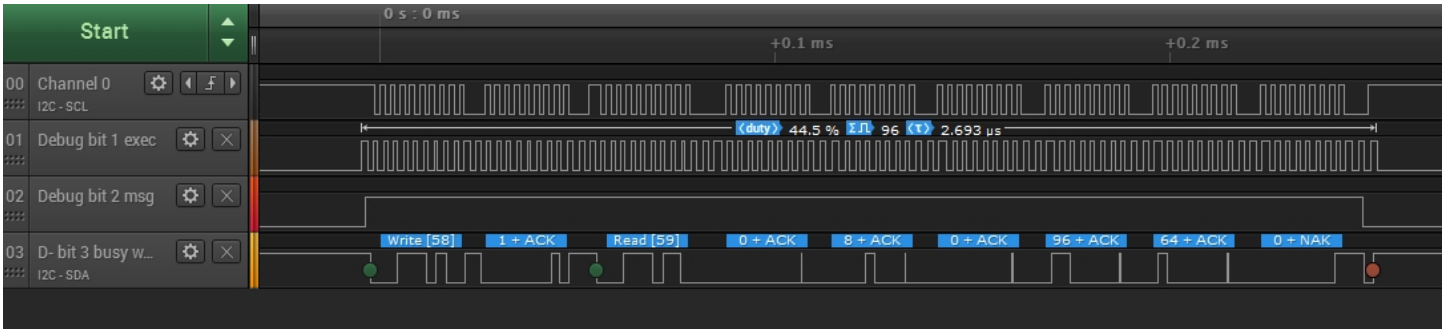


Figure4: FSM – Processor time – FSM

For FSM without delay

From the signal diagram it is seen that

Processor time = Average Duty Cycle x Average Time period x Number of pulses.

=0.45 x 2.7u x 96

= **116.64 usec**.

Thus FSM improved the response by reducing the processor time by 100usec

Finite State Machine Implementation

3. Explanation of how you pass input data to the FSM, and handshaking used (if any)

   In order to read data

   1. First device address and register address are written to slave
   2. A repeated start is made and data is read from the slave.

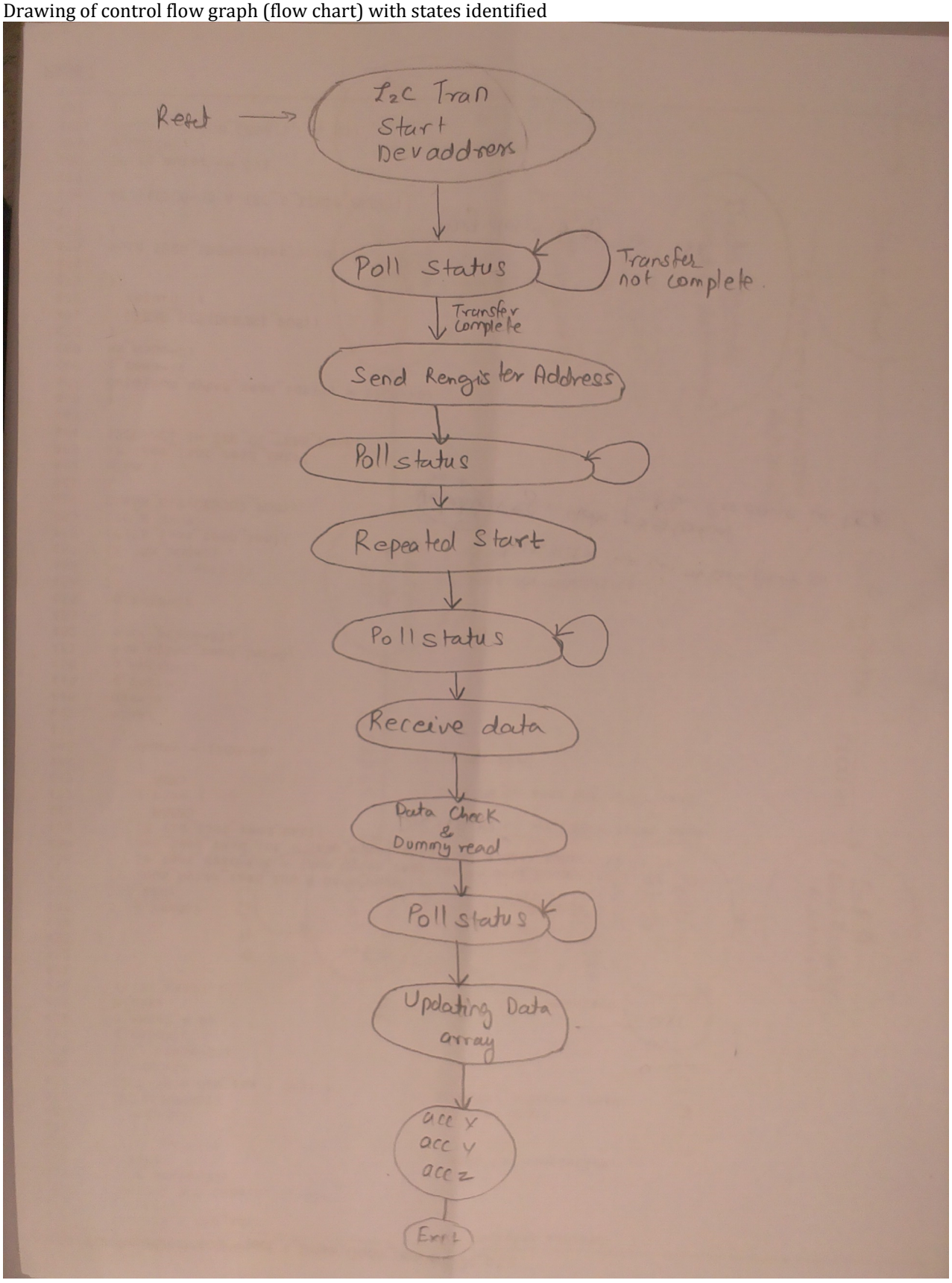
   For this data like device address, register address, number of bytes to be read is passed through the FSM function.

