ECEN 5803 – Mastering Embedded Systems Architecture – Project 2

Report on
Evaluation of ST STM32F401RET6 MCU
For the Keithley 2019 Signal Analyzer

Contents

1. Executive Summary 3
2. Problem Statement and Objectives 3
3. Approach and Methodology for Evaluation 3
    3.1 Hardware Evaluation – System Inputs 3
    3.2 Hardware Evaluation – System Outputs 4
4. Module Test Results 4
    4.1 Module 1: Hello World, What Time is it? 4
    4.2 Module 2: Button Read, ADC Read, LED PWM, and UART 4
    4.3 Module 3: RTOS Threads 5
    4.4 Module 4: Signal Analyzer 5
5. List of Project Deliverables 6
    5.1 Software Deliverables 6
    5.2 Hardware Deliverables 6
6. Recommendations 6
7. Appendix: Reference 7
8. Appendix: Frequently Asked Questions 7
    8.1 Module 1 7
    8.2 Module 2 8
    8.3 Module 3 8
    8.4 Module 4 8
9. Appendix: Project Staffing 8

1. Executive Summary

This report outlines the various tests that were performed on the ST STM32F401RET6 to test its feasibility for this project. Based on our analysis and rigorous tests, we conclude that this microcontroller is an ideal candidate for the Signal Analyzer by Keithley. The on-board processor is an ARM Cortex M4 which has a high range of optimized Thumb2 instruction set which greatly improves the throughput of the system while keeping the development process easy and fast. The ST STM32F401RET6 contains an FPU which helps significantly with calculations of complex algorithms of FFT in real time. The board has many interfaces like I2C, SPI and UART which allows connectivity with external devices. LAN connectivity can be provided to the system with the help of ethernet chips. The software support for this device is extensive and covers all the features offered by this microcontroller in order to accelerate the development process.

The ST STM32F401RET6 has prices averaging around $3.29 which is below the $4 MCU budget. The embedded system has and overall cost of about $19.45 which is below the $20 limit. There are externally sourced components to meet requirements, such as, ADC's, DACS, Differential amplifiers and Ethernet IC's. These parts are added to make up for some of the requirements. The STM32 lacks the presence of an onboard DAC which can be easily mitigated by using an external DAC chip like the PCM5102. The ST STM32F401RET6 has different package sizes, with the most common having 64 pinouts which is right on the edge of needed pin lines. There is also a 100-pinout package available. All the test requirements outlined by Keithley were performed and the board seem to have passed all the requirements.

Although this product is far from development, the current firmware along with more hardware enhancements and software optimizations will make this product ideal to drive the System Analyzer as shown by the justifications throughout this report.

After careful speculation of the requirements, the ST STM32F401RET6 is given a GO for the application of a System Analyzer.

2. Problem Statement and Objectives

To design an Embedded System as per the requirements, presented by Keithley, for Signal Analyzer using Cortex-M4 based microcontroller ST STM32F401RET6. In this project, the STM32 Nucleo evaluation board is used for the evaluation of the ST STM32F401RET6 for the requirements presented by Keithley.

3. Approach and Methodology for Evaluation

Below is the block diagram for the proposed system design.

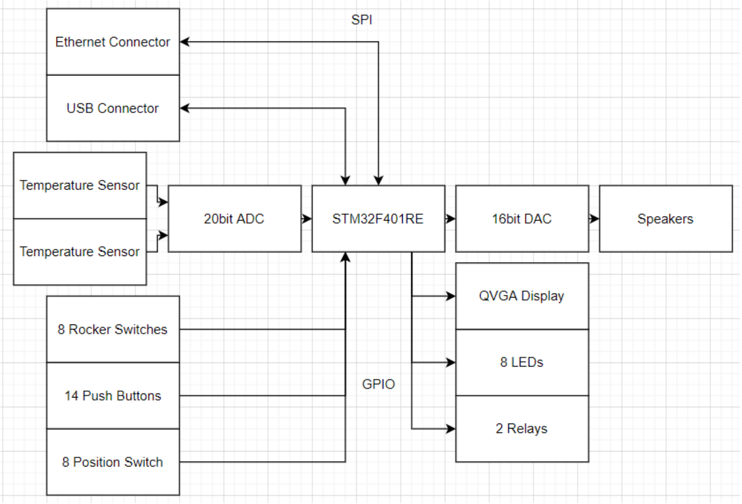


Figure 1: Block diagram for the proposed system

3.1 Hardware Evaluation – System Inputs

14 buttons
8-position switch
8 rocker buttons
Single ended analog Audio
Differential analog Audio
Hi Impedance Audio
SPDIF Digital Audio
I2S Digital Audio

