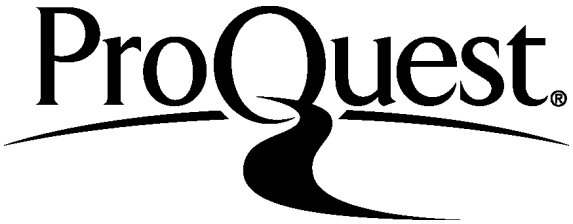
DESIGN AND DEVELOPMENT OF AN OPEN SOURCE PROGRAMMABLE DC ELECTRONIC LOAD

by

Yingbo Wang

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

Spring 2015



DESIGN AND DEVELOPMENT OF AN OPEN SOURCE PROGRAMMABLE DC ELECTRONIC LOAD

by

Yingbo Wang

Approved: _____

Fouad E. Kiamilev, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____

Kenneth E. Barner, Ph.D.
Chair of the Department of Electrical and Computer Engineering

Approved: _____

Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____

James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

First and foremost, I thank God for bringing me to the University of Delaware and for giving me the opportunity to obtain a Masters degree. It has been a difficult journey through academia, and I wasn't sure I was going to make it, but God has helped me persevere through my anxiety and self-doubt and helped bring this thesis to fruition.

Second, I want to thank my advisor, Dr. Fouad Kiamilev, for taking me into his research group ever since my early years as an undergraduate student at Delaware and for continuing to sponsor me for my graduate studies. Dr. K has always been supportive of my education and motivated me to finish my degree—despite my packing up and leaving for a full-time job in the midst of it. I would have"dropped out" if it weren't for Fouad's persistent "pestering". I am blessed to have been part of his crazy, fun, and quirky research group that is CVORG.

I definitely want to thank my dear friend Karl Bowers for his guidance and tremendous help in completing the design of the FreeDum Load. I would not have completed my thesis without him. Even though "hardware" isn't my forte, Karl's deep passion for all things electronics inspires me to continue to learn and tinker with the harder stuff. Karl will always be the number one electronics engineer in my book.

Thanks to Rodney McGee, Corey Lange, Rob Haislip, and Rob Rehrig, for their theses, which served as templates and as evidence that writing a thesis of my own is indeed possible.

Thanks to the fellow girls of CVORG, Michelle Pettinella and Yulia Karymova, for their friendship. Thanks Michelle, the mother hen of CVORG, for her awesome birthday cupcakes, and Yulia for our entertaining conversations of cats.

Lastly, I want to thank my roommate Melissa Tang and my coworker, Shawn Oberoi, for their continued encouragement in helping me finish this paper.

TABLE OF CONTENTS

LIST OF TABLES viii
LIST OF FIGURES ix
ABSTRACT xi

Chapter

1 INTRODUCTION 1

1.1 Background 1
1.2 Motivation 1

2 EXISTING MODELS AND DESIGNS 4

2.1 Commercially Available Models 4
2.2 Open Source Designs 5

2.2.1 EEVblog Episode #102 6
2.2.2 Martin Lorton's Electronic DC Load 7
2.2.3 Re:load 10
2.2.4 Other Designs 12

2.3 Design Summary 13

3 FREEDUM LOAD HARDWARE DESIGN 15

3.1 The Basic Electronic Load Circuit 15
3.2 Expansion of the Basic Circuit 16

3.2.1 Parallel Load Resistors 17
3.2.2 Op-Amp 17

3.2.3 Op-Amp Buffer Circuit . . . 17

3.3 Peripherals . . . 19

3.3.1 Microcontroller . . . 19
3.3.2 LCD Display . . . 20
3.3.3 The USB Port . . . 20
3.3.4 Controller Dial . . . 22
3.3.5 External Power Supply . . . 23
3.3.6 Fan Controller and Fans . . . 24

3.4 Manufacturing and Assembly . . . 25

3.4.1 PCB Manufacturing . . . 25
3.4.2 Enclosure . . . 29

4 FIRMWARE . . . 31

4.1 IDE and Programmer . . . 31
4.2 Source Code . . . 33

5 DESIGN PITFALLS . . . 34

5.1 Op-Amp Rail . . . 34
5.2 Heatsink . . . 35
5.3 Microcontroller . . . 35
5.4 Design Cost . . . 36

6 CONCLUSION . . . 37

REFERENCES . . . 39

Appendix

A FIRMWARE . . . 41

A.1 LCD Driver . . . 41

A.1.1 LCD_Support.h . . . 41

A.1.2 LCD_Support.c 43

A.2 Fan Controller Driver 48

A.2.1 SMBus_Support.h 48
A.2.2 SMBus_Support.c 53

A.3 Quadrature Encoder Driver 57

LIST OF TABLES

2.1 A comparison of prices and specifications of various commercial DC electronic loads. 5

LIST OF FIGURES

2.1 An Array brand electronic load (Array, n.d.). 4

2.2 Screen shot of Dave Jones' adjustable electronic load design as seen from EEVblog episode #102 (Jones, 2010). 7

2.3 Screenshot from Martin Lorton's video showing the front and back of his Electronic DC Load (Lorton, 2013). 9

2.4 Photo of the two available versions of Re:load posted on the Arachnid Labs website (Johnson, 2014). 11

2.5 Photo of ITead Studio's electronic load board (Itead, 2010). 12

2.6 Photo of the internals of Wigman's electronic load as posted in his Instructables guide (Wiggins, 2013). 14

3.1 The schematic of the basic circuit. 16

3.2 The schematic of the expanded circuit. 18

3.3 The schematic of the microcontroller. 21

3.4 The schematic of the lcd screen. 22

3.5 The schematic of the usb port. 23

3.6 The schematic of the rotary encoder. 23

3.7 The schematic of the power supply and voltage regulator. 24

3.8 The schematic of fan controller circuit. 25

3.9 The PCB layout. 26

3.10 The front of the board. 27

3.11 The front of the board with LCD attached. 27

3.12 The back of the board. 28

3.13 The front panel of the enclosure. 28

3.14 The AC to DC converter slotted inside the case. 30

3.15 The back panel of the enclosure. 30

4.1 The jtag connection to the microcontroller. 32

6.1 The FreeDum Load complete in case and powered on. 38

ABSTRACT

A DC electronic load is a test instrument that is used to simulate loading on an electronic circuit or device. DC electronic loads can be bought as commercial units. These units can be expensive with prices ranging from several hundred to a few thousands of dollars. They tend to be robust and offer more features and power ratings than what is needed by the average embedded electronics design engineer. As a result, there have been several electronics design engineers and hobbyists who have chosen the more cost effective and more rewarding route of designing their own DIY DC electronic loads. One design that started the trend is demonstrated by Dave Jones in episode 102 of his EEVblog on YouTube. In this video, Jones explained the circuitry of the electronic load circuit operating in constant current mode and suggested some improvements that can be made to the circuit. Many electronics hobbyists replicated and improved upon Jones' design in various ways and published their design files over the Internet as part of the open source hardware movement.

This thesis documents the design of an open source DIY DC electronic load called the FreeDum Load. The FreeDum Load aims to aggregate some of the good features of the previous DIY designs while making improvements to areas that were lacking. In addition, the FreeDum Load is designed to be a self-sufficient bench top device to have the form factor of commercial electronic loads. This feature is currently lacking in existing DIY designs as the majority of them are bare PCB boards. The FreeDum Load features a streamlined aluminum enclosure and runs off of AC power. It is programmable and the peripherals on board are controlled by a Freescale Kinetis K20 microcontroller with a Cortex M4 core. Peripherals include a 3.2 inch back-lit LCD touchscreen display, a quadrature encoder with RGB LED feedback, and a fan controller with DC fans for cooling. In addition, some firmware has been written for

the Kinetis K20 microcontroller to drive the LCD, quadrature encoder, and the fan controller. The firmware proved to be challenging as existing libraries and sample code for the Kinetis K20 microcontroller were non-existent in the open source community. Freescale, the manufacturer of the microcontroller, offered some support in its CodeWarrior IDE, but the IDE itself is proprietary and has a high licensing cost.

As a result of this project, the first hardware revision of the FreeDum Load has been completed and a prototype unit has been constructed. A second revision is needed to correct some of the hardware design mistakes made. Major changes would include starting with a completely new microcontroller that would be easier to program with an open source IDE, adding a heatsink to the device, and cutting down costs of the components so the FreeDum load can be made much cheaper than commercial units. Overall, this thesis documents the knowledge gained from the full prototype design life cycle including knowledge of analog circuit design, PCB layout, soldering, PCB fabrication, Cortex M4 firmware development, and mechanical construction.

Chapter 1

INTRODUCTION

1.1 Background

An electronic load is a test instrument that is used to simulate loading on an electronic circuit or device. It is particularly useful for gaging the performance of power sources such as power supplies, batteries, photovotaic cells, and generators. The instrument simulates the loading of a power source by acting as a current sink that can be set and varied electronically. It is a substitute to the conventional way of connecting the tested circuit to various fixed value load resistors in order to produce various load currents. Simple electronic loads are controlled using a potentiometer for adjusting the load. More advanced electronic loads are programmable stand-alone devices that are controlled via a microcontroller or computer. They are capable of performing more measurements and can operate in various modes such as constant current, voltage, resistance, or power modes. There are both AC and DC electronic loads. In its most basic form, the electronic load circuit can even be integrated into a more complex circuit design when a variable current sink is required. Overall, the electronic load is an invaluable tool for electronics design and testing, especially in applications where it is important to define and validate the reliability and efficiency of the power sources used in a design.

1.2 Motivation

The goal of this project is to design a relatively low-cost programmable DC electronic load that is enclosed in a presentable and functional package. The target users for this device are embedded hardware designers and electronics hobbyists who work with embedded systems involving microcontroller circuits running at low DC

voltages. Currently, commercial DC electronic loads are available on the market, but these units tend to be large, more powerful than needed, and expensive. The lowest priced commercially available unit costs around $350 with higher-end models costing up to $3,000. Although these units are robust and useful in industrial settings, their capabilities over-exceed the needs of the average embedded electronics design engineer. As a result, many electronics hobbyists and researchers do not have the budget to purchase such an expensive piece of specialized test instrument. Instead, they have decided to design their own do-it-yourself DC electronic loads.

Several open source, do-it-yourself, DC electronic load projects already exist in the public domain which are published on forums and blogs by electronics hobbyists and researchers. These previously published projects served as inspiration and motivation for the FreeDum Load, the name given to the project discussed in this paper. The goal is to improve upon these designs because many of the existing designs lack reproducibility or robustness.

For example, some of these designs incorporate and reuse existing "scrap" components that are found around the designer's work bench. Fiscally, these designs work well for the individual designer's budget because they use materials which the individual already possesses. However, the use of scrap parts that are on hand could limit the functionality and specifications of the design. Furthermore, the design might not be easily reproducible by others who do not have the components already on hand. Instead, others who wish to reproduce the design would have to acquire the parts used. This would mean extra costs that were not factored into the original design. It is also possible that a component used in the design is difficult to acquire because it is no longer manufactured or is an OEM only component salvaged from another circuit. Another example would include a published design that does come with a bill of materials consisting of off-the-shelve parts and even design files for easy reproducibility, but the design itself lacks robustness. It could be that the design is a bare-board circuit or lacks a functional or "hacked" enclosure to contain the project. Often, crude holes are cut out of makeshift enclosures to fit a component, or a prototyping board is laid

around a mess of wires. The problem with projects like these is that they lack the transportability of commercial units.

This project hopes to eliminate the cost of commercial DC electronic loads and the lack of reproducibility and robustness found in many existing DIY designs. The goal is to design a device that looks streamlined and marketable like commercial devices but also reproducible by the DIY electronics hobbyist community. Although many design principles and best practices were adapted from existing electronic load projects, novel electronic components were specifically selected and purchased to best meet the requirements and budget for this project. The components used are readily available through major electronic component retailers such as Digikey or Mouser. This allows anyone to reproduce the design with ease. Details regarding the project will be made available to the public as open source hardware, which means all design files and source code will be published on the Internet so anyone may reproduce or improve upon the existing work free of royalties. This is in the open hardware spirit of the existing designs that have already been published. Some of these preceding open source projects will be discussed in the next chapter.

Chapter 2

EXISTING MODELS AND DESIGNS

2.1 Commercially Available Models

Figure 2.1: An Array brand electronic load (Array, n.d.).

Most commercial models are capable of testing both AC and DC, but this paper will be only discussing DC electronic loads. Several commercial DC electronic loads currently exist on the market. The form factor of these models are similar in appearance to the typical brick-like power supply. They have control dials and buttons to adjust various settings, and plus and minus terminals for the input. They also have a digital display showing ideal and actual settings measured by the device. Some can be

controlled programmatically via a PC and measurements can be stored. Most importantly, commercial electronic loads have built in protective circuitry to protect against over current, over voltage, over temperature, and reverse polarity.

Table 2.1 lists the prices and features of four different brands of the commercially available models of DC electronic loads. The cost of these units range from several hundred dollars up to several thousand. The unit costing the least is the TP3710A from Tekpower at $399.99. It is interesting to note that Tekpower devices are the same as those offered by ARRAY but sold under a different brand and at a lower price. Despite this, $399 is still a hefty fee to pay for an electronic load. Embedded circuit engineers might not require the full range of current and power ratings offered by commercial units so it might be viable to try to design a cheaper alternative.