   The handshaking signals used are

   1. ACK- for data receipt
   2. NACK to indicate not to send any more bytes
   3. Stop signal to break the communication.

4. Explanation of how you get result data from the FSM, and handshaking used (if any).
   1. The data that is read is passed through a pointer array of unsigned integer.
   2. Each cycle checks for the number of maximum bytes to be read and then updates the data array with the data from the data register.
   3. Each data read is accompanied by an acknowledgement signal.
   4. For last read an NACK and STOP signal is generated,

5. Drawing of control flow graph (flow chart) with states identified

6. 

6. Drawing of finite state machine showing states and labeled transitions.

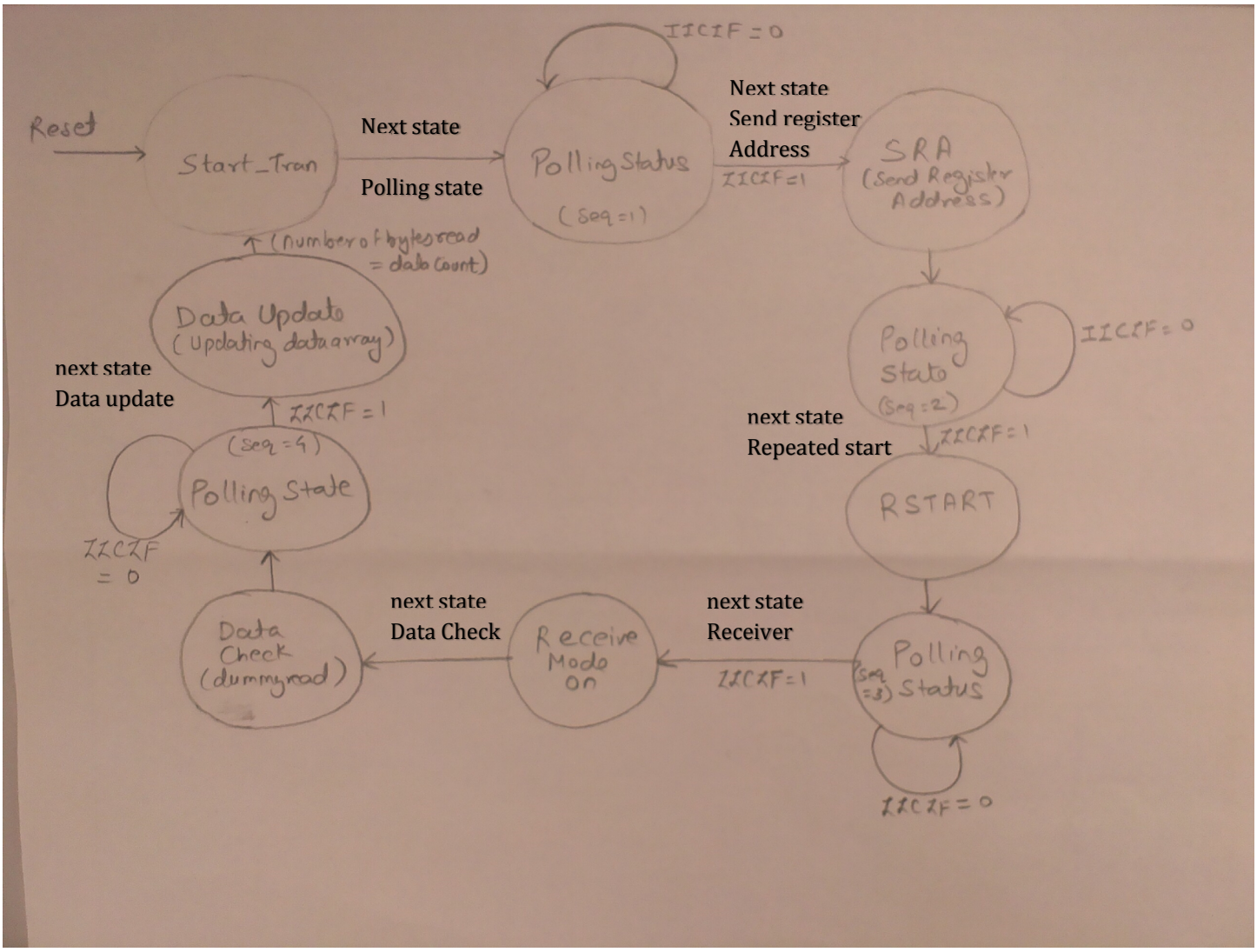


Figure5: State Diagram for read FSM

7. Screenshot showing three debug bits over duration of a read message with no delay between calls to FSM

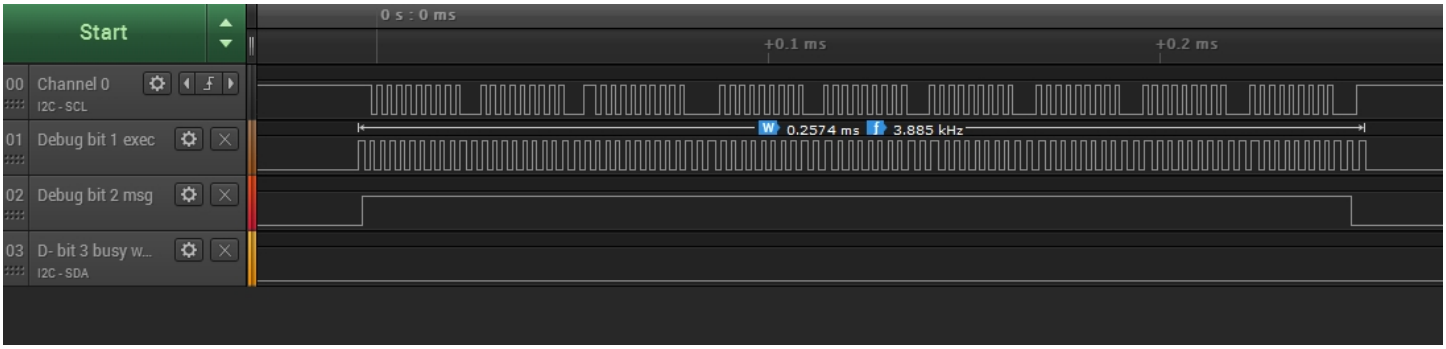


*Figure 6: SDA, Bit 1-exec, Bit 2-message, **Bit 3-busy wait** – FSM with no delay*