2 Temperature Sensors
20 bit ADC required at 192 kSps
16 bit DAC required at 192 kSps

3.1.1

The 14 buttons, 8-position switch, and the 8 rockers buttons will interface via 30 GPIO pins to the ST STM32F401RET6. Should there be any need for more IO, the higher package version of the controller can be used or an IO expander can be employed here.

3.1.2

The Single ended analog Audio, and the Differential analog Audio will operate with an external Differential Audio Amplifier interfaced with an external DAC chip which will in turn interface with the ST STM32F401RET6 over I2S for higher bit rate.

3.1.3

The SPDIF Digital Audio input will be will interface with the ST STM32F401RET6 via I2S and an external DAC designed to generate/identify SPDIF modulated signals. The Hi Impedance Audio uses the DAC.

3.1.4

The ST STM32F401RET6 comes with an internal temperature sensor attached to the ADC_IN16 input channel. An external temperature sensor will connect via I2C or it could be a plain analog temperature sensor.

3.1.5

There isn't an internal DAC on the ST STM32F401RET6 instead and external DAC was sourced. For part information seem the BOM attached to the technical report.

3.1.6

The internal ADC is only 16 bit and the system requires a 20-bit ADC so an external ADC was sourced to meet the requirements. For part information seem the BOM attached to the technical report.

3.2 Hardware Evaluation – System Outputs

8 LEDs
QVGA display
2 relays
Stereo Audio Speakers

3.2.1

The QVGA display can consume up to 24 pin lines but can be interfaced directly with the ST STM32F401RET6. There are also other QVGA displays that interface with the controller using SPI or other high-speed protocols.

3.2.2

The relays and the LEDs will need about 16 pins to operate. The ST STM32F401RET6 will drive them through GPIO pins.

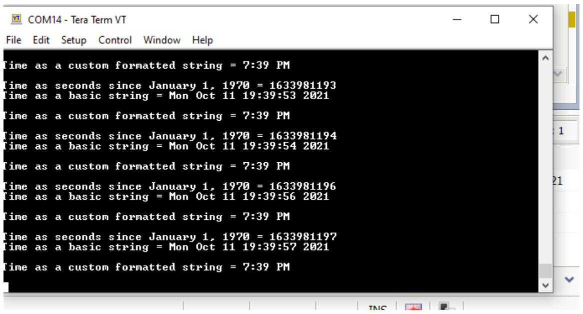
3.2.3

The Stereo Audio Speakers can be driven off of the external DAC in conjunction with the ST STM32F401RET6.

4. Module Test Results

4.1 Module 1: Hello World, What Time is it?

In module 1, the onboard real-time clock of the ST STM32F401RET6 is used to set the time to the current time and then in conjunction with printing code, print the current time to the terminal window on the PC.



```
Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981193
Time as a basic string = Mon Oct 11 19:39:53 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981194
Time as a basic string = Mon Oct 11 19:39:54 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981196
Time as a basic string = Mon Oct 11 19:39:56 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981197
Time as a basic string = Mon Oct 11 19:39:57 2021

Time as a custom formatted string = 7:39 PM
```

Figure 2: RTC on STM32

4.2 Module 2: Button Read, ADC Read, LED PWM, and UART

4.2.1 Programming using mbed API

In this section the goal was to interface the GPIOs using digital IO functions in the mbed API and implement the interrupt handler using the interrupt libraries in the mbed API.

4.2.2 Analog Input and Output

Using PWM and ADC, potentiometers and speakers were interfaced together in order to generate sound out of the speaker with the potentiometers controlling volume and pitch of the audio. The idea is that the when the period of the PWM changes, the pitch of the resultant analog wave will change. When the Ton time of the PWM changes, the Amplitude of the resultant analog waveform changes. Hence one potentiometer was used to vary the pitch and the other to vary the amplitude. The pitch has to be between 320Hz to 8000Hz. The amplitude can be between 0 and 3.3v.

