Project 1 Report

ECE 461

Zach Hannum

2017

Contents

[Blocking Code Implementation 2](#_Toc473788358)

[Finite State Machine Implementation 3](#_Toc473788359)

[Input Data 3](#_Toc473788360)

[Output Data 3](#_Toc473788361)

[Control Flow Diagram 3](#_Toc473788362)

[State Machine Diagram 4](#_Toc473788363)

[Logic Analyzer 5](#_Toc473788364)

[Interrupt Service Routine Implementation 6](#_Toc473788365)

[Input Data 6](#_Toc473788366)

[Output Data 6](#_Toc473788367)

[Logic Analyzer 6](#_Toc473788368)

[Figure 1: Blocking Code I2C message on logic analyzer 2](#_Toc473788369)

[Figure 2: FSM I2C Control Flow diagram 3](#_Toc473788370)

[Figure 3: FSM I2C State Diagram 4](#_Toc473788371)

[Figure 4: FSM I2C message with no delay 5](#_Toc473788372)

[Figure 5: FSM I2C message with ~17 μs delay 5](#_Toc473788373)

[Figure 6: FSM I2C message with ~19 μs delay - Missing NAK 5](#_Toc473788374)

[Figure 7: FSM I2C message with ~21 μs delay - Reading 11 bytes (broken) 5](#_Toc473788375)

[Figure 8: ISR I2C Full Message 6](#_Toc473788376)

[Figure 9: ISR I2C Message detail 1 6](#_Toc473788377)

[Figure 10: ISR I2C Message detail 2 6](#_Toc473788378)

[Figure 11: ISR I2C Message detail 3 6](#_Toc473788379)