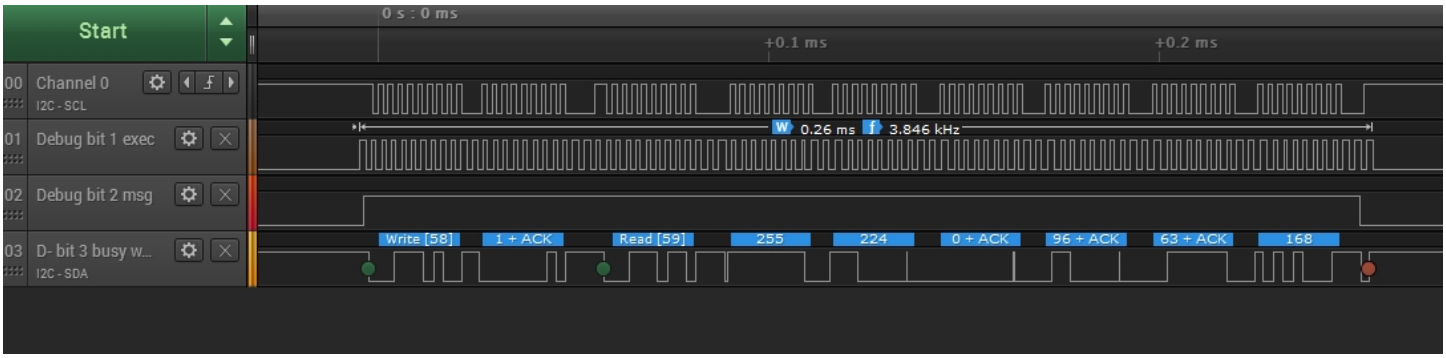


*Figure 7: SCL, Bit 1-exec, Bit 2-message, **SDA** – FSM with no delay*

8. Screenshot showing three debug bits over duration of a read message with maximum delay between calls to FSM

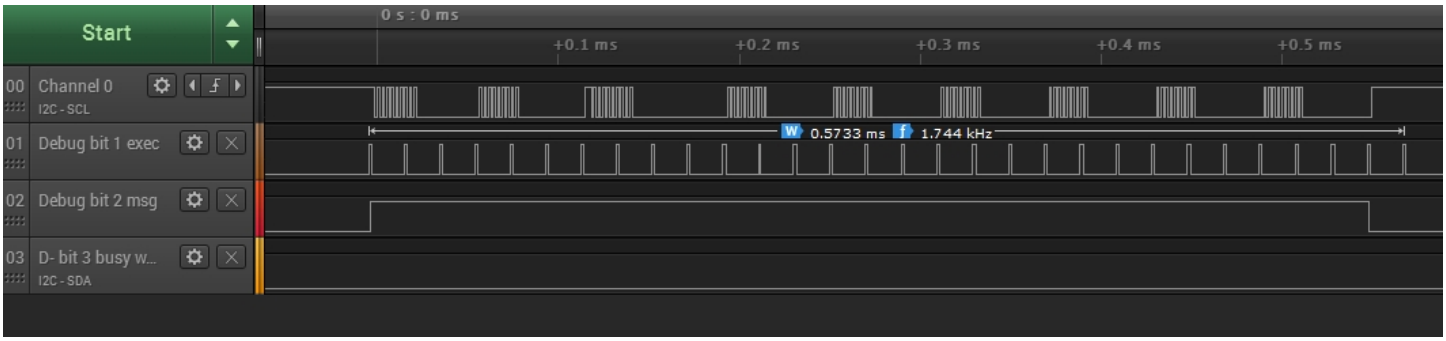


Figure 8: SCL, Bit 1-exec, Bit 2-message, Bit 3-busy wait – FSM with max delay

9. What are the durations of the shortest and longest calls to the FSM function?

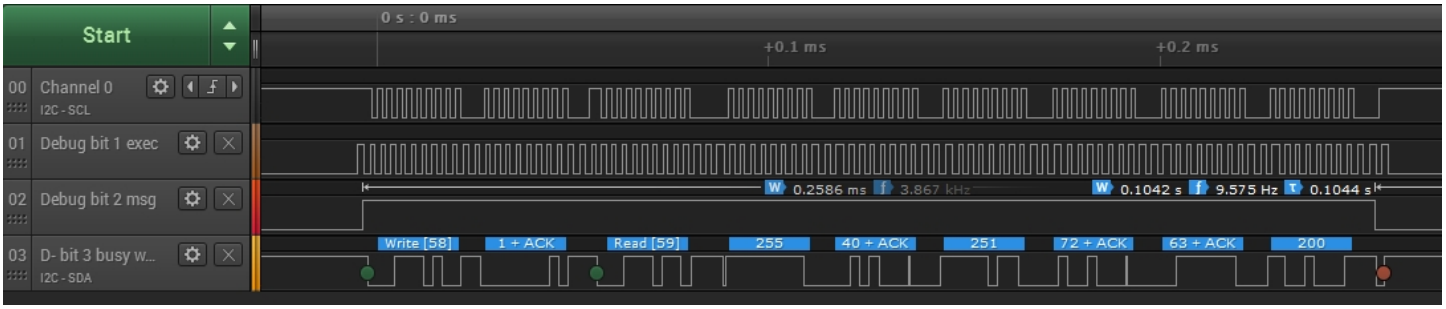


Figure 9: SCL, Bit 1-exec, Bit 2-message, SDA – FSM

Longest Call to FSM