4.2.3 Serial Communication

In this section the goal was to integrate three separate communication protocols to interface with three different peripherals. SPI is used to interface to an LCD display, I2C is used to interface to a temperature sensor, and UART is used to interface with a PC. After separate completion, an integrated application was created that used the LCD display to display the current temperature.

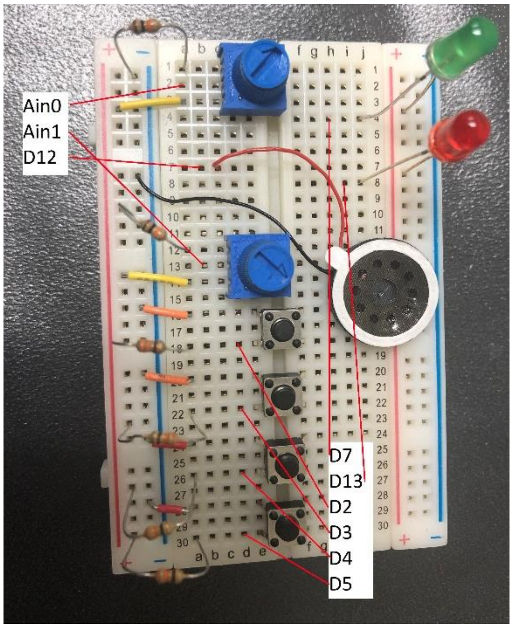


Figure 3: Circuit for 4.2.1 and 4.2.2.
For 4.2.3 See Figure 4

4.3 Module 3: RTOS Threads

The CMSIS RTOS (Keil RTX) was used to implement an RTOS. The RTOS ran and scheduled four threads meant to run concurrently. The four threads functionality are: Display the temperature on the LCD, Adjust the brightness of the external LED using a potentiometer, Display an incrementing counter on the LCD, and Blink an external LED. The RTOS utilized mutexes and semaphores to perform the necessary exclusions and protection of the resources.

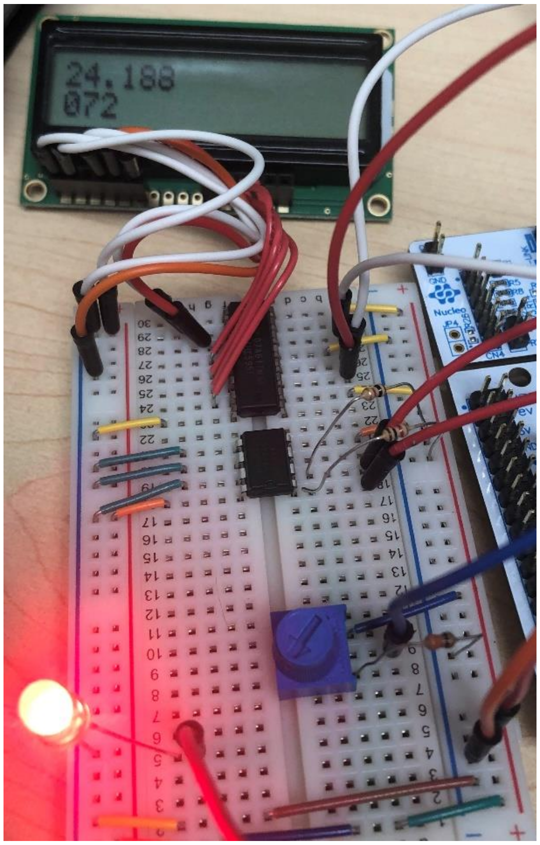


Figure 4: The Complete circuit for Module 3, including Temperature and Counter displayed on the LCD Display. The LCD circuit and Temperature circuit (Top half of Breadboard) also for 4.2.3

4.4 Module 4: Signal Analyzer

The software DAC was used to generate a 1004 Hz tone. The samples from the ADC were used to preform Harmonic analysis using FFT powered by the highly efficient FPU on the STM32. A user LED was then blinked at a rate proportional to the ADC input fundamental frequency. This essentially creates an overall signal analyzer that can help stand as a proof of concept for the system.