# Blocking Code Implementation

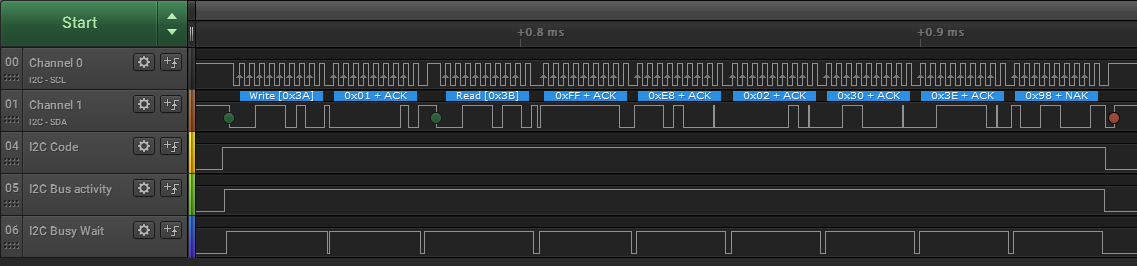


Figure 1: Blocking Code I2C message on logic analyzer

# Finite State Machine Implementation

## Input Data

## Output Data

## Control Flow Diagram



Figure 2: FSM I2C Control Flow diagram

## State Machine Diagram



Figure 3: FSM I2C State Diagram

## Logic Analyzer

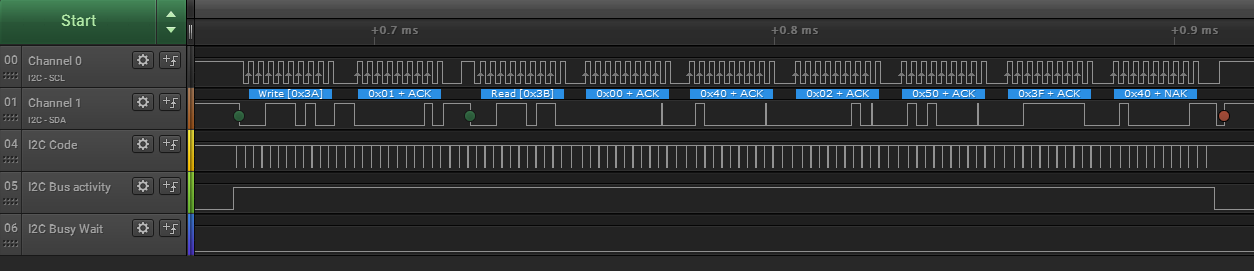


Figure 4: FSM I2C message with no delay

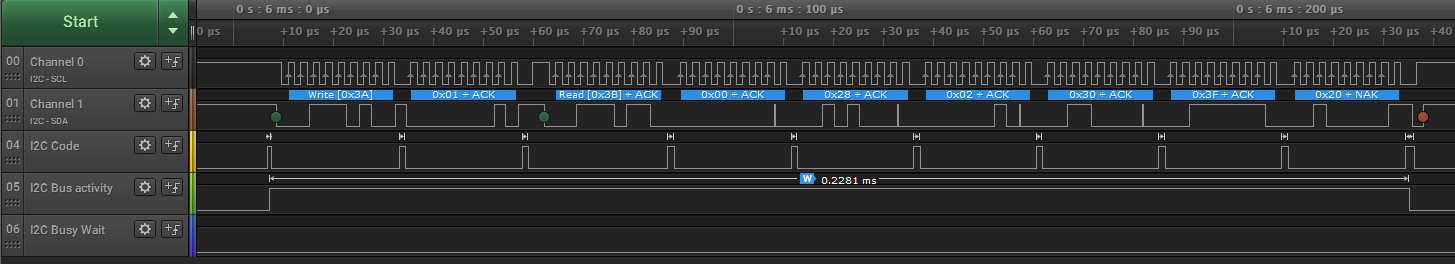


Figure 5: FSM I2C message with ~17 μs delay

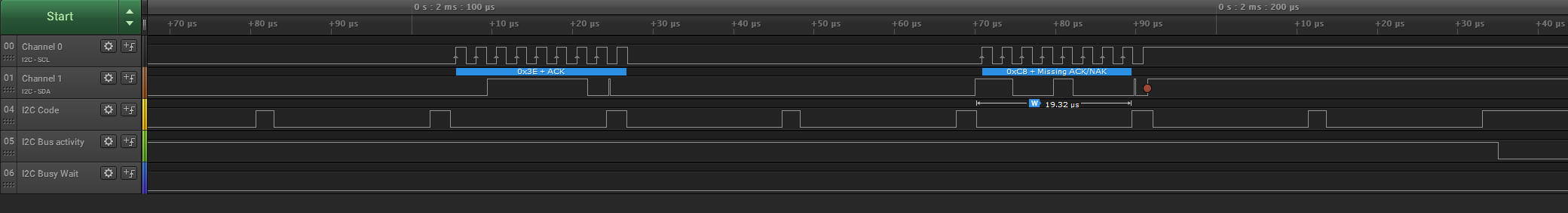


Figure 6: FSM I2C message with ~19 μs delay - Missing NAK

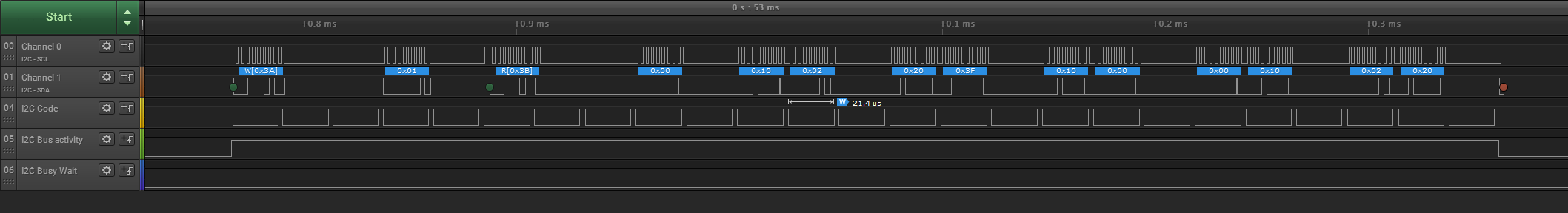


Figure 7: FSM I2C message with ~21 μs delay - Reading 11 bytes (broken)

As seen in Figures 5-7, polling the I2C State Machine at ~17 μs intervals is the largest delay without causing issues with the communications. This corresponds to ShortDelay(8) in code. Increasing the delay to ShortDelay(9) or ~19 μs, The I2C message is missing the NAK at the end of reading all 6 bytes. Interestingly though, the program still seems to function correctly. At ShortDelay(9) or ~21 μs, the program completely breaks, reading out 11 bytes of data. The data[] array changes on every iteration, causing the LED to flash white constantly.