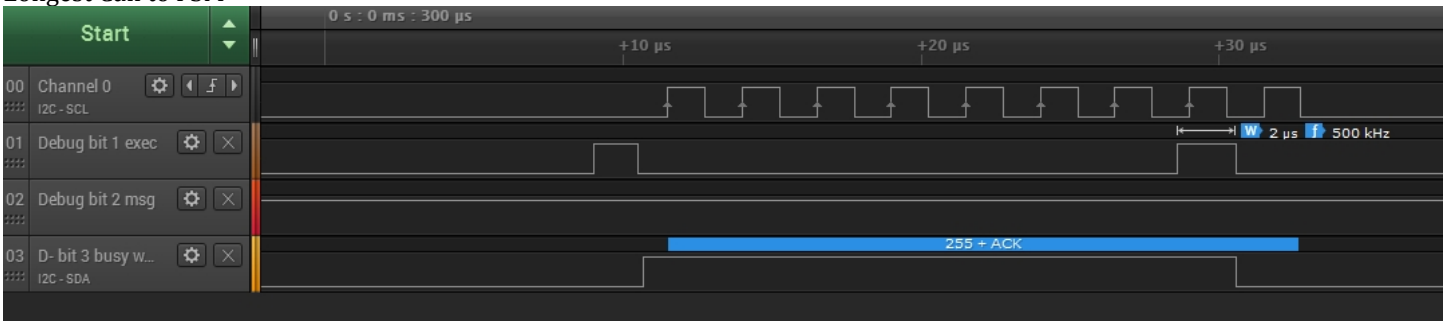


Figure 10: SCL, Bit 1-exec, Bit 2-message, SDA – FSM – Max delay

The duration of longest call is **2usec**.

Shortest call to FSM

Figure 11: SCL, Bit 1-exec, Bit 2-message, SDA – FSM – No delay

The duration of shortest call is **1.167usec.**

10. What is the maximum delay between FSM calls which works?

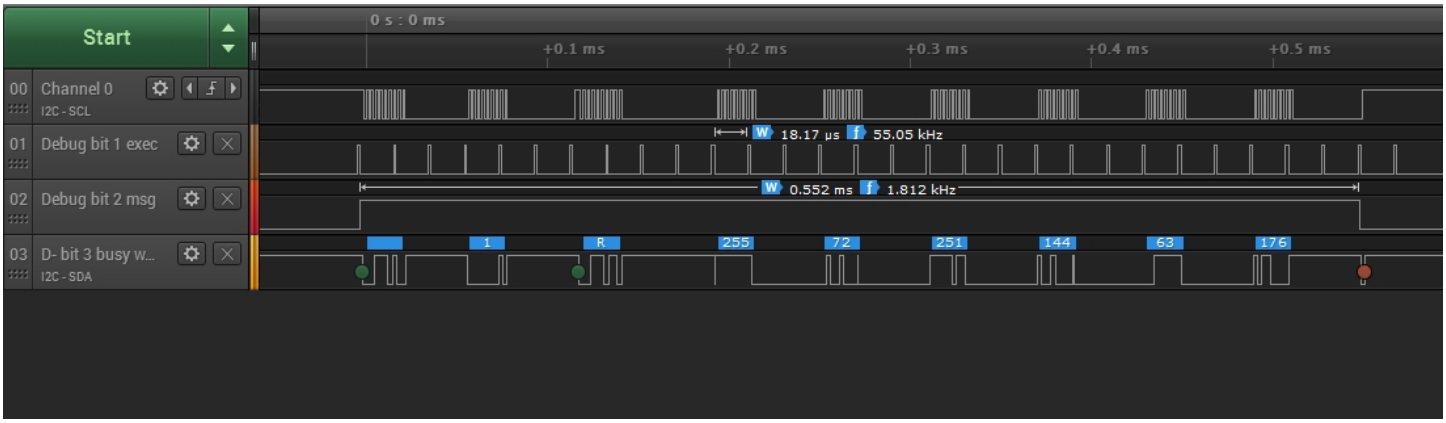


Figure 12: SCL, Bit 1-exec, Bit 2-message, SDA – FSM – Maximum delay

The delay between FSM calls is = **18.17usec**
This is for short_delay(8).