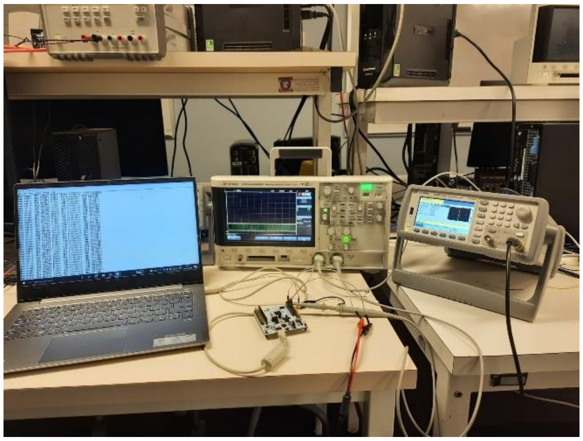
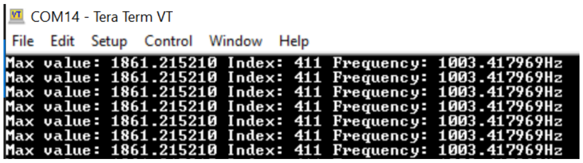


Figure 5: Lab Test Setup

The lab setup for doing FFT on the STM32. The input to the STM32 is the waveform from the function generator. The resultant waveform can be seen on the Oscilloscope. The output of the STM32 is the frequency of the fundamental in the input wave which is printed on the console.



```
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
Max value: 1861.215210 Index: 411 Frequency: 1003.417969Hz
```

Figure 6: FFT on the STM32 output.

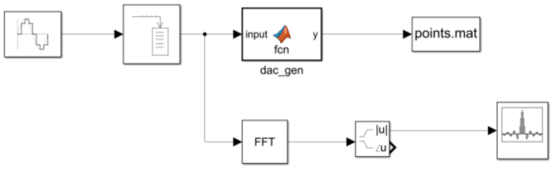


Figure 7: Simulink Simulation of the DAC/ADC => FFT

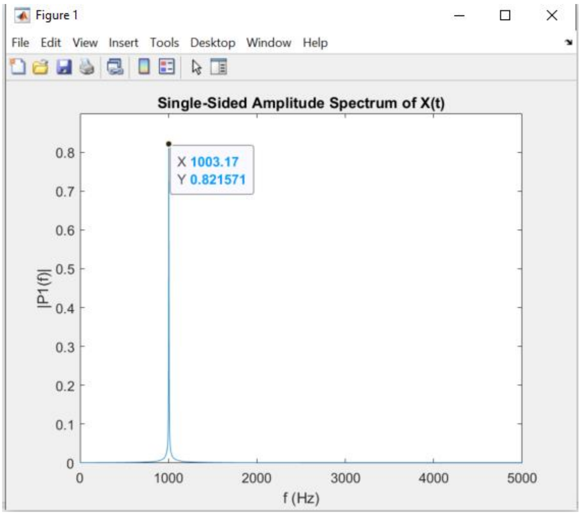


Figure 8: FFT of the input sine wave of 1004Hz

5. List of Project Deliverables

5.1 Software Deliverables

Module code files, simulation, block diagram and test results are compressed to ZIP file and attached with this report. Doxygen report files are also provided with the respective module directory.

5.2 Hardware Deliverables

Bill of materials for key parts of the project provided with this report. Pictures of the hardware test circuits provided with individual modules.

6. Recommendations

The required parameters while designing the Signal Analyzer for Keithley are Speed, Cost and Functionality. ST STM32F401RET6 performs well on a lot of the parameters because of its well laid out architecture and the presence of an FPU. After the evaluation and testing, it can be concluded that the ST STM32F401RET6 can be used as a microcontroller for the Keithley Signal Analyzer. The ST STM32F401RET6 can be equipped with an external DAC, a 20bit ADC and LAN capabilities with the help of external chips. This makes the development of this product feasible using the ST STM32F401RET6 in terms of functionality and also cost. The overall cost of the system will be about $19.45 which is below $20. Hence, the recommendation of the use of ST

STM32F401RET6 in the Keithley 2019 Signal Analyzer product has been given a GO.