Brand	**Model**	**Cost**	**Max Input Ratings**			**Data Link**
			Voltage	**Current**	**Power**	
ARRAY	3720A	$695	80V	30A	250W	RS232, USB, GPIB
ARRAY	3721A	$695	80V	40A	400W	RS232, USB, GPIB
ARRAY	3722A	$695	200V	20A	200W	RS232, USB, GPIB
ARRAY	3723A	$769	200V	30A	350W	RS232, USB, GPIB
ARRAY	3710A	$349	360V	30A	150W	RS232, RS485, USB
ARRAY	3711A	$499	360V	30A	300W	RS232, RS485, USB
BK Precision	8512	$2,595	500V	30A	600W	RS232, RS485, USB
BK Precision	8500	$1,095	120V	30A	300W	RS232, RS485, USB
BK Precision	8540	$525	60V	30A	150W	None
Tekpower	TP3710A	$399.99	360V	30A	150W	RS232, RS485, USB
Tekpower	TP3711A	$499.99	360V	30A	300W	RS232, RS485, USB
Tekpower	TP3723A	$999.95	360V	30A	350W	RS232, RS485, USB
Agilent	6060B	$2,910	60V	60A	300W	GPIB

Table 2.1: A comparison of prices and specifications of various commercial DC electronic loads.

2.2 Open Source Designs

Several open source hobbyists have shared and discussed their design of the electronic load over the Internet. These open source projects served as inspiration and

background research for the design presented in this paper so it is pertinent to describe some of these projects in the following section.

2.2.1 EEVblog Episode #102

The prototype that started the popularity of a do-it-yourself electronic load and a design which is referenced by many later electronic load projects is presented in a YouTube video made by Dave Jones of EEVblog. In this video, Jones (2010) goes over the basics of a constant current load circuit that he constructed using basic parts he "had lying around". Dave had a premade PCB board that he had designed for a battery capacity logger, which had the appropriate connections and footprints, so he populated this new electronic load circuit on top of this board.

For the circuit, he used a MTP3055 N-channel logic MOSFET and 1Ω power resistor fed by a LM324 op-amp to make an adjustable constant current load based an input voltage (Jones, 2010). The mechanics of how the basic load circuit works will be discussed in section 3.1. This constant current load circuit is prefaced by a voltage divider and another voltage follower op-amp connected to a 10-turn potentiometer. The 10-turn pot allows for a fine-grain adjustment of the input voltage (0 to 5 volts) (Jones, 2010). This voltage is then divided in half after the voltage follower giving 0 to 2.5 volts. In a perfect situation, the circuit should be able to generate from 0 to 2.5 amps at the output, but in reality the output is about 0 to 1.25 amps (Jones, 2010). Jones (2010) explains this is due to the LM324 having "issues with it can't go to the supply rail" and not being able to drive the MOSFET to achieve a higher output voltage across the load resistor. He also explains that this issue can be solved with a precision, rail-to-rail, op-amp. This advice was used in the design of the FreeDum Load as one can see later in the description of the design. At the output of the circuit, Jones uses another spare part, a cx101 3-digit LCD panel meter, in common-ground configuration, at the output to display the set current. Another design choice Jones points out is that for the 1Ω shunt resistor, he chose to use ten 10Ω resistors in parallel. The reasons are discussed in subsection 3.2.1 as the FreeDum load adopts this design

advice, as well. To prevent overheating, Jones attaches a heatsink to the MOSFET to dissipate the heat that is going to be generated from the current that is going to be sunk.

Lastly, Jones (2010) discusses his original battery capacity logger board which included an intelligent microcontroller to control and log the current output. The PWM input from the microcontroller simulates the potentiometer and the output is inputted back to the microcontroller via an ADC. With some programming, this set up allows for generation of different operating modes such as constant current, constant power, constant resistance, or pulse loads (Jones, 2010).

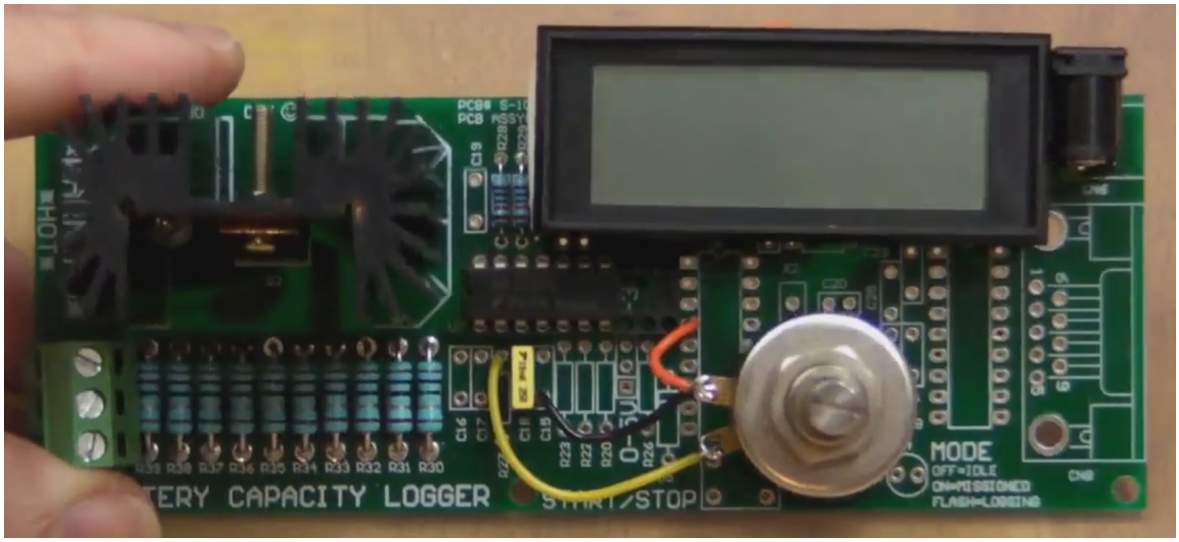


Figure 2.2: Screen shot of Dave Jones' adjustable electronic load design as seen from EEVblog episode #102 (Jones, 2010).

2.2.2 Martin Lorton's Electronic DC Load

Another YouTube blogger, Martin Lorton, known as mjlorton on YouTube, decided to create his own electronic load after a forum member on his website shared a modified design of Dave Jones' electronic load. Lorton wanted an electronic load for himself because he is a electronics hobbyist who works with solar power and electronic measurement equipment. In 2012, Lorton made a series of YouTube videos sharing the

design process of his electronic load. Lorton's design made some improvements to Dave Jones' circuit design based on his own power measurement needs, mostly by replacing the key components in Dave Jones' electronic load circuit with components that have the required specifications. For the N-channel MOSFET, which is the component responsible for sinking the majority of the current, Lorton chose the BUZ31L H-ND from Infineon Technologies, which is rated to handle maximum of 200V and 13.5A. He decided to put two of these MOSFETs in parallel in order for them to share the load and the heat dissipation (Lorton, 2012). For his load resistor, Lorton decided to use a single 1Ω power resistor rated at 50W. The input to the circuit is changed from one 10-turn pot to two 10-turn pots (50K and 5K) in series for finer control (Lorton, 2012). The op-amps in the circuit remained unchanged. An LM7806 linear regulator powers the op-amps. Overall, Lorton (2012) designed his electronic load to handle a max load of 30V and 2A or 60W.

Lorton (2013) took his design one step further and decided to put his completed circuit in an aluminum enclosure, and substituted the panel meter component in Jones' design with two backlit-LCD displays, which shows the input voltage and the current being sunk. The backlit-LCDs and the rest of the circuit require an external, dedicated power supply, so a separate DC input exists on the enclosure (Lorton, 2013). There are external connection terminals with access to the power supply inputs and current outputs so the current data can be logged via an multimeter. A large heatsink is mounted to the back of the enclosure and an L-shaped bracket is mounted to the heatsink to make sure the unit can stand upright (Lorton, 2013). Despite the improvements, the circuits in this design are still completely analog without any digital components. Lorton (2013) mentioned in his video that for his next revision of the device, he plans to add a low-voltage cut-off feature, to prevent over draining of lithium-ion batteries, and to replace the potentiometers with an Arduino control.

Figure 2.3: Screenshot from Martin Lorton's video showing the front and back of his Electronic DC Load (Lorton, 2013).

2.2.3 Re:load

In 2013, Nick Johnson of Arachnid Labs designed his own DC electronic load PCB which he named the Re:load. Even though Arachnid Labs does not explicitly reference or explain the origin of the circuit, the internal dummy load circuit used in the Re:load is very similar to that of the one presented by Dave Jones in his video. The MCP6002 low-power rail-to-rail op-amps in the circuit are powered by a LP2950 linear regulator. The MOSFET that Johnson (2014) chose was the BTS117. He specifically chose the BTS117 because it has built in ESD, over-current, over-voltage, and over-temperature protection. Like Lorton, Johnson chose to place one single power resistor instead of ten in parallel acting as the shunt resistor connected to the MOSFET capable of handling 3W. Current control is handled by a 10kΩ potentiometer although the potentiometer used seems to be cheaper and does not seem to provide as much granular control as those used by Jones and Lorton. Johnson laid out his whole circuit on a PCB on which there is a footprint for the heatsink to be attached. Re:load also does not include a dedicated power supply so an external power supply is needed to provide power to the linear regulator, provided that it is also not the supply to be tested by the electronic load. It is rated to handle 3.3 to 30V, 0-3A (0-5A for a bigger version with a bigger heatsink), 12W (20W for bigger version) (Johnson, 2014). Overall, Re:load is designed from the ground up with completely new components compared to Dave Jones' design but much simpler in functionality; it does not have an output display and relies completely on external lab equipment to monitor the output.[1] Arachnid Labs currently sells Re:load as a DIY bare-board kit priced at $15 ($20 for the bigger version). Source design files with the bill of materials are also provided should anyone wish to reproduce the board without buying the kit.

[1] In late 2014, Arachnid labs started work on a more advanced, programmable version of the Re:load, called Re:load Pro. It features a dedicated external enclosure, back-lit LCD display, and USB powered. It was released in early 2015 and available for sale for $150.

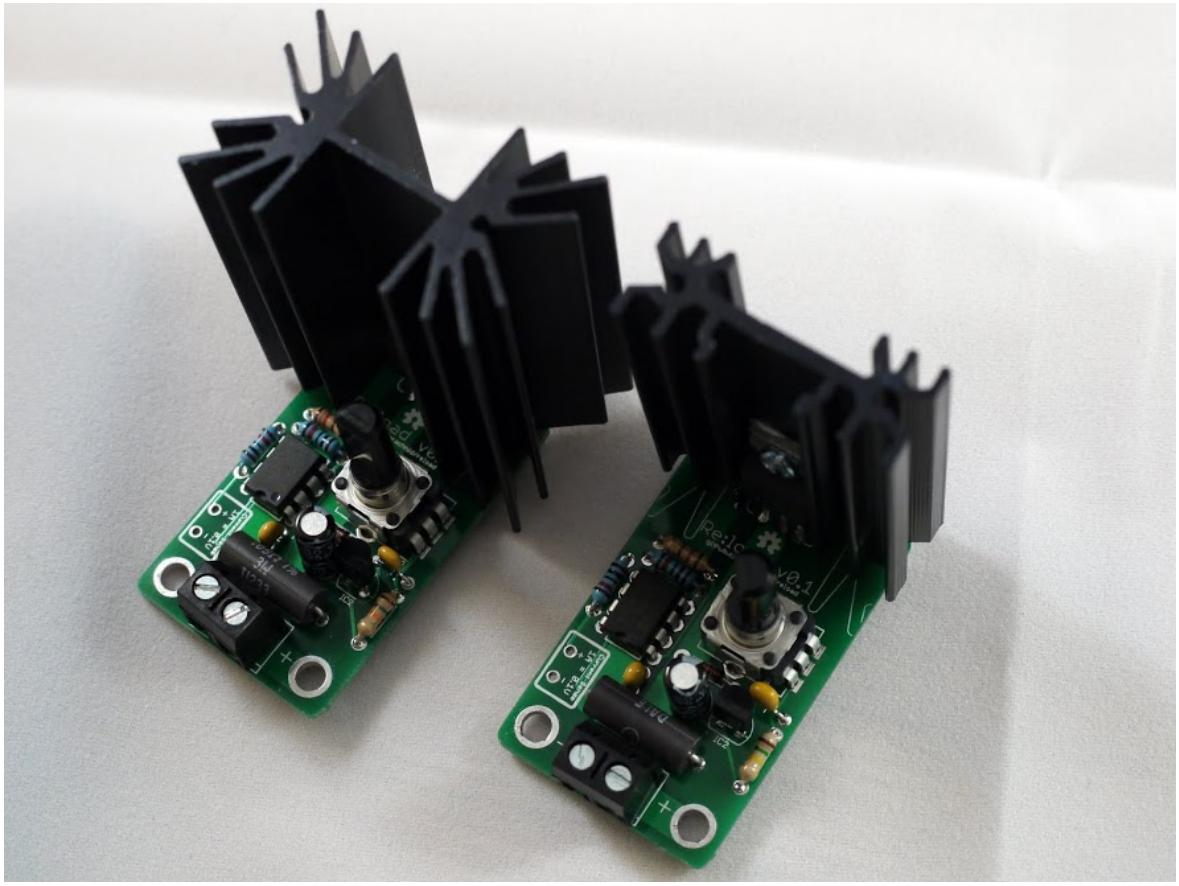


Figure 2.4: Photo of the two available versions of Re:load posted on the Arachnid Labs website (Johnson, 2014).

2.2.4 Other Designs

ITead Studio's C8051F Electronic Dummy Load Kit is a bare PCB board but it includes some advanced features: C8051F330 microcontroller, Nokia 5110 LCD that displays real time data, and constant current and constant voltage operating modes (Itead, 2010). It is capable of handling up to 2A. This kit currently sells on the ITead Studio's website for $35.

Figure 2.5: Photo of ITead Studio's electronic load board (Itead, 2010).