11. With that maximum delay, how much processor time is used to execute the i2c_read_bytes_fsm function to read the accelerations? What is the duration of the message on the bus?

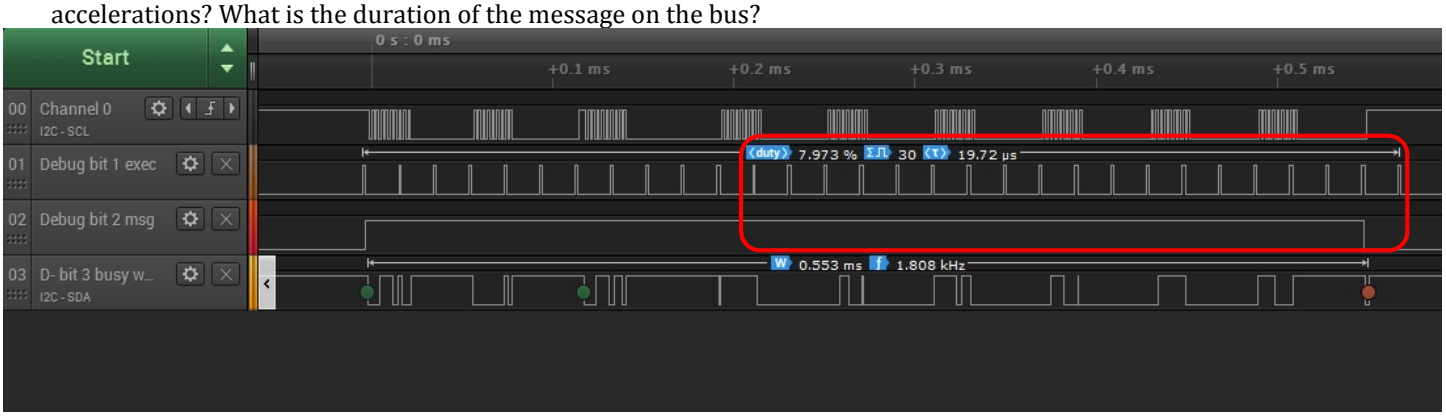


Figure 13: SCL, Bit 1-exec, Bit 2-message, SDA – FSM – Processor time

Processor time = Average Duty Cycle x Average Time period x Number of pulses.
=0.0797 x 19.72u x 30
= **47 usec**.

12. Extra Credit: Reduce the number of states which need to be executed to read the acceleration. Explain your approach and quantify the time benefits.

    In first design FSM was built with wait function and separate states for each operation of I2C communication (Start, Transmit, Device address, register address, repeated start, device address, Receive mode and so on) for transmission and reception of data. With such FSM i2c_read_FSM function was still being called for a considerable time.

    To gain improvement in this design, number of states were clubbed together in such a manner that the operation demanding wait function marked the end of one state and a polling state was created that kept checking the data transfer flag instead of waiting. As soon as the flag was set the program flow routed to the next state. Thus, a considerable reduction of processor time was achieved.

    The processing time improved from 108 usec to 47 usec. Thus with this a free time of 60 usec was achieved.

Interrupt Service Routine Implementation

13. Explanation of how you pass input data to the ISR, and handshaking used (if any)

    The interrupt service routine was enabled only for data reception part in this project design. The initial part of transmission of device and register address was carrying out through polling state in i2c_read_byte_ISR function. Thus the design is a combination of polling state FSM with interrupt.
    For this data like device address, register address, number of bytes to be read is passed through the read_byte_ISR function function.
    The handshaking signals used are
    1. ACK- for data receipt
    2. NACK to indicate not to send any more bytes
    3. Stop signal to break the communication.
    4. Also flags were used to create appropriate logic control between ISR and i2c_read_byte_ISR function,

14. Explanation of how you get result data from the ISR, and handshaking used (if any)
    1. During each read ISR updates an array called irq_data[6]. At end of Final_State, this data was updated to array data[6].
    2. The data that is read is passed through a pointer array of unsigned integer.
    3. Each cycle checks for the number of maximum bytes to be read every time ISR is called and then updates the array irq_data[6]with the data from the data register.
    4. Each data read is accompanied by an acknowledgement ACK signal.
    5. For last read an NACK and STOP signal is generated.