7. Appendix: Reference

7.1 Project 2 Guide, which describes the 4 Test Modules to be completed.
7.2 ST STM32F401RE Datasheet.
7.3 ST STM32F401RE Reference Manual.
7.4 ST Nucleo 401 Product Brief
7.5 ST Nucleo 401 User's Guide.
7.6 Keithley 2015-2016 Datasheet

8. Appendix: Frequently Asked Questions

8.1 Module 1

8.1.1

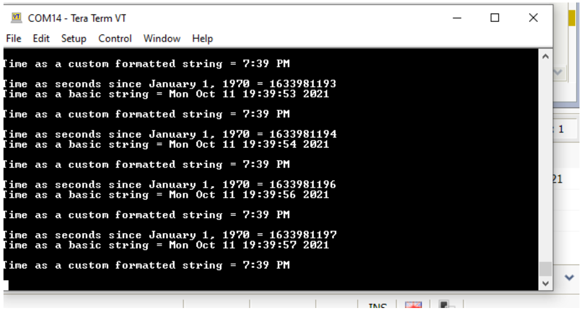
Where (at what address) does the Reset handler begin in the memory map?
Answer: 0x08000458

8.1.2

How much memory is used by the code (Led blinking code for the homework)?
Answer: 40.576Kb

8.1.3

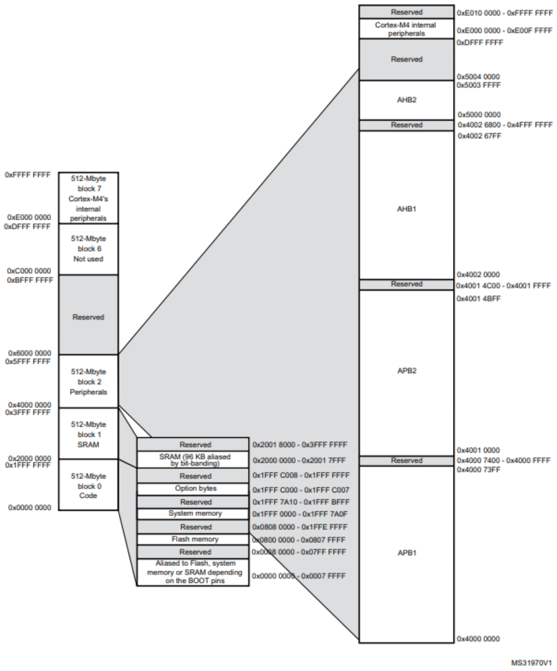
Run the mBed Nucleo Example (display_time). Set the time to the current time, and combine this with you're the mBed Nucleo Example (printf) to print the current time to a terminal window on your PC. Capture a screen shot of the terminal window. How much memory is used by this code?
Answer:
Memory consumption: 24.4Kb



```
Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981193
Time as a basic string = Mon Oct 11 19:39:53 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981194
Time as a basic string = Mon Oct 11 19:39:54 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981196
Time as a basic string = Mon Oct 11 19:39:56 2021

Time as a custom formatted string = 7:39 PM

Time as seconds since January 1, 1970 = 1633981197
Time as a basic string = Mon Oct 11 19:39:57 2021

Time as a custom formatted string = 7:39 PM
```

8.1.4

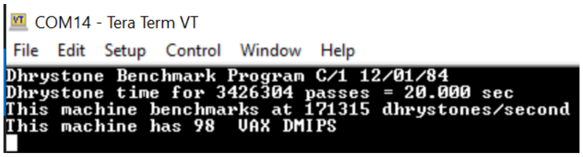
Explain the memory model of ARM Cortex-M4 with respect to the code memory, data memory, IRQ handlers and peripherals. Explain with the help of a diagram where required.
Answer:


0xFFFF FFFF
512-Mbyte block 7 Cortex-M4's internal peripherals
0xE000 0000
0xDFFF FFFF
512-Mbyte block 6 Not used
0xC000 0000
0xBFFF FFFF
Reserved
0x6000 0000
0x5FFF FFFF
512-Mbyte block 2 Peripherals
0x4000 0000
0x3FFF FFFF
512-Mbyte block 1 SRAM
0x2000 0000
0x1FFF FFFF
512-Mbyte block 0 Code
0x0000 0000
Reserved
0x2001 8000 - 0x3FFF FFFF
SRAM (96 KB aliased by bit-banding)
0x2000 0000 - 0x2001 7FFF
Reserved
0x1FFF C008 - 0x1FFF FFFF
Option bytes
0x1FFF C000 - 0x1FFF C007
Reserved
0x1FFF 7A10 - 0x1FFF BFFF
System memory
0x1FFF 0000 - 0x1FFF 7A0F
Reserved
0x0808 0000 - 0x1FFE FFFF
Flash memory
0x0800 0000 - 0x0807 FFFF
Reserved
0x0008 0000 - 0x07FF FFFF
Aliased to Flash, system memory or SRAM depending on the BOOT pins
0x0000 0000 - 0x0007 FFFF
Reserved
0xE010 0000 - 0xFFFF FFFF
Cortex-M4 internal peripherals
0xE000 0000 - 0xE00F FFFF
0xDFFF FFFF
Reserved
0x5004 0000
0x5003 FFFF
AHB2
0x5000 0000
Reserved
0x4002 6800 - 0x4FFF FFFF
0x4002 67FF
AHB1
0x4002 0000
Reserved
0x4001 4C00 - 0x4001 FFFF
0x4001 4BFF
APB2
0x4001 0000
Reserved
0x4000 7400 - 0x4000 FFFF
0x4000 73FF
APB1
0x4000 0000
MS31970V1