An electrical engineer, Lee Wiggins, by username of Wigman27, posted an extensive guide on the Instructables website explaining how he designed and constructed his programmable constant current power resistance dummy load. Like most of the

other designs mentioned thus far, his design was also inspired by Dave Jone's EEVblog video. It is, however, comparatively more sophisticated than the rest. To start off, the circuit is programmable and controlled by an Arduino. The use of the microcontroller enabled input to be done through a rotary encoder instead of a potentiometer, which most of the other designs used. The output is shown on a LCD backlit screen. Most of the components are surface mount and all laid out on a single PCB as opposed to through-hole soldered. Laying out all components on a single PCB allows for the reduction of wires and better cable management. Wiggins (2013) also had the idea of mounting a DC fan to the heatsink to cool the MOSFET. The only flaw to this design is that it lacks a dedicated power supply so the device itself will have to be powered off of an external source (Wiggins, 2013). Wiggins currently sells his PCB for $10 and has his bill of materials published on Instructables so others can use his board to make their own electronic load. According to the comments section, others have successfully created their own electronic load through the use of Wigman's PCB board, although they did not completely replicate the rest of his design such as using the same aluminum case, heatsink, and internal wiring scheme (Wiggins, 2013).

2.3 Design Summary

The do-it-yourself DC electronic load designs listed in the previous sections all have areas where they excel and areas where they are lacking. Microcontroller input via a rotary encoder, back-lit LCD output, and an aluminum enclosure are the good features of some of these designs. One common feature that many of DIY designs lacked that commercial devices have is a dedicated supply to power the test circuit and the peripherals on board the device. This prevents the electronic load from being an appliance that can function without any additional equipment. Another problem is that in most of these designs, the heatsink on the board is exposed. It can cause accidental burns and in general takes away from the aesthetics of the packaging. The design of the FreeDum Load aims to aggregate some of the good features of the previous designs while making improvements to areas that are lacking.

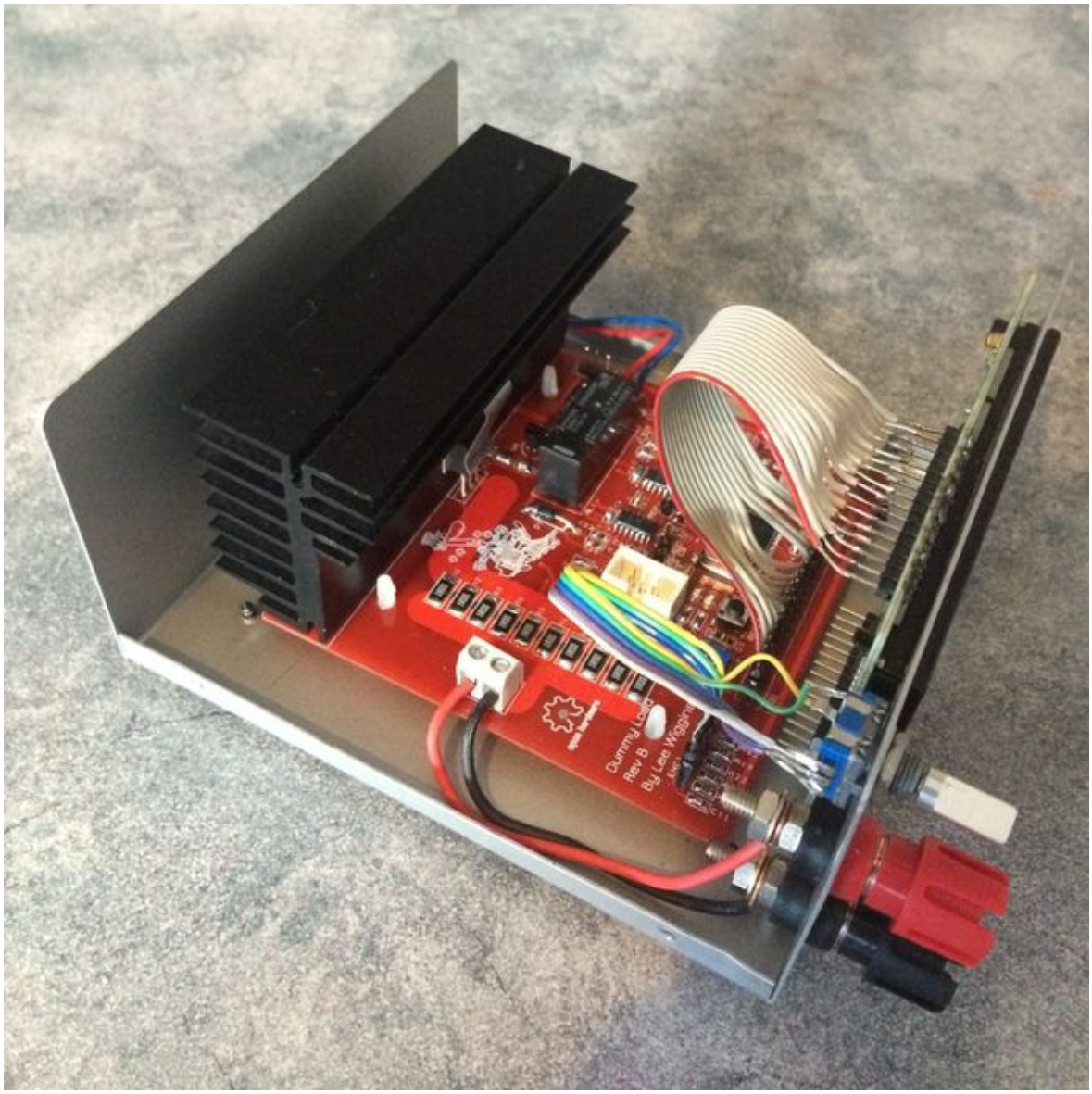


Figure 2.6: Photo of the internals of Wigman's electronic load as posted in his Instructables guide (Wiggins, 2013).

Chapter 3

FREEDUM LOAD HARDWARE DESIGN

3.1 The Basic Electronic Load Circuit

Like the designs discussed in the previous chapter, the circuit design of the FreeDum Load uses an extension of the basic electronic load circuit proposed by Dave Jones in his EEVblog episode. The electronic load circuit in its minimally functional form consists of an operational amplifier, an N-channel MOSFET, and a shunt resistor (Fig. 3.1). The basic functionality of this circuit acts as a voltage controlled current sink. The MOSFET and the resistor are arranged in high side configuration with the drain terminal of the MOSFET connected to the power source to be tested, the source of the MOSFET connected to the shunt resistor, and the resistor connected to common ground. In this configuration, the MOSFET acts as a variable resistor in series with the shunt resistor. Together, these two components act as the current sink.

The output of the op-amp in the circuit is connected to the gate of the MOSFET, which controls the resistance of the MOSFET and thus the amount of current that flows though the MOSFET and the shunt resistor. The voltage input of the circuit that controls the amount of current to be sunk is connected to the positive terminal of the op-amp. The voltage relative to ground at the junction of the load resistor and the drain of the MOSFET is connected to the negative terminal of the op-amp. Due to the operational characteristics of the op-amp, the op-amp will attempt to drive the circuit at its output so the voltages at its positive and negative input terminals match. In other words, the voltage fed into the input at the positive terminal will result at the negative terminal. Using a load resistor of 1Ω and an input voltage of X volts at the positive input of the op-amp, the op-amp will cause the voltage difference across

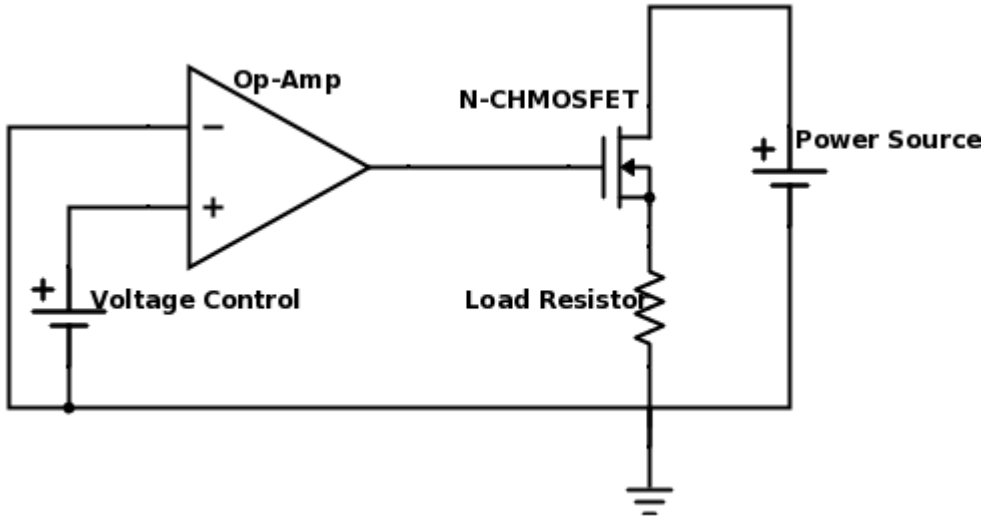


Figure 3.1: The schematic of the basic circuit.

the 1Ω resistor to also be X volts. Using Ohm's Law, X volts dropped over 1Ω results in X amps of current flowing through the resistor. Assuming the power source to be tested (labeled "Power Source" in Fig. 3.1) connected to this circuit is providing more than X volts, the voltage in excess of X volts will be dropped across the MOSFET.

3.2 Expansion of the Basic Circuit

The previous section describes the most basic form of the electronic load circuit for the purpose of explaining the operation of such a circuit and how it functions as a current sink. This basic circuit can be added onto more complicated circuit designs whenever a current sink is required in context. However, the goal of this design project was not to incorporate this circuit into an existing circuit but to design a stand-alone bench top test device that is capable of acting as a variable load to any power source that needs to be tested. Rather, the aim is to have an electronic load that is not only programmable and customizable but is also self-contained in its own packaging. Some modifications and several components were added to the basic circuit to allow for greater versatility, functionality, and also programmability. Each modification or

addition to the basic circuit that is found in the resulting design is described in more detail in the following subsections along the reasons for the design decision.

3.2.1 Parallel Load Resistors

As suggested by Dave Jones in his video blog, the single 1Ω resistor in the basic electronic load circuit is replaced by the equivalent circuit of ten 10Ω resistors in parallel. The tolerance of the resistors chosen is 1%. The reason behind this configuration is threefold. First, it is much more cost effective to obtain ten 1% 10Ω 2W resistors than one 1% 1Ω 20W resistor. Second, even though the equivalent resistance is the same, the actual resistance of the equivalent circuit would be closer to the specification due to the averaging of the manufacturing errors present in the ten resistors (Jones, 2010). Lastly, having ten resistors in a parallel configuration allows the circuit to sink more current and more power than one resistor by itself. Sharing the current amongst the ten resistors will allow for better heat dissipation. The specific power resistors used in the circuit are WHC10RFET manufactured by Ohmite. They are rated to handle 2W each so ten resistors together can handle a maximum 20W of power.

3.2.2 Op-Amp

The op-amp used in the circuit is the ADA4528 made by Analog Devices. It is a precision rail-to-rail op-amp with a low offset voltage. This op-amp is chosen in order to use the full range of rail to rail voltages available to drive the MOSFET. Figure 3.2 shows that the op-amp is powered off of a single supply of 5V. The positive terminal of the op-amp is driven by the output from the digital to analog converter of the microcontroller used in the FreeDum Load's design.

3.2.3 Op-Amp Buffer Circuit

The ADA428-2 contains two op-amps and the FreeDum Load utilizes both. The most basic electronic load circuit only require one op-amp to control the MOSFET. The second op-amp has been wired in a feedback buffer configuration. The input of the buffer is tied to the output of the electronic load circuit, at the junction between the

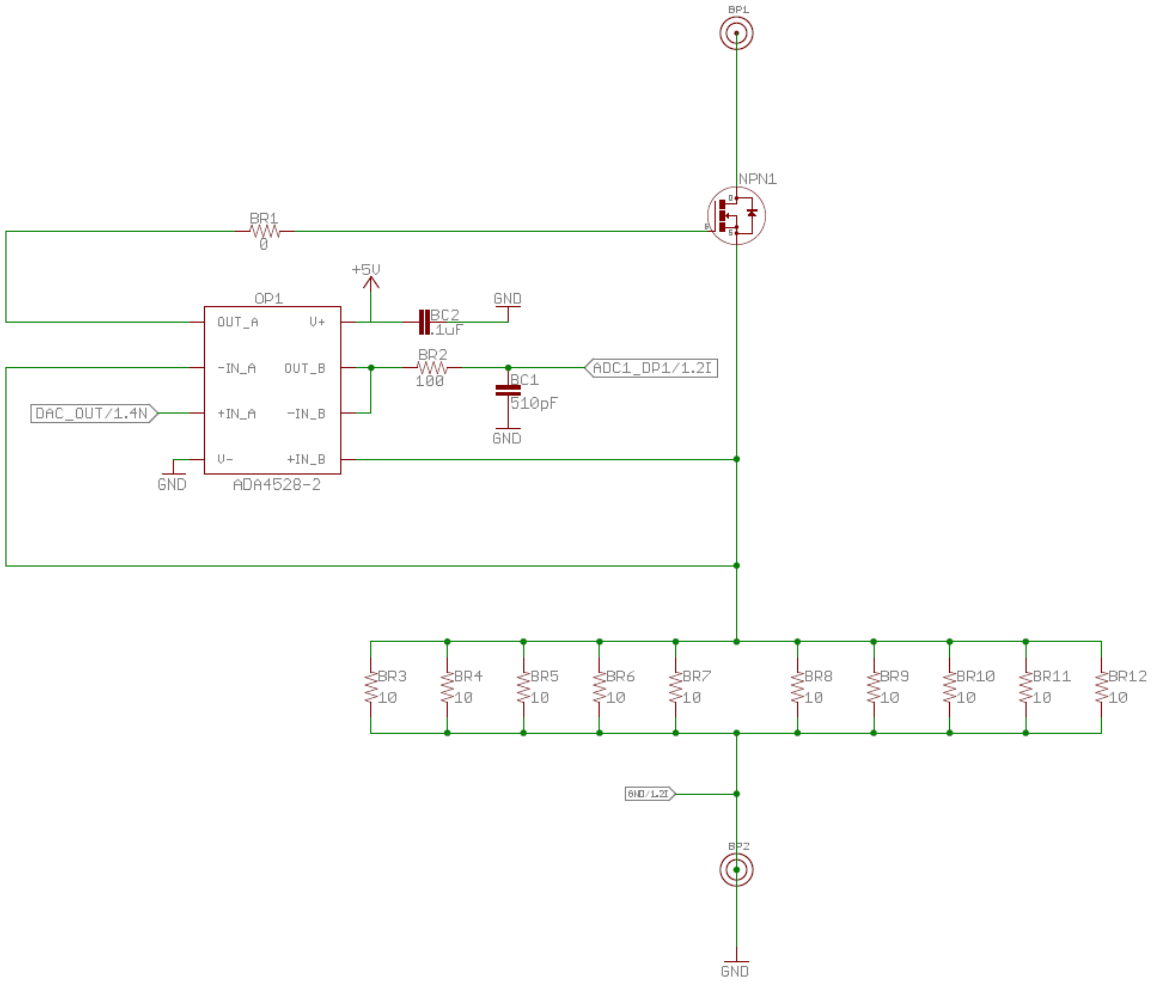


Figure 3.2: The schematic of the expanded circuit.

source of the MOSFET and the parallel resistors. The output of the buffer is connected to the analog to digital converter on the microcontroller to be read back via feedback. Since the supply rail to the op-amp is from 0 to 5V, the buffer is only able to output a maximum of 5V. This should be no problem as the load circuit is not designed to sink more than 3A making the theoretical voltage at the output to be 3V max.