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

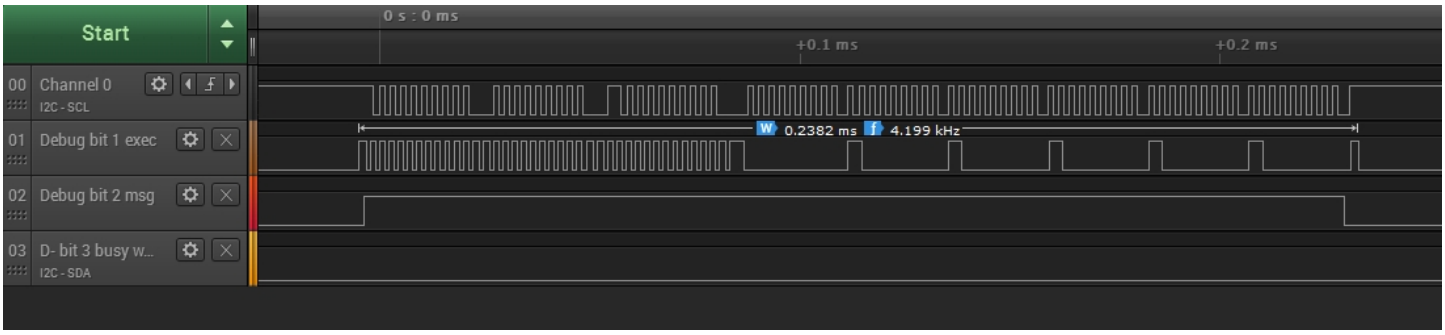


Figure 14: SCL, Bit 1-exec, Bit 2-message, Bit 3-busy wait – Interrupt Service Routine

16. How much processor time is used to execute the I2C ISR to read the accelerations? What is the duration of the message on the bus?
    Processor time = Average Duty Cycle x Average Time period x Number of pulses.
    =0.46 x 5u x 30
    = **69 usec**.
17. However, the interrupt was enabled only for the data read part and the improvement of processing time to read acceleration is
    Processor time = Average Duty Cycle x Average Time period x Number of pulses.
    =0.12 x 24u x 7
    = **20 usec**.
    Against previous of 38 usec for data read part for FSM without interrupt. Thus giving an Improvement of 18 usec to read acceleration data.

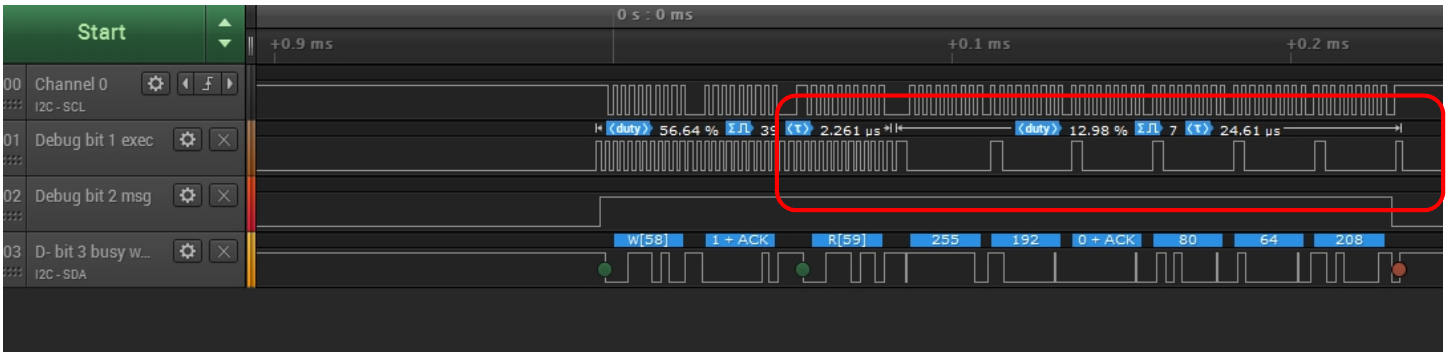


Figure 15: SCL, Bit 1-exec, Bit 2-message, SDA – Interrupt Service Routine-Data read time.