# Interrupt Service Routine Implementation

## Input Data

## Output Data

## Logic Analyzer

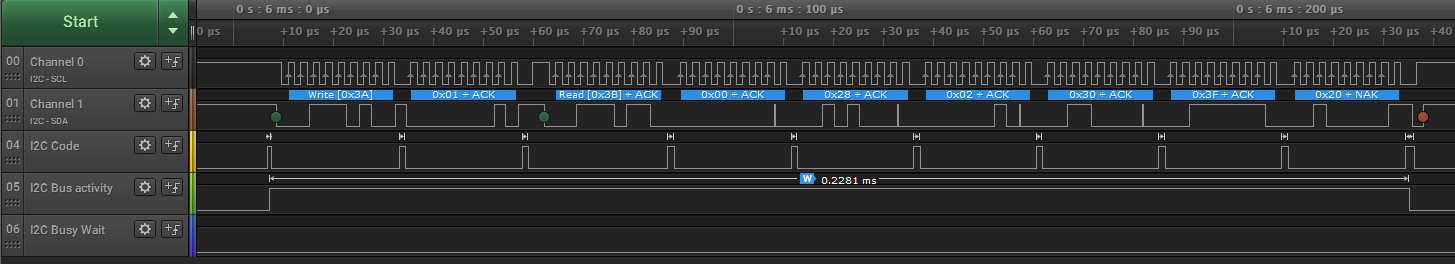


Figure 8: ISR I2C Full Message

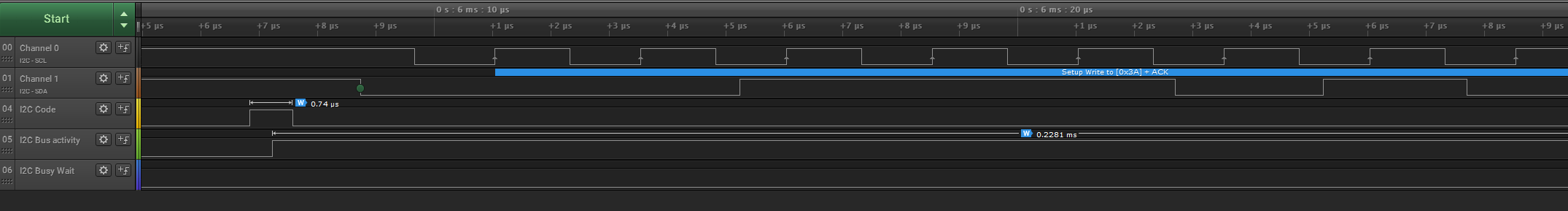


Figure 9: ISR I2C Message detail 1

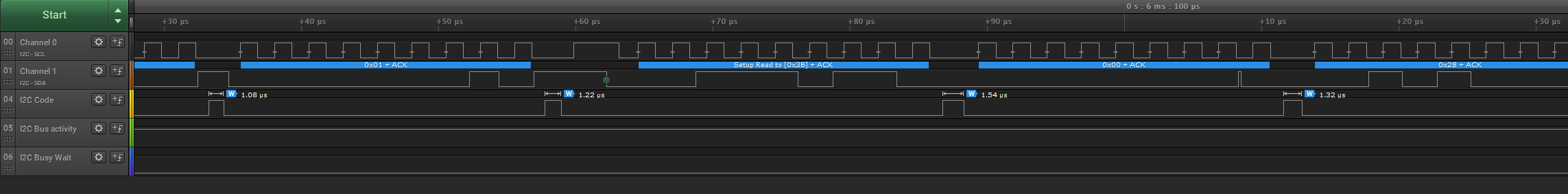


Figure 10: ISR I2C Message detail 2

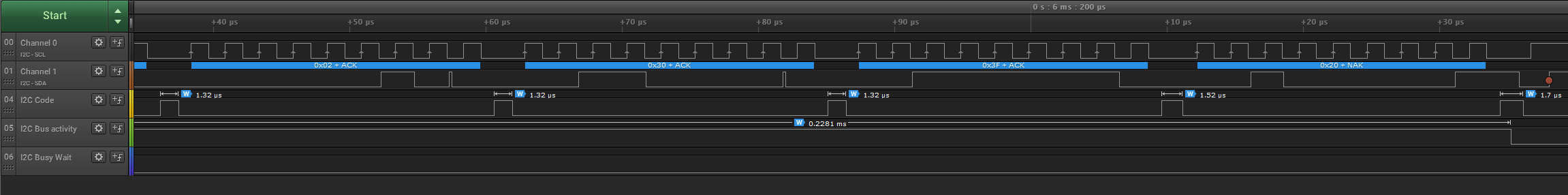


Figure 11: ISR I2C Message detail 3

As seen the detailed I2C message Figures (9-11), the total processor time executing the I2C code is

0.74 μs + 1.08 μs + 1.22 μs + 1.54 μs + 1.32 μs + 1.32 μs + 1.32 μs + 1.32 μs + 1.52 μs + 1.7 μs = **13.08 μs**

Figure 8 shows that the total duration of the message on the bus is **228.1 μs**. So the processor only needed to run ~6% of the time that the message was on the bus. ­