3.3 Peripherals

3.3.1 Microcontroller

In order to control the current being dropped across the load, some type of voltage control needs to be implemented at the positive terminal of the op-amp circuit. In Dave Jones' (2010) simple electronic load design, he used a 10-turn pot. In order to make the FreeDum Load's design more sophisticated, a microcontroller was employed. A microcontroller allows greater functionality to be added to the circuit via software controls. The microcontroller can be made to drive the circuit via PWM or a digital to analog converter port. The microcontroller can also be used to log data via an analog to digital converter, and it can drive additional electronic peripherals in the circuit.

The microcontroller used in this project is the MK20DX256ZVLQ. It is part of Freescale's Kinetis K20 series of MCUs built on the ARM Cortex-M4 core. The Kinetis K20 family of microcontroller was chosen because it had all the peripheral buses and ports needed for the project already built-in. The MK20DX256ZVLQ features a 144-pin LQFP package with a 100 MHz core operating frequency, 256 KB of flash, and 64KB internal RAM. The 100MHz operating frequency would allow the microcontroller to quickly drive a LCD to display graphics, read input from its sensors, and sample data from the ADC at a higher frequency. Sufficient flash memory and RAM are needed to store LCD graphics and any data points that are logged. The features that stood out regarding this microcontroller was that it has both a 16-bit analog to digital converter and a 12-bit digital to analog converter built into the chip. These are rather uncommon features not found in other microcontrollers at the time. Having all the

desired peripherals in one part makes it more cost efficient and also more space efficient for the PCB layout.

Figure 3.3.1 shows the schematic of the microcontroller. 0.1μF decoupling capacitors were put between all the power and ground pins of the microcontroller. A reset switch labeled SW3 is tied to the reset pin. An 8 MHz external crystal is attached to the microcontroller using the proper connection circuit as documented in the microcontroller's reference manual. 22pF capacitors were used as part of the crystal connection circuit because that value is more commonly used and close enough to the calculated value of 26pf, assuming C_{stray} of 5pF and C_L of 18pF according to the crystal's data sheet (Abracon, 2011).

3.3.2 LCD Display

An LCD display is employed to display information regarding user input, the mode of operation, and circuit output. The LCD in the FreeDum Load is a third party 3.2 inch touch LCD with a 320 by 240 resolution. It has a SSD1289 LCD controller and a XPT2046 touch screen controller. The screen is backlit. The LCD controller has a 16 bit data line input. The touch screen interface is controlled via SPI. It is important to note that the LCD includes a 5V to 3.3V voltage regulator, the 3.3V which is in turn used to power the microcontroller. Without the LCD board, the microcontroller would not work since there are no additional voltage regulators found on the main board. The decision to use the voltage regulator on the LCD was due to space efficiency. The LCD break out board snaps onto the main board via a row of headers.

3.3.3 The USB Port

Since the MK20DX256ZVLQ supports USB peripherals, a USB port is included in the design. The USB port can be used to download data from the device or be used to configure the device. The USB port lay out includes 2 line ESD protection. 33Ω termination resistors are placed on the D+ and D- lines. The USB lines go to a 5 pin

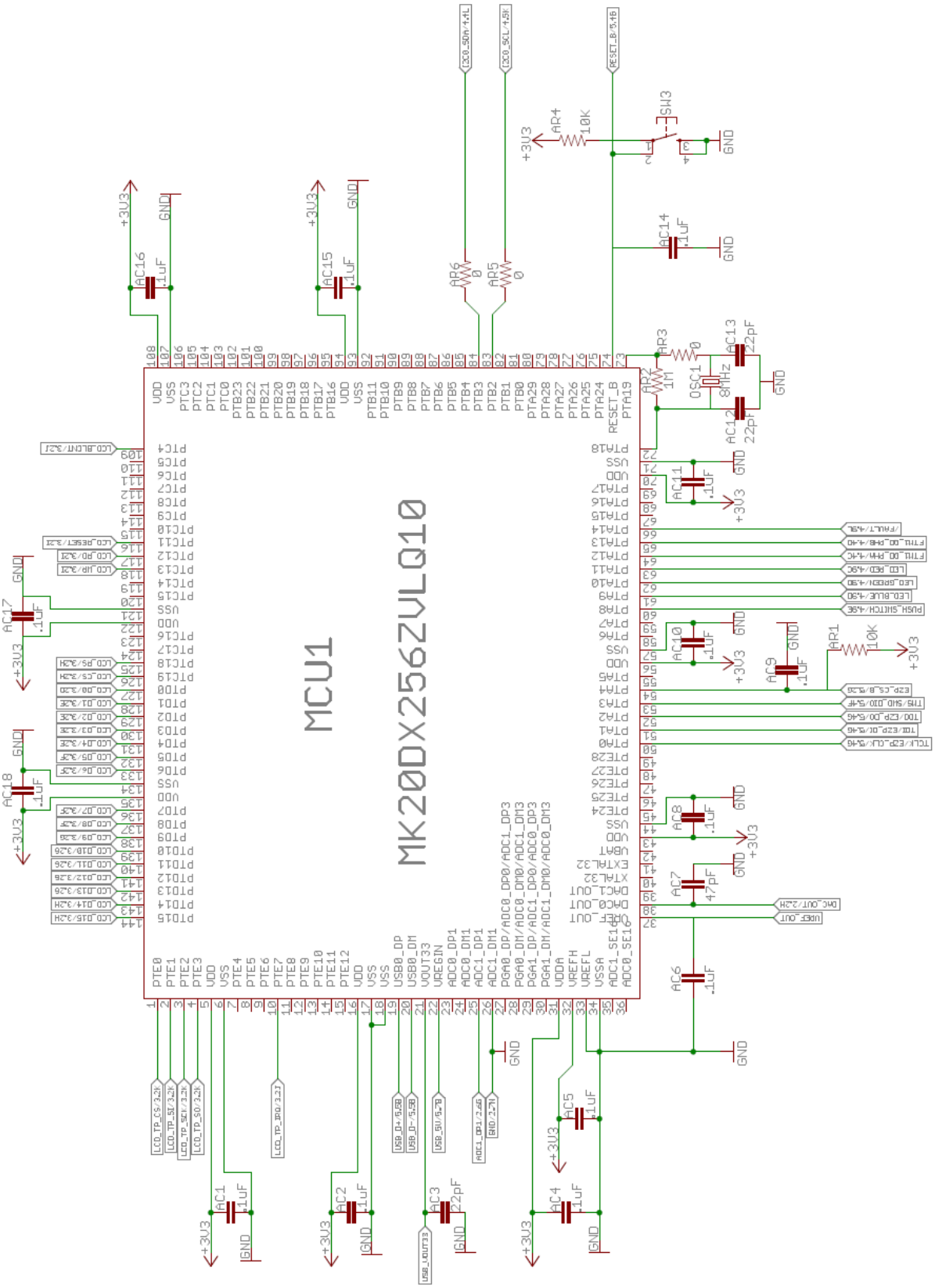


Figure 3.3: The schematic of the microcontroller.

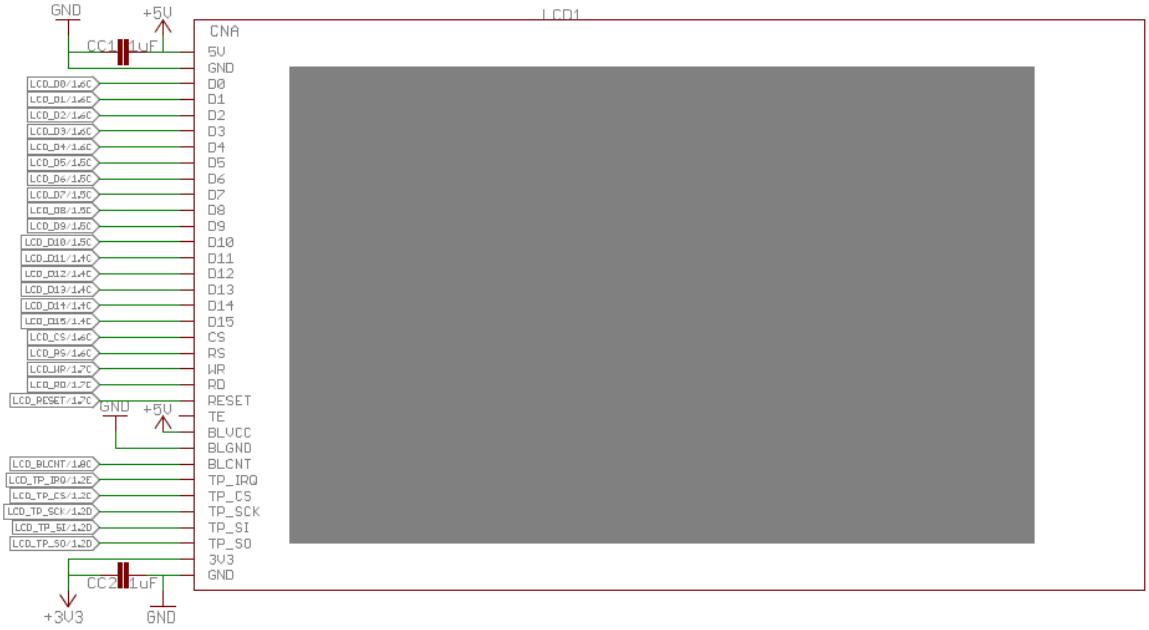


Figure 3.4: The schematic of the lcd screen.

connector on the board, which then connects to a USB-B style port that is mounted on the back panel of enclosure via a cable.

3.3.4 Controller Dial

The controller dial on the FreeDum Load is a PEL12T incremental rotary encoder with a RBG LED illuminated shaft that offers colored feedback. The shaft also acts as a push-switch. This dial would be used in conjunction with the touchscreen display to adjust the operating mode and settings of the device. Figure 3.6 shows the schematic of the rotary encoder. On the left side are terminals A, B, and C with A and B being the outputs of the quadrature that would tell the microcontroller of the relative position and direction of rotation of the shaft. The schematic includes a filtering circuit as suggested by the manufacturer's data sheet (Bourns, 2014). Terminals 1 to 4 are the microcontroller inputs to the RGB LEDs and the push switch. Current limiting resistors with values depending on the different color LED forward voltages

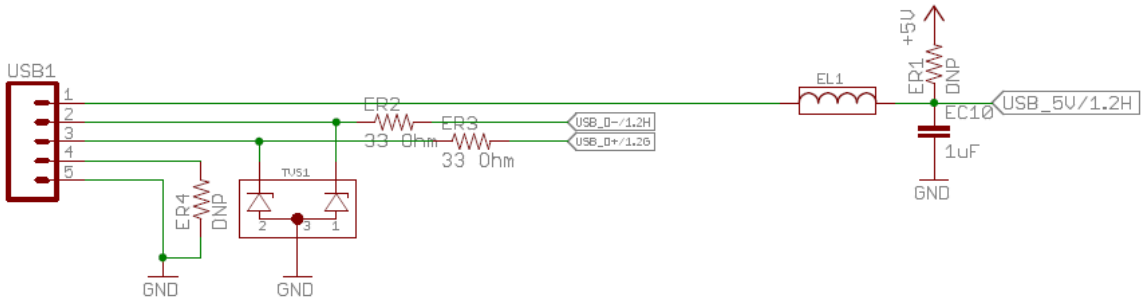


Figure 3.5: The schematic of the usb port.

are attached to the LED inputs and a pull down resistor is attached to the push switch input.

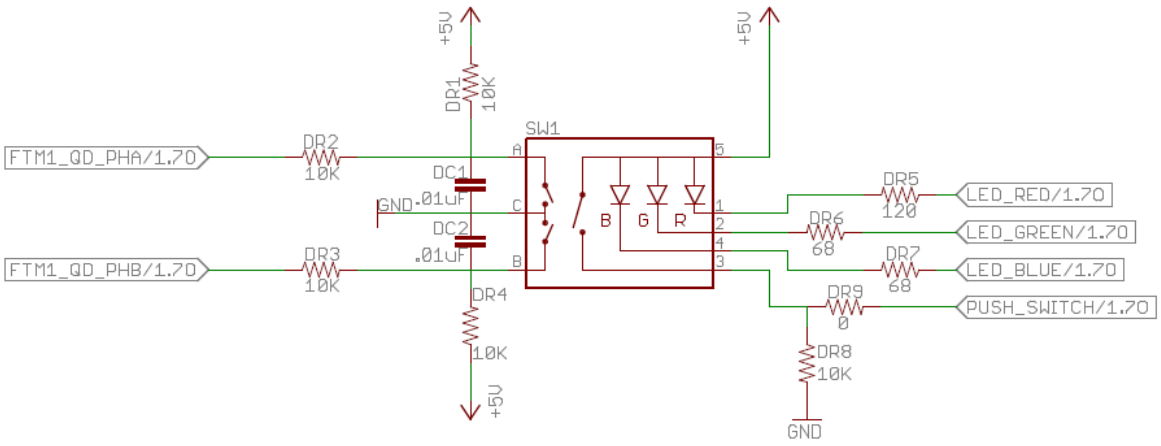


Figure 3.6: The schematic of the rotary encoder.