The memory model of the Coretx m4 implemented on the STM32F401RET6 is displayed above, As you can see the different sections belonging to different set of peripherals. The total memory is about 4GB. The fist block, Block 0 code consists of memory for storing and executing the program. It contains the flash region. Block 1 contains the SRAM region. This is the RAM for the code. It will also be the space in which the memory is assigned for atomic operations. Block 2 is divided into different regions like APB1, APB2 and AHB1 etc. This is also known as the peripheral memory region. The peripherals are connected to the APB busses. Each peripheral is given it's set of registers for configuration.

8.1.5

As a separate project, run either the Dhrystone or the Whetstone benchmark program on your target processor using the code provided, which may need to be modified. If running the Dhrystone, calculate the number of VAX DMIPS.
Answer: 98 DMIPS



```
Dhrystone Benchmark Program C/1 12/01/84
Dhrystone time for 3426304 passes = 20.000 sec
This machine benchmarks at 171315 dhrystones/second
This machine has 98  VAX DMIPS
```

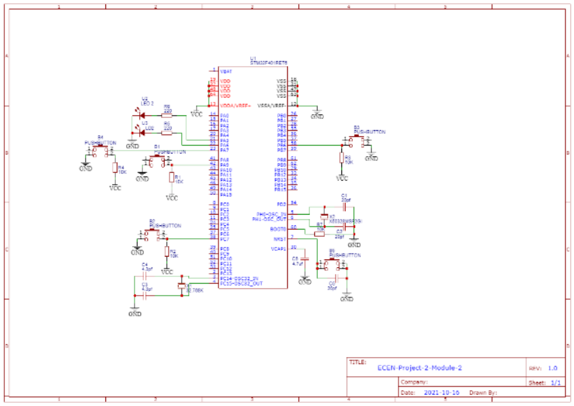
8.2 Module 2

8.2.1

Try to issue an interrupt on different signal edges (rising edge or falling edge). What changes?
Answer: If the interrupt is on the rising edge, then the interrupt will occur upon releasing the button. If the interrupt is on the falling edge, then the interrupt will occur upon pushing the button down.

8.2.2

Draw a schematic of your final circuit (nothing fancy, can be done by hand and scanned).
Answer: See GitHub “/Module_2/sch.png”



8.2.3

What changes when you adjust the amount by which variable <u>*i*</u> is incremented/decremented?
Answer: Changing the variable **i** will get a higher pitch upon increase and a lower pitch upon decrease. The IRQs are passed through the interrupt vector table which is placed at the start of the

8.3 Module 3

No Questions Presented

8.4 Module 4

8.4.1

Run Splint or an equivalent code checker (like CppCheck or Cpplint) on your code in 1 of this module. Resolve any errors, explain the warnings.
Answer: Ran cppcheck on the entire directory. No error or warnings in the code written by the user. The library mbed-os has warnings.

8.4.2

Auto-generate documentation using Doxygen on your code in 1 of this module. Provide either an HTML directory or PDF file documenting your codebase
Answer: In the git repo under Module 4.

8.4.3

Estimate the processor load in % of CPU cycles.
Answer:
fft computation: 6824us
total cpu execution time for 1 cycle of the code: 10633us
% load = 64.11%

9. Appendix: Project Staffing

Sankalp Agrawal
Graduate Student
University of Colorado Boulder
Email: saag2511@colorado.edu
Phone: +1 (721)-761-3030

Kanin McGuire
Under-Graduate Student
University of Colorado Boulder
Email: kamc9940@colorado.edu
Phone: +1 (720)-326-6241