3.3.5 External Power Supply

One major design consideration for the FreeDum Load was that its circuit components such as the microcontroller and the op-amp that drives the MOSFET should be powered off of a dedicated built-in power supply. As mentioned previously, many

other open source electronic loads require power from an external supply. This external power supply should not be the same as the power source being tested. Omission of a dedicated supply is likely done for simplicity's sake but adds the hassle of requiring extra equipment to power the electronic load. To make the FreeDum Load a self-sufficient device, all components on the board are powered by a AC to DC converter that supplies 12V. The specific AC to DC converter model is the LS25-12 made by TDK-Lambda Americas Inc. The 12V output from the AC to DC converter is fed to the 5 voltage linear regulator, which drives the op-amps in the circuit and the LCD. The connection from the AC to DC converter to the voltage regulator is controlled by a push switch that determines whether power is delivered to the device. The AC input of the converter is connected to a 3-prong AC port, which is found on the back of the FreeDum Load enclosure.

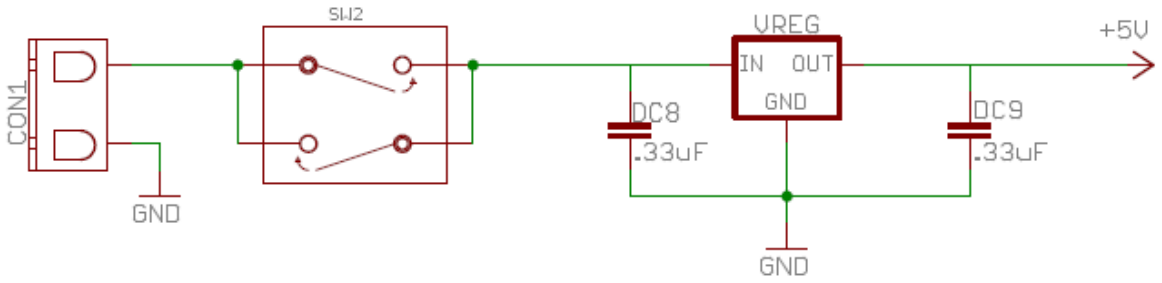


Figure 3.7: The schematic of the power supply and voltage regulator.

3.3.6 Fan Controller and Fans

To dissipate the heat dropped across the transistor, the FreeDum Load includes brushless DC fans and a PWM fan speed controller. The fan controller used is the TC655. Temperature data from a thermistor connected to CON4 is fed to Vin of the fan controller. The PWM output at V_{out} is adjusted between 30% and 100% based on the voltage at V_{in}. Sense 1/Sense 2 are used to monitor the actual RPM of the fans. Data can be read and fan speed can be controlled via the SMBus lines which are

connected to the microcontroller. The rest of the fan controller circuit (resistor and capacitor values and configurations) are taken from the Typical Application Circuit from the TC655 data sheet (Microchip, 2014). The fans are 12V DC fans that are directly powered by the power supply.

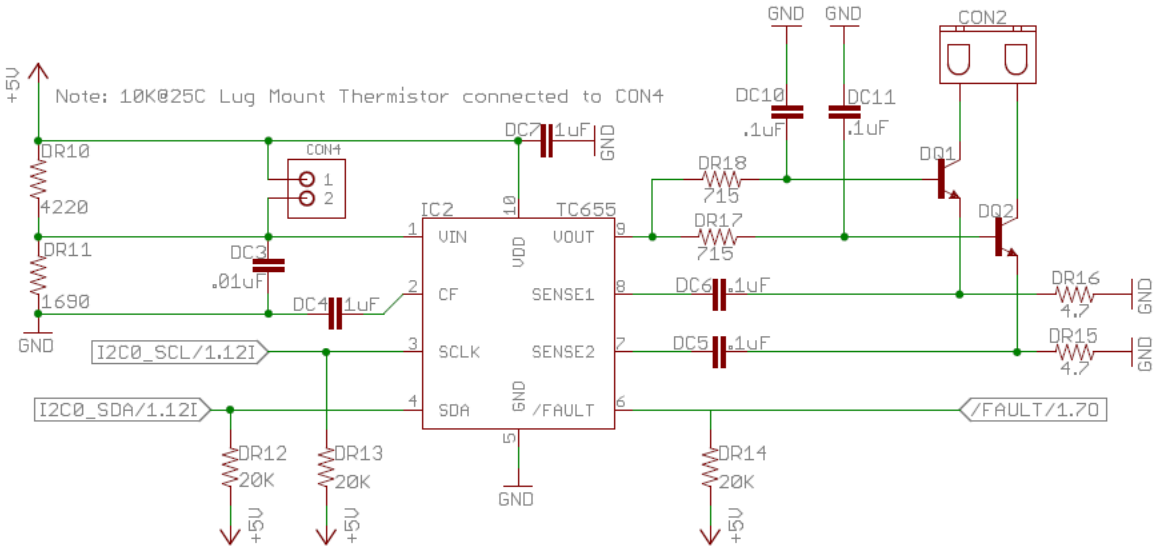


Figure 3.8: The schematic of fan controller circuit.

3.4 Manufacturing and Assembly

3.4.1 PCB Manufacturing

The schematic and PCB layout for the FreeDum board was done in Eagle Layout Editor 6.5. All of the circuits mention above were laid out on a double-sided PCB. The PCB layout is shown below. The PCB was fabricated by OSH Park. All components were soldered on by hand after fabrication. After the headers have been soldered on, the LCD break out board is snapped onto the main PCB. Photos of the completed PCB with and without the LCD break out board attached can be seen in Figures 3.10 and 3.11.

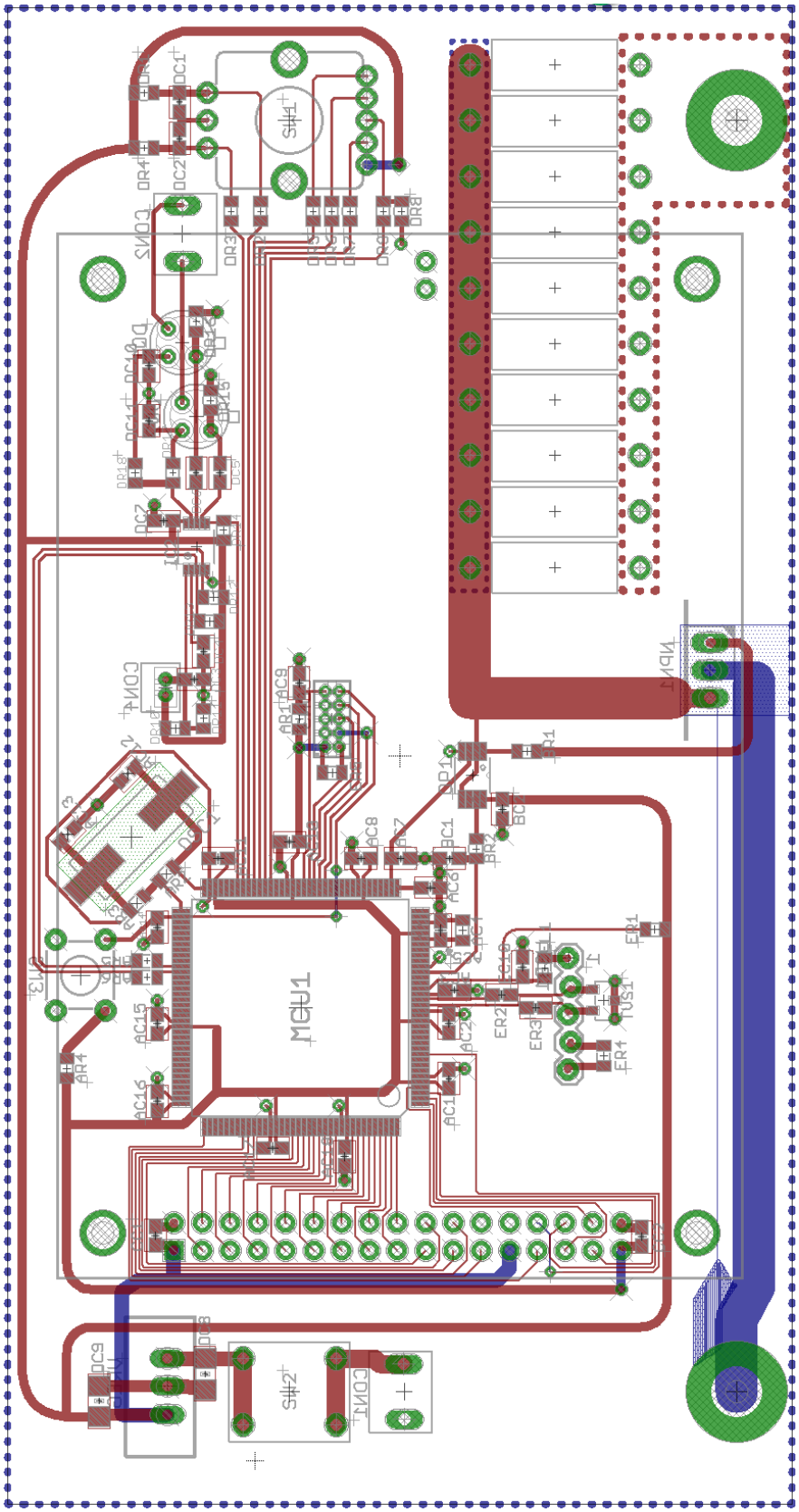


Figure 3.9: The PCB layout.

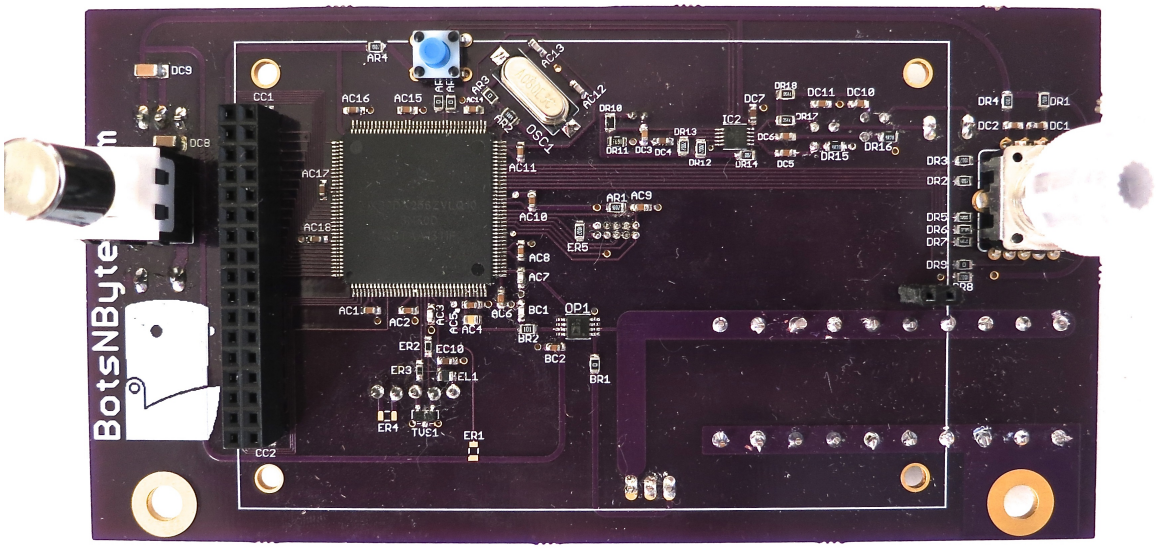


Figure 3.10: The front of the board.

Figure 3.11: The front of the board with LCD attached.

Figure 3.12: The back of the board.

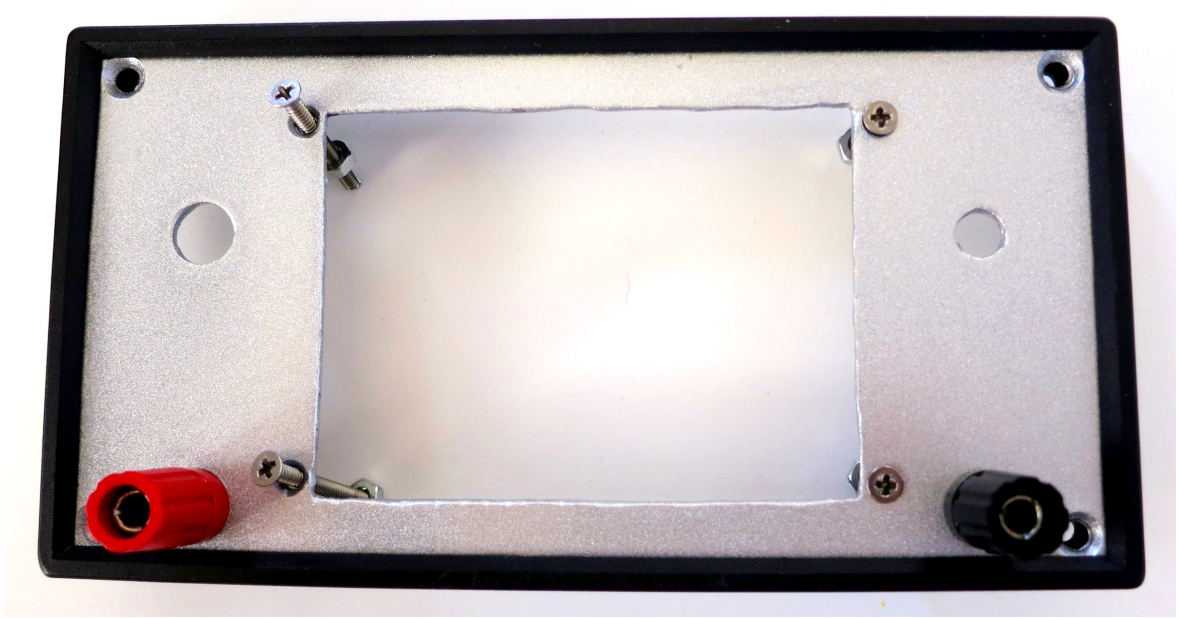


Figure 3.13: The front panel of the enclosure.

3.4.2 Enclosure

The FreeDum Load has a dedicated enclosure to make it a streamlined bench top device. It is a one piece extruded aluminum housing with removable aluminum end panels and internal PCB guides for easy of slotting of PCBs. The enclosure has external dimensions of 3.18in x 6.14in x 5.38in. The end panel of the the enclosure were modified and used as the front and the back side of the electronic load device. For the front panel, a rectangle the size of the LCD was cut out so that the screen sits flush with the panel (Fig. 3.13). The rest of the main board sits behind the front panel. Circular holes are cut out for the two push button switches and the input terminals. For the back panel, holes for the USB port, two 1.5in DC fans, and the power adapter port were cut out (Fig. 3.15). The external power supply was mounted to a aluminum panel and slotted into one of the enclosures slots (Fig. 3.14). The actual PCB of the FreeDum Load sits perpendicular to the aluminum panel and flush with the front panel of the enclosure. The PCB board is attached to the aluminum panel by the metal tab on the MOSFET. The intended idea is that the MOSFET would dispensate heat through the aluminum panel and through out the case, which would then be cooled off by the fans.

Figure 3.14: The AC to DC converter slotted inside the case.

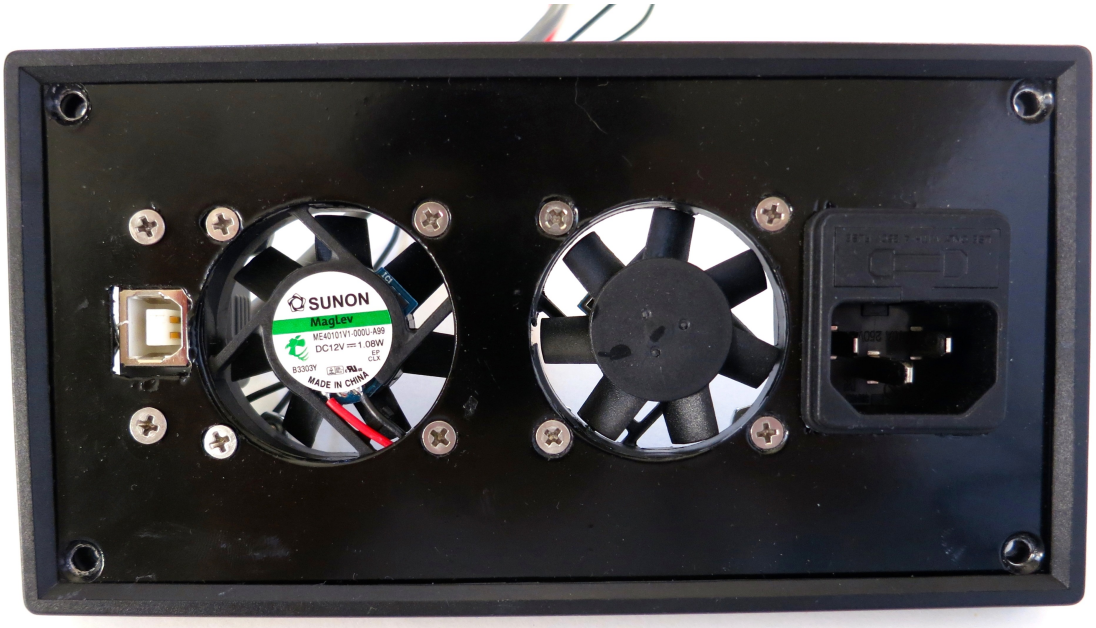


Figure 3.15: The back panel of the enclosure.

Chapter 4

FIRMWARE

Due to the scope and the time frame of the project, firmware for the Free-Dum Load has not been completed. Only some of the peripherals of the device were programmed to test whether the microcontroller and the accessories perform as they should. The following sections will outline the tools used to program the microcontroller and some of the the firmware written.

4.1 IDE and Programmer

In order to cut down the development costs of the project, many integrated development environments were evaluated. The first IDE used was the emIDE, a free IDE for embedded programming. Since there is no cost associated with this IDE, others who wish to reproduce the design of the FreeDum Load could download it and reuse the firmware. Unfortunately, the emIDE had limited support for the Freescale's Kinetis K20 series of microcontrollers. One of the major inconveniences from using the emIDE was its lack of customized start up code for the Kinetis K20. The start up code is the code run at the power on/reset of the microcontroller which is responsible for copying the code from ROM to RAM, clearing the BSS, and setting up the interrupt vector table of the microcontroller. From the Embedded Application Wizard, the emIDE was only capable of generating a very generic startup code for the Cortex M4 device. Furthermore, this startup code was written in assembly, making it difficult to update and understand. Additional startup code written in C was found for the Kinetis chip on the Freescale website but it was unclear how this code was to be integrated with the assembly already generated by emIDE.

Eventually, emIDE was given up and the Freescale CodeWarrior Development Studio was looked at as another option for development. One advantage to using CodeWarrior is that CodeWarrior is the official IDE for Freescale so it provides built-in support libraries for Freescale's microcontrollers. Despite this, CodeWarrior was not considered as an option because Freescale only offers the software on a 30-day evaluation license. After the trial expires, the license would cost $400 for the cheapest annual license. Attempts were made to "export" the support source code generated by CodeWarrior and "import" it into emIDE but it proved to be a time-consuming and difficult process as the project files generated by the CodeWarrior wizard were rather complicated.

In the end, the majority of the firmware for this project was developed using Rowley CrossWorks IDE for ARM. It proved easy to use and generated the proper startup code for the Kinetis K20 chips. However, the code size was restricted to 16KB for the trial version of the software. To break the restriction, a personal license for CrossWorks had to be purchased which cost $100.

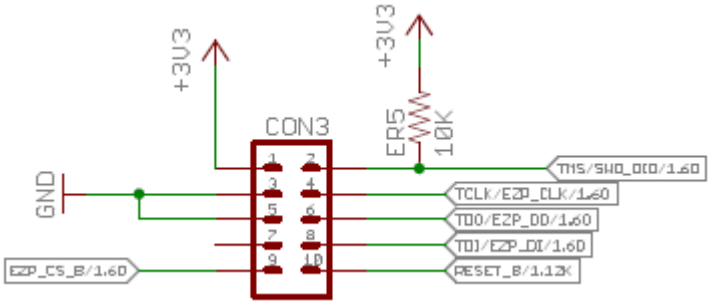


Figure 4.1: The jtag connection to the microcontroller.

The Seggar J-Link EDU debug probe was used with the Olimex ARM-JTAG-20-10 adapter to program and debug the firmware on the microcontroller. The debugger is connected via USB to the PC and connected to the FreeDum Load's PCB board via the 10-pin JTAG connection at Connector 3 (Fig. 4.1). The J-Link EDU costs $60 and the adapter costs about $5. This debugger combination worked with all three IDEs

mention above. Another debugger used was the ARM-USB-TINY-H by Olimex. This debugger is a cheaper alternative to the J-Link EDU (costing about $40), but did not seem to work with emIDE. The J-Link EDU is the more reliable option.

4.2 Source Code

One of the first tasks set out was to program the LCD display since it can be used to give I/O feedback to help test the rest of the peripherals. Fortunately, an electronics hobbyist by the name of Huub Smeitink shared a library that he wrote for the SSD1289. The library was written for a PIC microcontroller, so ports referenced in Huub's code had to be converted to the ports for the K20 microcontroller. LCD controller code for the MK20DX256ZVLQ can be found in Appendix A.1. Huub also packaged a small C# program along with his support library, which can be used to generate a RGB value array given a bitmap input. This array can then be plotted on the LCD using the LCD_Image() function allowing easy display of images on the screen. Another part of the firmware that was done was the driver for the TC655 fan controllers using SMBus. The code allows the SMBus to drive the DC fans in the FreeDum Load to run at various speeds. The SMBus code can be found in Appendix A.2. Lastly, code was written for the quadrature encoder and this is included in the appendix as well. Microcontroller port set up code is not included in the appendix for brevity's sake.

Chapter 5

DESIGN PITFALLS

Several mistakes were made during the design of the electronic load. This chapter will discuss the design pitfalls and any future changes that can be made for revision two of the FreeDum Load.

5.1 Op-Amp Rail

One design flaw in the FreeDum load circuit is that the rail voltage driving the op-amp is potentially not high enough. Currently the whole electronic load circuit is powered off of 5V DC from the linear regulator, including both op-amps in the circuit. This means that the maximum output at both op-amps would be 5V. The voltage output is especially important for the op-amp that is driving the N-channel MOSFET because V_{GS} at the gate of the MOSFET needs to be at a certain threshold in order to allow current to pass through the MOSFET. Usually N-channel MOSFETs are used in a low-side configuration in a design and V_G=V_{GS}, but in the basic electronic load circuit, the source of the MOSFET is not connected to ground but rather to the shunt resistor. This means that when the voltage at the resistor is increased, V_G at the gate must also increase to compensate for the increase in V_S in order to maintain the same V_{GS}.

With the circuit in its current state, the maximum possible V_G is 5V, limited by the rail of the op-amp. Looking at NXP's (2011) data sheet for the BUK9504-40A, a V_{GS} of 2V will lead to a drain current of 10A at V_{DS} of 25V. This is an optimistic rating. In reality, the electronic load circuit in the FreeDum Load's design cannot even drop 10A because that would mean V_G would need to be 12V, 7V higher than the op-amps output voltage of 5V. Although the FreeDum load is only designed to sink

a max of 3A of current, it is unclear whether a 5V rail for the op-amp is sufficient to even turn on the MOSFET enough to allow 3A to pass through. The rail voltage and the maximum op-amp output were not considered, and the MOSFET was assumed to work and chosen based on the MOSFET's maximum power ratings. In the second revision of the FreeDum Load, the rail voltage driving the op-amp might have to be increased or another MOSFET with different transfer characteristics might have to be chosen.

5.2 Heatsink

Another design flaw was the lack of the inclusion of a proper heatsink to dissipate heat across the transistor. The lack of a heatsink further limits the amount of current that can be dropped by the transistor. The heatsink was initially eliminated from the FreeDum Load's design to keep the enclosure of the FreeDum load compact. It was assumed that the DC fans and the aluminum case itself were sufficient enough to dissipate the heat. This set up might have been too optimistic. The thermal resistance of the aluminum case is unknown, which makes it impossible to do thermal calculations and find out the safe operating temperatures for the MOSFET. As a result, due to fear of overheating the MOSFET, not much testing was done on the load circuit to see how much current it can handle. For revision two of the FreeDum Load, a heatsink will have to be included and a larger enclosure will have to be chosen to accommodate for the heatsink.

5.3 Microcontroller

In addition to hardware design errors, the firmware writing process would have been made easier if a different brand of microcontroller as used in the design. Although the Kinetis K20 microcontrollers were the latest and greatest at the time, they proved to be too complicated to program. Open source support for the microcontroller, especially the Cortex M4 processor, was nonexistent. It is often more efficient to borrow and build off of existing open source libraries and example projects. Although some support

libraries and guides were found for STM branded Cortex core microcontrollers released by the open source hardware community, it was unclear how these resources translated to the Kinetis series. In addition, it was rather inconvenient that CodeWarrior, the IDE that comes with the proper support libraries for the microcontroller, was not free to use. The high cost of the IDE is counterproductive to making a cost-effective DIY electronics project.

5.4 Design Cost

Finally, one of the goals of this project was to design an electronic load that would be cheaper to make than to buy a commercial unit. In table 2.1, the lowest costing unit is about $400. All parts and materials used in the FreeDum Load, including the LCD, enclosure, AC/DC converter, and the electronic components cost about $130 total. To fabricate the PCB, the cost would be about $23 per board. This material cost does not include the JTAG debugger needed to program the microcontroller on the board and the accompanying adapter, which cost an additional $65. Assuming once the firmware is completed and released to the public, the firmware can be loaded onto the microcontroller via the free trial version of CodeWarrior. This would bring the total hardware costs to about $223 should someone wish to reproduce the design. Although the total cost is lower than $400, this cost is still higher than expected. To make the design worthwhile to reproduce, the total cost would have to be around $100.

Chapter 6

CONCLUSION

Although some errors and mistakes were made in the design, a functional prototype for the FreeDum Load was produced as the result of this project. The prototype has complete packaging as intended and it is capable of running off of 120V AC power from a wall socket. The unit is capable of displaying graphics on the LCD and drive the fans based on the input on the rotary encoder. It is a functional stand-alone appliance and not a bare board PCB like so many of the projects that inspired the FreeDum Load.

Firmware for the ADC has been started although not complete. Revision two of the hardware is needed before any more firmware can be written. Revision two of the FreeDum Load would include amendments to the design pitfalls described in the previous chapter. Two major changes would include starting with a completely new microcontroller that would be easier to program and adding a heatsink to the device. Overall, many embedded electronic design skills were gained through out the design process including knowledge of analog circuit design principles, PCB layout, soldering, PCB fabrication, Cortex M4 firmware development, and mechanical construction. This project involved all the steps of a full prototype design life cycle from start to finish.

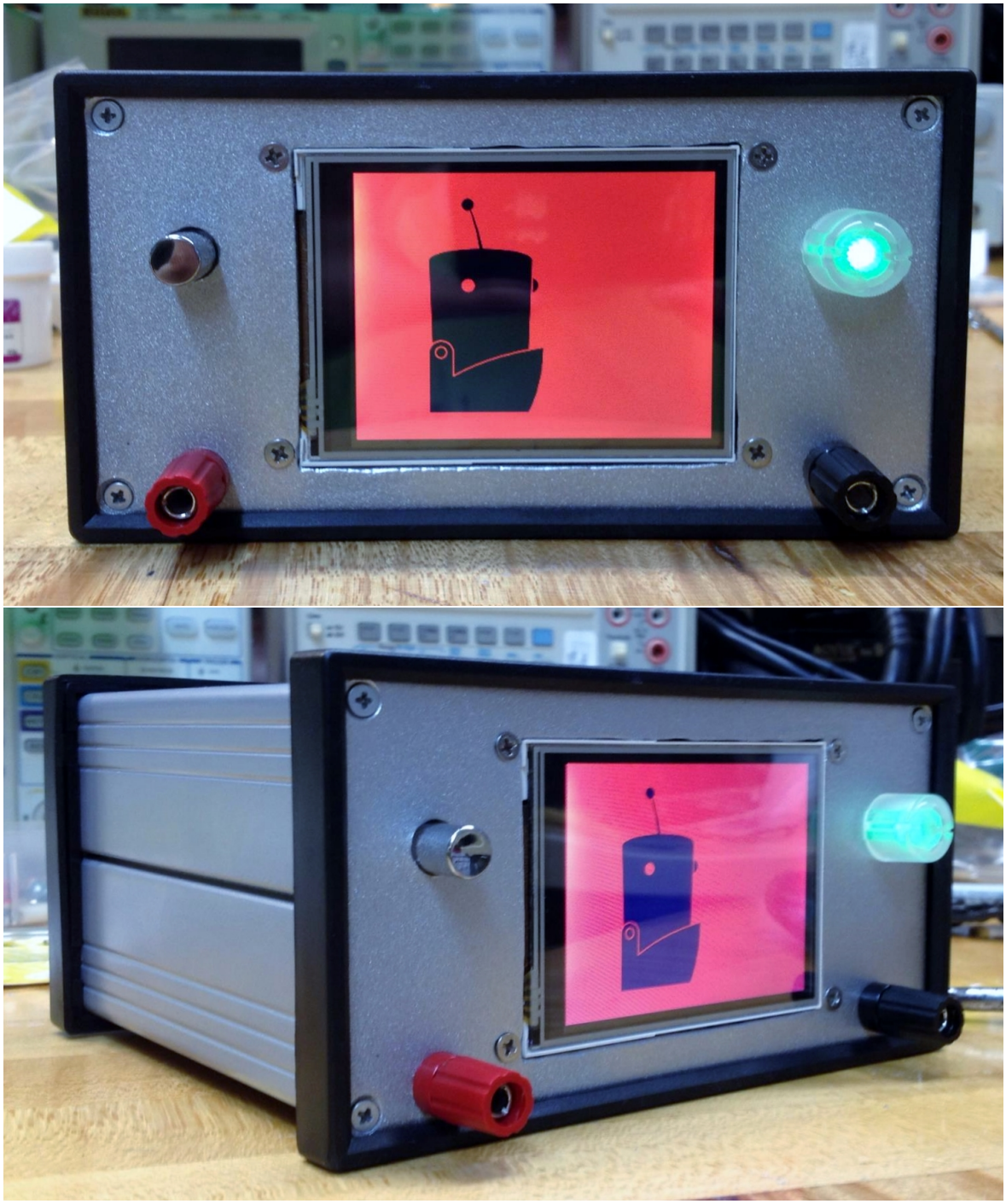


Figure 6.1: The FreeDum Load complete in case and powered on.

REFERENCES

Abracon. (2011, March 11). *HC/49US (AT49) low profile surface mount microprocessor crystal* [Data sheet]. Retrieved from http://www.abracon.com/Resonators/ABLS2.pdf

Array. (n.d.). An Array electronic load [Online image]. Retrieved April 6, 2015 from http://www.array.sh/yq-3721e.htm

Bourns. (2015, March). *PEL12T - 12 mm encoder with switch and illuminated shaft* [Data sheet]. Retrieved from http://www.mouser.com/ds/2/54/PEL12T-48721.pdf

Itead. (2010, November 18). ITead C8051F electronic dummy load kit [Weblog]. Retrieved from http://blog.iteadstudio.com/itead-c8051f-electronic-dummy-load-kit

Johnson, N. (2014). Introducing re:load [Weblog]. Retrieved from http://www.arachnidlabs.com/blog/2013/02/05/introducing-re-load

Jones, D. [EEVblog]. (2010, July 31). *EEVblog #102 - DIY constant current dummy load for power supply and battery testing* [Video file]. Retrieved from https://www.youtube.com/watch?v=8xX2SVcItOA

Lorton, M. J. [mjlorton]. (2012, September 24). *Electronic DC load #5 - Final circuit before build* [Video file]. Retrieved from https://www.youtube.com/watch?v=cvc0XsB7LMk

Lorton, M. J. [mjlorton]. (2013, February 8). *Electronic DC load #7 - Final assembly and build* [Video file]. Retrieved from https://www.youtube.com/watch?v=afM1aujgAF8

Microchip. (2014, August 7). *Dual SMBus PWM fan speed controllers with fan fault detection* [Data sheet]. Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/20001734C.pdf

NXP. (2011, February 7). *BUK9504-40A N-channel trenchMOS logic level FET* [Data sheet]. Retrieved from http://www.nxp.com/documents/data_sheet/BUK9504-40A.pdf

Wiggins, L. [wigman27]. (2013). *Arduino programmable constant current power resistance dummy load*. Retrieved from http://www.instructables.com/id/Arduino-Programmable-Constant-Current-Power-Resist/

Appendix A

FIRMWARE

A.1 LCD Driver

A.1.1 LCD_Support.h

```
//PTC4 - PIN109 = LCD_BLCNT
#define LCD_BLCNT 4
//PTC11 - PIN116 = LCD_RESET
#define LCD_RESET 11
//PTC12 - PIN117 = LCD_RD
#define  LCD_RD      12
//PTC13 - PIN118 = LCD_WR
#define  LCD_WR      13
//PTC18 - PIN125 = LCD_RS
#define  LCD_RS   18
//PTC19 - PIN126 = LCD_CS
#define  LCD_CS      19
//PTD0 - PIN127 = LCD_D0
#define LCD_D0       0
//PTD1 - PIN128 = LCD_D1
#define  LCD_D1      1
//PTD2 - PIN129 = LCD_D2
#define LCD_D2       2
//PTD3 - PIN130 = LCD_D3
#define LCD_D3       3
//PTD4 - PIN131 = LCD_D4
#define LCD_D4       4
//PTD5 - PIN132 = LCD_D5
#define LCD_D5       5
```

```
//PTD6 - PIN133 = LCD_D6
#define LCD_D6        6
//PTD7 - PIN136 = LCD_D7
#define LCD_D7        7
//PTD8 - PIN137 = LCD_D8
#define LCD_D8        8
//PTD9 - PIN138 = LCD_D9
#define LCD_D9        9
//PTD10 - PIN139 = LCD_D10
#define LCD_D10    10
//PTD11 - PIN140 = LCD_D11
#define LCD_D11    11
//PTD12 - PIN141 = LCD_D12
#define LCD_D12    12
//PTD13 - PIN142 = LCD_D13
#define LCD_D13    13
//PTD14 - PIN143 = LCD_D14
#define LCD_D14    14
//PTD15 - PIN144 = LCD_D15
#define LCD_D15    15
#define LCD_DATA_PORT    PTD->PDOR
#define SET_BLCNT     PTC->PSOR |= GPIO_PSOR_PTSO(GPIO_PIN(LCD_BLCNT))
#define  SET_LCD_RESET  PTC->PSOR |=
   GPIO_PSOR_PTSO(GPIO_PIN(LCD_RESET))
#define SET_LCD_RD       PTC->PSOR |= GPIO_PSOR_PTSO(GPIO_PIN(LCD_RD))
#define SET_LCD_WR       PTC->PSOR |= GPIO_PSOR_PTSO(GPIO_PIN(LCD_WR))
#define SET_LCD_RS       PTC->PSOR |= GPIO_PSOR_PTSO(GPIO_PIN(LCD_RS))
#define SET_LCD_CS       PTC->PSOR |= GPIO_PSOR_PTSO(GPIO_PIN(LCD_CS))
#define CLR_BLCNT     PTC->PCOR |= GPIO_PCOR_PTCO(GPIO_PIN(LCD_BLCNT))
#define  CLR_LCD_RESET  PTC->PCOR |=
   GPIO_PCOR_PTCO(GPIO_PIN(LCD_RESET))
#define CLR_LCD_RD       PTC->PCOR |= GPIO_PCOR_PTCO(GPIO_PIN(LCD_RD))
#define CLR_LCD_WR       PTC->PCOR |= GPIO_PCOR_PTCO(GPIO_PIN(LCD_WR))
#define CLR_LCD_RS       PTC->PCOR |= GPIO_PCOR_PTCO(GPIO_PIN(LCD_RS))
#define CLR_LCD_CS       PTC->PCOR |= GPIO_PCOR_PTCO(GPIO_PIN(LCD_CS))
```

```
#define  MAX_X  320
#define  MAX_Y  240

void Write_Command(uint32_t Wcommand);
void Write_Data(uint32_t Wdata);
void Write_Command_Data(uint32_t Wcommand,uint32_t Wdata);
void LCD_Set_Address(uint32_t PX1,uint32_t PY1,uint32_t PX2,uint32_t
   PY2);
void LCD_Init();
void LCD_Fill(uint32_t color);
void LCD_Image(uint32_t pos_x,uint32_t pos_y,uint32_t dim_x,uint32_t
   dim_y,const uint32_t *picture);
```

A.1.2 LCD_Support.c

```
#include "LCD_Support.h"
void LCD_Init()
{
   //Turn on LCD backlight
   //PTC4 - PIN109 = LCD_BLCNT
   SET_BLCNT;
   //PTC12 - PIN117 = LCD_RD
   SET_LCD_RD;
   //PTC11 - PIN116 = LCD_RESET
   //Reset is active low
   //Set LCD Reset high
   SET_LCD_RESET;
   //Delay 5 milliseconds
   time_delay_ms(5);
   //Set LCD reset low
   CLR_LCD_RESET;
   //Delay 15 milliseconds
   time_delay_ms(15);
   //Set LCD Reset high
```

```
    SET_LCD_RESET;
    time_delay_ms(15);
    //PTC19 - PIN126 = LCD_CS
    CLR_LCD_CS;

    time_delay_ms(5);
    Write_Command_Data(0x0000,0x0001);
    Write_Command_Data(0x0003,0xA8A4);
    Write_Command_Data(0x000C,0x0000);
    Write_Command_Data(0x000D,0x080C);
    Write_Command_Data(0x000E,0x2B00);
    Write_Command_Data(0x001E,0x00B7);
    Write_Command_Data(0x0001,0x293F);
    Write_Command_Data(0x0002,0x0600);
    Write_Command_Data(0x0010,0x0000);
    Write_Command_Data(0x0011,0x60B8);
    Write_Command_Data(0x0005,0x0000);
    Write_Command_Data(0x0006,0x0000);
    Write_Command_Data(0x0016,0xEF1C);
    Write_Command_Data(0x0017,0x0003);
    Write_Command_Data(0x0007,0x0233);
    Write_Command_Data(0x000B,0x0000);
    Write_Command_Data(0x000F,0x0000);
    Write_Command_Data(0x0041,0x0000);
    Write_Command_Data(0x0042,0x0000);
    Write_Command_Data(0x0048,0x0000);
    Write_Command_Data(0x0049,0x013F);
    Write_Command_Data(0x004A,0x0000);
    Write_Command_Data(0x004B,0x0000);
    Write_Command_Data(0x0044,0xEF00);
    Write_Command_Data(0x0046,0x013F);
    Write_Command_Data(0x0030,0x0707);
    Write_Command_Data(0x0031,0x0204);
    Write_Command_Data(0x0032,0x0204);
    Write_Command_Data(0x0033,0x0502);
```

```
    Write_Command_Data(0x0034,0x0507);
    Write_Command_Data(0x0035,0x0204);
    Write_Command_Data(0x0036,0x0204);
    Write_Command_Data(0x0037,0x0502);
    Write_Command_Data(0x003B,0x0302);
    Write_Command_Data(0x0023,0x0000);
    Write_Command_Data(0x0024,0x0000);
    Write_Command_Data(0x0025,0x8000);
    Write_Command_Data(0x004f,0x0000);
    Write_Command_Data(0x004e,0x0000);
    Write_Command(0x0022);
    time_delay_ms(1);
    SET_LCD_CS;
    time_delay_ms(1);
}

void Write_Command(uint32_t wcommand)
{
    //TFT_RD = 1;
    SET_LCD_RD;
    //TFT_RS = 0;
    CLR_LCD_RS;
    LCD_DATA_PORT = wcommand;
    //TFT_WR = 0;
    CLR_LCD_WR;
    Short_Pause(1000);
    //TFT_WR = 1;
    SET_LCD_WR;
}

void Write_Data(uint32_t wdata)
{
    //TFT_RD = 1;
    SET_LCD_RD;
```

```
    //TFT_RS = 1;
    SET_LCD_RS;

    LCD_DATA_PORT = wdata;

    //TFT_WR = 0;
    CLR_LCD_WR;

    Short_Pause(1000);

    //TFT_WR = 1;
    SET_LCD_WR;
}

void Write_Command_Data(uint32_t Wcommand,uint32_t Wdata)
{
    Write_Command(Wcommand);
    Write_Data(Wdata);
}

void LCD_Set_Address(uint32_t PX1,uint32_t PY1,uint32_t PX2,uint32_t
    PY2)
{
    Write_Command_Data(0x44,(PX2 << 8) + PX1);  //Column address
       start2
    //Write_Command_Data(0x45,PX1);      //Column address start1
    Write_Command_Data(0x45,PY1);      //Column address start1
    //Write_Command_Data(0x46,PX2);  //Column address end2
    Write_Command_Data(0x46,PY2);  //Column address end2
    //LCD IN PORTRAIT MODE
    //4E => ROW
    Write_Command_Data(0x4E,PX1);      //Column address end1
    //4F => COLUMN
    Write_Command_Data(0x4F,PY1);  //Row address start2
    Write_Command(0x22);
```

```
}

void LCD_Fill(uint32_t color)
{
   uint32_t i,j;
   //TFT_CS  = 0;
   CLR_LCD_CS;
   time_delay_ms(1);
   LCD_Set_Address(0,0,239,319);
   Write_Data(color);
   for(i = 0; i <= 319; i++)
   {
      for(j = 0; j <= 239; j++)
      {
         //TFT_WR = 0;
         CLR_LCD_WR;

         //time_delay_ms(1);

         //TFT_WR = 1;
         SET_LCD_WR;

         //time_delay_ms(1);
      }
   }
   //TFT_CS  = 1;
   SET_LCD_CS;
   time_delay_ms(1);
}

void LCD_Image(uint32_t pos_x,uint32_t pos_y,uint32_t dim_x,uint32_t
   dim_y,const uint32_t *picture){
unsigned int x, y;
   //TFT_CS  = 0;
   CLR_LCD_CS;
```

```
    time_delay_ms(1);
    LCD_Set_Address(pos_x, pos_y, pos_x + dim_x - 1, pos_y + dim_y -
        1);
    for(y = pos_y; y < (pos_y + dim_y); y++ ) {
        for(x = pos_x; x < (pos_x + dim_x); x++ ) {
            Write_Data(*picture++);
        }
    }
    //TFT_CS  = 1;
    SET_LCD_CS;
    time_delay_ms(1);
}
```

A.2 Fan Controller Driver

A.2.1 SMBus_Support.h

```
#define TC655_ADDRESS_WRITE      0x36
#define TC655_ADDRESS_READ    0x37
//RPM Output 1
#define RPM1      0x00
//RPM Output 2
#define RPM2      0x01
//Fan Fault 1 Threshold
#define FAN_FAULT1   0x02
//Fan Fault 2 Threshold
#define FAN_FAULT2   0x03
//Configuration
#define CONFIG       0x04
//Status Register
#define STATUS       0x05
//Fan Speed Duty Cycle
#define DUTY_CYCLE   0x06
//Manufacturer Identification
#define MFR_ID       0x07
```

```
//  Version Identification:
//  X =    0    TC654, X =    1    TC655)
#define VER_ID       0x08
// ====================================================
//    TC655 Duty Cycle Register Specific Defintions
// ====================================================
#define DC30      0x00
#define DC34_67      0x01
#define DC39_33      0x02
#define DC44      0x03
#define DC48_67      0x04
#define DC53_3    0x05
#define DC58      0x06
#define DC62_67      0x07
#define DC67_33      0x08
#define DC72      0x09
#define DC76_67      0x0A
#define DC81_33      0x0B
#define DC86      0x0C
#define DC90_67      0x0D
#define DC95_33      0x0E
#define DC100     0x0F
// ============================================================
//      TC655 Fan Fault Threshold Register Specific Defintions
// ============================================================
//2
#define Threshold_100RPM   0x02
//5
#define Threshold_250RPM   0x05
//10
#define Threshold_500RPM   0x0A
//20
#define Threshold_1000RPM  0x14
//40
#define Threshold_2000RPM  0x28
```

```
//60
#define Threshold_3000RPM  0x3C
//80
#define Threshold_4000RPM  0x50
//100
#define Threshold_5000RPM  0x64
//120
#define Threshold_6000RPM  0x78
//140
#define Threshold_7000RPM  0x8C
//160
#define Threshold_8000RPM  0xA0
//180
#define Threshold_9000RPM  0xB4
//200
#define Threshold_10000RPM 0xC8
//220
#define Threshold_11000RPM 0xDC
//240
#define Threshold_12000RPM 0xF0
// ================================================
//          TC655 and General SMBus Functions
// ================================================
/*
   Function:      SMBus_Init(void)
   Description:
   Configures the SMBus controller to use I2C0 SDA and SCLK
 */
void SMBus_Init(void);
/*
   Function:      void SMBus_StartTX_Write(void)
   Description:
   Starts a SMBus Transmission in Master Mode Write
 */
void SMBus_StartTX_Write(void);
```

```
/*
   Function:      void SMBus_StartTX_Read(void)
   Description:
   Starts a SMBus Transmission in Master Mode Read
 */
void SMBus_StartTX_Read(void);
/*
   Function:      void Wait_For_ACK(void)
   Description:
   If in Write mode:
   Wait until slave device acknowledges receipt of transmission from
      master
   If in Read mode:
   Wait until I2C peripheral acknowledges receipt of data from slave
 */
void Wait_For_ACK(void);
/*
   Function:      void SMBus_Write_Byte(uint8_8)
   Description:
   Write Byte of Data to the SMBus
 */
void SMBus_Write_Byte(uint8_t);
/*
   Function:      void SMBus_StopTX(void)
   Description:
   Sends Stop command onto the SMBus
 */
void SMBus_StopTX(void);
// ============================================
//       TC655 Register Specific Functions
// ============================================
/*
   Function:      uint8_t TC655_Read_Register(uint8_t)
   Arguments:     TC655 register to be read
   Return:        Contents of that register
```

```
   Description:
   Reads one of the various registers of the Microchip TC655
 */
uint8_t TC655_Read_Register(uint8_t);
/*
   Function:      uint8_t TC655_Write_Register(uint8_t, uint8_t)
   Arguments:     TC655 register to be written to
                 Data to be written to the specified register
   Description:
   Writes an 8 bit value to one of the various registers of the
      Microchip TC655
 */
void TC655_Write_Register(uint8_t,uint8_t);
// ==========================================
//    TC655 Operational Specific Functions
// ==========================================
/*
   Function:      void TC655_Fans_Full_On(void)
   Description:
   Turns both fans on to 100%
 */
void TC655_Fans_Full_On(void);
/*
   Function:      void TC655_Fans_Adj_On(uint8_t)
   Arguments:     Duty cycle from 30% to 100%
   Description:
   Turn both fans on with a duty cycle from 30% to 100%
 */
void TC655_Fans_Adj_On(uint8_t);
/*
   Function:      void TC655_Fans_Full_Off(void)
   Description:
   Places TC655 in shutdown mode. Effectively turns off both fans
 */
void TC655_Fans_Full_Off(void);
```

```
/*
   Function:      void TC655_Fans_Temp_Controlled(void)
   Description:
   Both fans' speed is determined by value of temperature sensor
 */
void TC655_Fans_Temp_Controlled(void);
```

A.2.2 SMBus_Support.c

```
void SMBus_Init(void)
{
    //Turn on clock to I2C0 module
   SIM->SCGC4 |= SIM_SCGC4_I2C0_MASK;
   //Setup I2C baud rate and hold times
   //SDA hold time = bus period (s)    mul    SDA hold value
   //SCL start hold time = bus period (s)    mul    SCL start hold
      value
   //SCL stop hold time = bus period (s)    mul    SCL stop hold
      value
   //I2C baud rate = bus speed (Hz)/(mul    SCL divider)
   //     75KHz    =      48MHz    /(  1       640     )
   // To achieve this ICR = 0x2D. Mul = 1
   //             or
   //   75KHz    =      48MHz    /(   2       320     )
   // To achieve this ICR = 0x28. Mul = 2
   //             or
   //   75KHz    =      48MHz    /(   4       160     )
   // To achieve this ICR = 0x20. Mul = 4
   //             or
   //     Current Implementation is: Mul = 1, ICR = 2D
    I2C0->F = 0x2D;

    //Enable I2C module
    I2C0->C1 = I2C_C1_IICEN_MASK;
```

```
    return;
}



void SMBus_StartTX_Write(void)
{
    I2C0->C1 |= I2C_C1_TX_MASK;
     I2C0->C1 |= I2C_C1_MST_MASK;
     //Note TC655 Address = 0b0011 011
     //For a write operation 8th bit = 0
     //Therefore resulting value to write  = 0b00110110
    SMBus_Write_Byte(TC655_ADDRESS_WRITE);
    return;
}

void SMBus_StartTX_Read(void)
{
    I2C0->C1 |= I2C_C1_TX_MASK;
     I2C0->C1 |= I2C_C1_MST_MASK;
     //Note TC655 Address = 0b0011 011
     //For a read operation 8th bit = 1
     //Therefore resulting value to write  = 0b00110111
    SMBus_Write_Byte(TC655_ADDRESS_READ);
    return;
}

void Wait_For_ACK(void)
{
    while((I2C0->S & I2C_S_IICIF_MASK)==0){}
    I2C0->S |= I2C_S_IICIF_MASK;
}


void SMBus_StopTX(void)
{
    I2C0->C1 &= ~I2C_C1_MST_MASK;
```

```
    I2C0->C1 &= ~I2C_C1_TX_MASK;
}

void SMBus_Write_Byte(uint8_t write_data)
{
    I2C0->D = write_data;
}

// ========================================
//    Begin: TC655 SMBus Specific Functions
// ========================================
uint8_t TC655_Read_Register(uint8_t read_register)
{
    uint8_t temp;
    SMBus_StartTX_Write();
    Wait_For_ACK();
    SMBus_Write_Byte(read_register);
    Wait_For_ACK();
    /* Do a repeated start */
    I2C0->C1 |= I2C_C1_RSTA_MASK;
    //Resend TC655 address with LSB = 1 for read operation
    SMBus_Write_Byte(TC655_ADDRESS_READ);
    Wait_For_ACK();
    //Put in Rx Mode
    I2C0->C1 &= (~I2C_C1_TX_MASK);
    //Turn off ACK
    I2C0->C1 |= I2C_C1_TXAK_MASK;
    //Dummy read to clear superfluous data in register
    temp = I2C0->D;
    //Wait until data from slave is received in register
    Wait_For_ACK();
    //Send stop signal
    SMBus_StopTX();
    //Read the actual data that the slave sent
    temp = I2C0->D;
```

```
    return temp;
}

void TC655_Write_Register(uint8_t write_register, uint8_t data)
{
    SMBus_StartTX_Write();
    Wait_For_ACK();
    //Transmit the register to be written to
    SMBus_Write_Byte(write_register);
    Wait_For_ACK();
    //Send data to be written to specified register
    I2C0->D = data;
    Wait_For_ACK();
    //Send stop signal
    SMBus_StopTX();
    //Wait a little while before writing again
    Short_Pause(1000);
    return;
}

void TC655_Fans_Full_On()
{
    //Make sure device is not in Shutdown Mode
    //Set Control of Fan Speed to SMBus
    //0010 1010
    TC655_Write_Register(CONFIG,0x2A);
     //Set Duty Cycle of fans to 100%
    TC655_Write_Register(DUTY_CYCLE, DC100);
    return;
}

void TC655_Fans_Adj_On(uint8_t duty_cycle_percent)
{
    //Make sure device is not in Shutdown Mode
     //Set Control of Fan Speed to SMBus
```

```
    //0010 1010
     TC655_Write_Register(CONFIG,0x2A);
     //Set Duty Cycle of fans to duty_cycle%
     TC655_Write_Register(DUTY_CYCLE, duty_cycle_percent);
    return;
}

void TC655_Fans_Full_Off()
{
    //Place TC655 in Shutdown Mode
     //Set Control of Fan Speed to SMBus
    //0010 1011
     TC655_Write_Register(CONFIG,0x2B);
    return;
}

void TC655_Fans_Temp_Controlled()
{
    //Make sure device is not in Shutdown Mode
    //Set Control of Fan Speed to Thermistor
    //0000 1010
    TC655_Write_Register(CONFIG,0x0A);
    return;
}
```

A.3 Quadrature Encoder Driver

```
void Quad_Enc_Init(void){
    //enable the clock for FTM1
    SIM->SCGC6 |= SIM_SCGC6_FTM1_MASK;
    //enable the counter
    FTM1->MODE |= FTM_MODE_FTMEN_MASK;
    //enable the counter to run in the BDM mode
    FTM1->CONF |= FTM_CONF_BDMMODE(3);
```

```
    //load the Modulo register and counter initial value
    //Use the entire 16bit register before triggering an overflow
    //Maximum Decimal Value of Count before overflow = 65,535
    FTM1->MOD = 0xFFFF;
    FTM1->CNTIN = 0x00;
    //configuring FTM for quadrature mode
    //Turns on the Quadrature encoder and turns on the input filters
    //FTM1->QDCTRL |= FTM_QDCTRL_PHAFLTREN_MASK |
       FTM_QDCTRL_PHBFLTREN_MASK |
    //FTM_QDCTRL_QUADEN_MASK;
     FTM1->QDCTRL |= FTM_QDCTRL_PHBPOL_MASK | FTM_QDCTRL_PHAPOL_MASK
        | FTM_QDCTRL_QUADEN_MASK;
    // Starts the clock timer.
    // Clock Source = System Clock / 4 = 24MHz
    FTM1->SC |= FTM_SC_CLKS(1);
    //configuring the input pins:
    //PORTA  - PTA12 - PIN64 = FTM1_QD_PHA - ALT7 Function
    PORTA->PCR[12] = PORT_PCR_MUX(7); // FTM1 CH0
    //PORTA - PTA13 - PIN65 = FTM1_QD_PHB - ALT7 Function
    PORTA->PCR[13] = PORT_PCR_MUX(7); // FTM1 CH1
    return;
}
```